

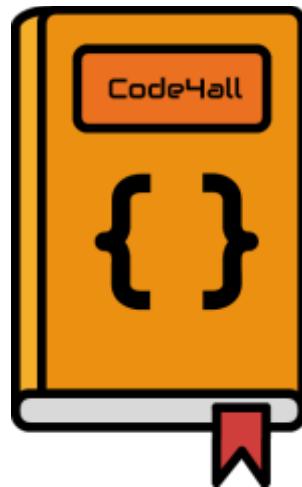


TÉCNICO DE GESTÃO E PROGRAMAÇÃO DE SISTEMAS INFORMÁTICOS

TRIÉNIO 2018/2021

Prova de Aptidão Profissional

Code4All



Prova Realizada por:

José Xavier Nabão Nº 16 12ºTGPSI

Cantanhede,
Ano Letivo 2020/2021

Cofinanciado por:



UNIÃO EUROPEIA
Fundo Social Europeu

Escola Técnico-Profissional de Cantanhede

TÉCNICO DE GESTÃO E PROGRAMAÇÃO DE SISTEMAS INFORMÁTICOS

Code4All

José Xavier Nabão Nº16 12ºTGPSI

Prova de Aptidão Profissional

Equipa de Acompanhamento

Michael Teixeira | Sónia Moço | Ana Raquel | Alexandre Silva | Paulo Soares

Cantanhede,

Ano Letivo 2020/2021

Agradecimentos

Quero, antes de tudo, agradecer à minha família (avós e tios) por me terem apoiado de uma forma incansável, ao longo deste percurso, e por me terem dado as condições perfeitas para que tivesse o melhor desempenho possível.

Gostaria, também, de agradecer ao meu coordenador de curso, Professor Michael Teixeira, pela orientação, coordenação e dedicação. Um dos melhores professores que tive até aos dias de hoje, visto que é empenhado em tudo o que faz, incluindo ensinar, e como é óbvio, o acompanhamento dele neste projeto foi um dos mais essenciais para o bom sucesso do mesmo.

De seguida, passo o agradecimento à minha diretora de turma, Professora Sónia Moço pelo que me sempre ajudou, deu-me a motivação para estar onde estou e dar-me a esperança para me fazer acreditar de que eu sou capaz, o que me fez pensar em prosseguir os meus estudos.

Em terceiro lugar, quero agradecer também à Professora Ana Raquel, por me ter ajudado a corrigir o meu português em todos os aspetos possíveis, e por sempre ter respondido aos meus urgentes pedidos de ajuda.

Falando agora do Professor Alexandre Silva, um dos professores que me deu “mais na cabeça” e sempre com razão, apesar de não termos aulas com ele no 12º ano, os restantes dois anos foram de grande aprendizagem sob todos os aspetos.

Por último, mas não menos importante, o Professor Paulo Soares que me deu os conhecimentos que precisava em relação a desenvolvimento Web e, graças a ele, a participação no *SiteStar* que de certa forma me fez ganhar mais outra paixão pela programação.

A todos, dou o meu mais sincero obrigado!

90% do sucesso
se baseia
simplesmente
Em insistir
(Woody Allen, 2014)

Índice Geral

Introdução	1
Introduction	5
Capítulo I – Pré Projeto	10
Fundamentação.....	10
Objetivos	13
Atividades.....	14
Fase de Definição	14
Fase de Desenvolvimento.....	18
Fase de Manutenção.....	21
Software	22
Android Studio.....	22
Adobe Photoshop CC 2019.....	22
Adobe Illustrator CC 2019	23
Adobe XD.....	23
Capítulo 2 – Projeto.....	24
Instalação do Software	24
Android Studio.....	24
Adobe Photoshop CC 2019.....	32
Adobe Illustrator CC 2019	35
Adobe XD.....	37
Logos.....	39
Base de Dados	42
Diagrama	42
Modelo de Dados	42
Esquema (versão 1).....	43

Esquema (versão 2)	43
Esquema (versão final)	44
Layout.....	45
SplashActivity.....	45
IntroActivity	46
EntrarActivity	50
LoginActivity	51
CriarContaActivity	52
PrincipalActivity	54
MenuActivity.....	55
Plugins Android Studio	56
ADB Idea.....	57
Code Time	58
GitToolBox	59
IDE Features Trainer.....	60
Key Promoter X.....	61
Nyan Progress Bar.....	62
Configuração do Dispositivo Android	63
Desenvolvimento via Emulação do Android	63
Desenvolvimento via Wi-Fi (ADB)	65
Codificação da Aplicação.....	72
Criação do Projeto	72
Configuração do Logo	76
Ligação ao Firebase (Base de Dados)	79
SplashActivity.....	91
IntroActivity	146
“EntrarActivity” (Layout da IntroActivity).....	182

LoginActivity	231
EsquecerPasswordActivity	255
CriarContaActivity	256
InscricoesActivity	374
PrincipalActivity	378
NotificacoesActivity	407
InicioFragment	409
QuizesFragment	411
QuizInfoActivity	416
QuizActivity	421
AvaliacaoQuiz	426
ForumFragment	428
PostActivity	432
PostsActivity (Posts do Utilizador)	439
CriarPostActivity	441
PontuacaoFragment	444
ContaFragment	448
EditarPerfilActivity	450
ContaActivity	454
OfertasFragment	456
DefinicoesFragment	459
SobreFragment	462
AdminActivity	463
InicioAdminFragment	465
ContasFragment	467
PostsVerificacaoFragment	471
CriarQuizFragment	473

ListarQuizFragment.....	482
EditarQuizActivity	486
CriarPostFragment.....	488
ListarPostsFragment	491
EditarPostActivity	493
CriarTagsTemasFragment	495
ListarTagsTemasFragment	498
CriarNotificacaoFragment.....	503
ListarNotificacoesFragment.....	506
EditarNotificacaoActivity	509
ModeradorActivity	511
InicioModFragment	513
EmpresaConfiguracaoActivity	516
EmpresaActivity	519
InicioEmpresaFragment	521
OfertasEmpresaVerificacaoFragment	523
EmpresasFragment.....	526
CriarOfertaFragment	529
CriarOfertaActivity	532
ListarOfertasEmpresaFragment	534
OfertaActivity.....	537
EditarOfertaActivity	539
EmpresaVisualizacaoActivity.....	541
Publicação da App (Google Play Store).....	542
Criação do repositório Code4All (GitHub)	549
Conclusão	550
Bibliografia	552

Anexos	553
Glossário	557

Índice de Ilustrações

Figura 1: Modelo de cascata com retorno	3
Figura 2: Cascade model with return.....	8
Figura 3: Cronograma do projeto.....	18
Figura 4: Android Studio - Início da Instalação	24
Figura 5: Android Studio - Escolha dos Componentes	25
Figura 6: Android Studio - Localização da Instalação do Android Studio	25
Figura 7: Android Studio - Criação da Pasta do Menu Inicial e Atalhos	26
Figura 8: Android Studio - Instalação do Android Studio	26
Figura 9: Android Studio - Instalação Finalizada	27
Figura 10: Android Studio - Ecrã Final.....	27
Figura 11: Android Studio - Instalação do SDK (Software Development Kit)	28
Figura 12: Android Studio - Seleção dos componentes do SDK necessários	28
Figura 13: Android Studio - Revisão da instalação do SDK	29
Figura 14: Android Studio - Realização do download dos componentes SDK	29
Figura 15: Android Studio - Finalização do download dos componentes SDK.....	30
Figura 16: Android Studio – Abertura do Programa	30
Figura 17: Android Studio – Plugins	31
Figura 18: Android Studio - Plugins necessários para o projeto	31
Figura 19: Adobe Photoshop CC 2019 - Ecrã Inicial.....	32
Figura 20: Adobe Photoshop CC 2019 - Opções de Instalação.....	32
Figura 21: Adobe Photoshop CC 2019 - Início do Processo de Instalação.....	33
Figura 22: Adobe Photoshop CC 2019 - Instalação Finalizada.....	33
Figura 23: Adobe Photoshop CC 2019 - Abertura do Programa	34
Figura 24: Adobe Illustrator CC 2019 - Ecrã Inicial	35
Figura 25: Adobe Illustrator CC 2019 - Início do Processo de Instalação	35
Figura 26: Adobe Illustrator CC 2019 - Instalação Finalizada	36
Figura 27: Adobe Illustrator CC 2019 - Abertura do Programa	36
Figura 28: Adobe XD CC 2020 - Ecrã Inicial da Instalação.....	37
Figura 29: Adobe XD CC 2020 - Progresso da Instalação.....	37
Figura 30: Adobe XD CC 2020 - Instalação Finalizada.....	38
Figura 31: Adobe XD CC 2020 - Primeira Inicialização	38

Figura 32: Logo inicial	39
Figura 33: Logo versão 2 (hexagonal)	39
Figura 34: Logo versão 3 (circular)	39
Figura 35: Logo versão 3 (hexagonal)	39
Figura 36: Logo versão 3 (circular)	39
Figura 37: Logo versão 4 (hexagonal)	39
Figura 38: Logo versão 4 (circular)	39
Figura 39: Logo final.....	39
Figura 40: Logo final (livro aberto)	39
Figura 41: Diagrama - Versão 1	43
Figura 42: Diagrama - Versão 2	43
Figura 43: Diagrama - Versão Final.....	44
Figura 44: SplashScreen.....	45
Figura 45: IntroActivity - Página 1.....	46
Figura 46: Introdução - Página 2	47
Figura 47: IntroActivity - Página 3.....	48
Figura 48: IntroActivity - Página 4.....	49
Figura 49: EntrarActivity	50
Figura 50: LoginActivity	51
Figura 51: RegistrarActivity – Utilizador	52
Figura 52: RegistrarActivity - Empresa	53
Figura 53: PrincipalActivity	54
Figura 54: MenuActivity	55
Figura 55: Página dos Plugins do Android Studio.....	56
Figura 56: Plugin "ADB Idea"	57
Figura 57: Pesquisa das funções do Plugin "ADB Idea"	57
Figura 58: Plugin "Code Time"	58
Figura 59: Plugin "Code Time" aberto pela a 1 ^a vez	58
Figura 60: Plugin GitToolBox.....	59
Figura 61: Plugin "IDE Features Trainer"	60
Figura 62: Plugin "Key Promoter X"	61
Figura 63: Plugin “Nyan Progress Bar”	62
Figura 64: Plugin "Nyan Progress Bar" em utilização	62
Figura 65: Página Inicial do Android Studio	63

Figura 66: Opção de configurar os dispositivos android virtuais	63
Figura 67: Dispositivo Android Virtual.....	64
Figura 68: Visualização do site referido acima	65
Figura 69: Localização do arquivo "zip"	65
Figura 70: Explicação de como abrir a linha de comandos.....	66
Figura 71: Depois de pressionar o ENTER, deve surgir esta janela	66
Figura 72: Execução do comando referido acima.....	67
Figura 73: Opção "Sobre o telefone"	68
Figura 74: Sobre o telefone	68
Figura 75: Visualização da opção "Estado" nas Definições "Sobre o telefone"	69
Figura 76: Endereço IP do Dispositivo Android	69
Figura 77: Conexão de ADB entre o computador e o Dispositivo Android	70
Figura 78: Verificação do dispositivo Android.....	71
Figura 79: Aviso de depuração USB	71
Figura 80: Página Inicial do Android - Criar um novo projeto.....	72
Figura 81: Escolha do modelo do projeto	73
Figura 82: Configuração do projeto em si.....	74
Figura 83: Projeto Code4All	75
Figura 84: Pasta mmap com os logos pré-definidos	76
Figura 85: Aspetto final do logo da App.....	77
Figura 86: Conteúdo da pasta transferida anteriormente.....	77
Figura 87: Pasta “mipmap” após a introdução do logo Code4All	78
Figura 88: Fotografia do meu dispositivo Android com o Code4All instalado.....	78
Figura 89: Página inicial da consola do Firebase	79
Figura 90: Nome do Projeto no Firebase	79
Figura 91: Verificação do nome do projeto na Firebase	80
Figura 92: Ligação ao Google Analytics na Firebase.....	80
Figura 93: Configuração do Google Analytics na Firebase	81
Figura 94: Finalização da criação do projeto	81
Figura 95: Página inicial do projeto na Firebase	82
Figura 96: Registro da app na Firebase	82
Figura 97: Chave SHA-1 do projeto "Code4All" no Android Studio	83
Figura 98: Colocação da chave "SHA-1" dentro do registo da app no firebase	84
Figura 99: Download do "google-services.json" para o meu computador	84

Figura 100: Colocação do ficheiro "google-services.json" dentro da pasta "firebase" ..	85
Figura 101: Importação do "google-services.json" para dentro do projeto	85
Figura 102: Ficheiro "google-services.json"	86
Figura 103: Inserção da dependência do Firebase, dentro do "build.gradle" (projeto) .	86
Figura 104: Inserção do código necessário para dentro do ficheiro "build.gradle" (projeto)	87
Figura 105: Inserção da dependência Firebase "BoM" e da "firebase-analytics" (app) 88	88
Figura 106: Abertura do ficheiro "build.gradle" ao nível da "app"	88
Figura 107: Implementação do "Firebase BoM" e do "Firebase Analytics"	89
Figura 108: Ligação da app à Firebase finalizada	90
Figura 109: Criar Atividade "SplashActivity"	91
Figura 110: Escolha do Tipo de Atividade	91
Figura 111: Configuração da Atividade.....	92
Figura 112: "SplashActivity" criada com sucesso	92
Figura 113 Ficheiro AndroidManifest.xml	93
Figura 114: Código para definir a SplashActivity como principal.....	94
Figura 115: Código anteriormente referido para o SplashActivity	94
Figura 116: Parte inicial da SplashActivity	95
Figura 117: Site que usei para colocar as fontes no projeto	95
Figura 118: Site usado para fazer o download da fonte "Montserrat"	96
Figura 119: Arquivo zip com a fonte "Montserrat"	96
Figura 120: Passos para criar uma nova diretória de recursos do Android.....	97
Figura 121: Criação da diretoria do tipo "font"	97
Figura 122: Resultado da pasta "font" com as duas fontes necessárias	98
Figura 123: Ficheiro "colors.xml" com os valores pré-definidos do Android Studio	98
Figura 124: Ficheiro "colors.xml" após da modificação de cores para a SplashActivity	99
Figura 125: Ficheiro "themes.xml" no seu aspeto inicial	100
Figura 126: Ficheiro "themes.xml" após de algumas modificações para a SplashActivity	100
Figura 127: Logo Code4All em PNG	101
Figura 128: Escolher o destino do ficheiro PNG	101
Figura 129: Confirmação da inclusão do logo.....	102
Figura 130: Visualização da pasta "res/drawable" com o logo.....	102
Figura 131: ImageView no layout da SplashActivity	103

Figura 132: Criação do “ImageView” e seleção do logo como Imagem	103
Figura 133: Inserção do Logo no layout	104
Figura 134: Logo no layout com 200dp x 200dp.....	104
Figura 135: Alinhando o logo no centro do layout.....	105
Figura 136: Aviso para colocar uma descrição no logo	105
Figura 137: Criar um valor onde seja armazenado uma string	106
Figura 138: Configuração do valor “String” em relação à descrição da imagem	107
Figura 139: Seleção do recurso logo como descrição da imagem.....	107
Figura 140: Visualização do atributo "contentDescription" no logo	108
Figura 141: Modificação da cor de fundo do layout	108
Figura 142: Configuração da cor de fundo	109
Figura 143: Seleção e confirmação da cor de fundo.....	109
Figura 144: Seleção do TextView como componente.....	110
Figura 145: Colocação do "TextView" no layout	110
Figura 146: Criação de um novo recurso String para o TextView	111
Figura 147: Configuração do recurso String para o TextView.....	111
Figura 148: Definição do recurso anteriormente criado no TextView	112
Figura 149: Resultado após as alterações no layout da SplashActivity	113
Figura 150: Ficheiro "strings.xml" após de ter adicionado os recursos "String"	113
Figura 151: Ficheiro "strings.xml" após das modificações para a "SplashActivity"	113
Figura 152: Erro na propriedade "android:label" visto que o nome do recurso foi alterado	114
Figura 153: Correção da propriedade "android:label"	114
Figura 154: Atualização dos plugins	115
Figura 155: Página do ícone "loader" que transferi.....	115
Figura 156: Ícone de carregamento.....	116
Figura 157: Visualização do ícone dentro do Photoshop	116
Figura 158: Passo de Abrir a "Timeline" no Photoshop	117
Figura 159: GIF de Carregamento finalizado.....	117
Figura 160: Guardar o GIF para a Web	118
Figura 161: Janela para escolher o tipo de exportação	118
Figura 162: Página do StackOverflow - Esconder e mostrar barra de notificações....	119
Figura 163: Código de esconder a barra de superior de ações	120
Figura 164: Código para tornar a atividade em modo de ecrã inteiro	120

Figura 165: Importação da classe "WindowManager"	121
Figura 166: Importação realizada com sucesso	121
Figura 167: Visualização da app após de algumas alterações da UI	122
Figura 168: Téorica do modo de Visualização "FullScreen" de um layout	123
Figura 169: Código para colocar a aplicação em ecrã inteiro	123
Figura 170: Resultado do Layout, após da configuração por código Java	124
Figura 171: Bug do layout, após de bloquear e desbloquear o dispositivo Android ...	125
Figura 172: Solução do bug da barra preta no "Notch" do dispositivo Android	126
Figura 173: Método "esconderInterface" da SplashActivity.....	126
Figura 174: Evocação do método "esconderInterface" no método “onResume”	127
Figura 175: Método “esconderInterface” da "SplashActivity"	127
Figura 176: Layout da "SplashActivity" após da implementação do método "esconderInterface"	128
Figura 177: Solução de colocar um GIF num layout a funcionar corretamente	129
Figura 178: Criação de uma nova classe de Java	130
Figura 179: Definição do nome para a classe a vir a ser criada	130
Figura 180: Abertura pela 1 ^a vez do ficheiro "GifImageView"	131
Figura 181: Colocação do código para colocar o GIF a funcionar corretamente.....	131
Figura 182: Código XML retirado do StackOverFlow	132
Figura 183: Configuração do Componente GIF	132
Figura 184: Alinhamento e tamanho do GIF	133
Figura 185: Ficheiro "carregamento.gif"	133
Figura 186: Código necessário para definir um GIF como recurso do componente...134	134
Figura 187: GIF já pronto e a funcionar corretamente	134
Figura 188: Definição da atividade "SplashActivity" em modo “portrait”.....	135
Figura 189: Método "carregamento()"	136
Figura 190: Invocação do método "carregamento()" dentro do método "onCreate" ...136	136
Figura 191: Verificação ao utilizador para ver se está "logado" ou não	137
Figura 192: método introducao(), com a verificação de autenticação do utilizador ...138	138
Figura 193: Criação do método "buscarUtilizador()"	139
Figura 194: Método "buscarUtilizador()"	139
Figura 195: Criação da Interface "RecuperarIntent"	140
Figura 196: Atribuição da "Intent" para com a Interface "RecuperarIntent"	140

Figura 197: Configuração do novo argumento dentro do método "buscarUtilizador()"	141
Figura 198: Alteração do método "carregamento()"	141
Figura 199: Android Gif Drawable (GitHub)	142
Figura 200: Implementação da libraria "GifImageView"	142
Figura 201: Alteração do elemento para "GifImageView"	143
Figura 202: Solução de como verificar a conexão à Internet	143
Figura 203: Implementação do método de verificar internet	144
Figura 204: Se a Intent fosse "null" finalizava a atividade em si, evitando um crash da app.....	144
Figura 205: Definição da Intent como null, se nenhum grupo fosse encontrado	145
Figura 206: Definição da orientação do ecrã como vertical	145
Figura 207: Libraria "material-intro"	146
Figura 208: Requisitos da libraria "material-intro"	147
Figura 209: Configuração da libraria "material-intro" no ficheiro "build.gradle" relacionado ao módulo	147
Figura 210: Configuração da libraria "material-intro" no ficheiro "build.gradle" do projeto	148
Figura 211: Código disponibilizado na página do "material-intro"	148
Figura 212: Código anteriormente referido após de algumas modificações	149
Figura 213: Correção da propriedade "android:theme" dentro da atividade	149
Figura 214: IntroActivity teste de execução	150
Figura 215: Criação do primeiro layout de introdução ao app.....	150
Figura 216: Configuração do primeiro slide de introdução à app	151
Figura 217: Colocação de um TextView no layout "intro_um.xml"	151
Figura 218: Configuração das três cores básicas no "colors.xml".....	152
Figura 219: Configuração final do primeiro slide na atividade.....	152
Figura 220: Demonstração do primeiro layout	153
Figura 221: Imagem descritiva (intro_um.xml).....	154
Figura 222: Definição do tamanho e largura da imagem	154
Figura 223: Centralização da imagem	155
Figura 224: Configuração do TextView, fonte, tamanho, texto.....	155
Figura 225: Solução de colocar a imagem com bordas redondas	156
Figura 226: Criação do ficheiro xml "efeito_redondo"	156

Figura 227: Modificação da etiqueta "selector" para "shape"	157
Figura 228: Definição do background com o efeito redondo.....	157
Figura 229: Alinhamento da imagem ao centro	158
Figura 230: Alinhamento do texto ao centro	158
Figura 231: Resultado do layout com a imagem e o texto devidamente configurados	159
Figura 232: Remoção do atributo "background" da ImageView	160
Figura 233: Colocação e configuração do CardView	161
Figura 234: Modificação da cor de fundo do layout "intro_um"	161
Figura 235: Definição de uma descrição para a ImageView	162
Figura 236: Primeiro layout de introdução finalizado com sucesso.....	162
Figura 237: Criação do segundo layout de introdução.....	163
Figura 238: Primeira abertura do ficheiro "intro_dois.xml"	163
Figura 239: Criação da cor de fundo do segundo layout de introdução	164
Figura 240: Ficheiro "colors.xml" com as cores iniciais e as duas cores dos layouts.	164
Figura 241: Inserção de uma ImageView, com a imagem anteriormente importada..	165
Figura 242: Seleção do código do CardView do layout anterior	165
Figura 243: Colocação da ImageView dentro do CardView.....	166
Figura 244: Resultado da ImageView, após de ser colocado o CardView	166
Figura 245: Colocação do TextView dentro do segundo layout	167
Figura 246: Criação e definição de um recurso string "descricao_intro_dois"	167
Figura 247: Configuração do segundo layout finalizada	168
Figura 248: Inserção de um novo slide, com o segundo layout	168
Figura 249: Slide um da parte introdutória.....	169
Figura 250: Transição entre o layout um e dois.....	169
Figura 251: Segundo layout de introdução	170
Figura 252: Remoção do atributo "background" do primeiro layout	170
Figura 253: Remoção do atributo "background" do segundo layout	171
Figura 254: Criação do terceiro e último layout	171
Figura 255: Abertura do layout "intro_tres.xml" pela a primeira vez.....	172
Figura 256: Criação do terceiro slide	172
Figura 257: Criação da cor "corIntroTres" para uso no terceiro layout.....	173
Figura 258: Modificação da cor de fundo do terceiro slide.....	173
Figura 259: Seleção da imagem para a ImageView	174

Figura 260: Estado do layout após a inserção de uma ImageView.....	174
Figura 261: Seleção do CardView colocado no layout "intro_dois.xml"	175
Figura 262: Inserção e configuração do CardView e ImageView.....	175
Figura 263: Criação do recurso String com a descrição para o TextView.....	176
Figura 264: Alinhamento do TextView do terceiro layout.....	176
Figura 265: Configuração e ajustes no TextView	177
Figura 266: Colocação da descrição na ImageView	177
Figura 267: Comportamento do terceiro layout após da sua execução	178
Figura 268: Remoção do código para que a IntroActivity deixe de atuar em ecrã inteiro	178
Figura 269: Resultado do layout, com a remoção do código "setFullscreen(true);"	179
Figura 270: Inserção de um novo slide para a "EntrarActivity"	179
Figura 271: Configurações para impedir o utilizador de andar para trás e para a frente	180
Figura 272: Código para esconder os botões de andar para a frente e para trás	180
Figura 273: Código geral na IntroActivity	180
Figura 274: Permissão "READ_EXTERNAL_STORAGE" para permitir a visualização das imagens da galeria	181
Figura 275: Método "validarPermissoes()" da classe "Permissao"	181
Figura 276: Importação do logo do code4all (livro aberto)	182
Figura 277: Inserção de um ImageView para colocar o logo (livro aberto)	182
Figura 278: Configuração do tamanho da ImageView	183
Figura 279: Alinhamento da ImageView	183
Figura 280: Definição da descrição da imagem.....	184
Figura 281: Possível resposta para colocar barras pretas num CardView.....	184
Figura 282: Seleção do CardView no layout "intro_dois.xml"	185
Figura 283: Colocação de um CardView no layout da atividade	185
Figura 284: Criação do ficheiro "borda_do_card_view.xml"	186
Figura 285: Configuração e edição do ficheiro "borda_do_card_view.xml".....	186
Figura 286: Definição do atributo "background" dentro do CardView.....	187
Figura 287: Importação da fonte "Montserrat ExtraBold"	187
Figura 288: Importação da fonte para o projeto com a modificação do seu nome	188
Figura 289: Definição do texto TextView e Modificação do CardView para LinearLayout	189

Figura 290: Modificação do nome do ficheiro "borda_do_card_view" para "borda_preta"	189
Figura 291: Criação do recurso String "slogan"	190
Figura 292: Resultado do TextView (slogan)	190
Figura 293: Configuração do botão "Email"	191
Figura 294: Criação da cor hexadecimal "facebook"	191
Figura 295: Criação da cor hexadecimal "twitter"	192
Figura 296: Configuração das cores de fundo dos botões e pequeno ajuste no texto	193
Figura 297: Execução da atividade com as cores dos botões e com o texto ajustado	193
Figura 298: Ajuste de espaçamento no LinearLayout.....	194
Figura 299: Colocação, configuração e alinhamento do TextView.....	194
Figura 300: Criação do recurso String "criar_conta"	195
Figura 301: Criação de uma cor "corTextoClique"	195
Figura 302: Configuração e alinhamento do TextView	196
Figura 303: Alinhamento do LinearLayout e dos dois TextView's previamente colocados	196
Figura 304: Alinhamento do logo (livro aberto)	197
Figura 305: Site dos ícones open-source (feather-icons)	197
Figura 306: Pasta com os 3 ícones, email, facebook e Twitter	198
Figura 307: Importação dos três ficheiros .svg para dentro do meu projeto	199
Figura 308: Visualização da correta importação dos três ficheiros .svg.....	199
Figura 309: Remoção dos três ícones anteriormente importados	200
Figura 310: Site usado para converter SVG para PNG	200
Figura 311: Importação das três imagens já convertidas para PNG	201
Figura 312: Aumento no "padding" dos três botões.....	201
Figura 313: Configuração dos ícones dentro dos botões.....	202
Figura 314: Execução da atividade, para visualizar os botões e os seus ícones	202
Figura 315: Abertura do Photoshop para aumentar o dpi dos ícones	203
Figura 316: Abertura de um projeto com umas determinadas configurações	204
Figura 317: Exportação do ícone com um tamanho de 24x24 e 150dpi	204
Figura 318: Substituição dos ícones de má qualidade (PNG).....	205

Figura 319: Substituição dos ícones com má qualidade, pelos os ícones com 150dpi	205
Figura 320: Captura de ecrã com os novos ícones melhorados a 150dpi.....	206
Figura 321: Site com a explicação de como importar ficheiros .svg para um projeto no Android Studio.....	207
Figura 322: Remoção dos ficheiros PNG dentro da pasta "desenho/icones"	207
Figura 323: Criação de um novo "Vector Asset" para o projeto no Android Studio	208
Figura 324: Configuração de um novo "Vector Asset"	208
Figura 325: Importação do ficheiro SVG para dentro do projeto no Android Studio ...	209
Figura 326: Configuração do atributo "drawable" do botão email	209
Figura 327: Captura do ecrã do erro que ocorreu ao tentar executar a aplicação	210
Figura 328: Realização de uma reconstrução do projeto.....	210
Figura 329: Modificação do valor "currentColor" para "@android:color/white"	211
Figura 330: Visualização do ícone dentro do botão de email.....	212
Figura 331: Modificação do recurso String "utilizador_novo"	212
Figura 332: Alinhamento dos TextView's localizados na parte inferior do layout	213
Figura 333: Captura de ecrã do layout nesta parte do desenvolvimento	214
Figura 334: Configuração de um "Vector Asset" para o Facebook	214
Figura 335: Correção na cor do ícone do Facebook.....	215
Figura 336: Configuração de um "Vector Asset" para o Twitter	215
Figura 337: Correção na cor do ícone do Twitter.....	216
Figura 338: Atribuição do ícone "ic_facebook"	216
Figura 339: Atribuição do ícone "ic_twitter"	217
Figura 340: Remoção dos três ícones (PNG)	217
Figura 341: Colocação de um "padding" de 8dp no LinearLayout	218
Figura 342: Alteração do "padding" dentro da "Autenticação"	218
Figura 343: Captura do ecrã de como se encontrava o layout.....	219
Figura 344: Colocação de um ouvinte na "IntroActivity"	220
Figura 345: Execução da aplicação para verificar o código escrito.....	220
Figura 346: Modificação do ID do TextView para "textoCriarConta"	221
Figura 347: Criação do método "criarConta" dentro da "IntroActivity"	222
Figura 348: Colocação do atributo "onClick" com o método "entrarConta"	223
Figura 349: Remoção da atividade "EntrarActivity"	223
Figura 350: Modificação do nome do respetivo layout.....	224

Figura 351: Criação do método criarConta().....	224
Figura 352: Atribuição do método "criarConta()" dentro do TextView "Criar conta" ...	225
Figura 353: Desativação total do indicador de páginas de introdução	225
Figura 354: Print do vídeo que visualizei para configurar o "ripple effect".....	226
Figura 355: Criação do ficheiro "borda_preta.xml".....	226
Figura 356: Definição do atributo "background" com o ficheiro "borda_preta"	227
Figura 357: Solução de prevenir uma atividade de lançar mais que uma vez.....	227
Figura 358: Definição do atributo "launchMode" com "singleInstance"	228
Figura 359: Solução de lançar apenas uma atividade	228
Figura 360: Adição da flag, "REORDER_TO_FRONT" dentro de cada uma das Intent's	229
Figura 361: Se executou a app mais que uma vez, apresenta apenas o layout de entrar, se não, apresenta todos.....	229
Figura 362: Reformulação no layout.....	230
Figura 363: Colocação de uma ImageView com o logo (livro aberto)	231
Figura 364: Pesquisa da solução para mudar a cor do botão para laranja	232
Figura 365: Definição do atributo "app:backgroundTint" com a cor primária escura ..	233
Figura 366: Importação do ícone (ic_password)'	233
Figura 367: Modificação da cor do ícone password para branco	234
Figura 368: Captura de ecrã com a atividade em execução	234
Figura 369: PrintScreen do código XML e do Layout.....	235
Figura 370: Configuração do tipo de dados do campo de Email.....	236
Figura 371: Configuração do tipo de dados do campo da Password	236
Figura 372: Configuração da atividade para atuar apenas em modo vertical.....	237
Figura 373: "Firebase Authentication" provedores de login	238
Figura 374: Ativação do "Firebase Authentication" relacionado com "Email/senha" ..	238
Figura 375: Página inicial da documentação do Firebase.....	239
Figura 376: Código de implementação do "Firebase Authentication"	239
Figura 377: Implementação da "Firebase Auth"	240
Figura 378: Criação e atribuição dos EditText (email e password)	241
Figura 379: Modificação do ID do botão "Entrar" para "buttonLogin"	241
Figura 380: Configuração e atribuição do botão "Entrar"	242
Figura 381: Configuração do ouvinte de cliques do botão e recuperação dos textos dos campos	243

Figura 382: Criação da estrutura "if" e colocação dos Toast's de informação.....	243
Figura 383: Criação da classe "DefinicaoFirebase"	244
Figura 384: Criação do método "recuperarAutenticacao()" dentro da classe "DefinicaoFirebase"	245
Figura 385: Condição "if" para verificar se a "task" executou sem problemas	245
Figura 386: Try Catch para as exceções do Firebase Auth (login)	246
Figura 387: Autenticação do utilizador e tratamento de erros.....	247
Figura 388: Criação da atividade "PrincipalActivity".....	248
Figura 389: Definição do método onClick "entrarConta"	248
Figura 390: Criação de um utilizador na FireBase	249
Figura 391: Teste de utilizador não criado.....	250
Figura 392: Teste de campos vazios	250
Figura 393: Teste de login com email e password.....	251
Figura 394: Autenticação do utilizador para a PrincipalActivity	252
Figura 395: Método "buscarUtilizador()"	252
Figura 396: Uso do método "buscarUtilizador()"	253
Figura 397: Solução para trocar a cor de fundo de um botão	253
Figura 398: Troca do elemento "Button" para o "AppCompatButton".....	254
Figura 399: Configuração do método "onClick" do TextView EsquecerPassword	254
Figura 400: Configuração do layout.....	255
Figura 401: Configuração do método "onClick" para reposicionar a password	255
Figura 402: Adição da linha de código para esconder a barra de ações.....	256
Figura 403: Configuração do LinearLayout e EditText.....	257
Figura 404: FeatherIcons - Ícone Utilizador	257
Figura 405: Modificação da cor do ícone	258
Figura 406: Definição de 8dp para as margens do EditText	259
Figura 407: Definição de 16dp para o "padding" do EditText.....	260
Figura 408: Ícone de Calendário (Feather Icons)	260
Figura 409: Modificação do atributo "strokeColor" dentro do ícone (calendário)	261
Figura 410: Resultado do layout após a configuração dos EditText's	262
Figura 411: Solução do Calendário - StackOverflow	263
Figura 412: Solução StackOverflow - EditText não editável	264
Figura 413: Alterações no código de abertura do calendário do EditText – Parte 1... <td>265</td>	265
Figura 414: Alterações no código de abertura do calendário do EditText – Parte 2... <td>266</td>	266

Figura 415: Eliminação dos espaços "vazios" dentro do ficheiro "strings.xml"	267
Figura 416: Alterações do layout consoante o ficheiro "strings.xml"	267
Figura 417: Desativação do EditText (data de nascimento).....	268
Figura 418: Definição do ouvinte "setKeyListener" como vazio "null" do EditText.....	269
Figura 419: Captura de ecrã do Calendário.....	270
Figura 420: Captura do ecrã após de ter selecionado a data de nascimento.....	270
Figura 421: Colocação do Componente "Switch" dentro do layout	271
Figura 422: Cópia do componente (Switch Utilizador) do Adobe XD	271
Figura 423: Importação do ícone "ic_botao_utilizador"	272
Figura 424: Importação do botão para a seleção da empresa	272
Figura 425: Solução de colocar uma cor consoante o estado do botão switch	274
Figura 426: Criação do ficheiro "configuracao_switch"	274
Figura 427: Definição de dois estados do Switch "checked=true" e "checked=false"	275
Figura 428: Troca do "Switch" pelo o "ToogleButton"	276
Figura 429: Recomendação de solução para esconder o texto de estado do "ToogleButton"	277
Figura 430: Configuração do "ToogleButton".....	278
Figura 431: Execução da aplicação para observar novas atualizações	279
Figura 432: Inserção de duas TextView's e alinhamento das mesmas com o "Switch"	279
Figura 433: Configuração do botão "Registar"	280
Figura 434: Colocação do padding do botão registar como "16dp"	280
Figura 435: Alinhamento dos TextView's e da criação de uma "Horizontal Chain"	281
Figura 436: Efeito da "Horizontal Chain" no “rodapé”	281
Figura 437: Definição do atributo "layout_marginTop" como "16dp"	282
Figura 438: Colocação de um "ScrollView" dentro do layout	283
Figura 439: Execução da aplicação para testar o "Scroll"	283
Figura 440: Definição do atributo "padding" de 8dp.....	284
Figura 441: Modificação da cor do botão "Registar"	284
Figura 442: Execução da app, no meu SmartPhone	285
Figura 443: Aviso "size" do ScrollView	285
Figura 444: Organização do código com a criação do método "configuracoesIniciais"	286
Figura 445: Restante código presente na atividade "CriarConta"	286

Figura 446: Criação do método "registarUtilizador"	287
Figura 447: Definição do atributo "onClick" do botão Registar, pelo o método "registarUtilizador"	288
Figura 448: Obtenção dos dados de registro.....	288
Figura 449: Verificação dos campos de registro, se estão vazios ou não.....	289
Figura 450: Criação da variável privada "autenticacao" e definição da mesma	290
Figura 451: Condição "password=confirmarPassword" e código para criar utilizador	291
Figura 452: Exceções para a criação de um utilizzador (Email e Password)	292
Figura 453: Configuração das verificações da FireBase.....	293
Figura 454: Revisão do código realizado.....	294
Figura 455: Registro de um novo utilizador para testar o código anteriormente escrito	295
Figura 456: Redirecionamento do utilizador da CriarContaActivity para a PrincipalActivity	295
Figura 457: Listagem dos utilizadores do FireBase (naquele instante).....	296
Figura 458: Ouvinte para o "ToogleButton"	297
Figura 459: Definição do ToogleButton e do seu ouvinte na atividade.....	297
Figura 460: Pesquisa do ícone para o NIF	298
Figura 461: Definição da cor do ícone NIF (branco)	298
Figura 462: Configuração Campo do NIF (Empresas).....	299
Figura 463: Configuração da cor do ícone "nif" como preto.....	299
Figura 464: Condição IF/SE do ToogleButton (clique)	300
Figura 465: Switch – Utilizador selecionado	301
Figura 466: Switch - Empresa selecionada	301
Figura 467: Ajuste do código no ouvinte do ToogleButton.....	302
Figura 468: Switch - Utilizador (corrigido)	302
Figura 469: Switch - Empresa (corrigido)	303
Figura 470: Avisos na tela com o tipo de utilizador a vir a ser criado.....	303
Figura 471: Definição da visibilidade do campo NIF, como "GONE"	304
Figura 472: Definição da visibilidade do campo "NIF" como "GONE" (layout da atividade)	304
Figura 473: Definição do atributo "inputType" como "number".....	305
Figura 474: Verificação do Tipo de Utilizador (Empresa/Utilizador Normal)	306

Figura 475: Definição do atributo "onClick" com o método "entrarConta" no TextView	307
Figura 476: Criação do método "entrarConta" e criação de dois parâmetros no "registroUtilizador"	308
Figura 477: Remoção dos comentários nas linhas de execução do método "registrarUtilizador"	309
Figura 478: Criação da classe "Utilizador"	309
Figura 479: Configuração das variáveis da classe Utilizador	310
Figura 480: Menu de gerar métodos e outros	310
Figura 481: Seleção das variáveis para a criação dos "Getter's and Setter's"	311
Figura 482: "Getter's and Setter's" da classe Utilizador	311
Figura 483: Renomeação da classe "Utilizador" para "Conta"	312
Figura 484: Criação de uma variável do tipo "Conta" e definição dos métodos	313
Figura 485: Importação da Firebase Database para dentro do projeto	313
Figura 486: Criação do método "recuperarBaseDados"	314
Figura 487: Criação do método "guardar" dentro da classe "Conta"	315
Figura 488: Página "Realtime Database" no painel de controlo do projeto na Firebase	316
Figura 489: Escolha das Regras de segurança	316
Figura 490: Página "Realtime Database"	317
Figura 491: Eliminação do utilizador "josexavier46@outlook.pt"	317
Figura 492: Captura de ecrã com o registo de um novo utilizador	318
Figura 493: Verificação do utilizador na Firebase Auth	318
Figura 494: Listagem dos "nós" vazia	319
Figura 495: Configuração de uma nova variável "id" e do seu "Getter and Setter"	319
Figura 496: Classe "Conta" após à criação da variável "id" e de um "Getter and Setter" para a mesma	320
Figura 497: Definição do "id" do utilizador	320
Figura 498: Alteração na linha 12 - child(this.Id)	321
Figura 499: Exclusão do utilizador criado anteriormente em testes	321
Figura 500: Criação de mais um utilizador de teste	322
Figura 501: Erro na linha 105 - atribuição de um valor "nulo/vazio"	322
Figura 502: Alteração da String fornecida dentro do método "setId"	323
Figura 503: Eliminação do projeto no FireBase	323

Figura 504: Criação de um novo projeto no Firebase	324
Figura 505: Ativação do Google Analytics para o projeto "Code4All"	324
Figura 506: Último passo para a criação do projeto.....	325
Figura 507: Página Inicial do projeto "Code4all"	325
Figura 508: Ativação do Provedor "Email/senha"	326
Figura 509: Configuração da localização do banco de dados (FireBase)	326
Figura 510: Configuração das regras de segurança relativas ao banco de dados.....	327
Figura 511: Dados da "Realtime Database"	327
Figura 512: Adesão do projeto Firebase à app Android (Code4All)	328
Figura 513: Arquivo de configurações "google-services.json"	328
Figura 514: Colocação do ficheiro "google-services.json" dentro do portfolio	329
Figura 515: Remoção do antigo ficheiro "google-services.json" do projeto Code4All (Android Studio)	329
Figura 516: Conclusão da adesão do projeto Firebase para a app (Android)	330
Figura 517: Inserção de uma linha de código para testar a Firebase Database	330
Figura 518: Criação de um utilizador de teste	331
Figura 519: Página "Authentication" com o utilizador de teste já criado.....	331
Figura 520: Visualização dos dados de teste na base de dados (FireBase).....	332
Figura 521: Eliminação da linha de código que enviava dados de teste	332
Figura 522: Alteração da Linha 139 - definição certa do uid do utilizador	333
Figura 523: Criação de um método "Construtor" e alterações no método "guardar()"	334
Figura 524: Criação de um utilizador para testar o envio de dados para a Firebase Database.....	334
Figura 525: Visualização dos dados.....	335
Figura 526: Limpeza da base de dados.....	335
Figura 527: Eliminação da conta de teste anteriormente criada	336
Figura 528: Criação da variável "tipo" e do seu método "Getter and Setter"	336
Figura 529: Criação de duas variáveis privadas do tipo "String"	337
Figura 530: Verificação e recuperação dos valores dos campos de NIF e data de nascimento.....	337
Figura 531: Definição do método "setTipo()" da variável utilizador	338
Figura 532: Firebase Authentication - Utilizador criado com sucesso.....	339
Figura 533: Realtime DataBase - Envio de dados	340
Figura 534: Configuração do método "setTipo()" da classe "Conta" como "empresa"	340

Figura 535: Criação de um utilizador Normal.....	341
Figura 536: Criação de um utilizador do tipo "empresa"	341
Figura 537: Realtime Database - Dados das contas.....	342
Figura 538: Criação do método "limparCampos()"	342
Figura 539: Escrita do código necessário para a limpeza dos campos.....	343
Figura 540: Criação da variável "nome" do tipo "String"	343
Figura 541: Definição da variável "nome" e alteração da condição “IF/SE” (linha 94).....	344
Figura 542: Alteração do método "setNome()" pela a variável "nome" nos dois tipos de conta.....	344
Figura 543: Inicialização das variáveis "nif" e "dataNascimento"	345
Figura 544: Criação de uma "empresa" de teste	346
Figura 545: Realtime Database - Dados do tipo de utilizador "empresa".....	346
Figura 546: Criação de um utilizador "normal" de teste	347
Figura 547: Realtime Database - as duas vertentes de contas.....	347
Figura 548: Listagem dos dois utilizadores criados anteriormente.....	348
Figura 549: Complementos ao método “registroConta()”.....	349
Figura 550: Mudança do código do botão switch do método "onCreate" para o "configuracoesIniciais"	349
Figura 551: Mudança do nome do método "registroConta()" para "registroConta()"	350
Figura 552: Modificação do nome do método "registarUtilizador()" para "verificarDados()"	350
Figura 553: Remoção do “logout” do utilizador e reencaminhamento para a “PrincipalActivity”.....	351
Figura 554: Definição do método "setCorPerfil" e "setCorFundoPerfil"	351
Figura 555: Implementação do Firebase Storage	352
Figura 556: Configuração do Firebase Storage	352
Figura 557: Solução de desenhar as iniciais de um nome como Bitmap	353
Figura 558: Método “drawText()”	353
Figura 559: Organização do código e criação de dois novos métodos	354
Figura 560: Método "guardarNome()"	354
Figura 561: Método guardarFoto().....	355
Figura 562: Criação de um novo campo String "foto"	355
Figura 563: Imagem do Utilizador dentro da Firebase Storage.....	356
Figura 564: Criação do Utilizador	356

Figura 565: Visualização da foto criada para o Utilizador	357
Figura 566: Atualização do perfil, com o URI da foto do utilizador.....	357
Figura 567: Definição da variável conta como global (privada).....	358
Figura 568: Organização do código.....	358
Figura 569: Definição da foto dentro da variável conta e envio dos dados	359
Figura 570: Solução de como perceber se o método “guardar()” foi finalizado	360
Figura 571: Criação da Interface "Validacao" e do método isValidacaoSucesso	360
Figura 572: Introdução da Interface "Validacao" dentro do método guardar().....	361
Figura 573: Modificações dentro do método "conta.guardar()"	361
Figura 574: Método "buscarUtilizador()" e início da próxima atividade	362
Figura 575: Definição do tipo de conta	362
Figura 576: Início da atividade "InscricoesActivity"	363
Figura 577: Objeto "Conta", adição do campo totalXP	363
Figura 578: Parâmetro "tipo" dentro do método "guardarFoto()"	364
Figura 579: Modificação na instância do método "guardarFoto()" com o "conta.getTipo()"	364
Figura 580: Condição se o utilizador for um "membro" ou "empresa"	365
Figura 581: Configuração do caminho da Firebase Database, na criação do utilizador	365
Figura 582: Definição do ID do utilizador, com o método "task.getResult().getUser().getUid()"	366
Figura 583: Configuração do utilizador membro e empresa	366
Figura 584: Solução de como definir um DrawableLeft em Java	367
Figura 585: Criação de um Drawable de uma casa e de uma pessoa	367
Figura 586: Associação dos “drawables” anteriormente criados.....	368
Figura 587: Criação do objeto "Empresa" e associação da empresa.....	368
Figura 588: Ajustes no início da “EmpresaConfiguracaoActivity”	369
Figura 589: Implementação do "Serializable" dentro do Objeto Empresa.....	369
Figura 590: Criação do método "obterIniciais()"	370
Figura 591: String iniciais recebe o retorno do método "obterIniciais()"	370
Figura 592: Verificação das iniciais do nome puder apenas conter o primeiro e último nome.....	371
Figura 593: Configuração do método "onClick" do TextView "Entrar"	371
Figura 594: Criação do campo sexo dentro do objeto "Conta"	372

Figura 595: Método "configurarSpinnerSexo()"	372
Figura 596: Definição do sexo e verificação desse mesmo campo.....	373
Figura 597: Configuração da visualização do SpinnerSexo.....	373
Figura 598: Layout da InscricoesActivity	374
Figura 599: Configuração dos atributos iniciais	374
Figura 600: Configurações iniciais e configuração do método “onItemClickListener” do “ListView”	375
Figura 601: Método "buscarInscricoes()"	375
Figura 602: Método guardarInscricoes()	376
Figura 603: Método limparInscricoes()	376
Figura 604: Captura de ecrã.....	377
Figura 605: Dialog de Carregamento	377
Figura 606: Linha de código para esconder a barra de ações	378
Figura 607: Inserção de um identificador no layout da atividade (TextView)	378
Figura 608: Alinhamento dos ícones de menu e notificações	379
Figura 609: Componente "ImageView" com o logo Code4All (livro aberto)	379
Figura 610: ImageView do logo (livro aberto) e alinhamento dos ícones (menu e notificações).....	380
Figura 611: Layout desta atividade após da configuração dos TextView's	380
Figura 612: Visualização do Layout dentro da app em Android.....	381
Figura 613: Criação de um ficheiro de layout nomeado de "toolbar"	381
Figura 614: Transferência do código que estava no layout "activity_principal" para o "toolbar" (cabecalho)	382
Figura 615: Inclusão do ficheiro "toolbar.xml" dentro do layout "activity_principal.xml"	382
Figura 616: Alinhamentos no TextView "Quizes"	383
Figura 617: Inserção da linha de código para esconder a barra de ações.....	384
Figura 618: Mudança do tema da aplicação	384
Figura 619: Definição da execução de todas as atividades em modo de "portrait" (vertical)	385
Figura 620: Criação da PrincipalActivity (NavigationDrawer)	386
Figura 621: Captura de ecrã (PrincipalActivity).....	386
Figura 622: Código automaticamente gerado na PrincipalActivity	387
Figura 623: Início da atividade (PrincipalActivity) dentro da LoginActivity	387

Figura 624: Início da atividade (PrincipalActivity) dentro da SplashActivity.....	388
Figura 625: Edição no ficheiro "nav_header_main.xml"	388
Figura 626: Solução de como obter a versão da app	389
Figura 627: Configuração do ficheiro "mobile_navigation.xml"	389
Figura 628: Configuração do menu da PrincipalActivity.....	390
Figura 629: Colocação dos ids dentro da AppBarConfiguration	390
Figura 630: Include do ficheiro "toolbar.xml".....	391
Figura 631: Solução de como abrir a Navigation Drawer no clique de um botão	391
Figura 632: Configuração "onClick" do botão para abrir o Navigation Drawer	392
Figura 633: Solução para esconder a Navigation Drawer.....	393
Figura 634: Solução retirada do StackOverflow - Botão para trás fecha o Navigation Drawer	393
Figura 635: Configuração do botão de notificações para abrir a "NotificacaoActivity".....	394
Figura 636: Solução de como mudar a cor das opções do menu (Navigation Drawer)	394
Figura 637: Criação de um novo style "configuracoes_navigation_view".....	395
Figura 638: Mudança de cor do menu (Navigation View)	395
Figura 639: Configuração da opção de sair.....	396
Figura 640: CircleImageView	396
Figura 641: Glide.....	397
Figura 642: Solução de como recuperar a View do Navigation View Header	398
Figura 643: Configuração do Navigation Header e dos seus elementos.....	398
Figura 644: Solução de colocar um rodapé com a versão da app, dentro do Navigation View	399
Figura 645: Rodapé do NavigationView	399
Figura 646: Configuração do TextView do rodapé.....	400
Figura 647: Solução de como guardar dados dentro do dispositivo de execução.....	401
Figura 648: Método "vezesExecucao()"	401
Figura 649: Ficheiro "borda_perfil.xml"	402
Figura 650: Atribuição do drawable "borda_perfil" como background do CircleImageView (Navigation Header)	402
Figura 651: Navigation View da PrincipalActivity	403
Figura 652: método "onResume" vai atualiza a foto e nome do utilizador	403
Figura 653: Método "onClick", sobre o TextView da versão da app.....	404

Figura 654: Método "quantidadeVezesExecucao()"	404
Figura 655: Adição do atributo "background" com o drawable "retangulo_laranja"	405
Figura 656: Execução da PrincipalActivity e visualização do TextView (Versão da App)	405
Figura 657: Fundo laranja na imagem do Perfil	406
Figura 658: Layout da NotificacoesActivity	407
Figura 659: Configurações iniciais e criação do método "buscarNotificacoes()"	407
Figura 660: Exemplo de uma notificação	408
Figura 661: Layout InicioFragment	409
Figura 662: Configurações Iniciais	409
Figura 663: Método "buscarQuizes()"	410
Figura 664: Layout do QuizesFragment	411
Figura 665: Configurações Iniciais	411
Figura 666: Método "buscarInscricoes()"	412
Figura 667: Método "buscarQuizesSubscricao()"	412
Figura 668: Configuração do layout	413
Figura 669: Código do Adapter	413
Figura 670: Código do Adapter 2	414
Figura 671: Código do Adapter 3	415
Figura 672: Código do Adapter 4	415
Figura 673: Google Cloud Platform - Youtube API	416
Figura 674: Download Youtube Android Player API - Download	416
Figura 675: Definição da API Key dentro de uma variável constante	417
Figura 676: Implementação da Libraria "YoutubeAndroidPlayerApi"	417
Figura 677: Configurações Iniciais	418
Figura 678: Método "buscarQuiz()"	418
Figura 679: Método "obterIDYoutube()"	419
Figura 680: Declaração da atividade, com a libraria YoutubeAndroidPlayerApi	419
Figura 681: Método "onInitializationSuccess" do YoutubePlayer	420
Figura 682: Captura de ecrã da QuizInfoActivity	420
Figura 683: Layout da QuizActivity	421
Figura 684: Configurações Iniciais	421
Figura 685: Método "preparacaoInterface()" (excerto de código)	422
Figura 686: Método "proximaPergunta()"	422

Figura 687: Método "limparTudo()"	423
Figura 688: Método "limparCheckBoxs()"	423
Figura 689: Método "limparRadioButtons()"	423
Figura 690: Método "buscarInfoQuizAtual()"	424
Figura 691: Método "buscarQuiz()"	424
Figura 692: Método "adicionarSolucao()" e "limparSolucaoMultipla()"	425
Figura 693: Uma das condições do caso de a solução ser múltipla.....	425
Figura 694: Layout da AvaliacaoQuizActivity.....	426
Figura 695: Método "partilharLinkedIn()"	426
Figura 696: Método "onClick" do FAB	427
Figura 697: Layout do ForumFragment	428
Figura 698: Configurações iniciais.....	428
Figura 699: Configuração do clique no RecyclerView.....	429
Figura 700: Métodos "cliqueCurto()", "filtrar()" e "buscarForum()".....	429
Figura 701: Layout do PostsAdapter	430
Figura 702: Método "onBindViewHolder"	430
Figura 703: Execução do ForumFragment	431
Figura 704: Layout da PostActivity	432
Figura 705: Configurações Iniciais	432
Figura 706: Método "onClick" do botão gosto.....	433
Figura 707: Método "onClick" do botão gosto 2	433
Figura 708: Método "onClick" do botão comentar.....	434
Figura 709: Opção positiva do AlertDialog anteriormente criado	434
Figura 710: Finalização da configuração do AlertDialog	435
Figura 711: Método "buscarComentarios()" e "buscarPost()"	435
Figura 712: Configuração do layout do ComentariosAdapter	436
Figura 713: Método "onBindViewHolder()"	436
Figura 714: Condição que possibilita a edição da administração de todos os comentários	437
Figura 715: Criação do AlertDialog para a edição do comentário	437
Figura 716: Configuração da remoção do comentário	438
Figura 717: Captura de ecrã da PostActivity.....	438
Figura 718: Layout da PostsActivity	439
Figura 719: Configurações Básicas	439

Figura 720: Clique curto e clique longo do RecyclerView	440
Figura 721: Configuração do clique do FloatingActionButton	440
Figura 722: Layout do CriarPostActivity.....	441
Figura 723: Configurações Iniciais	441
Figura 724: Método "verificarCampos()"	442
Figura 725: Método "buscarTags()"	442
Figura 726: Método "criarPost()"	443
Figura 727: Layout da PontuacaoFragment	444
Figura 728: Configuração do RecyclerView.....	444
Figura 729: Configuração do adapter	445
Figura 730: Método "buscarUtilizadores()"	445
Figura 731: Layout do PontuacaoAdapter	446
Figura 732: Método "onBindViewHolder()" e "getItemCount()"	446
Figura 733: Previlépios da app	447
Figura 734: Captura de ecrã da PontuacaoFragment.....	447
Figura 735: Layout ContaFragment.....	448
Figura 736: Configurações Iniciais	448
Figura 737: Método "recuperarDadosUtilizador()"	449
Figura 738: Layout do EditarPerfilActivity	450
Figura 739: Configurações Iniciais	450
Figura 740: Método "recuperarDados()"	451
Figura 741: Método "atualizarCorPerfil()"	451
Figura 742: Método "atualizarCorFundoPerfil()"	452
Figura 743: Método "confirmarAlteracoesPerfil()"	452
Figura 744: Método "atualizarDadosPerfil()"	453
Figura 745: Layout da ContaActivity	454
Figura 746: Método "buscarConta()"	454
Figura 747: Execução da ContaActivity (exemplo)	455
Figura 748: Layout (OfertasFragment)	456
Figura 749: Configurações Básicas do RecyclerView.....	456
Figura 750: Método "buscarOfertas()"	457
Figura 751: Layout (OfertasAdapter)	457
Figura 752: Método "onBindViewHolder()"	458
Figura 753: Layout (DefinicoesFragment)	459

Figura 754: Configurações Iniciais	459
Figura 755: Método "onClick" do botão de atualizar password	460
Figura 756: Método "onClick" do botão de remover conta.....	461
Figura 757: Layout (SobreFragment)	462
Figura 758: Configurações Iniciais e Definição das ações dos botões	462
Figura 759: Configuração da AppBarConfiguration, com os ids dos fragments utilizados nesta atividade	463
Figura 760: Configuração do ficheiro "mobile_navigation_admin.xml"	463
Figura 761: Configuração do ficheiro "menu_admin.xml"	464
Figura 762: Layout InicioAdminFragment.....	465
Figura 763: Configurações Iniciais	465
Figura 764: Método "buscarInfo()"	466
Figura 765: Método "buscarInfo" 2	466
Figura 766: Layout (ContasFragment).....	467
Figura 767: Configurações iniciais e do RecyclerView	467
Figura 768: Método "cliqueCurto()"	468
Figura 769: Método "preencherTipos()"	469
Figura 770: Método "buscarContas()".....	469
Figura 771: Layout (ContasAdminAdapter)	470
Figura 772: Método "onBindViewHolder()"	470
Figura 773: Layout (PostsVerificacaoFragment).....	471
Figura 774: Configurações Iniciais e o método "buscarPosts()".....	471
Figura 775: Layout (PostsVerificacaoAdapter)	472
Figura 776: Método "onBindViewHolder()"	472
Figura 777: Layout (CriarQuizFragment)	473
Figura 778: Parâmetros Iniciais.....	473
Figura 779: Configurações Iniciais dos elementos do Layout	474
Figura 780: Método "onClick" do botão passo seguinte.....	475
Figura 781: Método "onClick" do botão passo seguinte 2.....	475
Figura 782: Método "guardarFoto()" e "guardarPergunta"	476
Figura 783: Método "onActivityResult()"	476
Figura 784: Método "limparCampos()" e "novaPergunta()"	477
Figura 785: Método "verificarCamposPasso3()"	478
Figura 786: Método "abrirDialog()"	478

Figura 787: Método "abrirDialog()" 2	479
Figura 788: Método "definirSolucaoPergunta()" e "passoAnterior()"	479
Figura 789: Método "configuracaoSpinnerPergunta()"	480
Figura 790: Método "adicionarEditText()"	480
Figura 791: Método "removerOpciao()"	481
Figura 792: Método "buscarTemas()"	481
Figura 793: Layout (ListarQuizFragment)	482
Figura 794: RecyclerItemClickListener - GitHub	482
Figura 795: Configurações iniciais e do RecyclerView	483
Figura 796: Método "cliqueLongo()"	483
Figura 797: Método "cliqueCurto()"	484
Figura 798: Layout (QuizesEmpresaAdapter).....	484
Figura 799: Método "onBindViewHolder()"	485
Figura 800: Layout (EditarQuizActivity).....	486
Figura 801: Configurações Iniciais	486
Figura 802: Método "onClick" do botão confirmar alterações	487
Figura 803: Método "onClick" do botão confirmar alterações 2.....	487
Figura 804: Layout (CriarPostFragment)	488
Figura 805: Configurações Iniciais e configuração do método "onClick" do botão criar post.....	488
Figura 806: Método "onActivityResult".....	489
Figura 807: Método "abrirGaleria()"	489
Figura 808: Método "verificarCampos()"	490
Figura 809: Método "criarPost()"	490
Figura 810: Layout (ListarPostsFragment)	491
Figura 811: Layout (PostsAdminAdapter).....	491
Figura 812: Método "onBindViewHolder()"	492
Figura 813: Layout (EditarPostActivity).....	493
Figura 814: Configurações iniciais.....	493
Figura 815: Métodos "buscarTags()" e "buscarPost()"	494
Figura 816: Métodos "confirmarEdicaoPost()" e "atualizarPost()"	494
Figura 817: Layout (CriarTagsTemasFragment).....	495
Figura 818: Configurações Iniciais	495
Figura 819: Método "criarTT()"	496

Figura 820: Método "criarTT()" 2	496
Figura 821: Método "limparCampos()"	497
Figura 822: Layout (ListarTagsTemasFragment)	498
Figura 823: Solução de como criar um ListView	498
Figura 824: Clique curto ListViewTemas	499
Figura 825: Clique curto ListViewTags	499
Figura 826: Clique Longo ListViewTemas	500
Figura 827: Clique Longo ListViewTags	500
Figura 828: Método "buscarTemas()"	501
Figura 829: Método "buscarTags()"	501
Figura 830: Método "trocarTT()"	502
Figura 831: Layout (CriarNotificacaoFragment)	503
Figura 832: Configurações Iniciais	503
Figura 833: Método "onClick" botão de criar notificação	504
Figura 834: Método "onClick" botão de criar notificação 2	504
Figura 835: Método "onActivityResult()"	505
Figura 836: Método "abrirGaleria()"	505
Figura 837: Layout (ListarNotificacoes)	506
Figura 838: Configurações Iniciais	507
Figura 839: Método "buscarNotificacoes()"	507
Figura 840: Método "cliqueCurto()"	508
Figura 841: Método "cliqueLongo"	508
Figura 842: Layout (EditarNotificacaoActivity)	509
Figura 843: Configurações Iniciais	510
Figura 844: Método "buscarNotificacao()"	510
Figura 845: Definição dos ids dentro da AppBarConfiguration	511
Figura 846: Configuração do ficheiro "mobile_navigation_mod.xml"	511
Figura 847: Ficheiro "menu_mod.xml"	512
Figura 848: Layout (InicioModFragment)	513
Figura 849: Configurações Iniciais	513
Figura 850: Método "buscarInfo()"	514
Figura 851: Captura de ecrã	514
Figura 852: Condição para impedir o aparecimento de erros	515
Figura 853: Layout da EmpresaConfiguracaoActivity	516

Figura 854: Configurações iniciais.....	516
Figura 855: Método "onClick" do botão de guardar	517
Figura 856: Método "onActivityResult.....	517
Figura 857: Método “buscarEmpregados()”	518
Figura 858: Configuração dos ids da AppBarConfiguration	519
Figura 859: Configuração do ficheiro "mobile_navigation_empresa.xml".....	519
Figura 860: Ficheiro "menu_empresa.xml"	520
Figura 861: Layout InicioEmpresaFragment	521
Figura 862: Configurações Iniciais	521
Figura 863: Método "buscarInfo()"	522
Figura 864: OfertasEmpresaFragment Layout.....	523
Figura 865: Configuração do RecyclerView e do seu Adapter	523
Figura 866: Método "buscarOfertas()"	524
Figura 867: Layout do OfertasEmpresaVerificacaoAdapter	524
Figura 868: Método "onBindViewHolder()"	525
Figura 869: Configuração do Layout EmpresasFragment	526
Figura 870: Configurações Iniciais	526
Figura 871: Método "filtrar()"	527
Figura 872: Método "buscarEmpresas()"	527
Figura 873: Layout do EmpresasAdapter	528
Figura 874: Configuração do método "onBindViewHolder"	528
Figura 875: Layout do CriarOfertaFragment	529
Figura 876: Configurações Iniciais	529
Figura 877: Método "filtrar()" e "buscarContas()"	530
Figura 878: Layout do ContasEmpresaAdapter	530
Figura 879: Método "onBindViewHolder"	531
Figura 880: Método "filtrarDados()"	531
Figura 881: Layout da CriarOfertaActivity	532
Figura 882: Configurações Iniciais	532
Figura 883: Método "onClick" do botão de criar oferta.....	533
Figura 884: Dialog de Carregamento	533
Figura 885: Layout ListarOfertasEmpresaFragment	534
Figura 886: Configurações Iniciais	534
Figura 887: Layout do OfertasEmpresaAdapter	535

Figura 888: Método "onBindViewHolder()"	535
Figura 889: Função de remover/editar a oferta.....	536
Figura 890: Layout da OfertaActivity	537
Figura 891: Configurações iniciais da atividade.....	538
Figura 892: Layout da EditarOfertaActivity	539
Figura 893: Configurações Iniciais	539
Figura 894: Configuração do botão de confirmar alterações	540
Figura 895: Layout da EmpresaVisualizacaoActivity	541
Figura 896: Configurações iniciais e o método "buscarEmpresa()"	541
Figura 897: Tentativa falhada de envio da app.....	542
Figura 898: Modificação do nome do pacote	542
Figura 899: Reconfiguração da Firebase.....	543
Figura 900: Configuração do Acesso a apps	543
Figura 901: Configuração dos Anúncios.....	544
Figura 902: Classificações de Conteúdo - Categoria.....	544
Figura 903: Classificações de Conteúdo – Questionário	545
Figura 904: Classificações de Conteúdo – Resumo	545
Figura 905: Público-alvo e conteúdo - Faixa etária de segmentação.....	546
Figura 906: Público-alvo e conteúdo – Presença na Google Play Store	546
Figura 907: Público-alvo e conteúdo - Resumo	547
Figura 908: Apps de notícias.....	547
Figura 909: Definições da Loja.....	548
Figura 910: Lançamento da App - Em revisão.....	548
Figura 911: Code4All - Repositório GitHub.....	549
Figura 912: Linhas de código escritas (JAVA + XML).....	551
Figura 913: Ficha 1 - Parte 1	553
Figura 914: Ficha 1 - Parte 2.....	554
Figura 915: Ficha 1 - Parte 3.....	555
Figura 916: Certificado de conclusão - Curso Android.....	556

Índice de Tabelas

Tabela 1: Objetivos gerais e específicos	13
--	----

Introdução

A minha Prova de Aptidão Profissional (PAP) diz respeito à fase final do Curso Profissional de Técnico de Gestão e Programação de Sistemas Informáticos e a mesma teve como principal objetivo a utilização e o abranger dos conhecimentos que adquiri ao longo dos três anos do curso.

Para nome do meu projeto, escolhi “Code4All”. Esta escolha deveu-se a uma aplicação que instalei no meu smartphone, chamada de “Mimo”, que consiste em o seu utilizador realizar simples “quizes” sobre linguagens simples de programação. Com isto, cheguei à conclusão que seria criativo e engraçado, fazer uma aplicação que conseguisse colocar qualquer um a programar, e se auto intitular como um programador, visto que programar é uma coisa divertida, pois não é nada difícil, basta ter vontade de aprender.

O meu objetivo principal ao realizar este projeto, é fazer com que todos os seus utilizadores entrem na mesma e tenham curiosidade em aprender a programar, visto que é uma das maiores paixões que tenho atualmente.

Também queria compartilhar esta paixão para que todos, ao fazer os “quizes” disponibilizados na app, conseguissem encontrar se não o mesmo gosto, pelo menos algum entusiasmo pela programação como eu encontrei quando comecei este curso espetacular.

O projeto consiste então numa aplicação para o Android. Ao entrar na aplicação pela primeira vez, terá uns layouts de explicação do que a app realiza e de seguida, o utilizador irá visualizar uma interface de registo pelo E-mail.

Depois, será redirecionado para o ecrã principal dos “quizes” onde poderá então começar um quiz ou continuar o progresso do mesmo. Terá também no canto superior direito acesso às suas notificações e no canto superior esquerdo, o acesso ao menu...

Avançando mais um pouco, planeei também colocar uma espécie de um fórum de perguntas e respostas muito semelhante ao *stackoverflow*, que é um dos melhores fóruns para programadores, para que todos os utilizadores que tenham dúvida em algum quiz consigam facilmente passar essa dificuldade e entender no que estavam a errar, para de certa forma aprender com o seu erro, gostaria também de implementar uma função da aplicação em que seja permitido aos administradores criarem os seus próprios “quizes”.

Como é óbvio, terei uma parte da app em que irão estar apresentadas algumas configurações que o utilizador poderá fazer, alterar o nome, biografia, entre outros...

Penso ainda, em colocar uma atividade em que será possível ver a pontuação de todos os utilizadores da app, para assim haver um maior “despike” entre os mesmos.

Para dar o devido movimento ao projeto, foi preciso adquirir um curso na plataforma *Udemy*, sobre o desenvolvimento em android com um total de 109 horas de aulas. Este curso foi realizado na sua maior parte, durante as horas vagas que tinha depois das aulas, fins de semana, e foi graças a este curso que retirei as informações base para construir de raiz o meu projeto final de curso.

Quero ainda referir que este projeto será totalmente open source, ou seja, quem quiser modificar o código terá a total liberdade para o efetuar, pelo que no final do mesmo, será publicado no *GitHub*.

Por fim, irei fazer três tipos de interface, uma para os utilizadores normais, uma para os fundadores e outra para os administradores que poderão, na ausência do fundador realizar algumas operações, como criar “quizes”, como já referido anteriormente, quantos estão online, em que quiz estão e o seu devido progresso, o seu nível de experiência entre outros...

Para a realização deste projeto final de curso, decidi usar a metodologia de cascata com retorno, pelo simples facto de ser a melhor que de certa forma encaixa perfeitamente no desenvolvimento da minha aplicação.

Ao contrário da metodologia de cascata que obriga a quem a utiliza, só avançar para a seguinte fase, quando a anterior à mesma estiver totalmente finalizada e passa a não ser possível voltar para a fase anterior visto que não contém retorno, na metodologia de cascata com retorno, a mesma que apliquei no desenvolvimento desta app, é sempre possível voltar para a fase anterior e fazer as alterações necessárias.

Porém, existe sempre um porém, com este modelo existe o risco dessas alterações demorarem muito tempo para ser realizadas, o que provoca o alargamento do objetivo final, isto se não houver um planeamento bem feito antes de começar o projeto. As etapas desta metodologia de cascata com retorno são as seguintes:

- Análise e Definição dos Requisitos;
- Projeto do Sistema;
- Implementação;
- Teste do Sistema;
- Manutenção.

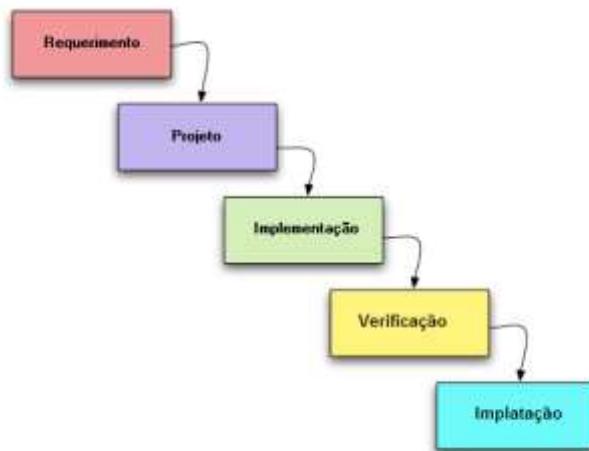


Figura 1: Modelo de cascata com retorno

Com este projeto pretendo ainda, testar os meus limites e consequentemente obter a devida experiência e não só, mas também, evoluir neste ramo da tecnologia, no desenvolvimento mobile, visto que sempre foi uma área que me deixou bastante curioso e também, de certo modo, melhorar a minha habilidade em contornar erros / obstáculos que possam surgir e com isto pretendo ainda melhorar a minha gestão de tempo, tendo sempre tudo planeado.

O relatório da Prova de Aptidão Profissional divide-se em dois capítulos. No primeiro capítulo nomeado de pré-projeto foi desenvolvido o enquadramento teórico do meu projeto, ou seja, onde foi abordado tudo o que diz respeito ao projeto, os objetivos, a fundamentação para a escolha do mesmo, alguns recursos necessários para a sua realização, as atividades com as quais vou trabalhar, e por fim, a calendarização, onde defino os prazos estimados para a conclusão das tarefas.

No segundo capítulo chamado de projeto, passa-se então ao enquadramento prático do projeto, onde apresento tudo o que foi realizado, exemplos disso são, o código realizado, onde tive mais dificuldades e como superei as mesmas, entre outros...

Na conclusão indicam-se algumas considerações finais sobre a realização e elaboração do trabalho realizado nesta Prova de Aptidão Profissional.

Na bibliografia encontram-se alguns sites onde pesquisei a informação necessária para o desenvolvimento da app.

Nos anexos, apresento a ficha 1 e o certificado do curso que realizei extra curricularmente, na plataforma de ensino nomeada de Udemy, este curso foi essencial para a coerente realização desta prova, visto que adquiri conhecimentos que me fizeram melhorar e dominar a programação em dispositivos móveis.

No glossário, indico o significado de alguns termos técnicos utilizados neste relatório e consequentemente uma explicação do mesmo.

Introduction

My Professional Aptitude Test (PAT) concerns the final phase of the Professional Course of Management Technician and Computer Systems Programming and it had as its main objective the acquisition and coverage of the knowledge I have learned over the three years of teaching.

For my project name, I chose “Code4All”. This choice was due to an application I installed on my smartphone, called “Mimo”, which consists of its user performing simple “quizes” on simple programming languages. With this, I came to the conclusion that it would be creative and funny, to make an application that could put anyone to program, and self-entitled as a programmer, since programming is a fun thing, it's not difficult at all, one just need to want to learn.

My main objective in carrying out this project is to make all its users enter the app and become curious to learn to program, as it is one of the biggest passions I currently have.

I also wanted to share this passion so that everyone, when doing the “quizes” available on the app, could find, if not the same taste, at least some enthusiasm for programming as I found when I started this spectacular course.

The project then consists of an application for Android. When entering the application for the first time, you will have some layouts explaining what the app does and then the user will see a registration interface via E-mail.

Then will be redirected to the main quiz screen where you can then start a quiz or continue its progress. You will also have access to your notifications in the upper right corner and in the upper left corner, access to the menu...

Going a little further, I also planned to put up a kind of question and answer forum very similar to stackoverflow, which is one of the best forums for programmers, so that all users who have questions about a quiz can easily get past this difficulty and understand if they are making mistakes, to somehow learn from that mistake, I would also like to implement an application function in which administrators are allowed to create their own "quizes".

Of course, I will have a part of the app that will present some settings that the user can make, change the name, biography, among others...

I am also thinking about putting an activity in which it will be possible to see the score of all users of the app, so that there is a greater "dispatch" between them.

To give the project proper movement, it was necessary to acquire a course on the Udemy platform, on android development, with a total of 109 hours of classes. This course was mostly taken during the free hours I had after classes, on weekends, and it was thanks to this course that I acquired the basic information to build my final course project from scratch.

I also want to mention that this project will be completely open source, that is, whoever wants to modify the code will have complete freedom to do so, so at the end of it, it will be published on GitHub.

Finally, I will make three types of interface, one for normal users, one for founders and one for administrators who will be able, in the absence of the founder, to perform some operations, such as creating "quizes", as mentioned above, how many are online , what quiz they are in and their progress, their level of experience among others...

To carry out this final course project, I decided to use the cascade with feedback methodology, for the simple fact that it is the best one that somehow fits perfectly into the development of my application.

Unlike the cascade methodology, which requires whoever uses it, to only proceed to the next phase, when the one before is fully completed and it is no longer possible to return to the previous phase as it contains no return, in the cascade methodology with return, the same I applied in the development of this app, it is always possible to go back to the previous phase and make the necessary changes.

However, there is always one however, with this model there is the risk that these changes take a long time to be carried out, which causes the extension of the final objective, if there is no good planning done before starting the project. The steps of this cascade with return methodology are as follows:

- Analysis and Definition of Requirements;
- System Project;
- Implementation;
- System Test;
- Maintenance.

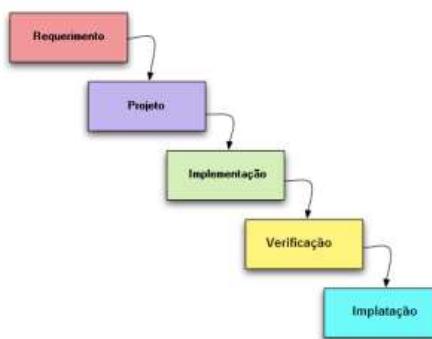


Figura 2: Cascade model with return

With this project I also intend to test my limits and consequently get the proper experience and not only, but also evolve in this field of technology, in mobile development, as it has always been an area that left me very curious and also, in a way, improve my ability to work around errors / obstacles that may arise and with this I intend to improve my time management, always having everything planned.

The Professional Aptitude Test report is divided into two chapters. In the first chapter named pre-project, the theoretical framework of my project was developed, that is, where everything related to the project was addressed, the objectives, the rationale for choosing it, some resources necessary for its realization, the activities I will work with, and finally, the schedule, where I define the estimated deadlines for completing the tasks.

In the second chapter called project, we move on to the practical framework of the project, where I present everything that was carried out, examples of it, the code carried out, where I had the most difficulties and how I overcame them, among others...

In the conclusion, some final considerations about the accomplishment and elaboration of the work carried out in this Professional Aptitude Test are indicated.

In the bibliography there are some sites where I researched the necessary information for the development of the app.

In the annexes, I present the form 1 and the certificate of the course I took extra-curricular, on the teaching platform named Udemy, this course was essential for the coherent completion of this test, as I acquired knowledge that made me improve and master programming on devices furniture.

In the glossary, I indicate the meaning of some technical terms used in this report and, consequently, an explanation of it.

Capítulo I – Pré Projeto

Para realizar e atingir os objetivos que defini para a criação desta aplicação para Android, tive que passar pelas três fases principais dentro da metodologia de cascata com retorno, tais como: Definição, Desenvolvimento e Manutenção.

Na fase de Definição foi desenvolvido todo o trabalho de pesquisa para que houvesse uma boa definição dos objetivos deste projeto, houve também uma análise das ferramentas necessárias para que fosse feito toda a gestão da aplicação corretamente.

Na fase de Desenvolvimento foi realizado o desenho, a codificação e todos os testes derivados ao bom funcionamento da aplicação, os chamados “debugs”.

Por fim, mas não menos importante, a fase de Manutenção, onde foram efetuadas as devidas correções ao código efetuado na fase anterior.

Fundamentação

Começo então por explicar onde tudo começou voltando no tempo, quase no final do 11º ano, instalei uma aplicação no meu dispositivo Android nomeada de “Mimo”, que consistia em “quizes” de várias linguagens de programação, mas infelizmente, esta app tem “in-app purchases”, ou seja, para vermos todos os conteúdos que a aplicação nos disponibiliza temos de os comprar.

Essa aplicação no início, não me chamou nada a atenção, mas quando eu realmente tirei um tempo para a mesma, vi que tinha bastantes funcionalidades, funcionalidades que eu conseguia vir a implementar, visto que estava a realizar um curso específico para desenvolvimento em android.

Curso esse, que já tinha planeado começar a fazer no 11º ano, visto que nessa altura já tinha planeado fazer uma aplicação android para a minha PAP.

Então, foi aí que surgiu o interesse pela aplicação, realizei alguns quizzes que a mesma tem disponível e adorei, é uma experiência magnífica, visto que todos os programadores normalmente fazem a codificação no computador, mas usar aquela aplicação como forma de lazer ou simplesmente para relembrar os conhecimentos, é totalmente diferente do que estar a programar num computador.

Com esta inspiração toda, decidi então criar uma aplicação semelhante e usá-la como projeto final de curso, mas com a diferença de ser totalmente grátis e assim fazer com que todas as pessoas tenham acesso à mesma e que aprendam a programar com apenas alguns cliques no seu dispositivo Android.

De certa forma, ao realizar este projeto, gostaria de aprofundar a área do desenvolvimento para dispositivos móveis, pois sempre foi uma coisa que me fascinou bastante e, não só, mas também ao fazê-lo irei adquirir e consolidar bastantes conhecimentos nesta mesma área, com a finalidade de evoluir neste ramo da programação.

Neste projeto, pretendo utilizar como linguagem principal de programação Java, uma linguagem lançada a 23 de janeiro de 1996 e criada pela empresa estadunidense Sun Microsystems e adquirida pela Oracle, uma empresa bastante famosa, por outro lado, a linguagem Kotlin, foi desenvolvida pela JetBrains em 2011, uma empresa conhecida no ramo de informática como desenvolvedora de ferramentas de desenvolvimento para engenheiros de software, esta é baseada em Dart, Java ou Swift.

Se tiver tempo que sobre, pretendo ainda trocar o código todo da aplicação de Java para Kotlin, visto que a Google está com alguns problemas com a Oracle, a “dona” da linguagem Java.

A IDE que pretendo usar em todo o código será o próprio Android Studio, visto que é o programa aconselhado e desenvolvido pela Google, a criadora do Android.

Um ponto que pretendo dizer ainda, é que esta app será totalmente “responsiva”, o que quero dizer com isto é, nos telemóveis Android mais recentes, a partir da versão 9 (Nougat), foram disponibilizados o modo escuro e o modo claro, e consoante cada um, as próprias cores e interface adaptam-se a cada um. Basicamente, também pretendo implementar isso nesta aplicação visto que já é considerado “habitual” em qualquer uma.

Por fim, uma das outras razões para escolher este projeto, é eu ter o objetivo pessoal de fazer algo que de certa forma influencie a comunidade, neste caso gostaria que as pessoas conseguissem, através de uma aplicação, aprender como programar de uma forma intuitiva, pois a programação em si, não é como o que muitos dizem um “bicho de 7 cabeças”, e fazer com que as pessoas deixem de ter essa ideia sobre a programação e realmente vejam, o quanto divertido é programar!

Objetivos

Objetivos	
Gerais	Específicos
Finalização do curso	<ul style="list-style-type: none"> • Ter uma boa nota da Prova de Aptidão Profissional; • Ter uma boa média no fim do curso.
Divulgação da aplicação	<ul style="list-style-type: none"> • Colocar a aplicação na Google Play Store (Loja de Apps do <i>Android</i>); • Ver a aplicação que criei a ser utilizada por várias pessoas; • Conseguir com que a aplicação tenha uma boa classificação na Play Store; • Partilhar o código fonte no GitHub, fazendo assim, com que a minha aplicação seja Open-Source.
Desenvolvimento da aplicação	<ul style="list-style-type: none"> • Aumentar os meus conhecimentos de programação; • Aprender Kotlin e Java; • Explorar a Base de dados Firebase; • Melhorar a minha capacidade de raciocínio lógico em programação.

Tabela 1: Objetivos gerais e específicos

Atividades

Fase de Definição

- Pesquisa Bibliográfica: Pesquisa de conteúdos e informação necessária para a realização do projeto;

A pesquisa bibliográfica é a base que sustenta qualquer pesquisa científica, neste caso não é diferente, foi aqui que realizei a aprendizagem da área em estudo (desenvolvimento para smartphones), onde realizei um estudo sobre quais as tecnologias iria precisar para o desenvolvimento deste projeto final de curso.

- Análise do Sistema: Análise dos recursos para a realização do projeto;

Laptop ASUS-X540LJ

- Processador: Intel i3 5005U com Intel Graphics (2.00Ghz Quad-Core);
- Disco de Armazenamento: Disco Rígido 1TB (Toshiba);
- RAM: 12 GB DDR3L (1600 MHz);
- Placa Gráfica (Dedicada): NVIDIA GeForce 920M 2Gb VRAM;
- Sistema Operativo: Windows 10 Pro de 64 bits;
- Ecrã: 1366x768.

Desktop Lenovo Idea Centre 720-18APR

- Processador: AMD Ryzen 5 2400G com Vega 11 (3.90Ghz Octa-Core);
- Disco de Armazenamento: SSD Team Group 240GB + Disco Rígido de 2TB (Seagate);
- RAM: 8 GB DDR4 (2667 MHz);
- Placa Gráfica (Integrada): Radeon Vega 11 1Gb VRAM;
- Sistema Operativo: Windows 10 Home de 64 bits;
- Fonte de Alimentação: HuntKey 80 PLUS BRONZE (180 Watts);
- Ecrã 1: 1920x1200;
- Ecrã 2: 1280x1024.

Huawei Mate 20 Lite – Android 10

- Processador: Kirin 710 Octa-Core (4 x 2.2 Ghz e 4 x 1.7 Ghz);
- Armazenamento: 64GB;
- RAM: 4 GB;
- Sistema Operativo: EMUI 10 (Android 10);
- Bateria: 3750 mAh;
- Ecrã: 6.3 IPS Full HD+.

- Análise dos Requisitos: Levantamento dos requisitos mínimos para a realização do projeto;

Android Studio 4.1

- Sistema Operativo: Microsoft® Windows® 7/8/10 (64-bit);
- RAM: 4 GB RAM mínimo, 8 GB RAM recomendado;
- Espaço em disco rígido: 2 GB no mínimo de disco disponível 4 GB recomendados (500 MB para a IDE + 1.5 GB para o Android SDK e a imagem do emulador android);
- Resolução de ecrã: 1280 x 800 de resolução de ecrã no mínimo.

Adobe Photoshop CC 2019

- Processador: Intel® ou AMD compatível com 64 bits; 2 GHz ou mais rápido
- Sistema operacional: Windows 7 com Service Pack 1 (64 bits) ou Windows 10 (atualização de outubro de 2018 – 64 bits – versão 1809 ou posterior)
- Memória RAM: 2 GB ou mais (8 GB recomendados)
- Placa de vídeo: NVIDIA GeForce GTX 1050 ou equivalente (recomenda-se NVIDIA GeForce GTX 1660 ou Quadro T1000);
- Espaço em disco rígido: 3,1 GB ou mais de espaço disponível em disco rígido para a instalação (requer espaço livre adicional durante o processo);
- Resolução do monitor: 1280x800;

Adobe Illustrator CC 2019

- Processador: Intel Pentium 4 ou AMD Athlon 64
- Sistema Operativo: Microsoft Windows 7 com Service Pack 1, Windows 8.1 ou Windows 10
- RAM: 1GB de RAM (recomenda-se 3 GB) para 32 bits; 2 GB de RAM (recomenda-se 8 GB) para 64 bits;
- Espaço em disco rígido: 2 GB de espaço livre em disco para a instalação; requer espaço livre durante a instalação (não é possível instalar em dispositivos de armazenamento removíveis flash)
- Ecrã: Resolução de 1024 x 768 (recomenda-se 1280 x 800)

Adobe XD 2020

- Processador: 2 GHz;
- Sistema Operativo: Atualização de Aniversário do Windows 10 (64 bits) - Versão 1607 (compilação 10.0.14393) ou posterior com as seguintes configurações mínimas:
- Memória: 4 GB;
- Espaço em Disco Rígido: 2 GB de espaço disponível no disco rígido para a instalação; requer espaço livre adicional durante a instalação;
- Ecrã: 1280 x 800;
- Gráficos: para GPU Intel, drivers mais recentes que a versão 8.15.10.2702.

Fase de Desenvolvimento

- Planeamento do Projeto: Elaboração do cronograma do projeto com atividades definidas temporalmente (Calendarização);

No planeamento do projeto, que fiz com ajuda do meu coordenador de curso, que me forneceu uma tabela no Excel, onde foram colocadas as datas do início deste ano letivo até ao final do mesmo, e neste documento encontram-se também as três fases da PAP, sendo elas, definição, desenvolvimento e manutenção.

Com isto, fui pintando os quadrados que eram referentes às semanas de cada mês, para que no final, tivesse uma estimativa de quanto tempo iria demorar cada fase.

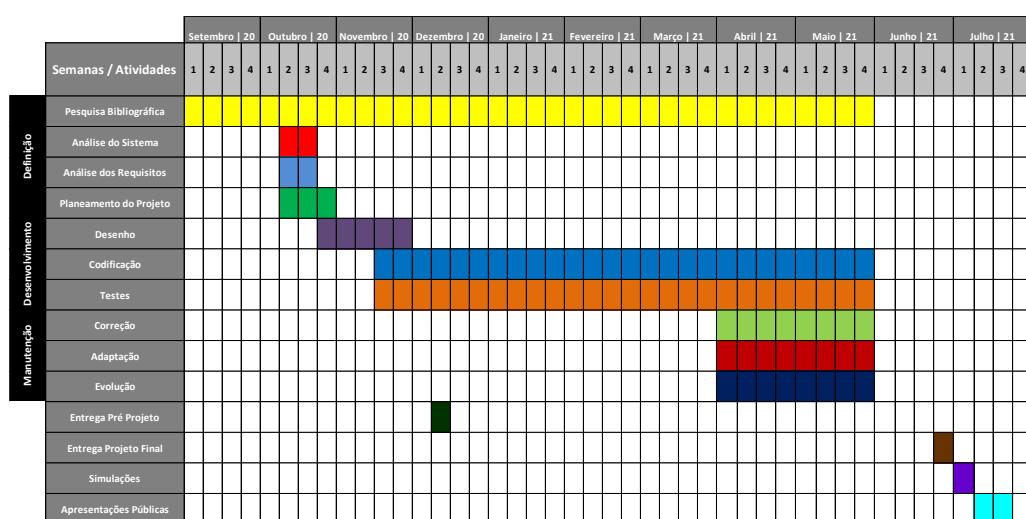


Figura 3: Cronograma do projeto

Neste cronograma realizado por mim, tive como objetivo prever o tempo que demoraria em cada fase deste projeto final de curso, começando então com a pesquisa bibliográfica, coloquei o tempo todo visto que é uma das atividades que vou estar constantemente a realizar, visto que quando surgir algum obstáculo vou ter de me “dirigir” à internet e procurar a solução do mesmo.

Continuando, tanto a análise do sistema como a análise dos requisitos têm o mesmo tempo, visto que basta apenas pesquisar pelo hardware que vou utilizar na realização deste projeto e também o software necessário e os seus requisitos mínimos.

No planeamento do projeto foi onde em três semanas realizei, por passos, por onde iria começar. Nesta atividade também realizei o cronograma como mostrado acima.

Na atividade seguinte, o desenho, foi aqui que realizei o logo, layout da aplicação e edições de foto que foram necessárias para o projeto.

Na codificação e nos testes, ambos com o mesmo tempo, foi aqui em que a aplicação ganhou a sua forma, na codificação fazia a construção do código e nos testes realizava os testes do código para que não houvesse nenhum erro.

Na correção, adaptação e evolução, mais uma vez todos com o mesmo tempo, é aqui que eu já com a aplicação “pronta” irei adicionar funcionalidades e tratar de possíveis erros e fazer as alterações necessárias para corrigir os mesmos. Deixei ainda o mês de junho livre, para tratar nesse mês de um site de suporte para completar a aplicação.

Por fim, tenho as datas importantes, tais como, a entrega do pré-projeto, a entrega do projeto final, as simulações e o mais importante as apresentações públicas.

- Desenho: Layout da aplicação, base de dados, criação de imagens e do logo;

O desenho para mim, é umas das fases mais importantes, visto que tem de ser realizada com muita concentração, de certa forma, pretende-se realizar o layout da aplicação, criação de algumas imagens e do tão importante logo, o que nomeei anteriormente são exemplos de coisas que estão constantemente à vista do utilizador, ou seja, ele terá contato com elas a maior parte do tempo, logo, tem que ser feita uma análise pormenorizada para que não haja falhas em nenhuma das partes.

Na criação da base de dados, optei por primeiramente realizar uma versão inicial da mesma, com as partes mais simples da aplicação, e à medida que vou evoluindo, aí sim, vou aumentando o grau de dificuldade da mesma e também as funções que a aplicação terá.

- Codificação: Código necessário para o funcionamento da aplicação.

A codificação será a parte que fará todo o projeto “ter pernas para andar” é aqui que irei realizar e criar o código necessário, não só, mas também, irei fazer a verificação do tempo de execução e otimizar de certa forma o código, para que não haja perdas de desempenho desnecessárias nem falhas de segurança.

- Teste: Realizar testes na aplicação e verificar o seu funcionamento.

Os tão famosos testes, é aqui que vou passar com certeza a maior parte do tempo, para que tudo fique no lugar e a funcionar a 100%, e para que à medida que vá fazendo a codificação vá também realizando testes e corrigindo o seu funcionamento e também alguns eventuais erros que poderão existir.

Fase de Manutenção

- Correção: Correção de falhas no código;

Na correção, irei fazer uma revisão do código escrito e então fazer melhoramentos para que o mesmo execute sem problemas, sem avisos nenhuns e também sem “quedas/perdas” de desempenho.

- Adaptação: Necessidade de mudar algo no projeto, o mesmo que um melhoramento;

Nesta parte da manutenção pretendo, modificar o código que foi anteriormente corrigido e adaptá-lo consoante a necessidade de ser mais responsivo e user-friendly, ou seja, adaptar o código que realizei numa interface bonita e apresentável.

- Evolução: Acrescimento de funcionalidades na aplicação.

Por fim, na evolução, pretendo evoluir a minha aplicação para colocar na mesma mais funcionalidades, aqui é onde será feita codificação à parte visto que vai ser utilizada para o melhoramento da app e mais tarde corrigida e revista.

Com isto, vai ter de haver alterações na base de dados e no resto das fases, mas como estou a utilizar o modelo de cascata com retorno, posso, sempre que quiser voltar para as fases anteriores ou vice-versa.

Software

Aqui, vou passar a explicar, o porquê da escolha dos programas que utilizei no desenvolvimento.

Android Studio

Escolhi este programa como o programa principal para codificar, visto que foi realizado pela a Google e esta é a criadora da Android, logo terei tudo “do bom e do melhor” na realização deste projeto e no que diz respeito a atualizações do Android e das funcionalidades mais recentes.

Este foi baseado no seu “rival”, IntelliJ Idea, criado pela a JetBrains, uma empresa bastante conhecida por distribuir IDE’s para os programadores.

Mesmo assim, continuei por optar pelo o Android Studio, visto que é a IDE nativa para o desenvolvimento em Android.

Adobe Photoshop CC 2019

Este programa é utilizado para edição de imagens, recorte de mesmas e ainda algumas alterações nas mesmas.

Coloquei este programa e não outro, porque sempre fui fã da Adobe e dos seus programas e para mim não existe melhor programa para editar fotos do que o Photoshop, o problema mesmo é o seu elevado preço, mas de resto é a melhor ferramenta que encontrei.

Adobe Illustrator CC 2019

O Adobe Illustrator, serve mais para a criação de desenhos e ArtWorks. Este programa foi bastante útil ao fazer os meus logos e sem este programa não tinha conseguido fazer o meu logo do projeto.

Optei por esta aplicação, visto que não conhecia outra que fizesse tão bem o trabalho que esta faz e mais uma vez também a escolhi por ser um dos produtos da Adobe.

Adobe XD

O Adobe XD é um programa bastante utilizado nos dias de hoje, não pelos programadores, mas sim pelos designers, visto que este programa permite construir layouts de aplicações de mobile ou websites, sem qualquer necessidade de nenhuma linha de código.

Com esta aplicação aprendi bastante e foi graças a ela que fiz o planeamento total da minha aplicação, em relação a layouts, o que me poupou bastante tempo, visto que com o layout planeado era só escrever código.

Esta aplicação sim, foi uma grande descoberta e é a mais indicada para fazer layouts sem qualquer tipo de código.

Capítulo 2 – Projeto

É neste capítulo que irei tratar tudo sobre o desenvolvimento do projeto, passo então ao enquadramento prático do projeto, onde apresento tudo o que realizei, os problemas que tive e como os solucionei.

Instalação do Software

Nesta parte, irei instalar os programas necessários para a realização deste projeto final de curso.

Android Studio

Utilizei este programa (IDE), para programar o código todo da minha aplicação, posso dizer então que esta aplicação é a principal para o desenvolvimento da mesma.



Figura 4: Android Studio - Início da Instalação

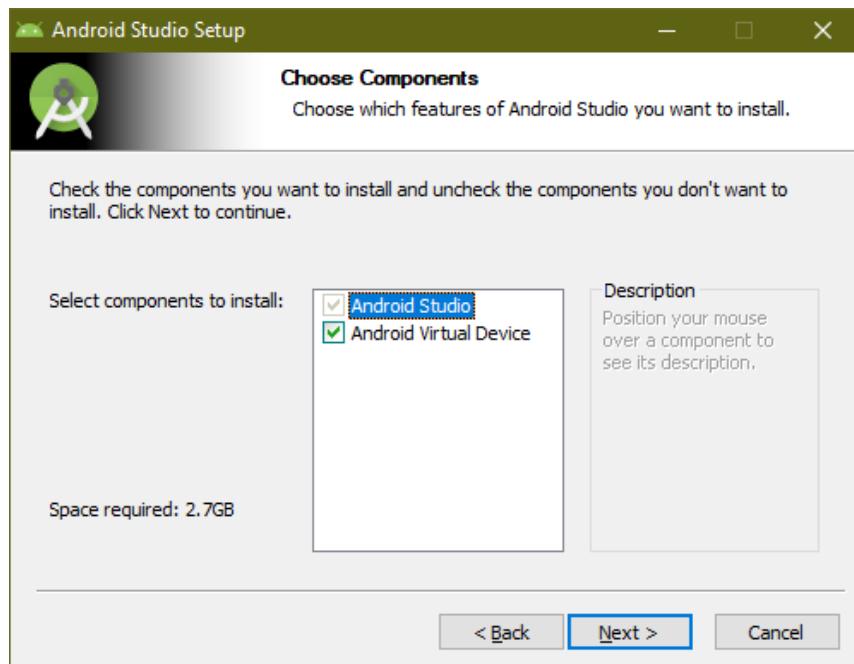


Figura 5: Android Studio - Escolha dos Componentes

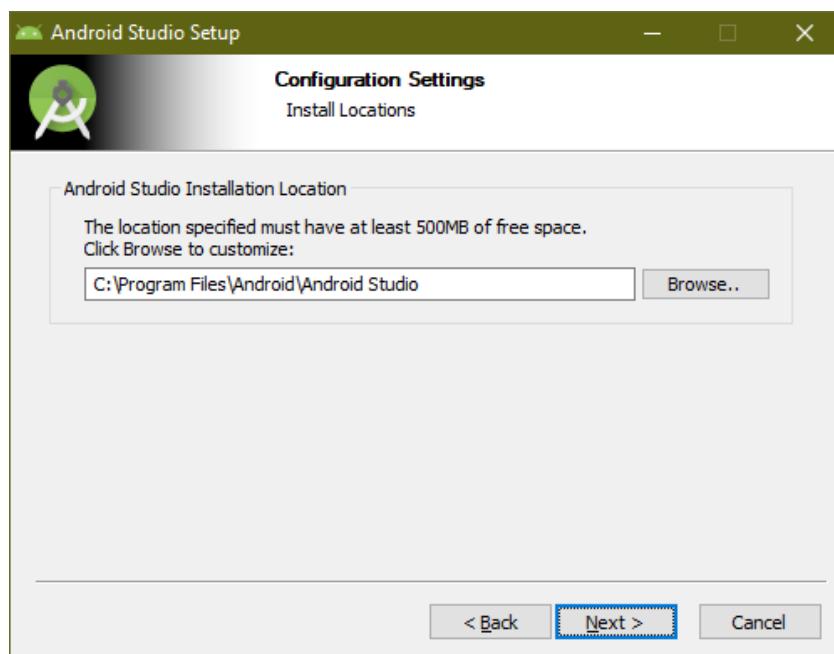


Figura 6: Android Studio - Localização da Instalação do Android Studio

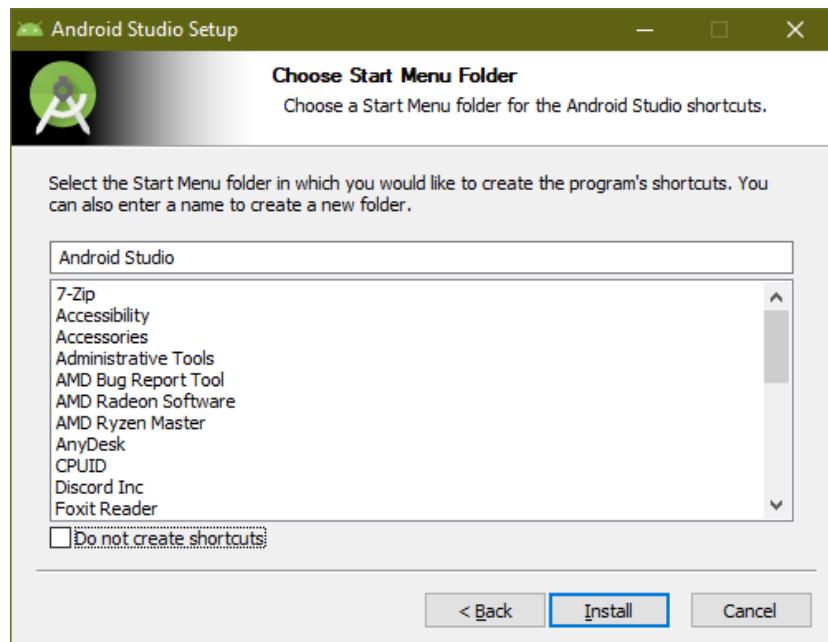


Figura 7: Android Studio - Criação da Pasta do Menu Inicial e Atalhos

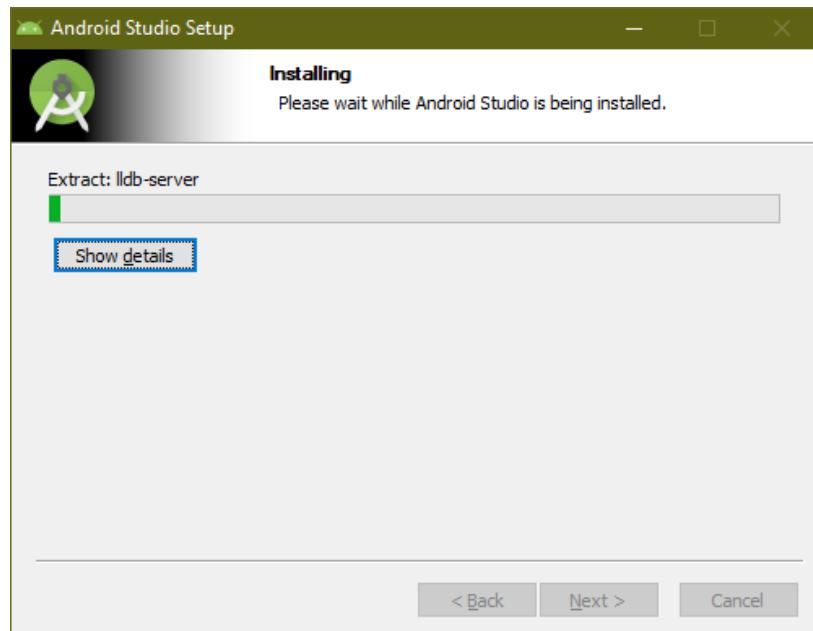


Figura 8: Android Studio - Instalação do Android Studio

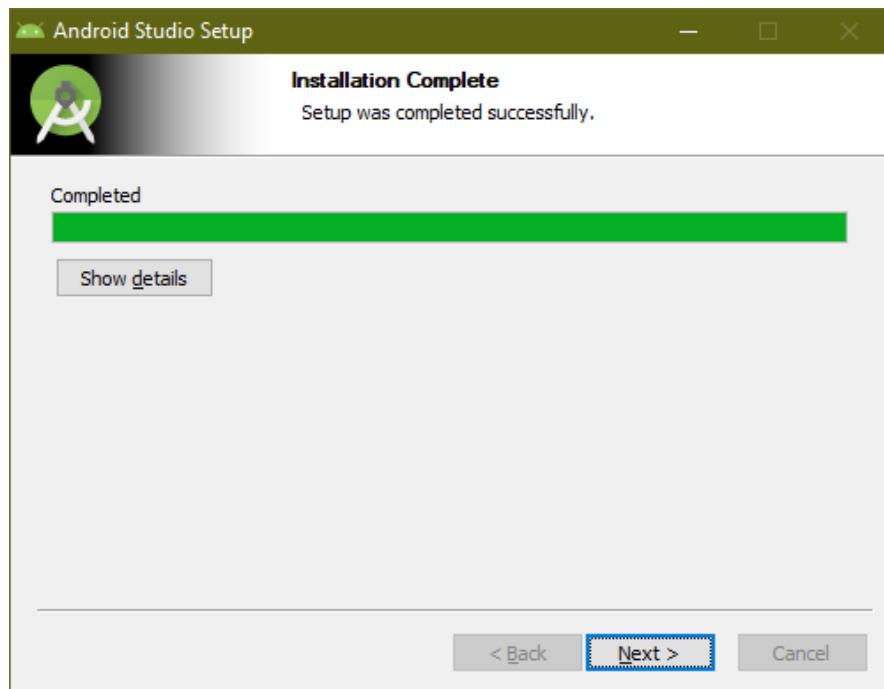


Figura 9: Android Studio - Instalação Finalizada

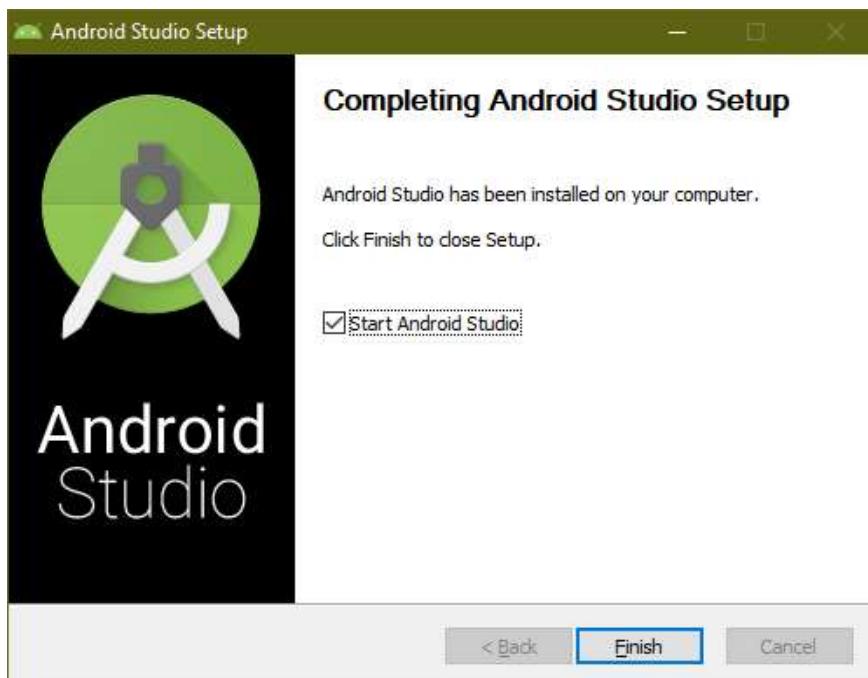


Figura 10: Android Studio - Ecrã Final

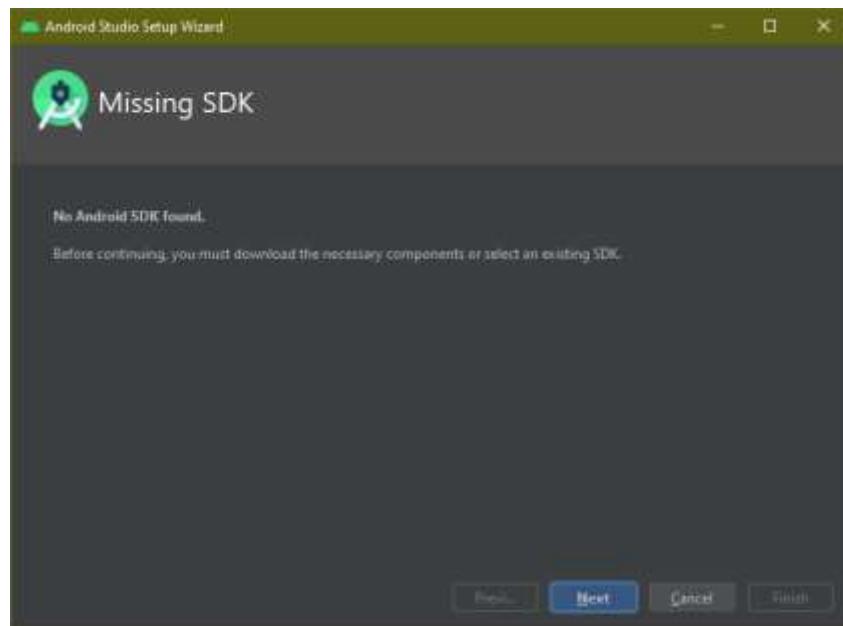


Figura 11: Android Studio - Instalação do SDK (Software Development Kit)

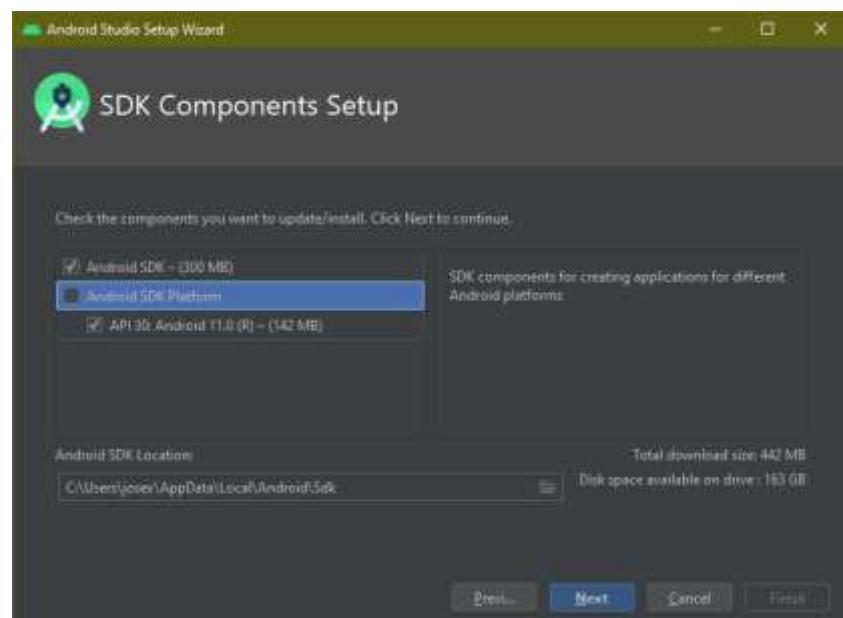


Figura 12: Android Studio - Seleção dos componentes do SDK necessários

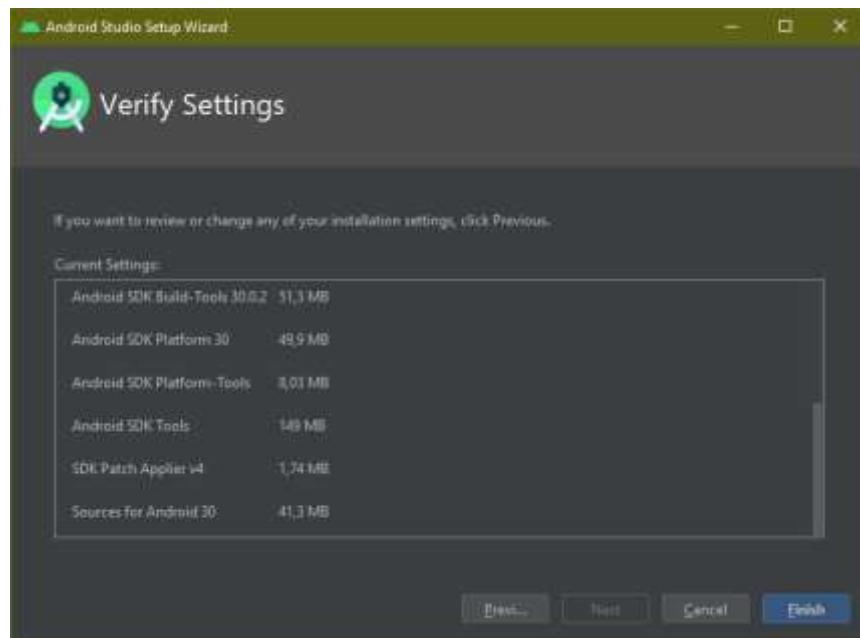


Figura 13: Android Studio - Revisão da instalação do SDK

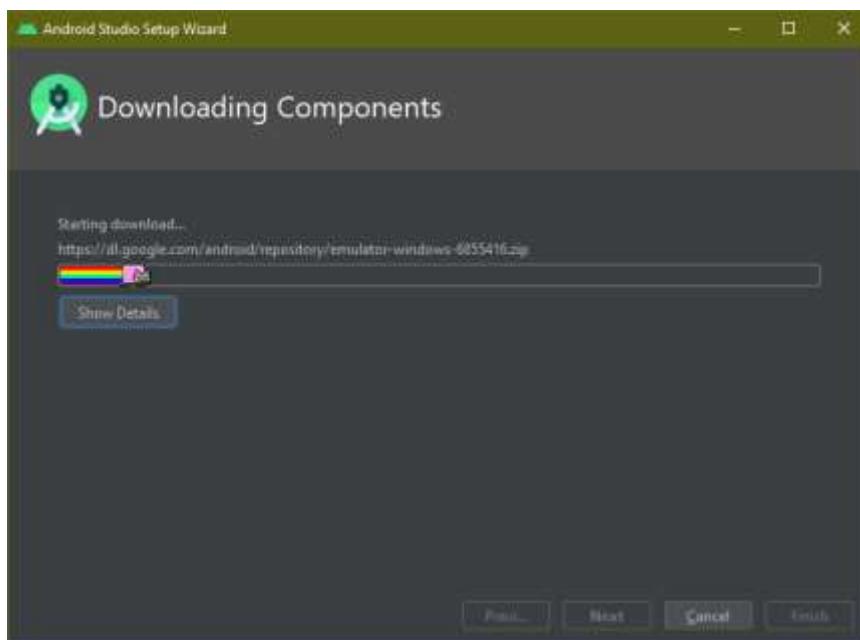


Figura 14: Android Studio - Realização do download dos componentes SDK

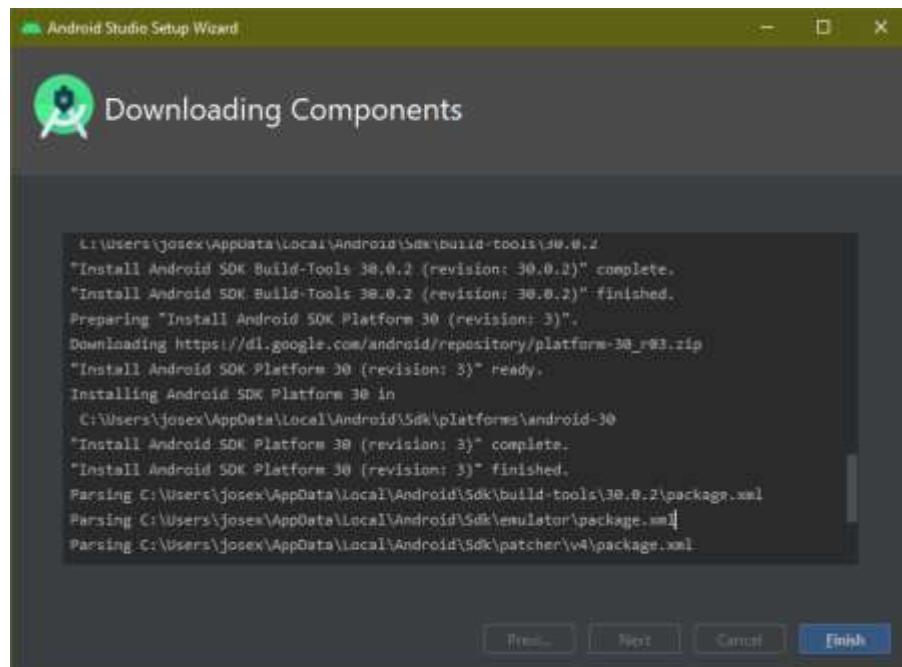


Figura 15: Android Studio - Finalização do download dos componentes SDK



Figura 16: Android Studio – Abertura do Programa



Figura 17: Android Studio – Plugins

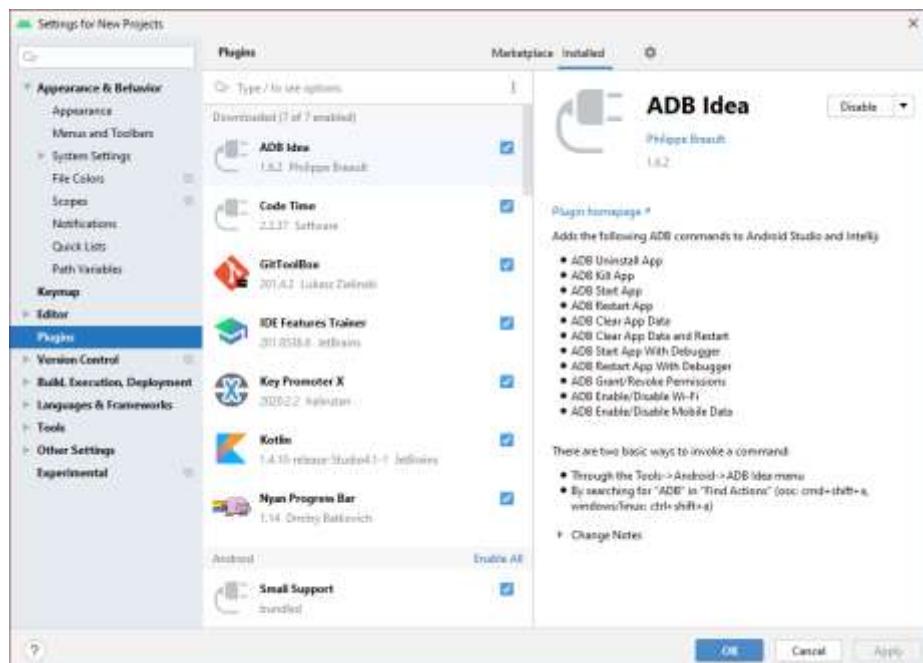


Figura 18: Android Studio - Plugins necessários para o projeto

Adobe Photoshop CC 2019

Este programa, foi essencial para a edição e recorte das imagens que precisei de modificar e colocar no meu projeto.



Figura 19: Adobe Photoshop CC 2019 - Ecrã Inicial



Figura 20: Adobe Photoshop CC 2019 - Opções de Instalação



Figura 21: Adobe Photoshop CC 2019 - Início do Processo de Instalação



Figura 22: Adobe Photoshop CC 2019 - Instalação Finalizada

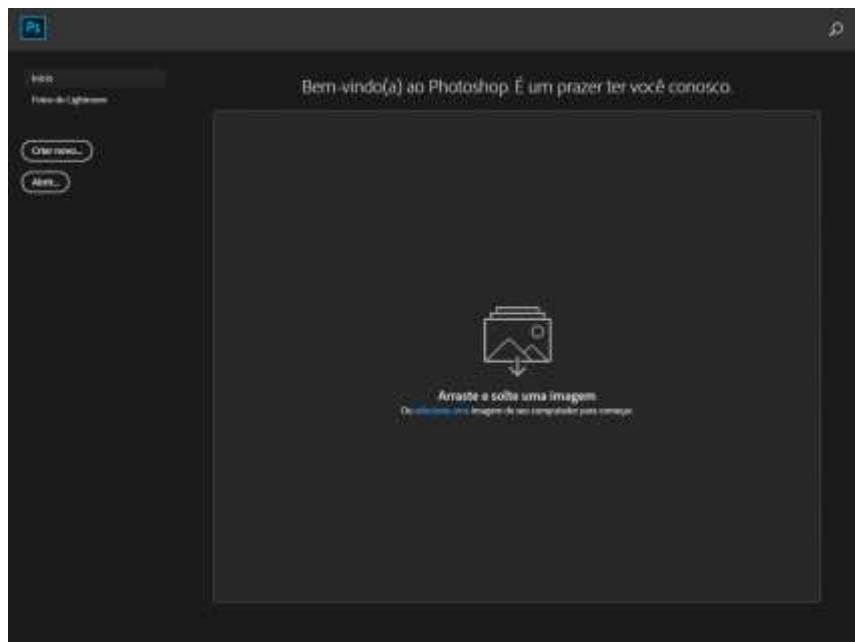


Figura 23: Adobe Photoshop CC 2019 - Abertura do Programa

Adobe Illustrator CC 2019

Foi neste programa que realizei os meus logos, visto que é um programa que me permite “desenhar” no computador.



Figura 24: Adobe Illustrator CC 2019 - Ecrã Inicial



Figura 25: Adobe Illustrator CC 2019 - Início do Processo de Instalação



Figura 26: Adobe Illustrator CC 2019 - Instalação Finalizada

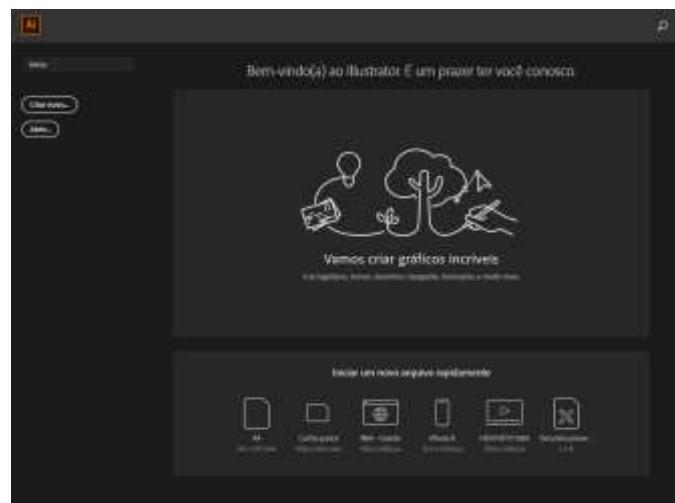


Figura 27: Adobe Illustrator CC 2019 - Abertura do Programa

Adobe XD

Este programa também foi bastante importante, visto que foi neste que realizei o planeamento e criação do layout da minha app.



Figura 28: Adobe XD CC 2020 - Ecrã Inicial da Instalação



Figura 29: Adobe XD CC 2020 - Progresso da Instalação



Figura 30: Adobe XD CC 2020 - Instalação Finalizada



Figura 31: Adobe XD CC 2020 - Primeira Inicialização

Logos



Figura 32: Logo inicial

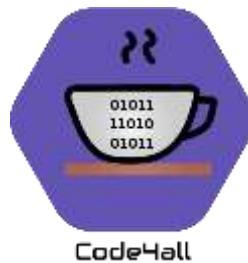


Figura 33: Logo versão 2
(hexagonal)



Figura 34: Logo versão 3
(circular)

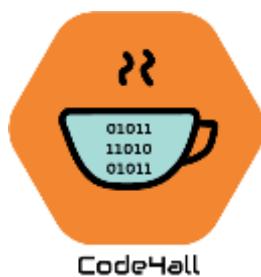


Figura 35: Logo versão 3
(hexagonal)



Figura 36: Logo versão 3
(circular)



Figura 37: Logo versão 4
(hexagonal)



Figura 38: Logo versão 4
(circular)

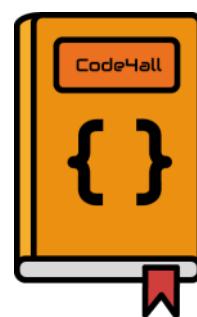


Figura 39: Logo final

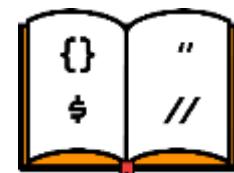


Figura 40: Logo final (livro
aberto)

Comecei então por realizar a criação do logo, que felizmente ainda teve algumas versões antes da versão final, como anteriormente visualizado, coloquei as sucessivas versões, com ajuda da minha equipa de acompanhamento, fui realizando as alterações e sugestões que a mesma me dava. Vou passar, então, à explicação.

No logo inicial, o primeiro logo, baseei-me no filme “Flash” e no seu próprio elenco, bastante conhecido, e como eu gosto muito do mesmo, decidi usá-lo como logo, mas de imediato a opção foi recusada pela equipa de acompanhamento, visto que não tinha muito a ver com o meu objetivo principal da PAP.

De seguida, decidi então implementar uma chávena de café, a razão é simples, visto que a minha aplicação é virada para a programação, pensei em programadores e nos seus hábitos, com uma leve pesquisa entendi que eles são conhecidos como noturnos, ou seja, eles ficam a codificar até de madrugada, e têm assim a fama de beber bastante café, e foi com isso que fui buscar a tal chávena de café.

Com isso em mente, mostrei novamente à minha equipa de acompanhamento, e levantou-se logo a questão do porquê a chávena de café e só depois de eu explicar o porquê é que sim entenderam e me deram razão, com isto, chegámos à conclusão que só os “intendentes” de programação é que saberiam minimamente o significado da chávena de café e que levou com que tivesse de refazer o meu logo mais uma vez.

Decidi então renovar totalmente o logo, e começar o mesmo de raiz, para isso fiz uma leve pesquisa das palavras chave programação / quizzes, o que me levou a retirar duas simples conclusões.

Para o meu logo precisava de uma simples característica que levasse a quem a estivesse a utilizar indicar que se trata de programação, assim sendo escolhi as chavetas {}, que são bastante conhecidas no ramo da programação para indicar ao compilador até onde ele vai executar o código, estas são mais utilizadas em condições “se/if”.

A outra e última característica, foi pela parte dos quizzes que pretendo realizar na minha aplicação, com mais uma pesquisa, conclui que, para além das chavetas, um caderno poderia de certa forma, dar a entender às pessoas que se trata de uma aplicação, na qual o seu objetivo principal é perguntas de programação.

Para finalizar, fiz ainda uma recriação do logo final, mas com o livro aberto, visto que o poderia vir a utilizar, com esta recriação dei então como finalizada esta parte.

Base de Dados

Diagrama

Aqui decidi colocar os diagramas que realizei no planeamento da minha base de dados, sendo a mesma diferente do comum, visto que é “NOSQL” e bases de dados com este tipo não contém qualquer relação, logo teve de haver um tratamento diferente de uma base de dados “MySQL” visto que a mesma tem relações.

Modelo de Dados

No modelo de dados, descrevi os campos que iria utilizar, consoante cada diagrama

utilizadores (id->email, estado, fotoPerfil, grupo, inscricoes, nivel, nome, nomeSimples, perguntas, pontuacao, seguidores, seguindo)

quizes (id -> classificacao, criadorQuiz, dificuldadeQuiz, experienciaQuiz, imagemQuiz, nota, statusQuiz, temaQuiz, tituloQuiz | membros -> email, fotoPerfil, idUtilizador, nome, percentagem)

taxaSucesso (idQuiz -> membrosTotal, sucesso | membrosConclusao -> idQuiz, statusQuiz

forum (idPergunta -> fotoPerfil, horas, mensagem, tag, titulo, tituloSimples)

pontuacao (idUtilizador -> mediaQuiz, nivel, nome, posicao)

Esquema (versão 1)

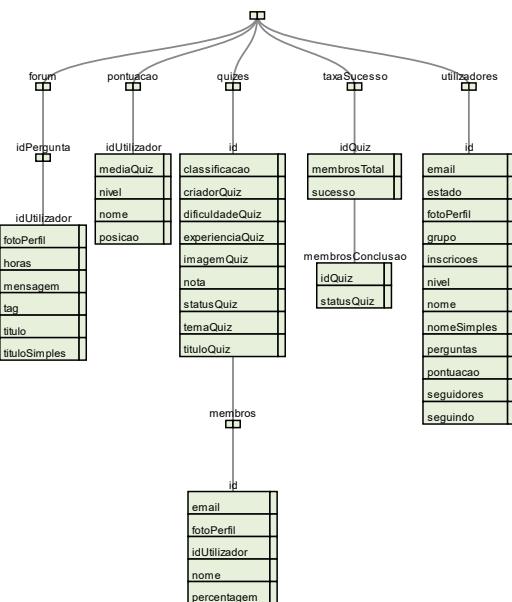


Figura 41: Diagrama - Versão 1

Esquema (versão 2)

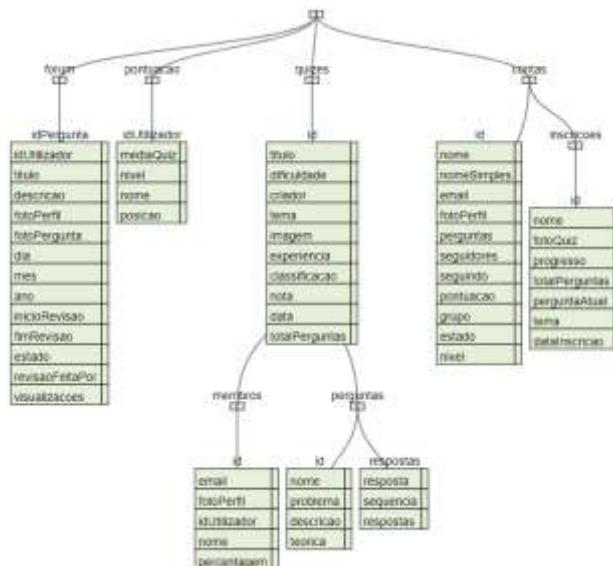


Figura 42: Diagrama - Versão 2

Esquema (versão final)

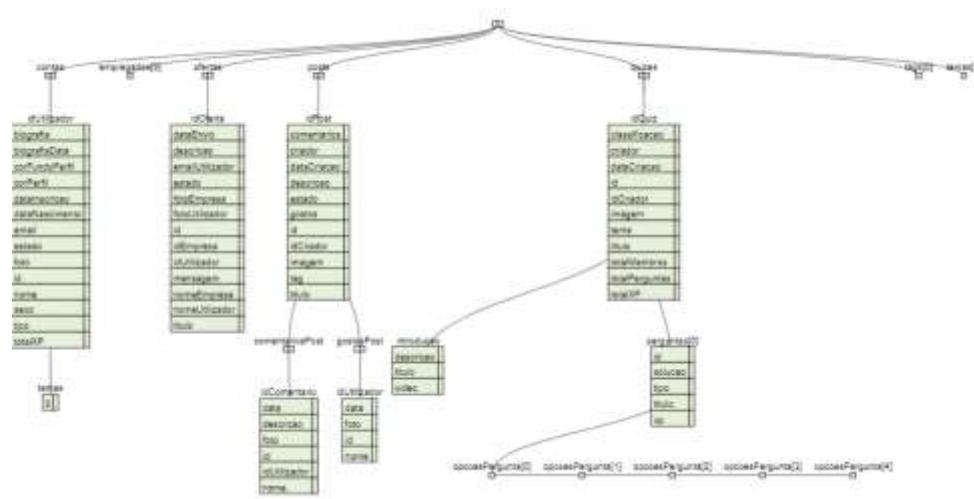


Figura 43: Diagrama - Versão Final

Layout

SplashActivity

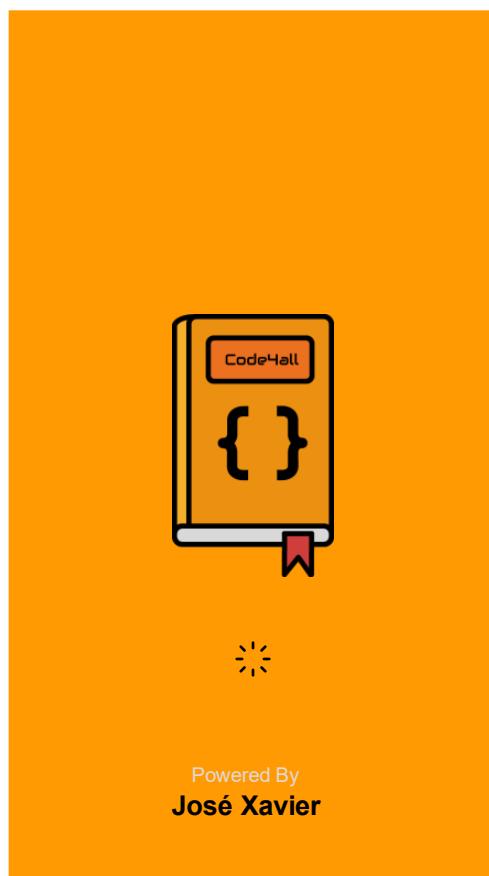


Figura 44: SplashScreen

No esboço desta atividade, decidi desenhar uma atividade com elementos básicos e simples, sendo ela apenas uma atividade de carregamento, ou seja, tem de conter poucos elementos, mas que de certa forma chamem a atenção. Estão presentes então 4 elementos, começando pelo logo, o GIF de carregamento (realizado por mim) e dois elementos de texto um a dizer “Powered By” sendo este para indicar “por quem foi feito” e por fim o “autor” deste projeto, estando então presente o meu nome.

IntroActivity



Figura 45: IntroActivity - Página 1

No desenho deste layout, comecei por colocar um botão de fechar, no canto superior direito. Seguidamente, coloquei uma imagem com os cantos arredondados e mais abaixo encontra-se um texto a descrever a mesma.

Na parte de baixo do layout, encontra-se um pequeno menu de navegação, para navegar entre os layouts desta atividade.



Figura 46: Introdução - Página 2

Nesta segunda página de layout o que realmente fiz de diferente, foi adicionar uma nova imagem, um novo texto introdutório e colocar o botão “Anterior”, para no caso do utilizador querer voltar para trás conseguir fazê-lo sem problemas.

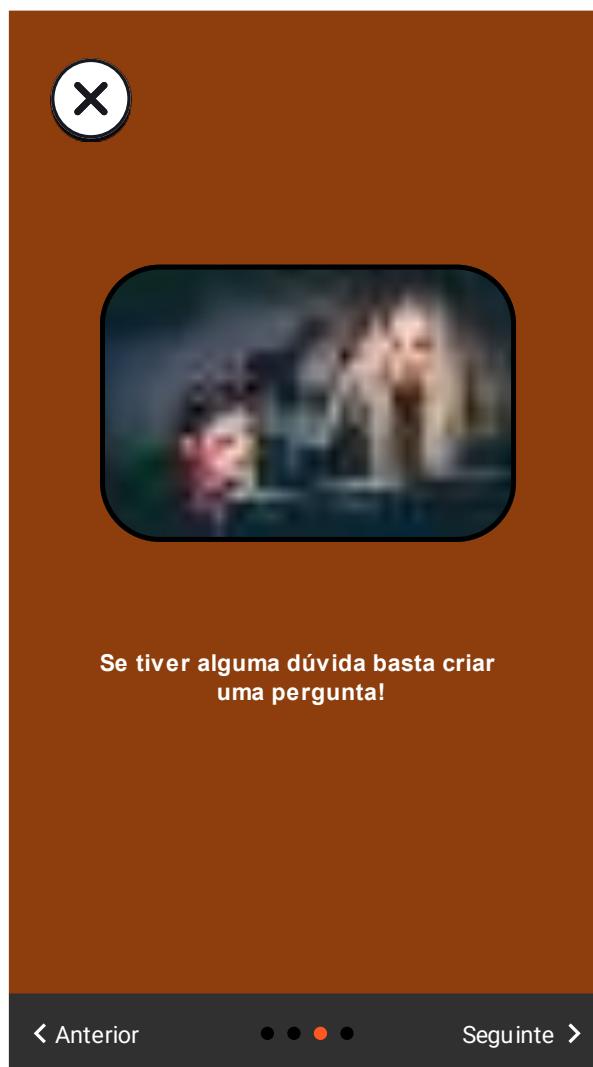


Figura 47: IntroActivity - Página 3

Neste layout, não foi diferente do anterior e decidi também colocar outra imagem com um texto descriptivo da mesma, como funcionalidade da app.



Figura 48: IntroActivity - Página 4

Por fim, realizei este layout, com mais uma vez a alteração da imagem e do seu texto descriptivo, queria ainda referenciar que ao longos destes quatro layouts que expliquei anteriormente, existem um efeito de degrade visto que a sua cor vai escurecendo.

EntrarActivity

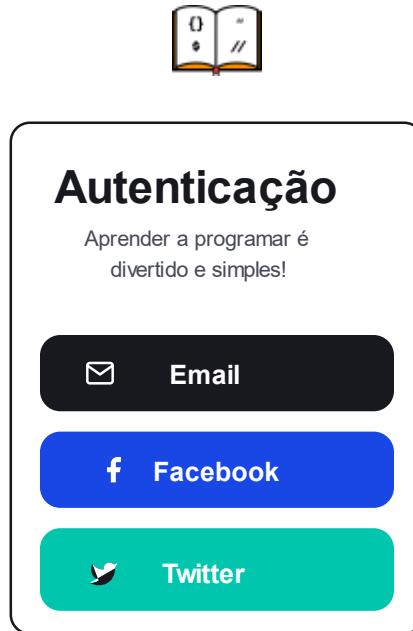


Figura 49: EntrarActivity

Esta atividade, terá então elementos básicos, começando como já foi referido logo com o livro aberto, anteriormente tinha dito que o mesmo poderia ser utilizado e decidi então colocar o mesmo neste layout.

Seguidamente, vem um texto “Autenticação” para que o utilizador saiba do que se trata esta atividade e seguidamente três botões, Email, Facebook e Twitter para ser realizado o login em cada uma, caso assim o utilizador pretenda. Encontram se ainda dois textos, em que um deles serve de função criar uma nova conta.

LoginActivity

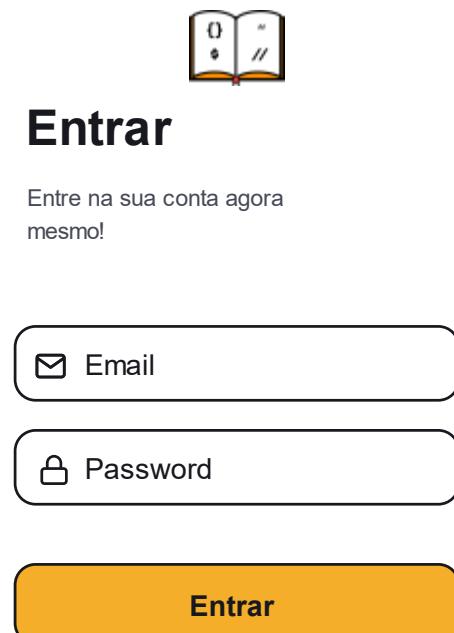


Figura 50: LoginActivity

Neste layout, foi colocado o mesmo logo que referi na atividade anterior, coloquei também um texto em “grande” para o utilizador saber onde está e também uma pequena descrição para não deixar aquele espaço em “vazio”.

Após isso, coloquei dois campos, um de email outro de password e seguidamente um botão “Entrar” e em baixo desses elementos todos, coloquei mais uma vez dois textos e neste caso o “Criar conta” que dirige o utilizador para a atividade.

CriarContaActivity



Registo

Crie uma conta agora
mesmo é grátis e rápido.

 Nome
 Email
 Password
 Confirmar Password
 Data de Nascimento

Empresa  Utilizador

Figura 51: RegistrarActivity – Utilizador

O layout acima representado, diz respeito ao registo de uma conta por email, como fiz nos layouts anteriores comecei por colocar um texto “grande” e uma descrição.

Na próxima página, como indica o layout, coloquei cinco campos, todos relacionados com o registo de um utilizador normal e em baixo coloquei ainda um “Switch” para o utilizador conseguir escolher se vai criar uma conta de utilizador normal ou de empresa.



Registo

Crie uma conta agora
mesmo é grátis e rápido.

 Nome

 Email

 Password

 Confirmar Password

 NIF

Empresa Utilizador

Figura 52: RegistarActivity - Empresa

Este layout é basicamente igual ao anterior, mas com a diferença do “Switch” estar virado para o lado da empresa e também o último campo ter mudado de “Data de Nascimento” para “NIF” onde será colocado então o NIF da empresa em questão.

PrincipalActivity



Quizes

Populares

**Quiz de HTML
- Iniciantes**

★★★★★ 3.2

José Xavier | Fundador

Data de Criação: 11/11/20

Matricular

Figura 53: PrinicipalActivity

O layout desta atividade, foi bastante confuso, mas com a ajuda da minha equipa de acompanhamento, lá consegui realizar a mesma. Começando então pela parte superior, onde se visualiza um ícone “hambúrguer” que vai abrir o menu, no meio o logo (livro aberto) e no lado direito um ícone de notificações.

Seguidamente, vem a colocação do assunto desta atividade “Quizes Populares” e após isso uma pré-visualização de um Quiz. Nesse quiz é apresentada uma imagem, um título, umas estrelas de avaliação, o seu criador e a data criação e por fim um botão “Matricular”.

MenuActivity



Figura 54: MenuActivity

O layout deste menu foi bastante simples de realizar, coloquei as atividades que pretendia utilizar como opções selecionáveis e a selecionada no momento teria a cor laranja.

Em baixo apresenta-se a versão atual da aplicação Code4All ,sendo ela inicialmente 1.0.

Plugins Android Studio

Para aceder à página dos “Plugins”, extensões em português, basta fazer o atalho “CTRL + ALT + S”, que de imediato abre uma janela semelhante a esta.

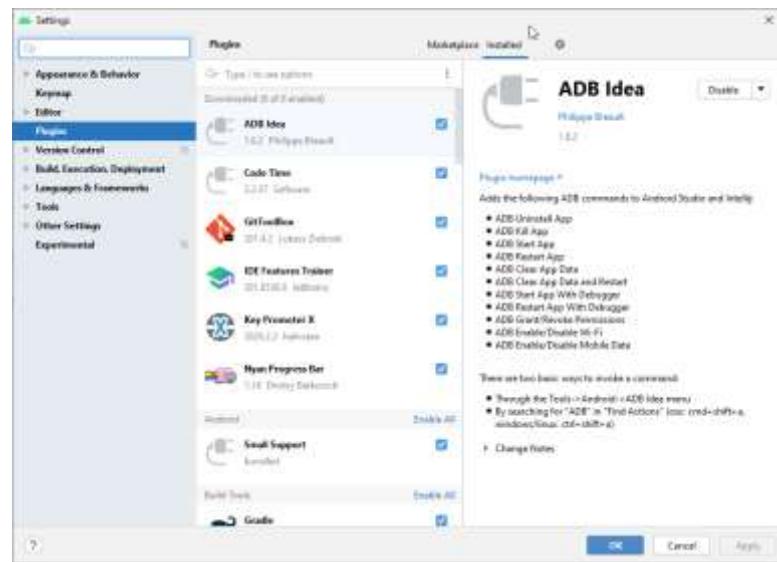


Figura 55: Página dos Plugins do Android Studio

Vou então passar à explicação do porquê de cada plugin e o que cada um faz ou pode fazer na realização deste projeto:

ADB Idea

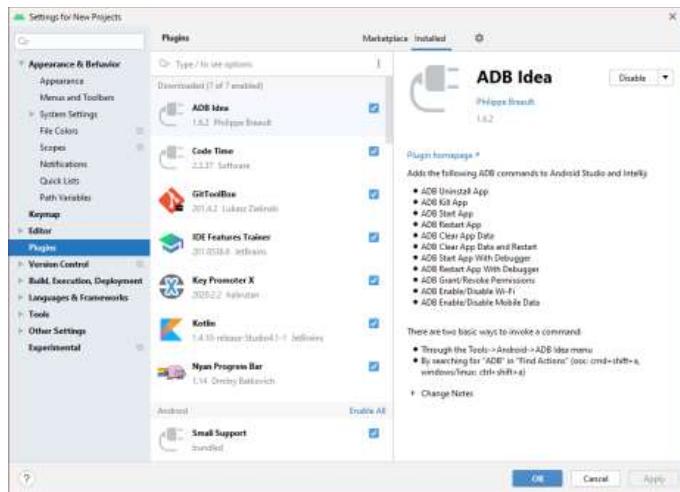


Figura 56: Plugin "ADB Idea"

Com este plugin, consigo gerenciar os dispositivos, que tenho conectados via WiFi, USB ou dispositivos virtuais.

Consegue-se ainda, desinstalar a aplicação através dele, desinstalar e instalar e abrir a mesma, abrir a aplicação com o “Debugger”, que é utilizado para descobrir o que o programa está a fazer e para encontrar possíveis erros, ativar e desativar o WiFi e os dados móveis, entre outras funções bastante interessantes. Para utilizar o mesmo, basta apenas fazer o atalho “CTRL + SHIFT + A” e pesquisar por “ADB”, como indicado na figura abaixo.

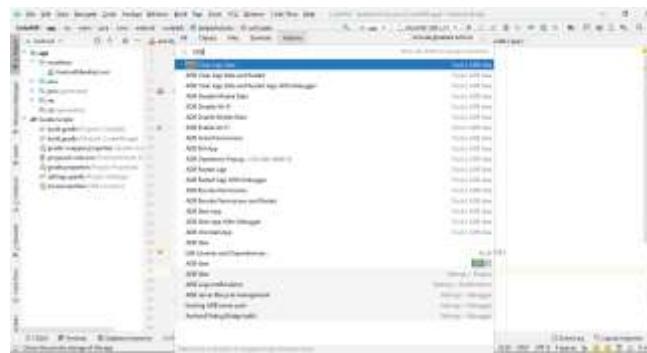


Figura 57: Pesquisa das funções do Plugin "ADB Idea"

Code Time

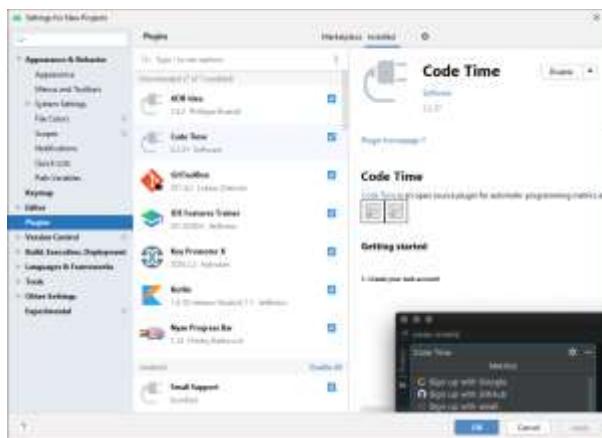


Figura 58: Plugin "Code Time"

Esta extensão, tem como objetivo, captar o tempo que o “programador” passa a codificar, com o requerimento de apenas ter de entrar com uma conta, para depois então os seus dados passarem a ser guardados num servidor.

Com isto pretendo então gerir o meu tempo melhor e perceber se estou a passar muito tempo num obstáculo ou na realização do código em si. E com esta extensão consigo fazer esta gestão muito mais facilmente.

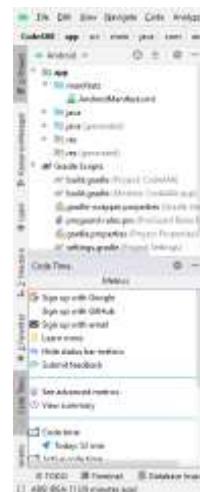


Figura 59: Plugin "Code Time" aberto pela 1^a vez

GitToolBox

Este “plugin” é uma espécie de pacote, que me permite, fazer os tão famosos “commits”, estes são uma espécie de relatório obrigatório que é necessário ao enviar os ficheiros para um repositório de GitHub, este é então bastante utilizado pelas empresas ou projetos tal como no meu caso.

Esta ferramenta, consegue dar-me mais opções ainda, para de certa forma facilitar o meu trabalho, basicamente é um complemento ao meu desenvolvimento, mas só vou utilizar este “plugin” mesmo a sério quando acabar a fase de desenvolvimento e chegar à versão 1.0 da minha aplicação para depois então realizar os “commits”, visto que me dá os complementos mais que necessários para mandar os ficheiros automaticamente e escrever o que realizei/melhorei em cada atualização da app.

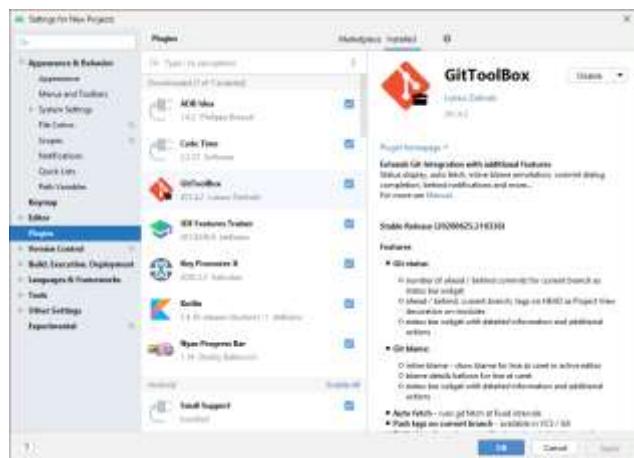


Figura 60: Plugin GitToolBox

IDE Features Trainer

Esta extensão é bastante simples, e informa sem ter que ler a documentação toda do Android Studio, os atalhos que posso utilizar, como melhorar a minha codificação, entre outros aspectos.



Figura 61: Plugin "IDE Features Trainer"

Key Promoter X

Faz quase a mesma coisa do que a extensão anteriormente referida, mas quando clico em alguma parte do Android Studio, em que é possível utilizar um atalho, ele avisa-me e diz-me logo o atalho que posso usar para substituir o mesmo, e para completar, até me diz quantas vezes já cliquei naquela parte do Android Studio.

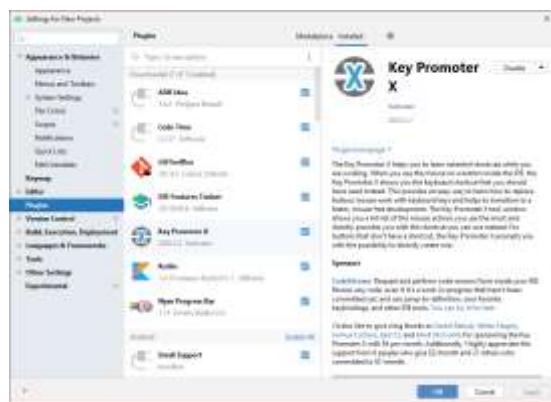


Figura 62: Plugin "Key Promoter X"

Nyan Progress Bar

Esta extensão modifica apenas as barras de carregamento, de um progresso “chato” para um gato fofo com arco-íris, um gato bastante conhecido na Internet que virou um “meme” há uns anos atrás.

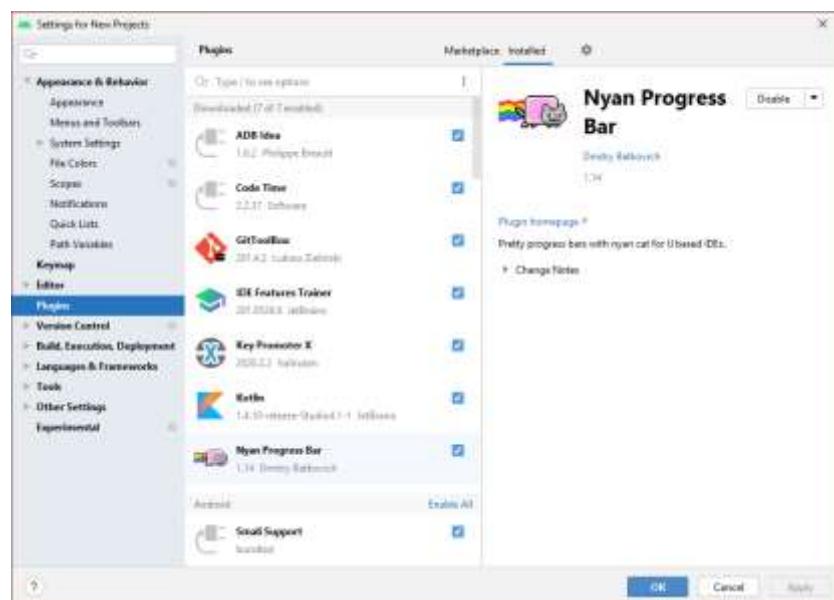


Figura 63: Plugin “Nyan Progress Bar”

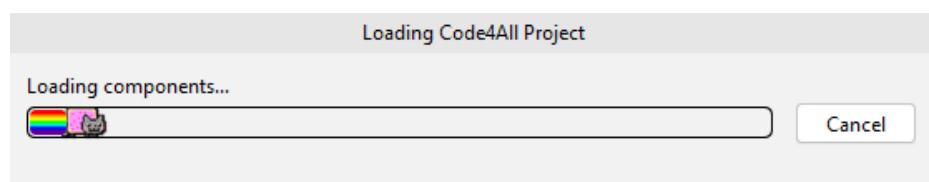


Figura 64: Plugin "Nyan Progress Bar" em utilização

Configuração do Dispositivo Android

Desenvolvimento via Emulação do Android

Inicialmente, é preciso iniciar o Android Studio e deixar o mesmo carregar completamente até chegar à página principal.

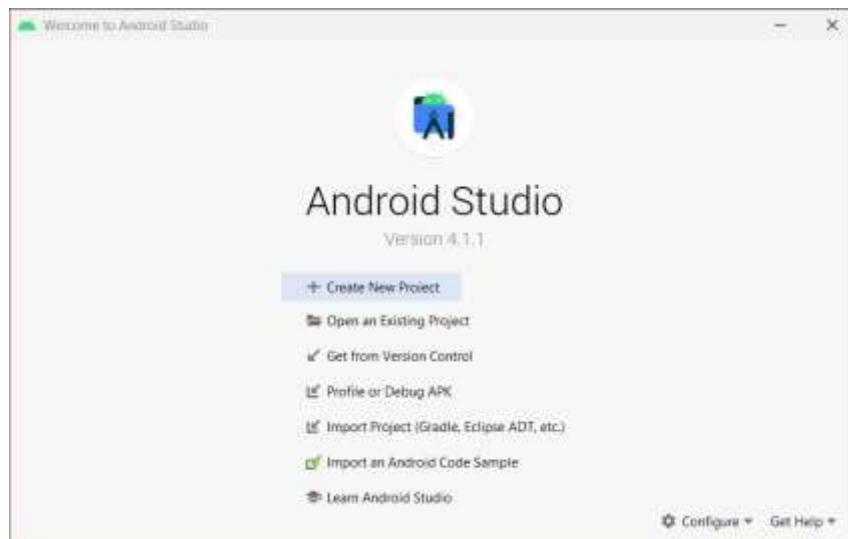


Figura 65: Página Inicial do Android Studio



Figura 66: Opção de configurar os dispositivos android virtuais

Após clicar na opção tal como foi referido anteriormente, irá ser aberta uma janela igual ou muito semelhante à mesma representada na página anterior, como demonstra a mesma, já temos um dispositivo virtual configurado, este foi automaticamente adicionado quando a instalação do Android Studio foi realizada, sem a necessidade de termos configurações adicionais, o mesmo também já está 100% operacional com o Android mais recente, o Android 11, e pronto para receber aplicações.



Figura 67: Dispositivo Android Virtual

Desenvolvimento via Wi-Fi (ADB)

Primeiramente, tive que fazer download das ferramentas da plataforma, neste caso do Android Studio, no seguinte link.

[https://developer.android.com/studio/releases/platform-tools:](https://developer.android.com/studio/releases/platform-tools)



Figura 68: Visualização do site referido acima

De seguida, extraí o conteúdo do arquivo "zip" na raiz do meu disco do Windows, o disco C:\, ficando desta forma.

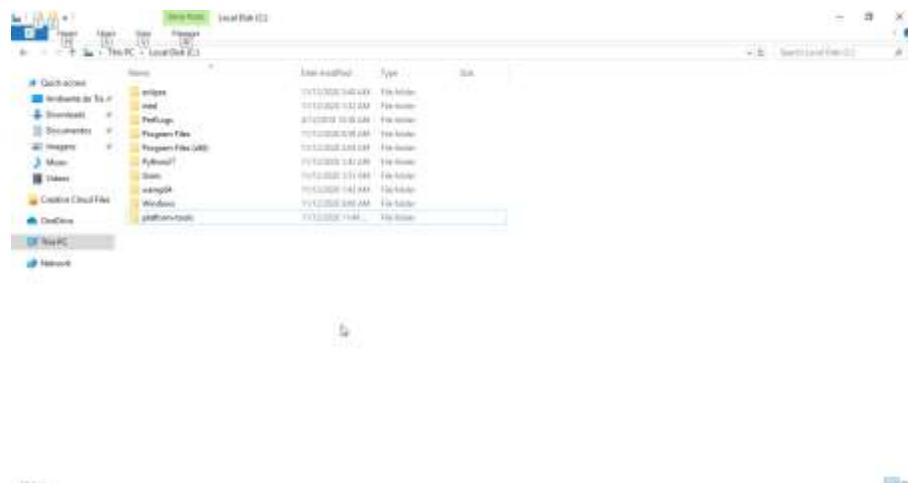


Figura 69: Localização do arquivo "zip"

Abre-se a pasta extraída, chamada no meu caso de “platform-tools”, seguidamente, é necessário abrir a linha de comandos do Windows, a forma mais fácil de fazer o mesmo, é escrever “cmd” como representado na figura abaixo.

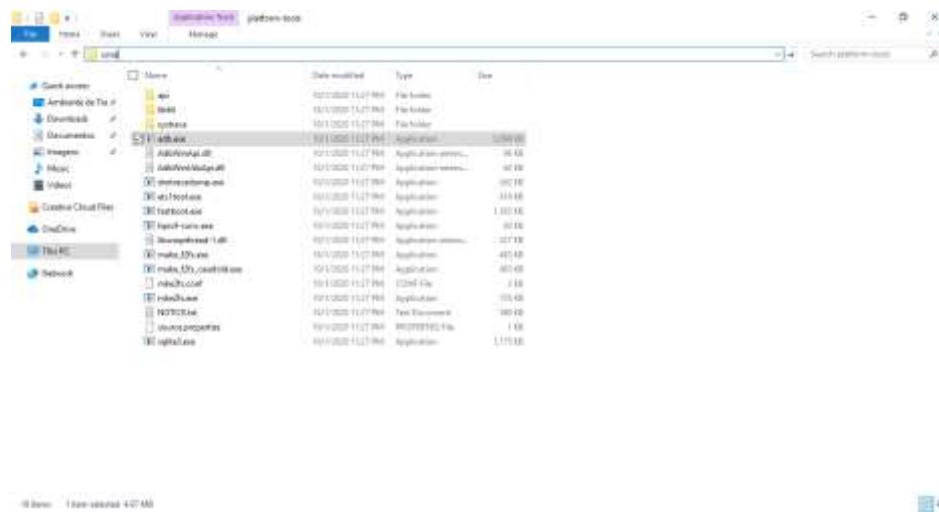


Figura 70: Explicação de como abrir a linha de comandos

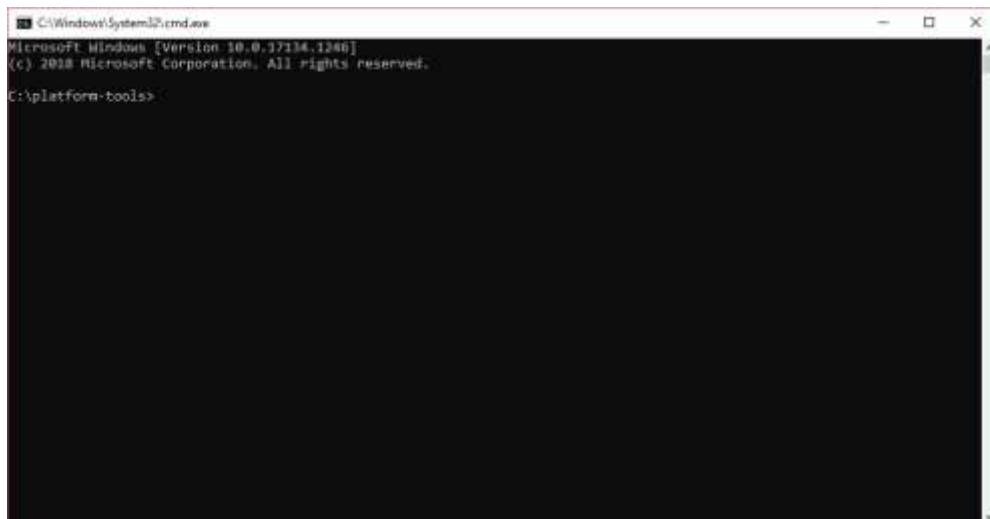


Figura 71: Depois de pressionar o ENTER, deve surgir esta janela

Após isso, basta conectar o dispositivo android no seu computador e uma informação que pretendo reforçar é que tanto o dispositivo como o computador têm de obrigatoriamente de estar na mesma rede, para assim terem conexão entre si, visto que a instalação da aplicação é realizada via WiFi.

Continuando, tem de se abrir uma porta no dispositivo, para que seja realizada a comunicação sem qualquer “barramento”, para isso utiliza-se o comando “adb tcpip 5555”, normalmente usa-se sempre a porta 5555.

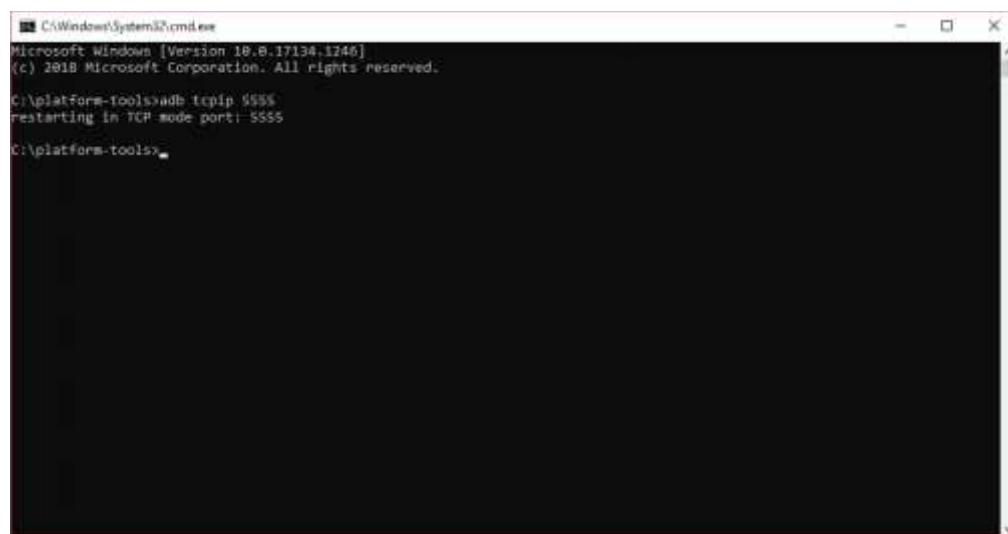


Figura 72: Execução do comando referido acima

De seguida, já se pode desconectar o dispositivo do computador, e passa-se para o telemóvel para ativar o modo de programador.

Para ativar o modo programador basta ir às definições do dispositivo e pressionar na última opção, no meu caso é “Sobre o telefone”, como mostrado nas figuras da página seguinte.



Figura 73: Opção "Sobre o telefone"



Figura 74: Sobre o telefone

Após se chegar a esta página, é necessário clicar sete vezes onde diz “Número da compilação”, para então colocarmos o telefone em modo programador. Com esta configuração, pode-se dizer que o telefone já está preparado para fazer instalações pelo o Android Studio, via WiFi.

Por fim, agora sim podemos fazer a ligação entre o computador e o telemóvel, com o comando “adb connect endereço-ip:5555”, este endereço IP é o que está presente no telemóvel como mostrado na figura abaixo, mais uma vez na opção do “Sobre o telefone” clica-se na opção de “Estado”, como mais uma vez mostrado abaixo.



Figura 75: Visualização da opção "Estado" nas Definições "Sobre o telefone"



Figura 76: Endereço IP do Dispositivo Android

Neste caso, como a ligação à internet está a ser efetuada através de internet da minha escola, o meu IP é um pouco diferente, mas se tivesse ligado numa rede doméstica “normal” muito provavelmente, se não houvesse modificações, começava por 192.168.1.xxx, este “xxx” pode ir de 1 a 255, se esta configuração for feita numa rede doméstica é provável que o “xxx” seja bastante baixo, com isto quero dizer que, o IP do dispositivo será muito provavelmente menor que 50.

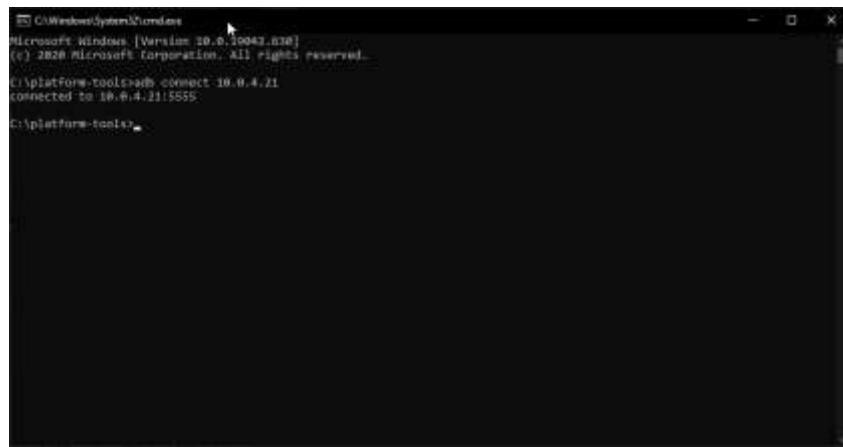


Figura 77: Conexão de ADB entre o computador e o Dispositivo Android

Agora sim, temos o nosso dispositivo totalmente configurado para receber as aplicações criadas pelo o código no Android Studio e ao fazer esta configuração o dispositivo irá no mesmo momento aparecer no Android Studio como opção de dispositivo para executar a aplicação.

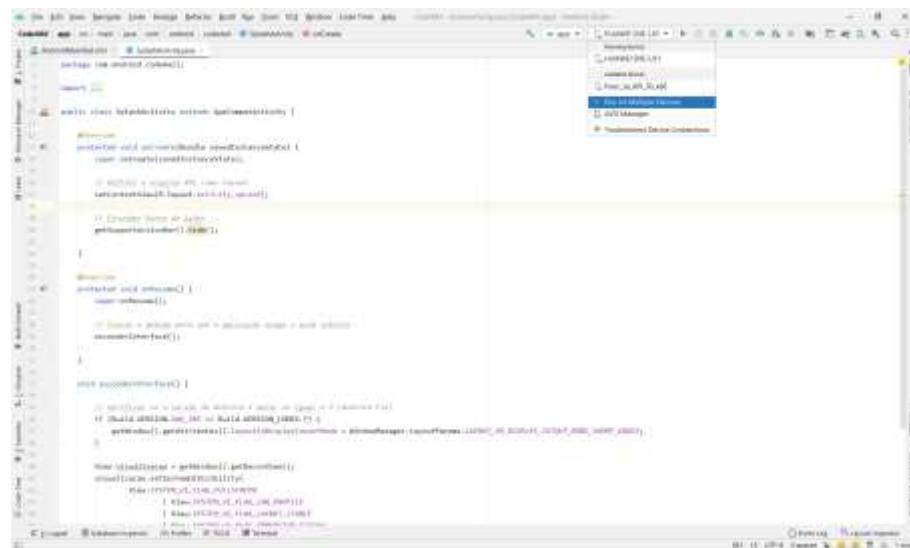


Figura 78: Verificação do dispositivo Android

Uma informação bastante importante, é que o dispositivo android (físico) estará com a porta (5555) aberta até que seja reiniciado/desligado o mesmo, portanto se o mesmo for desligado, este processo terá de ser feito de novo desde a parte de ligar o telemóvel por USB e executar o comando “adb tcpip 5555” após isso o telemóvel estará novamente pronto para receber comunicações, neste caso aplicações. E também se o Android Studio for desligado, a conexão “adb” terá de ser feita novamente.

Por fim, também poderá aparecer um aviso semelhante a este, no meu caso não apareceu, mas poderá eventualmente aparecer. A figura abaixo foi retirada da Internet.



Figura 79: Aviso de depuração USB

Codificação da Aplicação

Nesta parte do relatório é onde vou apresentar de certa forma, o que foi realizado desde o início até ao fim do seu desenvolvimento seja tanto de código, obstáculos, como passei os mesmos, entre outros...

Criação do Projeto

Para criar o projeto foi necessário abrir o Android Studio mais uma vez e clicar em “Create New Project”, o que traduzido em português significa “Criar um novo projeto”, tal como referido na figura.

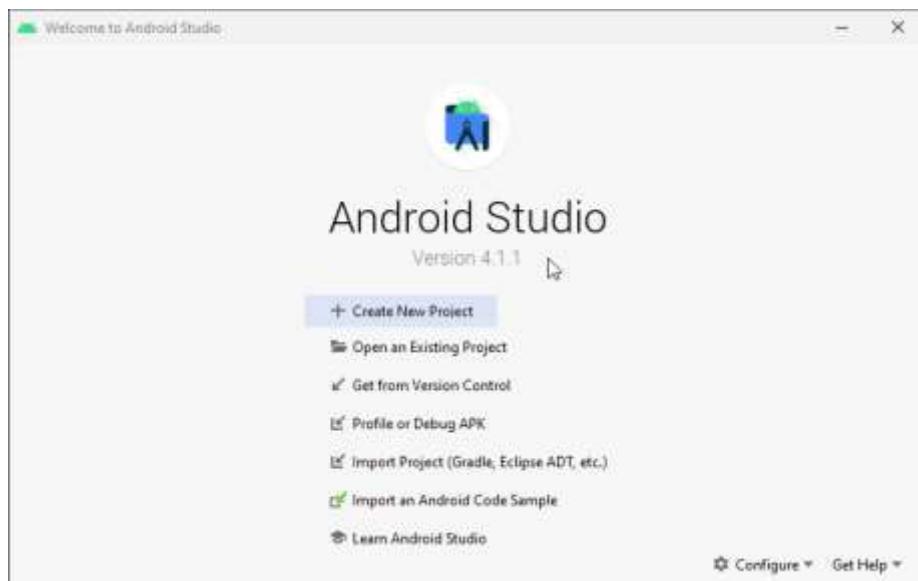


Figura 80: Página Inicial do Android - Criar um novo projeto

De seguida, vai ser aberta uma nova janela muito semelhante à aquela que se encontra abaixo.

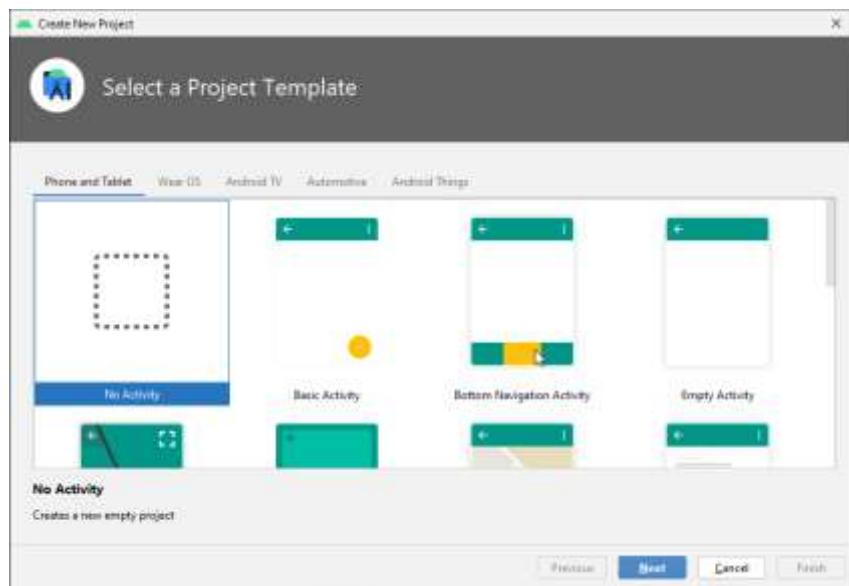


Figura 81: Escolha do modelo do projeto

Como é obvio nesta parte coloquei como modelo de projeto a “No Activity”, devido ao facto que neste modelo só são criados os ficheiros básicos para o funcionamento do projeto.

Ou seja, será feito tudo de raiz, com a menor ajuda possível por parte do Android Studio, visto que existem modelos que fazem bastante código, mas como isto é um projeto com estrutura e responsabilidade, tem de ser tudo feito de raiz.

Após a escolha do modelo de projeto cliquei no botão “Next”, o que me levou para a janela representada abaixo.

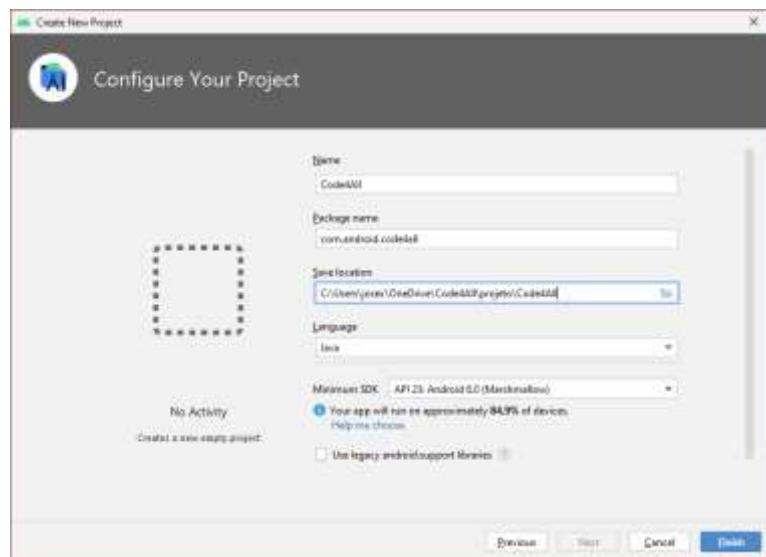


Figura 82: Configuração do projeto em si

No campo “Name”, que significa “nome”, coloquei Code4All, visto que é o nome do meu projeto final de curso, este nome será apresentado como o nome da aplicação.

No campo “Package name”, que significa “nome do pacote”, coloquei “com.android.code4all”, separando isto em três partes, coloquei primeiramente o “com” pois este nome de pacote, tem a normalização de ser um nome de website só que inverso, de seguida coloquei “android”, visto que isto é uma aplicação para android e por fim coloquei mais uma vez “code4all” visto que é o nome do projeto.

Coloquei também a linguagem de programação Java, visto que é a programação que domino mais e quem sabe como evolução posso trocar de Java para Kotlin, visto que é uma linguagem muito mais recente e com alguns melhoramentos.

Por fim, mas não menos importante, coloquei o campo “Minimum SDK”, como o Android 6.0 (Marshmallow), pois segundo os requisitos que planeio utilizar esta é a versão que todos os dispositivos que instalarem a aplicação deverão ter, se tiverem uma versão anterior não conseguirão instalar a aplicação. Seguidamente, pressionei para finalizar a criação do novo projeto, em “Finish” ao fazer isto levou me para a seguinte janela.

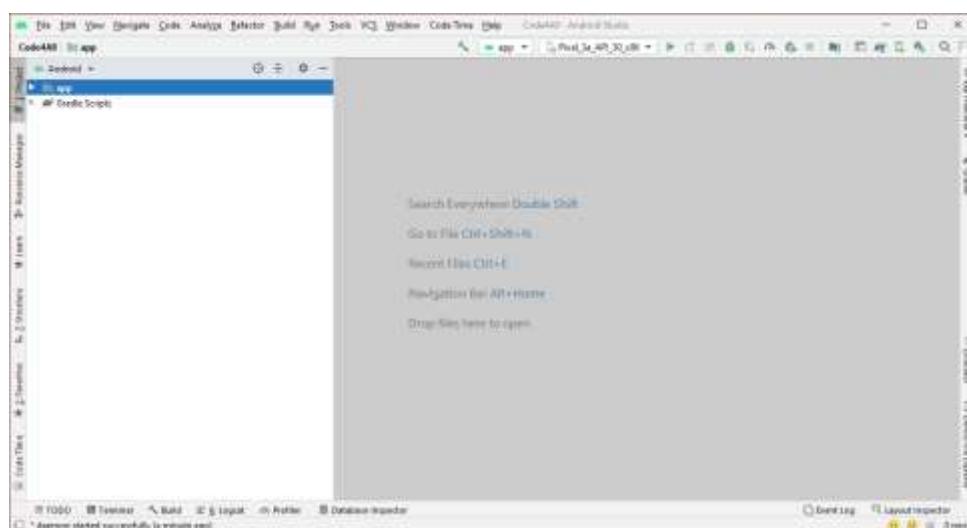


Figura 83: Projeto Code4All

Com isto, tenho então a criação do projeto base concluído com sucesso.

Configuração do Logo

Aqui foi realizado a configuração do logo, para que a imagem da aplicação selecionada por defeito seja substituída pelo logo que foi realizado para este projeto.

Para isso, obviamente é preciso ter o projeto já aberto, de seguida foi necessário ir à pasta “res” e depois “mipmap” e dentro dessa mesma pasta vão-se encontrar duas pastas por norma, uma chamada de “ic_launcher” e a outra chamada de “ic_launcher_round”, como se pode visualizar na imagem abaixo.

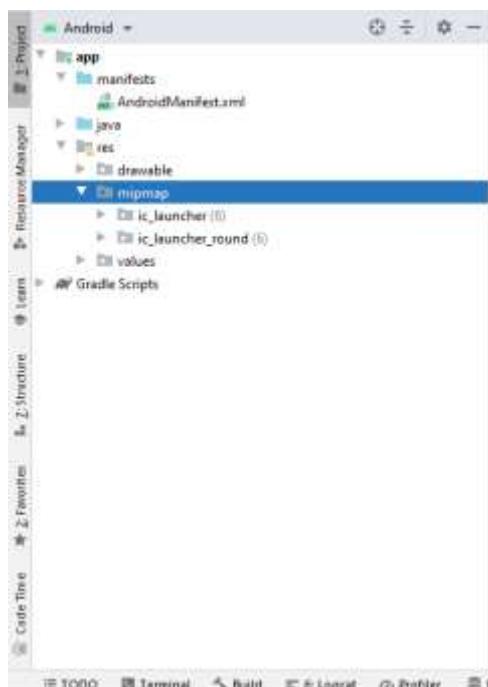


Figura 84: Pasta mipmap com os logos pré-definidos

Vamos então eliminar a pasta “ic_launcher_round” e a “ic_launcher”, deixando a pasta mipmap completamente vazia.

Para adicionar os ficheiros corretos foi preciso a ajuda de um site externo bastante útil, criado pelo Roman Nurik, um nome bastante conhecido na plataforma GitHub, mais propriamente, este link:

<https://romannurik.github.io/AndroidAssetStudio/icons-launcher.html/>



Figura 85: Aspetto final do logo da App

A única coisa que tive de fazer neste site foi apenas colocar o ficheiro PNG que já tinha anteriormente criado no Adobe Illustrator quando realizei o logo.

Após estar tudo completamente concluído, apenas pressionei o botão de transferir no canto superior direito, para iniciar a transferência dos ficheiros necessários para que tudo funcione corretamente.

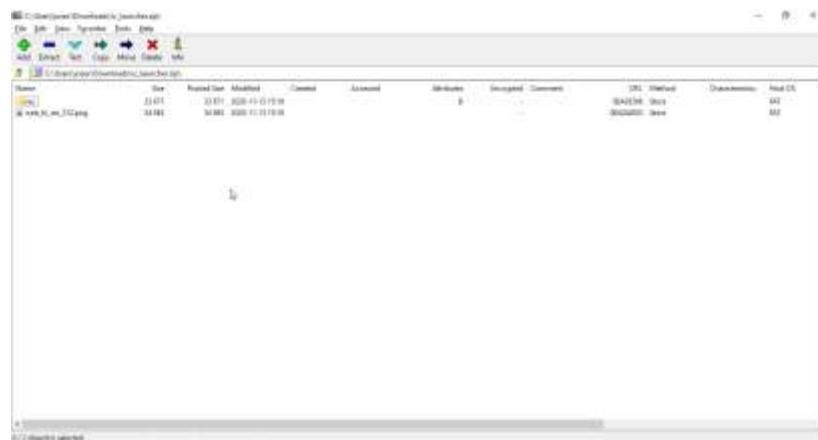


Figura 86: Conteúdo da pasta transferida anteriormente

Após extrair tudo, o processo foi bastante simples, apenas foi necessário copiar os ficheiros e colar na pasta mmap já referida anteriormente, ficando da seguinte maneira.

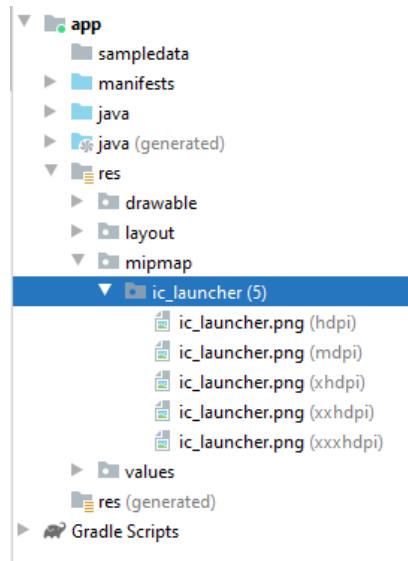


Figura 87: Pasta “mipmap” após a introdução do logo Code4All

E como não mudamos o nome do ficheiro porque permaneceu o mesmo, não é necessário fazer as devidas alterações no ficheiro `AndroidManifest.xml`. Agora vou apresentar uma foto da aplicação no meu dispositivo android.

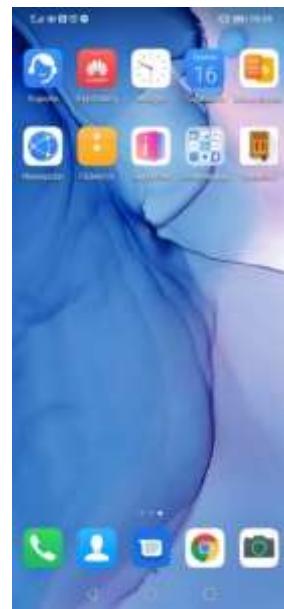


Figura 88: Fotografia do meu dispositivo Android com o Code4All instalado

Ligaçāo ao Firebase (Base de Dados)

Para fazer a ligação à base de dados, é preciso primeiro possuir uma conta Google, para passar à criação da mesma. Com uma conta Google criada, o procedimento é fácil, basta apenas entrar no link e logo será apresenta uma página semelhante à figura representada, com menos projetos provavelmente, visto que eu já utilizei a Firebase anteriormente. <https://console.firebaseio.google.com/>

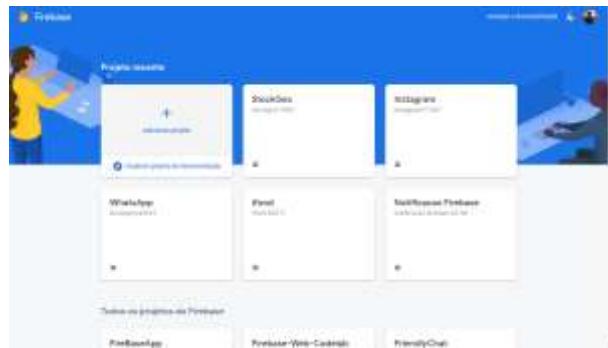


Figura 89: Página inicial da consola do Firebase

Seguidamente, comecei por criar um projeto novo, com o nome “Code4All” e com o código “code4all-pap”.

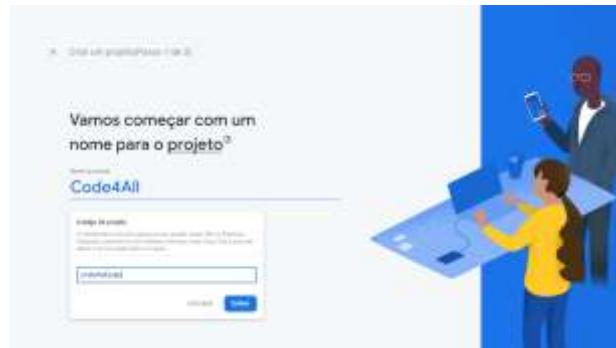


Figura 90: Nome do Projeto no Firebase

Com o nome do projeto definido, pressionei em “Continuar”.



Figura 91: Verificação do nome do projeto na Firebase

Logo de imediato, foi necessário configurar o Google Analytics, para fazer uma análise na minha app, para saber por exemplo quantas pessoas tenho diariamente na mesma, quantos dispositivos estão registados, entre outros...



Figura 92: Ligação ao Google Analytics na Firebase

Passei então à configuração da conta do Google Analytics, como não tenho, escolhi a opção “Default Account for Firebase”.

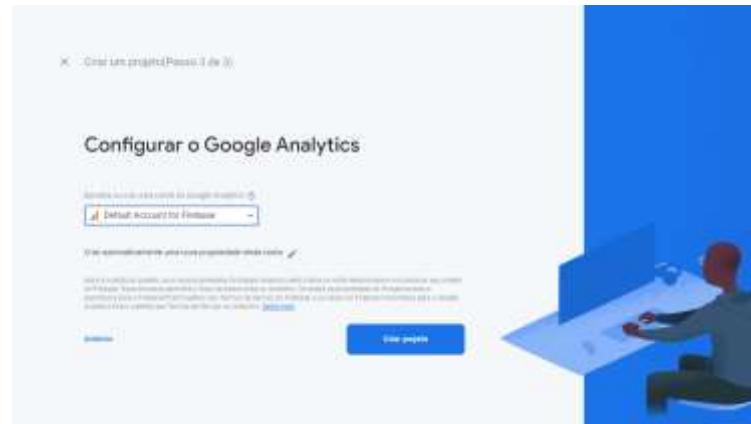


Figura 93: Configuração do Google Analytics na Firebase

Em baixo, encontra-se uma captura de ecrã referente à finalização da criação do meu projeto “code4all” no firebase, relativamente à base de dados.



Figura 94: Finalização da criação do projeto

Na figura representada, existem vários elementos que podem ser configurados, com isto dizendo que a Firebase é uma base de dados, que funciona em tempo real. Posso configurar então esta base de dados para operar com quatro aplicações, sendo elas iOS, Android, Web e Unity. Como a aplicação que pretendo realizar vai ser apenas em Android, comecei por configurar a mesma pelo mesmo.



Figura 95: Página inicial do projeto na Firebase

Com o clique, no botão para configurar a base de dados para Android, é necessário registrar de imediato a app. Começando então, com o nome do pacote que no meu caso é “com.android.code4all” e seguidamente do nome do projeto, onde coloquei “Code4All”.

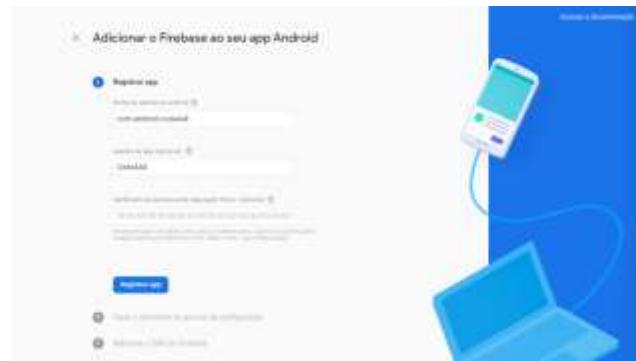


Figura 96: Registro da app na Firebase

Para o campo de certificação da assinatura de depuração SHA-1, foi preciso então voltar de novo para o Android Studio para descobrir o mesmo, visto que é opcional, mas gosto de ter tudo bem definido, para que no futuro não haja erros.

Como o Android Studio aberto, para descobrir a chave SHA-1, bastou apenas clicar no painel do lado direito em “Gradle”, abrir a diretoria “Code4All”, seguidamente de “app”, “Tasks” e por fim “android”. Após a abertura dos mesmos, cliquei duas vezes em “signingReport”, para então ser feito um relatório das informações do projeto.

Este valor da chave SHA-1, varia de projeto para projeto, visto que é uma identificação única.

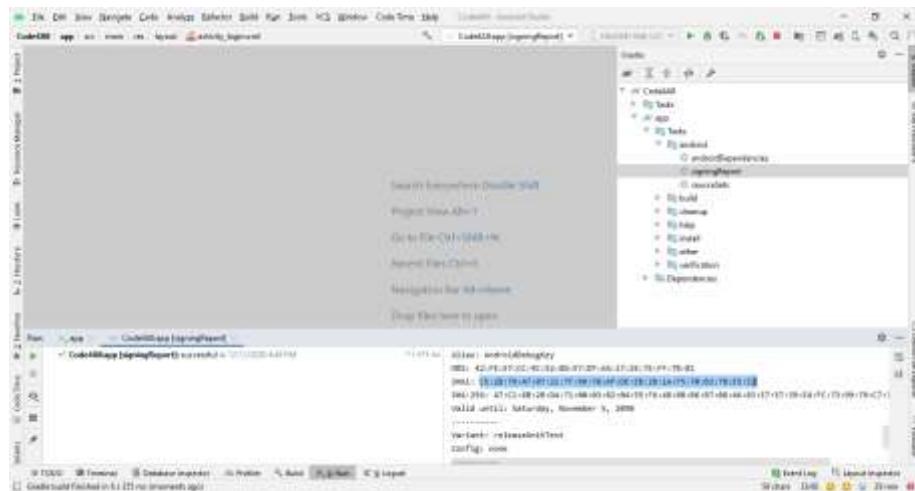


Figura 97: Chave SHA-1 do projeto "Code4All" no Android Studio

Após copiar o “SHA-1” retirado do Android Studio, coloquei o mesmo no formulário do Firebase e pressionei em “Registrar App”.

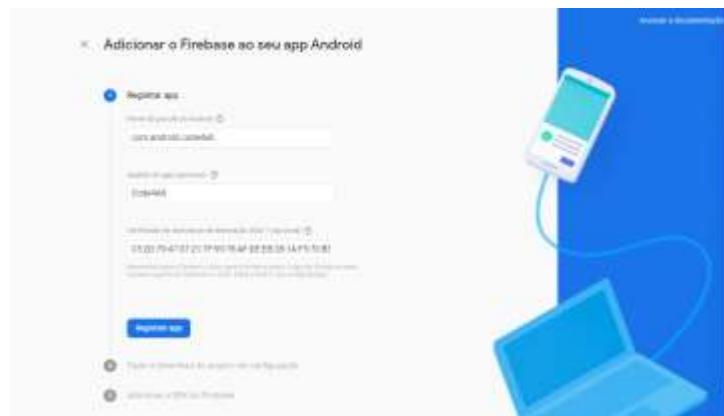


Figura 98: Colocação da chave "SHA-1" dentro do registo da app no firebase

Para adicionar as configurações da minha base de dados ao meu projeto Android, é preciso fazer download de um arquivo chamado de “google-services.json”, dentro da pasta “app”, para que assim o meu projeto no Android Studio, consiga aceder à base de dados Firebase.



Figura 99: Download do "google-services.json" para o meu computador

Coloquei então o ficheiro anteriormente transferido da Firebase, com as configurações da minha base de dados, para dentro do meu “dossier” onde guardo todos os ficheiros relacionados com a PAP.

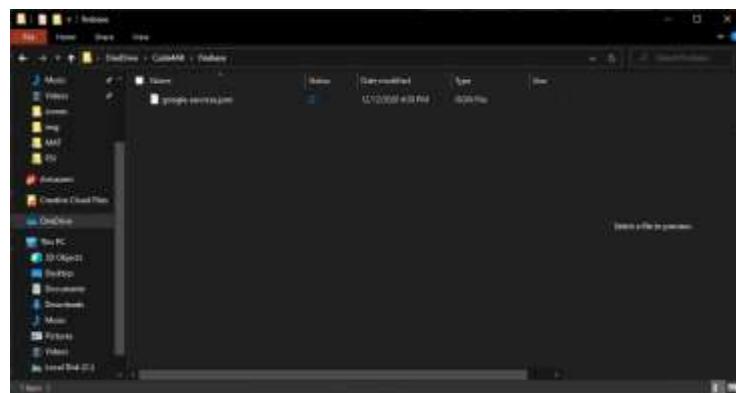


Figura 100: Colocação do ficheiro "google-services.json" dentro da pasta "firebase"

Com o ficheiro colocado dentro do meu “dossier”, bastou apenas copiar o mesmo, neste caso com o atalho “CTRL+C” e realizar um “CTRL+V” dentro do projeto, na pasta “app”. Observação importante, é preciso mudar a vista do projeto de “Android” para “Project”, visto que é preciso visualizar os ficheiros do projeto.

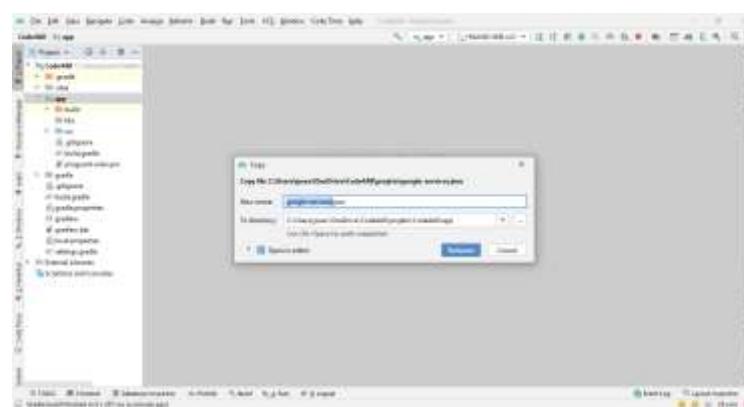


Figura 101: Importação do "google-services.json" para dentro do projeto

Na figura abaixo, representa-se o ficheiro que anteriormente importei, com as suas configurações

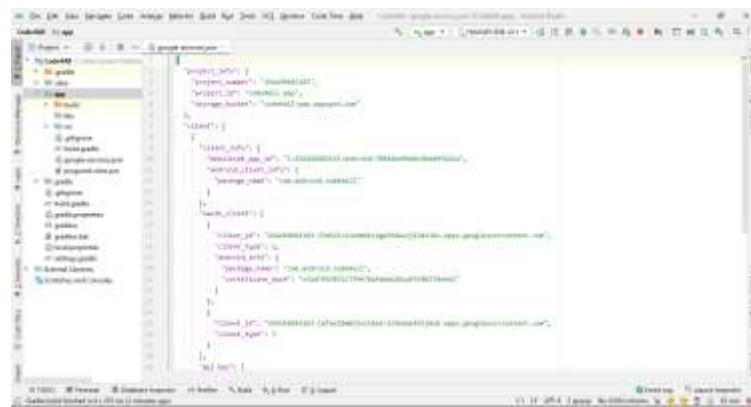


Figura 102: Ficheiro "google-services.json"

Seguidamente, passei ao adicionamento do SDK do Firebase para dentro do meu projeto, este SDK serve então para todas as librarias do Firebase, sejam importadas para o meu projeto, para depois as usar quando tiver a codificar as atividades onde irei precisar de criar utilizadores ou simplesmente de enviar dados dos mesmos para o Firebase.



Figura 103: Inserção da dependência do Firebase, dentro do "build.gradle" (projeto)

Adicionei então a dependência “classpath ‘com.google.gms:google-services:4.3.4’”, como dito na explicação anteriormente dada pelo próprio Firebase.

É normal, surgir aquela informação no topo a dizer que é preciso sincronizar as “Gradle files”, mas só vou realizar essa operação quando tiver todas as linhas que o Firebase me está a pedir para colocar, estiverem integradas no meu projeto.

Figura 104: Inserção do código necessário para dentro do ficheiro "build.gradle" (projeto)

Agora falta realizar a mesma inserção de código e librarias, para dentro do ficheiro “build.gradle”, mas desta vez a nível da “app”.



Figura 105: Inserção da dependência Firebase "BoM" e da "firebase-analytics" (app)

Abri então o ficheiro “build.gradle” mas direcionado para a “app”, para então passar a adicionar as duas importações que tinha por fazer.

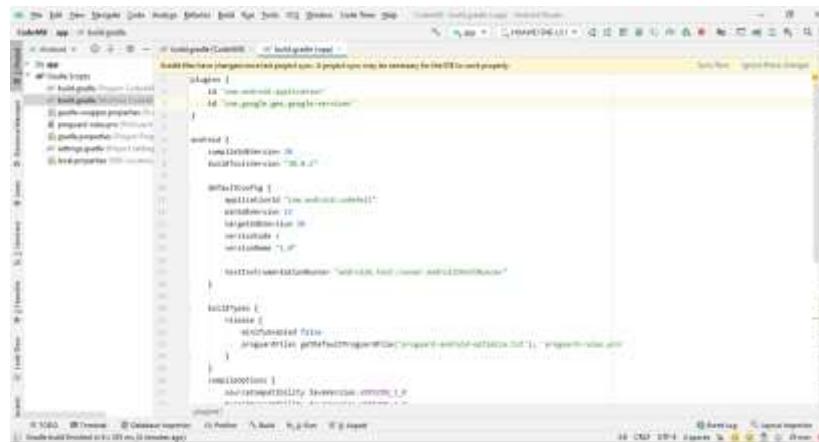


Figura 106: Abertura do ficheiro "build.gradle" ao nível da "app"

Passei então à colocação de um comentário a dizer “Firebase” para quando vier importar outras librarias, ter a mínima noção de qual é que faz parte de cada um dos “implementation”.

A implementação do “Firebase BoM”, serve para facilitar a vida do programadores e faz com que nós ao introduzir a implementação “com.google.firebaseio:firebase-analytics” por exemplo, antigamente, teríamos de indicar uma versão da mesma, mas com esta implementação já não é preciso, basta apenas colocar o elemento da Firebase que irá ser utilizado e o mesmo será atualizado automaticamente.

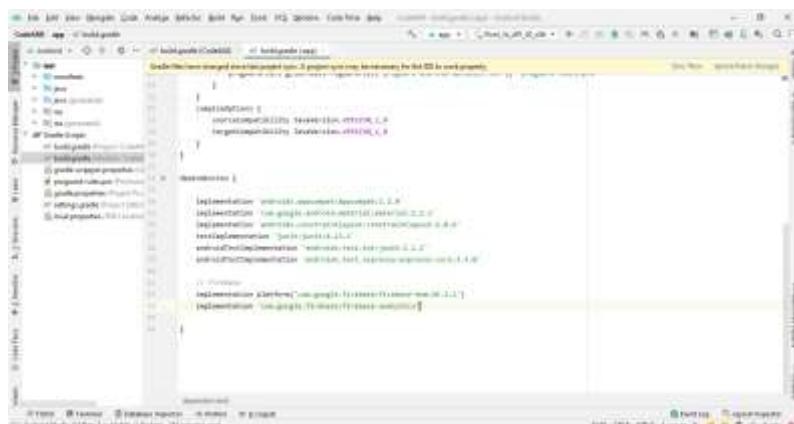


Figura 107: Implementação do "Firebase BoM" e do "Firebase Analytics"

Com a implementação das librarias necessárias dada como completa, posso dizer que a minha base de dados (Firebase) está completa e a ligação à mesma também.

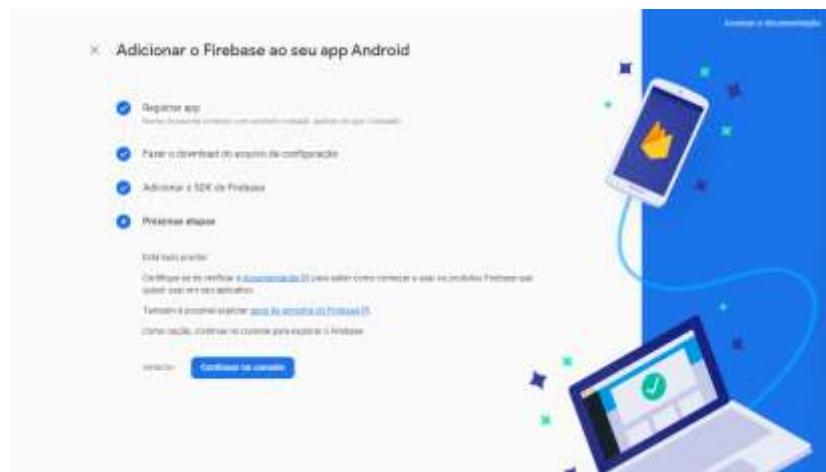


Figura 108: Ligação da app à Firebase finalizada

SplashActivity

Primeiramente foi preciso criar uma atividade em branco, visto que não vem incluído neste projeto em branco, logo só contém os ficheiros essenciais para a sua execução básica.

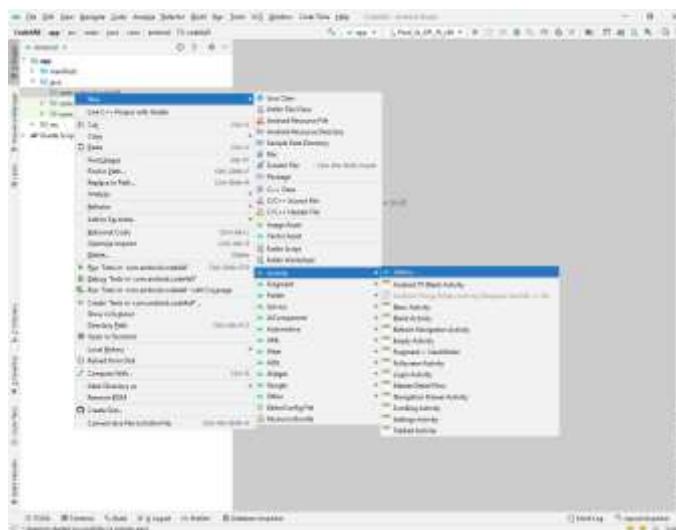


Figura 109: Criar Atividade "SplashActivity"

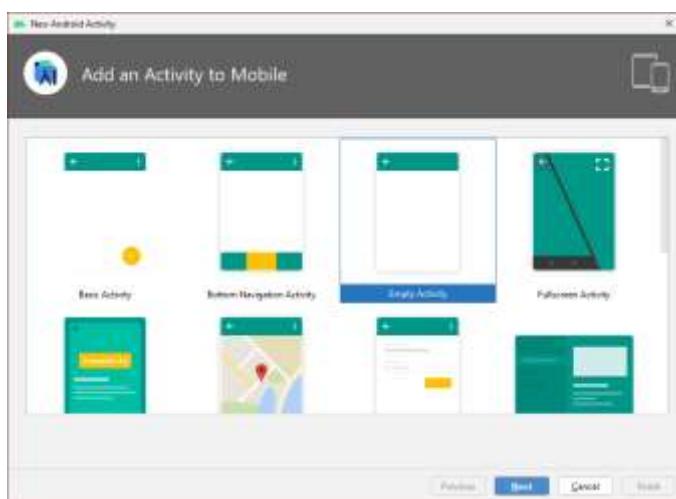


Figura 110: Escolha do Tipo de Atividade

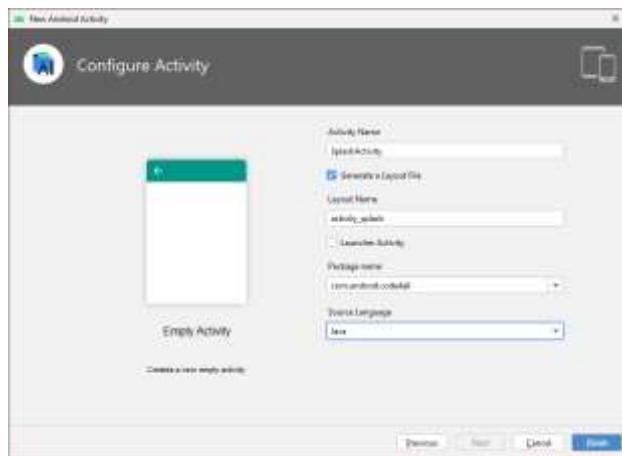


Figura 111: Configuração da Atividade

Na figura acima, pode-se configurar a atividade, como o seu nome, nome do layout agregado, nome do pacote, linguagem da atividade, podendo variar entre Java e Kotlin, passo a salientar, que tem-se ainda uma opção de “Launcher Activity” que é bastante útil e tem como função definir a atividade a criar como atividade principal deste projeto, neste caso não utilizei e usei antes o método manual de definir a atividade como principal.

Na figura abaixo, encontra-se em aberto o ficheiro da atividade.

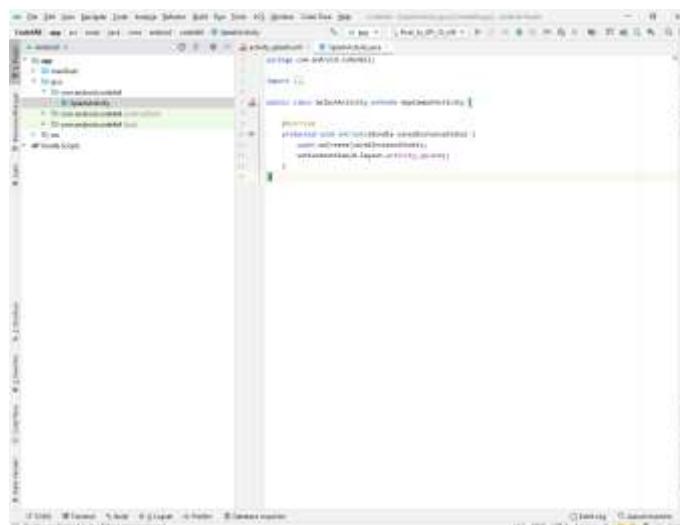


Figura 112: "SplashActivity" criada com sucesso

Agora sim, dei início ao desenvolvimento da `SplashActivity`, que neste caso significa uma atividade de introdução, a mesma foi usada para fazer o carregamento de componentes da aplicação. Uma ideia que precisa de ser reforçada é que o layout desta atividade vai ser igual ao que foi realizado anteriormente no Adobe XD, tal como as restantes atividades/fragmentos futuros deste projeto.

Com a criação desta atividade, foram criados dois ficheiros no meu projeto, o “`SplashActivity.java`”, o arquivo no qual irei realizar o código, e o ficheiro “`activity_splash.xml`”, o arquivo onde irá ficar o layout da atividade.

Nesta parte, surgiu um pequeno problema, como criei um modelo de raiz, ou seja, sem quase nenhum “apoio” por parte do Android Studio, vou ter que definir esta `SplashActivity`, como a atividade principal, ou seja, todas as vezes que esta aplicação for executada a mesma vai ter de passar sempre por esta atividade.

Para corrigir isso é necessário ir ao arquivo “`AndroidManifest.xml`”, o arquivo que leva algumas das configurações mais importantes do projeto, tais como nome, tema os ícones, entre outros...



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.code4all">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Code4All"
        android:supportsRtl="true"
        android:theme="@style/Theme.Code4All">
        <activity android:name=".SplashActivity">
            </activity>
    </application>
</manifest>
```

Figura 113 Ficheiro `AndroidManifest.xml`

Como representado na imagem anterior, já se encontra no mesmo a atividade que criei anteriormente, mas para corrigir o problema foi necessário adicionar um pequeno código que indicará que a SplashActivity será então a atividade principal e inicial deste projeto. Para isso foi necessário adicionar o seguinte código, dentro da etiqueta “<activity>”:

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Figura 114: Código para definir a SplashActivity como principal

O respetivo código “android.intent.action.Main” é para informar que esta atividade é sim a atividade principal, e o outro código “android.intent.category.LAUNCHER” é para informar que é com esta atividade que vai ser iniciada a aplicação.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package=".com.android.code4all">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Code4All"
        android:supportsRtl="true"
        android:theme="@style/Theme.Code4All">
        <activity
            android:name=".SplashActivity"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Figura 115: Código anteriormente referido para o SplashActivity

Nesta figura, dá para evidenciar que foi colocado o código anteriormente referido, dentro da etiqueta “<activity>” do SplashActivity.



Figura 116: Parte inicial da SplashActivity

Como dá para observar na figura, após iniciar a aplicação, a mesma redirecionou-me para a única atividade que tenho, ou seja, está a fazer aquilo que lhe planeei para fazer.

Depois de realizar então esta pequena correção no código, passei ao desenvolvimento do layout desta atividade. Para isso, foi preciso primeiro fazer uma pequena pesquisa, para saber se conseguia então adicionar umas fontes customizadas, tal como realizei no layout, no Adobe XD.

Com essa pesquisa, cheguei à conclusão que é possível sim, adicionar as fontes que eu quisesse, bastando então seguir o tutorial que se encontra na documentação dos desenvolvedores de android.

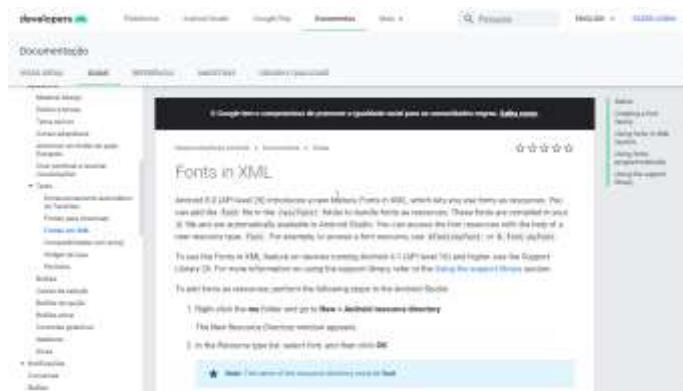


Figura 117: Site que usei para colocar as fontes no projeto

Com a imagem do site na página anterior referido, eu apenas tive de seguir as instruções dadas no mesmo.

Começando então, por ir buscar as fontes que precisava, que neste momento eram duas, a “Montserrat Bold” e a “Montserrat Medium”, para isso, precisava então de localizar e fazer download das mesmas para o meu computador, para isso foi preciso ir ao site <https://fonts.google.com/specimen/Montserrat>, como referido em baixo.

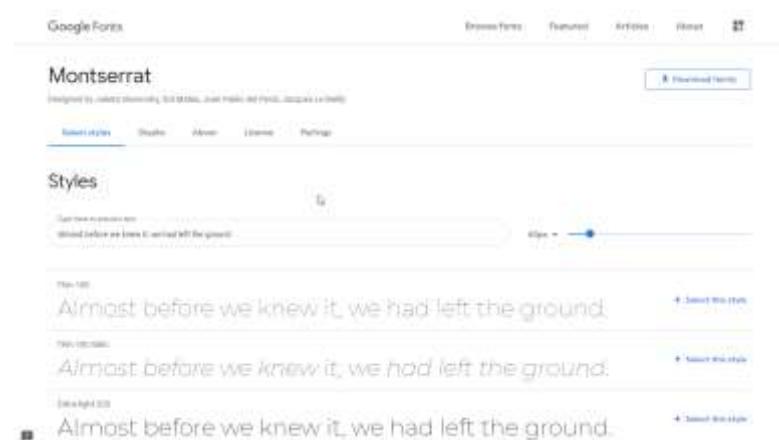


Figura 118: Site usado para fazer o download da fonte "Montserrat"

Passei, então ao download do pack completo da família “Montserrat”, como se apresenta na figura abaixo.

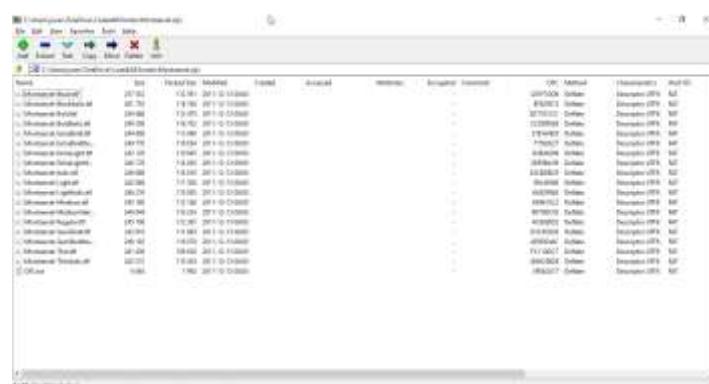

 A screenshot of a Windows file explorer window. The title bar says "U:\Users\jose.silva\Downloads\1_Montserrat\montserrat.zip". The file list shows various files and folders related to the Montserrat font family, including "Montserrat-Bold.woff", "Montserrat-Bold.woff2", "Montserrat-Medium.woff", "Montserrat-Medium.woff2", "Montserrat-Regular.woff", "Montserrat-Regular.woff2", "Montserrat-Semibold.woff", "Montserrat-Semibold.woff2", "Montserrat-Thin.woff", "Montserrat-Thin.woff2", "Montserrat-VeryLight.woff", "Montserrat-VeryLight.woff2", "Montserrat-VariableFontWeight.woff", "Montserrat-VariableFontWeight.woff2", "OTL.css", and "OTL.css.map". The file sizes range from 10.00 to 10.94 KB.

Figura 119: Arquivo zip com a fonte "Montserrat"

Cofinanciado por:







Página | 96

Depois passei então à extração da família “Montserrat”, para dentro de uma pasta no meu dossier da PAP, chamada “fontes”, após isso, bastou apenas selecionar as fontes que precisava para esta atividade, tal como referi anteriormente.

Seguidamente, fui ao Android Studio realizar o que foi mostrado anteriormente no site dos desenvolvedores de android, para então adicionar as fontes customizadas.

Para isso, foi preciso clicar com o botão direito do rato em cima da pasta “res” e ir em “New” após isso bastou apenas pressionar em “Android Resource Directory”, tal como mostrado na imagem.

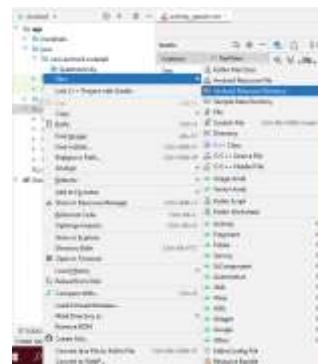


Figura 120: Passos para criar uma nova diretória de recursos do Android

Após realizar os passos referidos anteriormente, foi aberta uma nova janela, como mostrado na figura abaixo, agora bastou configurar a mesma conforme o que queria, neste caso é uma diretoria de fontes, para então colocar as fontes desejadas.

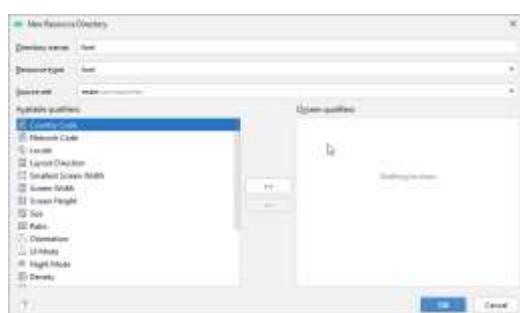


Figura 121: Criação da diretoria do tipo "font"

Após isso será criada uma nova pasta chamada “font”, com a pasta criada, foi só preciso selecionar as duas fontes que precisava, copiar as mesmas e colar na pasta “font” dentro da diretoria “res” ficando desta maneira.

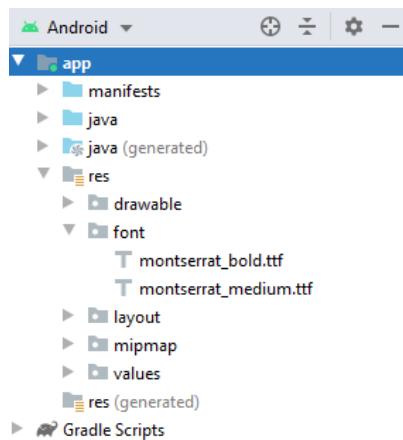


Figura 122: Resultado da pasta "font" com as duas fontes necessárias

Foi preciso, nesta altura efetuar algumas mudanças de cor, foi então necessário alterar alguns valores que estão por predefinidos no ficheiro “colors.xml”, onde ficam todas as cores customizadas gravadas. O ficheiro inicialmente teve este aspeto, como na figura abaixo se representa.



Figura 123: Ficheiro "colors.xml" com os valores pré-definidos do Android Studio

Após de ter observado este ficheiro, vi que tinha cores, que de certa forma, não eram de meu interesse e teria então de modificar este ficheiro para colocar apenas as cores que realmente necessitava, inicialmente, só precisava de três cores, uma cor primária, uma cor primária mais escura e uma cor diferente das outras e que se destacasse. Depois de ter realizado as alterações, o ficheiro ficou da seguinte forma.

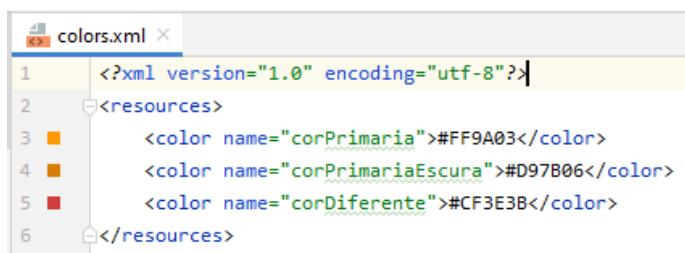


Figura 124: Ficheiro "colors.xml" após da modificação de cores para a SplashActivity

De certa forma, o ficheiro ficou visualmente mais apresentável, na minha opinião. Após ter feito esta modificação, foi preciso efetuar as mesmas modificações nos ficheiros "themes.xml" e no "themes.xml" (night), uma pequena informação, é que existem dois temas, um para que se o telefone estiver em modo branco siga umas cores e outra para que se o telefone estiver em modo escuro siga outras.

Basicamente é para isso que serve, por isso é que têm o mesmo nome, para que depois o modo de visualização do telefone se adapte à aplicação e vice-versa. Esta funcionalidade apareceu na última versão do Android Studio, até ao momento, neste caso a 4.1, lançada a 12 de outubro de 2020.

Estes ficheiros "themes.xml", como é de esperar, iam buscar as suas cores do tema, ao antigo ficheiro modificado "colors.xml", mas como apaguei e modifiquei o mesmo, se executasse a aplicação assim, ia falhar, visto que estou a evocar cores que foram apagadas, ou seja, cores customizadas que não existem. O aspetto inicial deste ficheiro era da seguinte forma, como se pode visualizar, na próxima página:



```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.Code4All" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor" tools:targetApi="1">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Figura 125: Ficheiro "themes.xml" no seu aspeto inicial

Após ter estruturado o código conforme dito anteriormente, fiz as respetivas alterações no ficheiro, ficando assim da seguinte forma: Fiz esta configuração, pois é mais simples e fácil de entender e é o recomendado dentro de um projeto.



```
<resources>
    <!-- Tema Base da Aplicação -->
    <style name="Theme.Code4All" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Cores Principais -->
        <item name="colorPrimary">@color/corPrimaria</item>
        <item name="colorPrimaryDark">@color/corPrimariaEscura</item>
        <item name="colorAccent">@color/corDiferente</item>
        <!-- Cor da Barra de Notificações. -->
        <item name="android:statusBarColor">@color/corPrimariaEscura</item>
        <!-- Customizar o Tema Aqui. -->
    </style>
</resources>
```

Figura 126: Ficheiro "themes.xml" após de algumas modificações para a SplashActivity

Seguidamente, tive de copiar o código acima referido e colocar o mesmo no ficheiro "themes.xml" para aquele que é para o modo escuro do telefone (night).

Após ter finalizado essa tarefa, fui para o layout da SplashActivity e dei início ao processo de fazer as devidas configurações.

Com esta configuração adicional finalizada, posso então dar início ao desenvolvimento do arquivo XML de layout.

Comecei por inserir o meu logo, a versão do meu logo em PNG, para ser mais concreto e mudar a cor do fundo, como feito na parte do layout. Seguidamente, passei por introduzir dois textos, um a dizer “Powered By” e o seguinte a dizer “José Xavier”.

Primeiramente, comecei por ir buscar o ficheiro PNG do meu logo, como se mostra na figura abaixo apresentada.



Figura 127: Logo Code4All em PNG

Após o ter ido buscar, foi preciso copiar o mesmo, usando o atalho “CTRL + C” e voltar novamente para o Android Studio, ir à pasta “res\drawable” e aplicar o atalho de colar, sendo ele “CTRL + V”, imediatamente, apareceu a janela que se representa abaixo.



Figura 128: Escolher o destino do ficheiro PNG

Após isso, apareceu-me a seguinte janela, para confirmar realmente a inclusão desta foto no meu projeto, podendo agora alterar-lhe o nome e reverificar o caminho do mesmo.

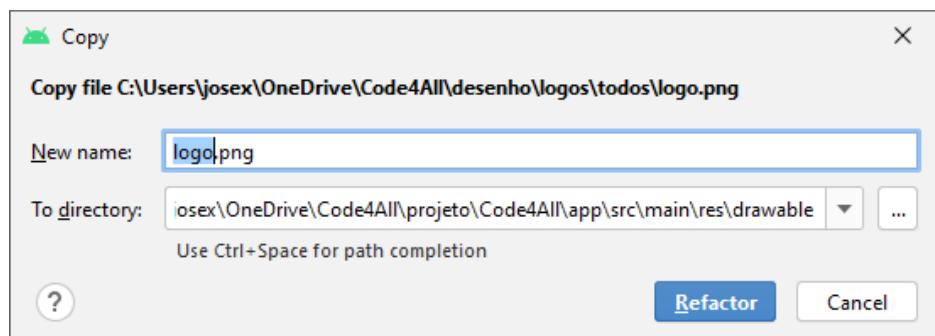


Figura 129: Confirmação da inclusão do logo

Para verificar, que o ficheiro estava realmente no projeto, abri a pasta “res/drawable” e realmente observei que o mesmo lá estava.

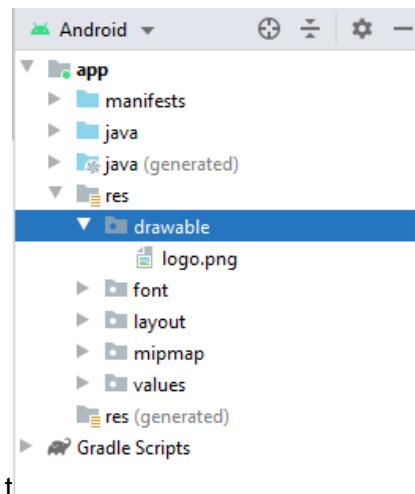


Figura 130: Visualização da pasta "res/drawable" com o logo

Após a inclusão do logo no meu projeto, pude logo passar a adicionar uma “ImageView” no layout desta atividade, como represento na figura abaixo, começando agora o desenvolvimento no ficheiro “activity_splash.xml” o ficheiro que é apresentado ao utilizador.

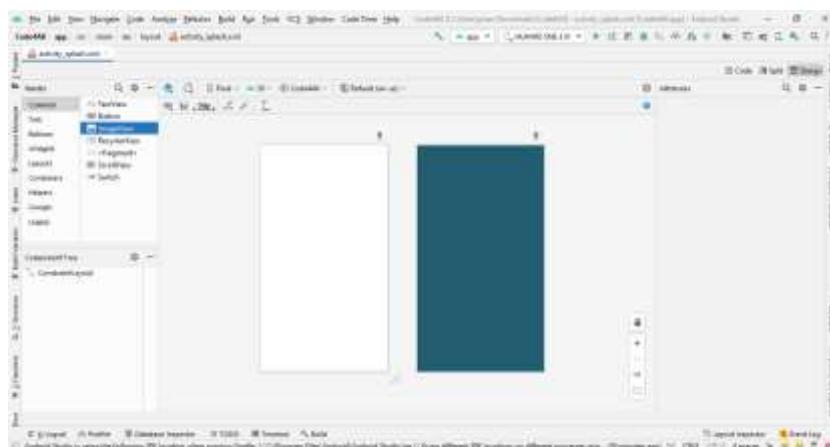


Figura 131: ImageView no layout da SplashActivity

Agora, basta apenas, arrastar a “imageView” da “Palette” para o layout em si, que de certa forma, irá abrir a seguinte janela.

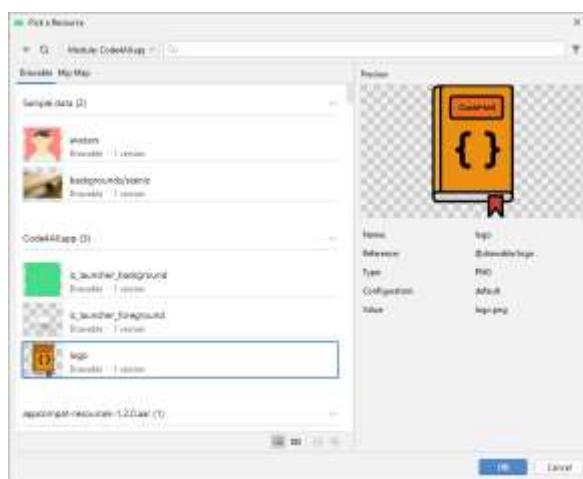


Figura 132: Criação do “ImageView” e seleção do logo como Imagem

Após clicar no botão “Ok”, foi de imediato colocado o logo da aplicação no layout, como se pode visualizar em baixo.

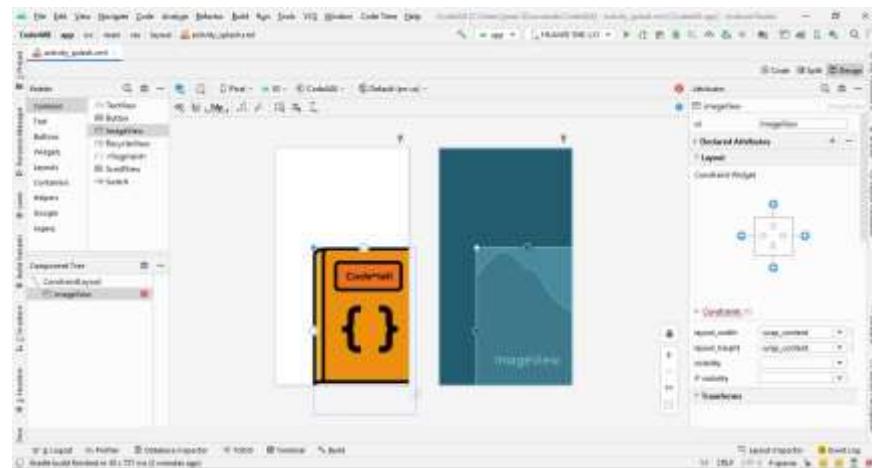


Figura 133: Inserção do Logo no layout

Após adicionar a imagem, foi preciso realizar as configurações consoante as suas dimensões, visto que inicialmente, quando uma “ImageView” é adicionada, tanto a largura como a altura, estão a “wrap_content”, ou seja, a imagem vai ocupar o seu tamanho total no layout e como não pretendo esse efeito, neste caso, prefiro definir tamanhos manuais e apliquei no logo o tamanho 200dp.

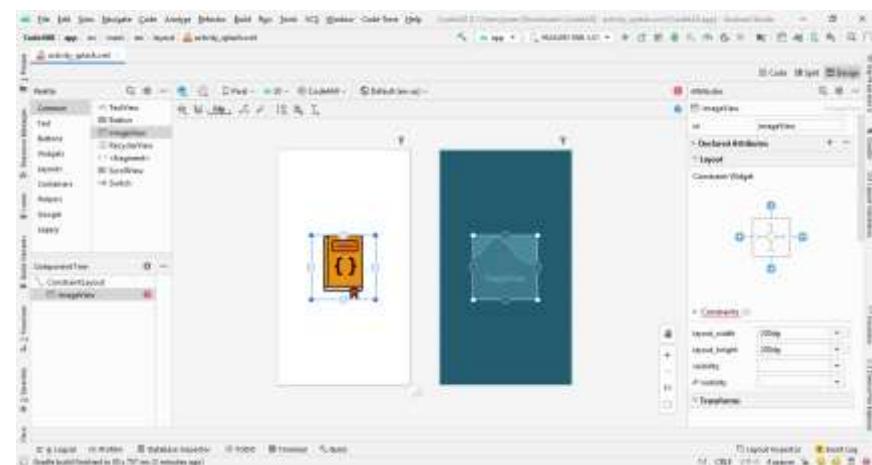


Figura 134: Logo no layout com 200dp x 200dp

Agora, passei ao alinhamento do logo no centro do layout, para que o mesmo fique bem centralizado e que em cada dispositivo se adapte ao centro do mesmo.

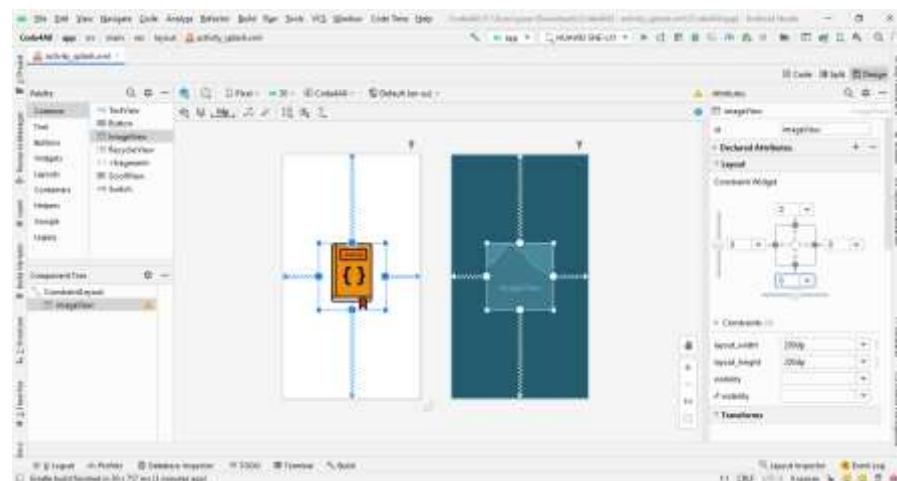


Figura 135: Alinhando o logo no centro do layout

Logo de imediato, surgiu um aviso, dizendo que deveria colocar um texto agregado à imagem para que de certa forma, os leitores de ecrã ou outras ferramentas de acessibilidade pudessem funcionar sem problemas.

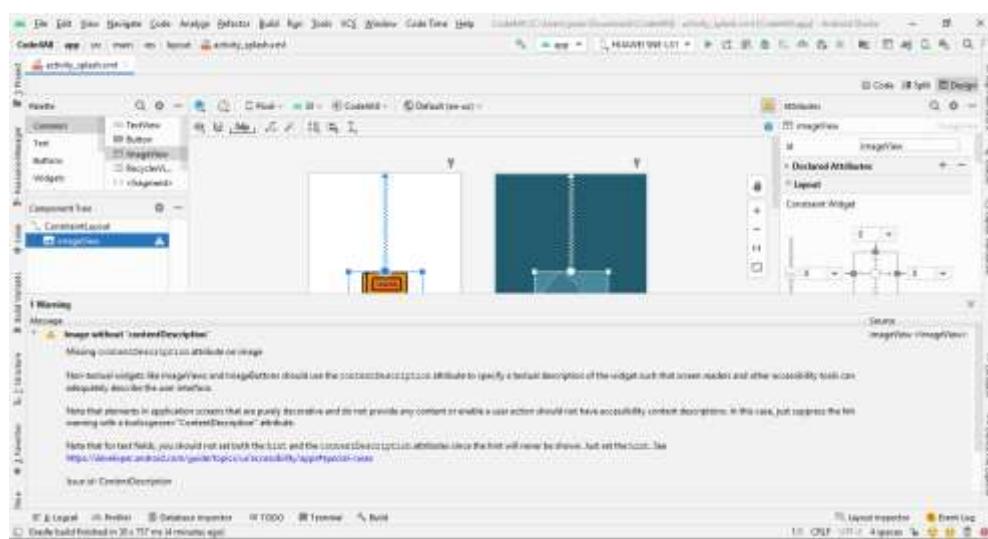
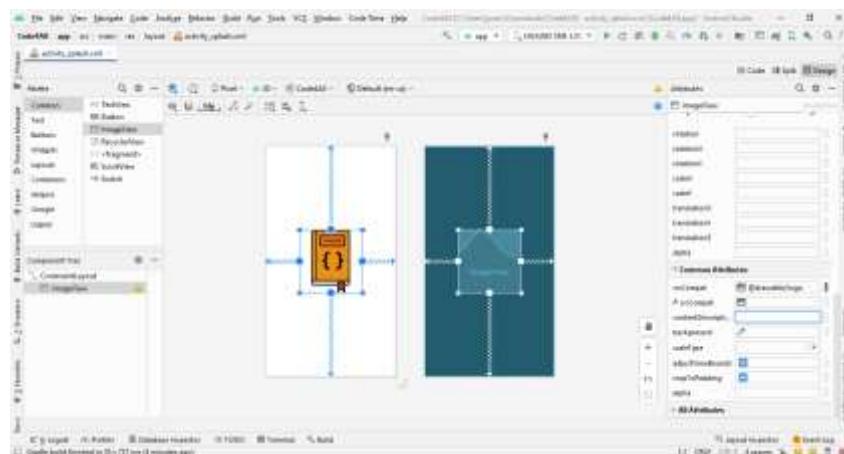


Figura 136: Aviso para colocar uma descrição no logo

Passei então para a colocação de uma descrição na imagem que contém o logo da minha aplicação.



Após ter clicado na barra que se encontra ao lado da caixa de texto, abriu a seguinte janela, para então colocar uma descrição na imagem, para isso, bastou apenas clicar no “+” e ir em “String Value”.

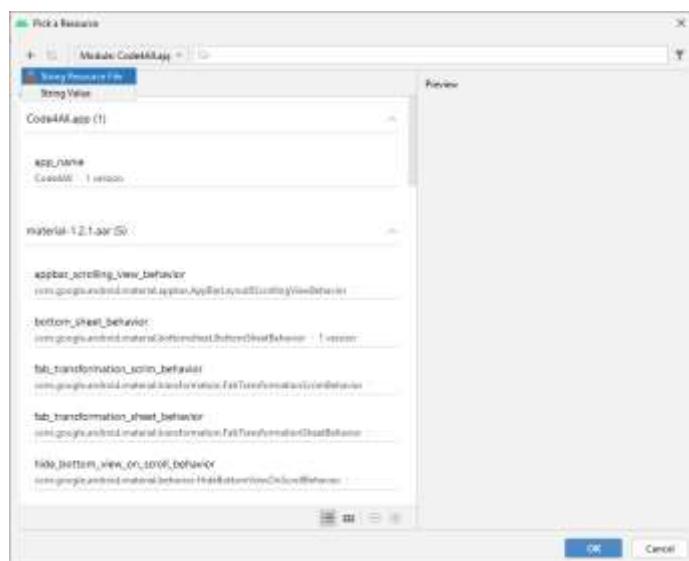


Figura 137: Criar um valor onde seja armazenado uma string

Após ter realizado os passos ditos anteriormente, foi necessário configurar o valor da “String” que iria ser armazenado e configurei o mesmo da seguinte forma.

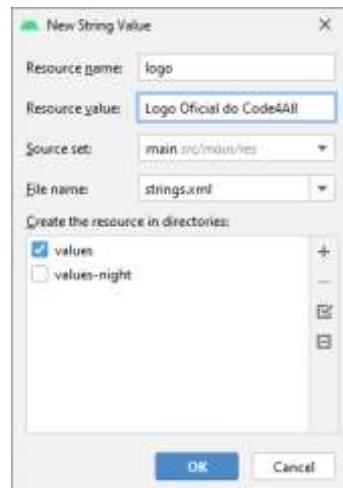


Figura 138: Configuração do valor “String” em relação à descrição da imagem

Após ter finalizado o realizado em cima, cliquei em “OK”, que depois me levou para a janela onde estive anteriormente, agora sim, bastou apenas selecionar o recurso chamado “logo”, como criado no passo anterior, e pressionar “OK”

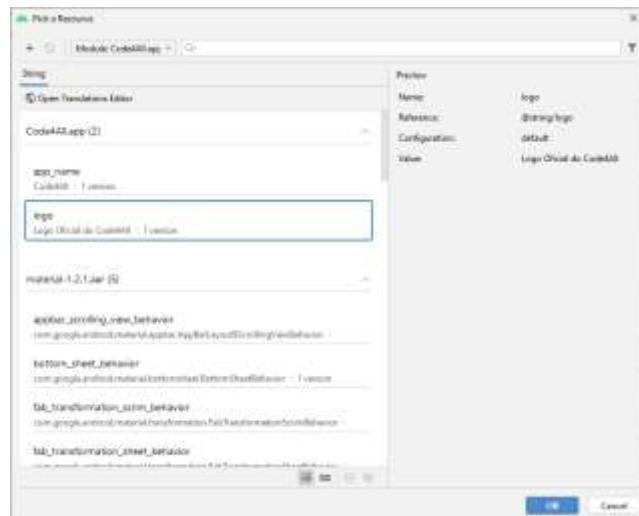


Figura 139: Seleção do recurso logo como descrição da imagem

Como se pode visualizar no canto direito, o atributo “contentDescription”, já tem um valor, o mesmo é o recurso “String” que criei anteriormente.

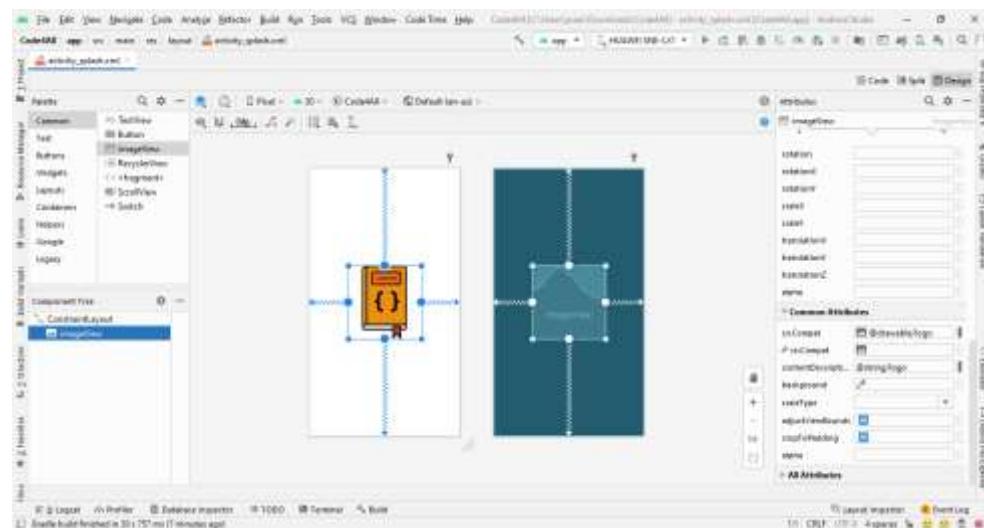


Figura 140: Visualização do atributo "contentDescription" no logo

Passando então à cor do fundo, tive de selecionar o componente “ConstraintLayout” e pesquisar pelo o atributo “background” e clicar na barra, que se encontra logo após a caixa de texto.

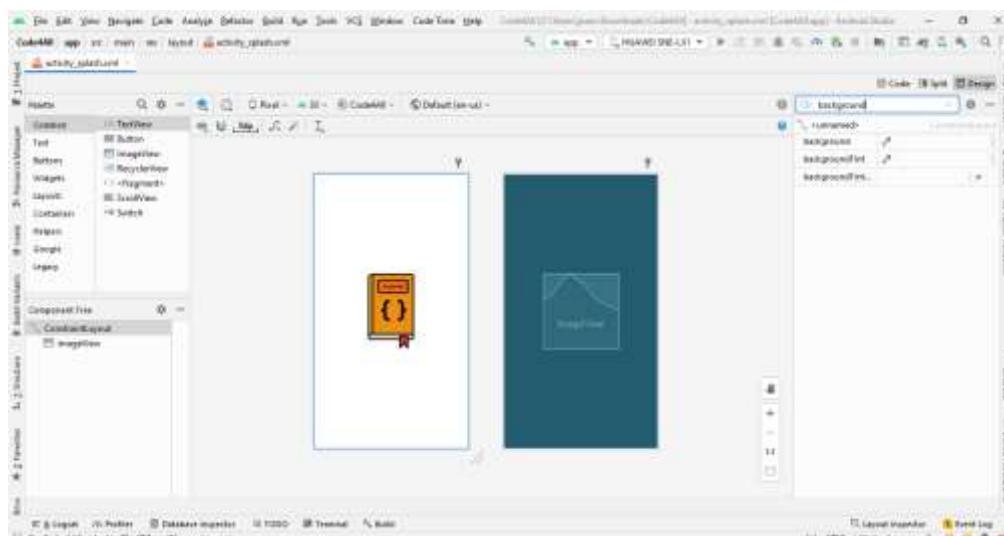


Figura 141: Modificação da cor de fundo do layout

Foi necessário dirigir-me à aba onde diz “Color”, para então selecionar uma cor já criada anteriormente no ficheiro “colors.xml”.

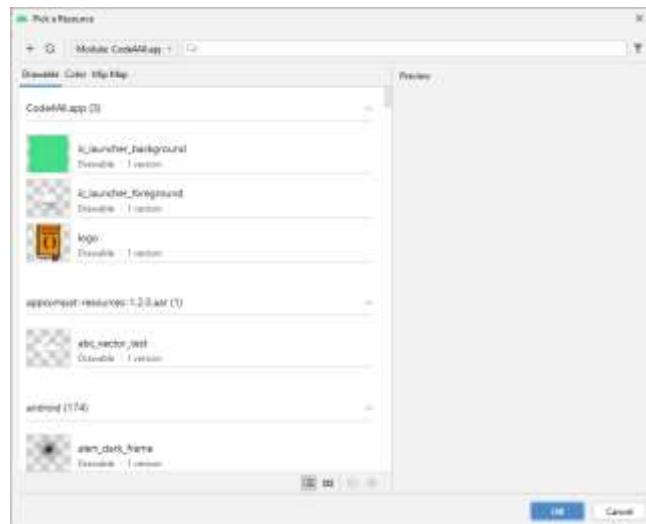


Figura 142: Configuração da cor de fundo

Após isso, bastou apenas selecionar o atributo “corPrimaria”, para então definir esta cor como cor de fundo do layout, para confirmar a alteração, bastou clicar em “OK”.

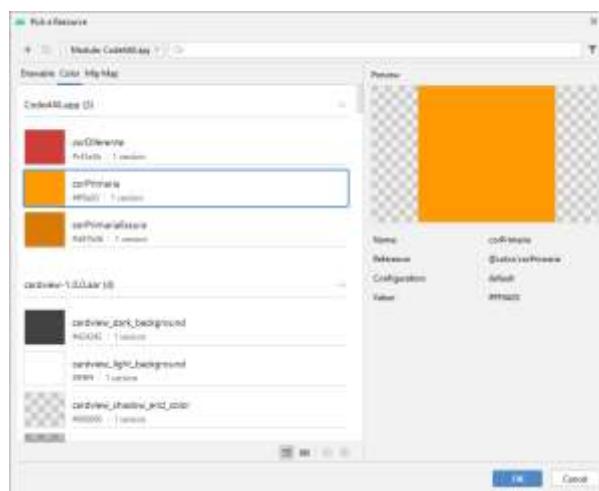


Figura 143: Seleção e confirmação da cor de fundo

Com esta configuração de fundo totalmente finalizada, passo então a adicionar o componente “TextView”, para adicionar o texto “Powered By”, como já dito anteriormente.

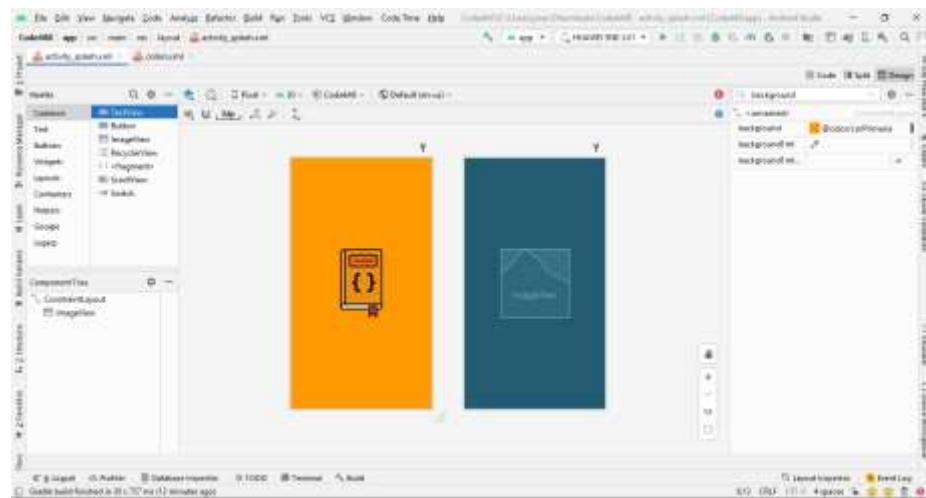


Figura 144: Seleção do TextView como componente

Para adicionar um “TextView”, basta apenas arrastar o mesmo para o layout.

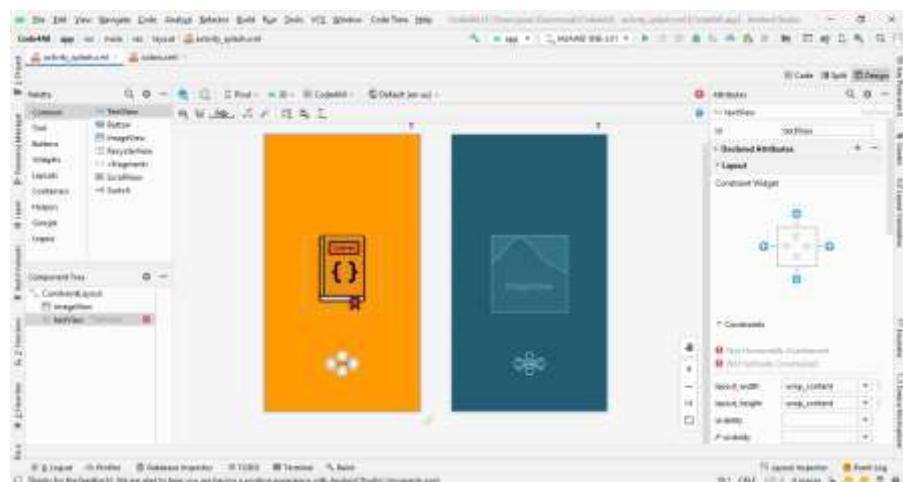


Figura 145: Colocação do "TextView" no layout

Agora, basta apenas realizar o que já foi feito anteriormente e criar um recurso “String”.

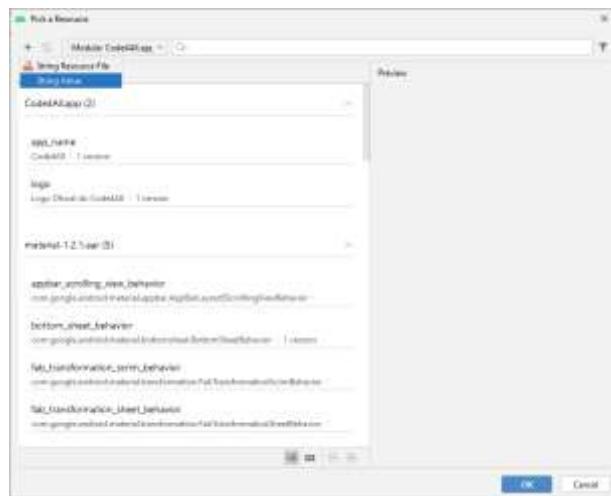


Figura 146: Criação de um novo recurso String para o TextView

Agora basta configurar esse mesmo recurso e eu configurei-o da seguinte maneira.

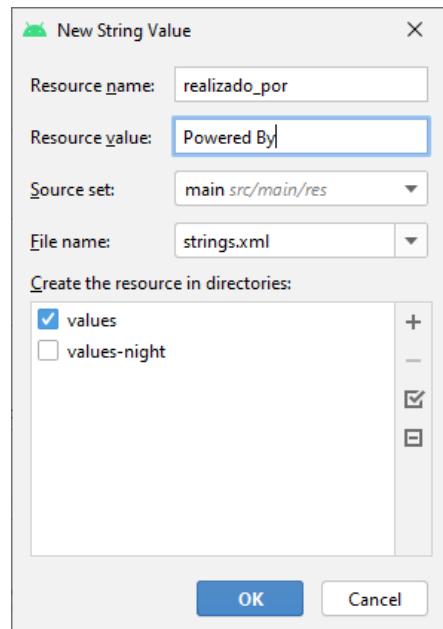


Figura 147: Configuração do recurso String para o TextView

Depois do recurso estar criado, basta selecionar o mesmo, para que este seja definido no TextView.

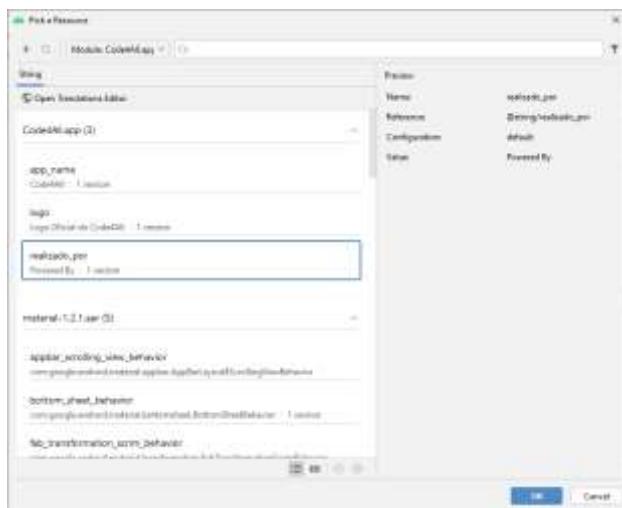


Figura 148: Definição do recurso anteriormente criado no TextView

Agora falta configurar os dois “TextView’s” com o seu tamanho e letra adequado. Para isso, é muito simples, basta selecionar o “TextView” e ir na propriedade “fontFamily” onde no primeiro TextView coloquei “fontserrat_medium” e no segundo “TextView” coloquei “fontserrat_bold”.

No tamanho, coloquei 16sp no “Powered By” e no “José Xavier” coloquei 20sp alinhando também ambos os componentes ao centro.

Por fim, foi preciso realizar as devidas configurações no layout destes “TextView’s”, visto que ambos têm de se adaptar consoante o dispositivo em que a aplicação está a correr.

No textView, liguei ao mesmo verticalmente ao componente textView2, usando uma distância de 24dp. No textView2, liguei-o ao inferior do layout e usei um espaçamento de 32 dp (verticalmente).

Para terminar, todos os “TextView” deste layout estão em “wrap_content” tanto na altura como na largura, visto que são componentes de texto, não faz sentido ter um tamanho definido.

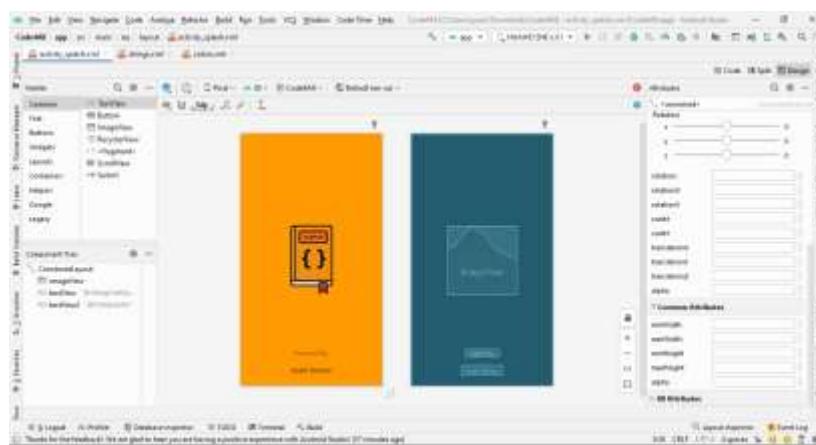


Figura 149: Resultado após as alterações no layout da `SplashActivity`

Seguidamente, passei à orientação e organização do ficheiro “`strings.xml`” localizado na pasta “`res/values`”, onde ficam todos os recursos “String” que criamos anteriormente, sendo o seu aspeto igual ao referenciado em baixo.

```
<resources>
    <string name="app_name">Code4All</string>
    <string name="logo">Logo Oficial do Code4All</string>
    <!-- <string name="realizado_por">Powered By</string>
        <string name="autor">José Xavier</string>
    -->
</resources>
```

Figura 150: Ficheiro “`strings.xml`” após de ter adicionado os recursos “String”

Depois de uma breve reorganização deste ficheiro, mudei o nome da string “`app_name`”, visto que pretendo utilizar sempre, ou quase sempre, elementos em português, mudei o mesmo para “`nome_da_app`” e adicionei dois comentários de separação, como se pode observar na figura abaixo, para que no futuro seja mais fácil de alterar os recursos e de encontrar a atividade em questão.

```
<resources>
    <string name="nome_da_app">Code4All</string>
    <!-- <string name="realizado_por">Powered By</string>
        <string name="autor">José Xavier</string>
    -->
</resources>
```

Figura 151: Ficheiro “`strings.xml`” após das modificações para a “`SplashActivity`”

Ao mudar o recurso “app_name” para “nome_da_app” é preciso também modificar o mesmo no ficheiro “AndroidManifest.xml” como se pode ver nas figuras abaixo.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.code4all">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.Code4All">
        <activity
            android:name=".SplashActivity"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Figura 152: Erro na propriedade "android:label" visto que o nome do recurso foi alterado



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.code4all">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/nome_da_app"
        android:supportsRtl="true"
        android:theme="@style/Theme.Code4All">
        <activity
            android:name=".SplashActivity"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Figura 153: Correção da propriedade "android:label"

Após este feito, estive a falar com a minha equipa de acompanhamento, para então analisar se valia a pena adicionar um “círculo” de carregamento.

No final de tudo, cheguei à conclusão que sim valia a pena, e mesmo eu o poderia fazer de uma forma fácil e intuitiva, utilizando o Photoshop.

Visto que foi aprovada esta ideia, foi o próximo passo que realizei, primeiramente, fui à procura de um círculo de carregamento, mas só uma imagem, para depois eu fazer a edição e tornar a mesma numa imagem de animação, o mesmo que GIF.

Mas primeiro, surgiu uma notificação dada pelo o Android Studio como ilustrado abaixo, que de certa forma são atualizações e sempre que existe uma é recomendado atualizar no momento.

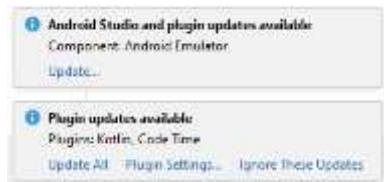


Figura 154: Atualização dos plugins

Após atualizar os plugins, como pedido pelo próprio Android Studio, passei então ao início do desenvolvimento do anteriormente referido GIF de carregamento.

Comecei por ir ao site <https://feathericons.com>, que são ícones “open-source” já que posso utilizar os mesmos para o desenvolvimento da minha aplicação, pois são completamente gratuitos. Passei ao download do ícone base de carregamento, sendo ele o seguinte.

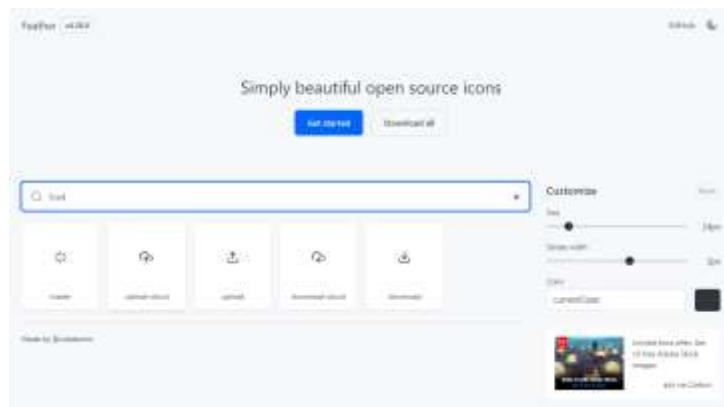


Figura 155: Página do ícone "loader" que transferi

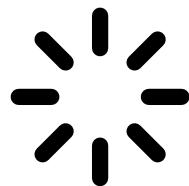


Figura 156: Ícone de carregamento

Um ícone bastante minimalista e que chama a atenção, os seus principais objetivos. Após isso, abri o Photoshop, e comecei por arrastar o ficheiro .SVG, do ícone para o mesmo. Mas deparei-me com um problema, visto que precisava de abrir uma “TimeLine” para então começar a criar o GIF, tal como indica a figura abaixo representada.



Figura 157: Visualização do ícone dentro do Photoshop

Após a pesquisa, cheguei à conclusão que até era bastante simples, apenas bastou ir à aba “Window” e depois clicar em “Timeline” como se pode observar na figura representada na seguinte página.

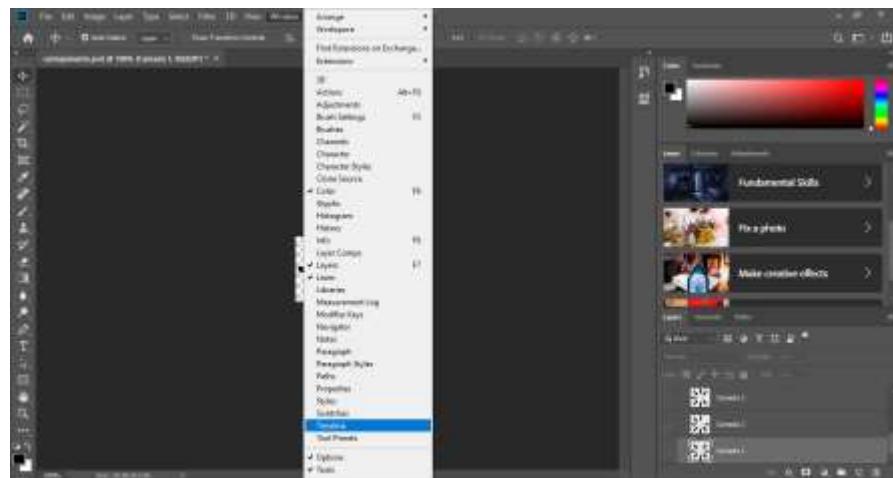


Figura 158: Passo de Abrir a "Timeline" no Photoshop

Depois de ter a “Timeline” aberta, foi bastante simples, apenas tive que copiar e colar as camadas, até que o círculo desse uma volta completa, e pintar de cinzento a parte da sucessão, o resultado final do GIF encontra-se em baixo.

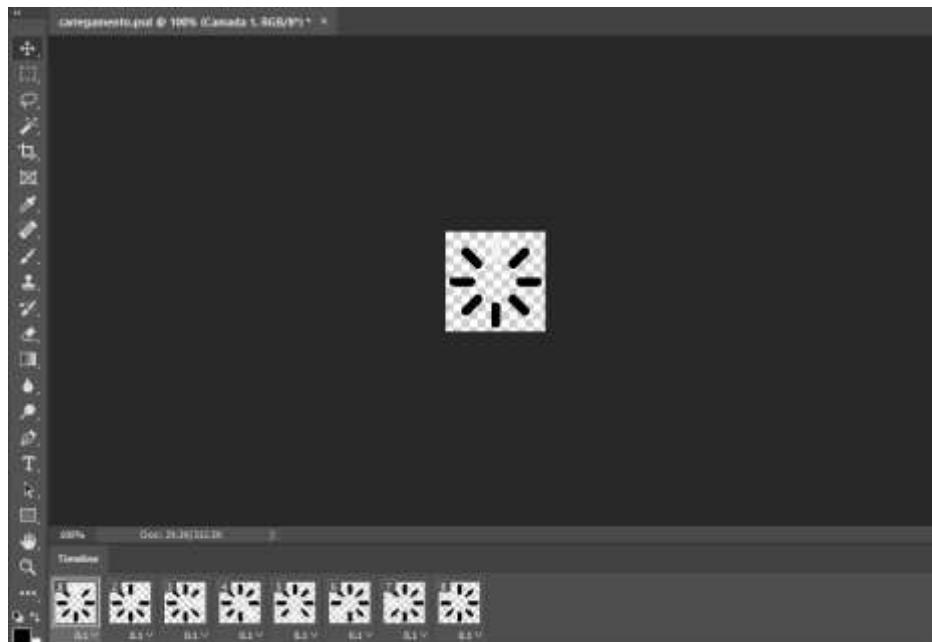


Figura 159: GIF de Carregamento finalizado

Agora, já com o GIF feito, foi feita a exportação do mesmo e foi realizada da seguinte forma.

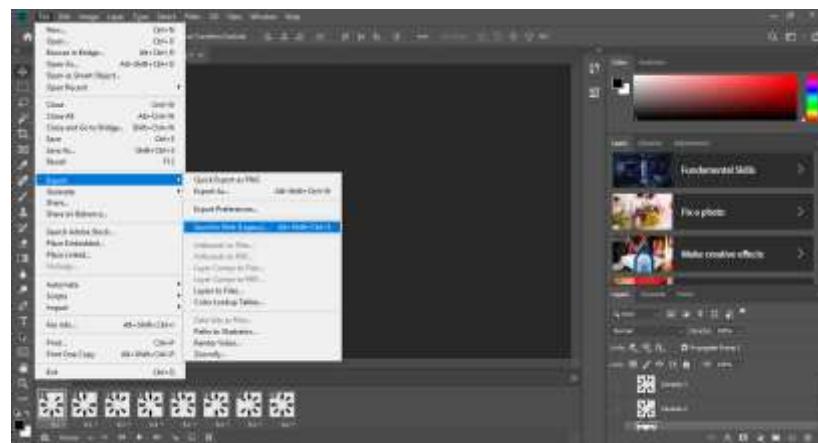


Figura 160: Guardar o GIF para a Web

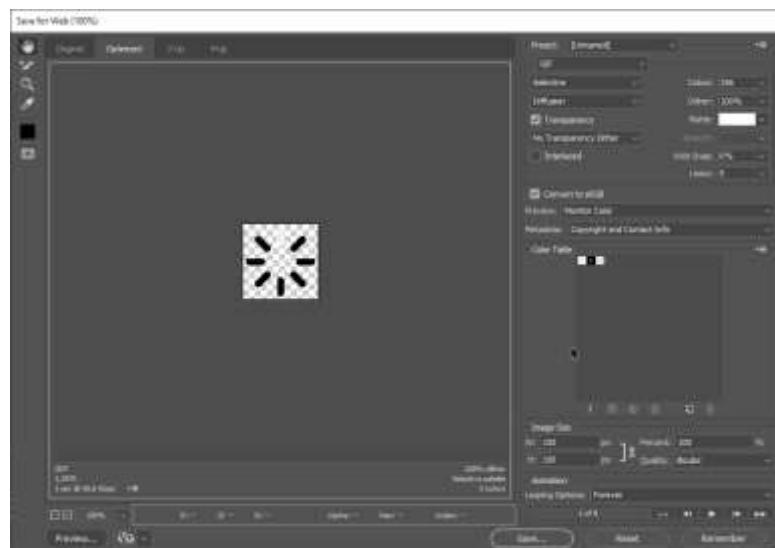


Figura 161: Janela para escolher o tipo de exportação

Após fazer as configurações, tal como na figura acima indicada, bastou apenas clicar em “Save...” e após isso escolher a localização onde pretendia guardar o ficheiro, neste caso guardei numa pasta própria que criei, para colocar tudo o que era relacionado com o projeto, o meu “dossier”.

Passando agora ao ficheiro `SplashActivity.java`, realizei nesta altura umas alterações ao nível do layout, mas com programação de Java.

Agora a minha ideia era colocar a `SplashActivity` como ecrã inteiro, ou seja, queria “tapar” de certa forma a barra superior de notificações, onde normalmente ficam informações básicas, tais como, as horas, a rede do utilizador, algumas notificações, o notch (no caso de o utilizador o possuir), entre outros...

Como não aprendi a fazer isso no curso que realizei na Udemy, foi preciso então realizar uma pesquisa de como se faria o mesmo e após algumas pesquisas encontrei este site chamado de StackOverflow, já mencionado anteriormente, um fórum que como disse no início do projeto, me vai dar muita ajuda neste projeto, como se apresenta na figura abaixo.

<https://stackoverflow.com/questions/57263678/hide-show-status-bar-with-notches>.

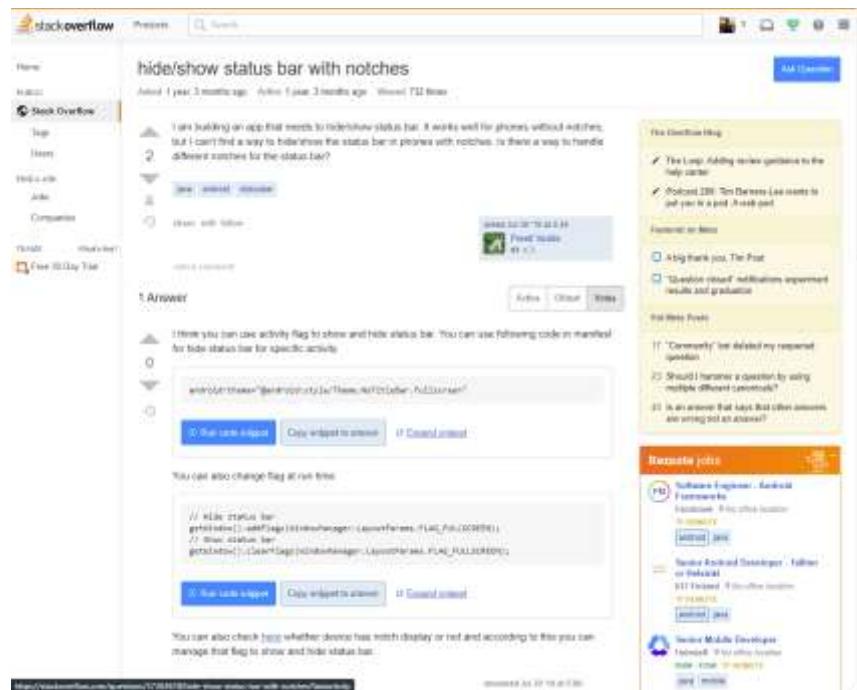


Figura 162: Página do StackOverflow - Esconder e mostrar barra de notificações

Com esta pesquisa, descobri que existem duas maneiras de solucionar o meu problema, a primeira é ir ao arquivo xml dos temas e modificar o tema inicial para um que não tenha uma barra de notificações e que seja de ecrã inteiro.

Na segunda opção, posso então fazer a mesma configuração, mas com código em Java, sendo ele colocado no ficheiro “SplashActivity.java”, visto que é o ficheiro que “manda” nesta atividade.

Concluindo, neste caso, escolhi a segunda opção, porque sempre que usei a primeira opção, em outras aplicações que já realizei, sempre me correu mal, por isso por segurança optei por este método, sendo os dois permitidos e usáveis e ambos fazem a mesma função.

Começando então a codificação em si, comecei por esconder a barra de notificações roxa, onde aparece o nome da aplicação (Code4All).

Utilizando o seguinte código, que me vai buscar a barra superior de ações, onde normalmente fica o nome de aplicação e também os famosos “três pontos” onde se consegue configurar para mais funcionalidades. O código é o apresentado na figura abaixo:

```
// Esconder barra de Ações  
getSupportActionBar().hide();
```

Figura 163: Código de esconder a barra de superior de ações

De seguida, tive o objetivo de tornar o ecrã em modo de “fullscreen”, ou seja, em modo de ecrã inteiro e para de certa forma esconder a barra de notificações superior. Para isso bastou apenas ir ao site, referido anteriormente e copiar e colar o código na IDE, tal como indicado na figura:

```
// Fazer com que a aplicação ocupe o ecrã inteiro, inclusive o "Notch" dos SmartPhones  
getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

Figura 164: Código para tornar a atividade em modo de ecrã inteiro

Mas surgiu um problema ao copiar e colar o código que se encontrava no StackOverflow, visto que existe ali uma palavra que está a vermelho, logo indica que algo não está certo.

Normalmente, a primeira abordagem que faço, é fazer o atalho “ALT+ ENTER” e perceber que erro é que eventualmente poderá ser e de imediato a IDE me apresenta as possíveis soluções, se no caso se aplicar.



Figura 165: Importação da classe "WindowManager"

Como neste caso, é uma classe que ainda não foi importada basta apenas clicar no “ENTER” para a classe ser adicionada automaticamente pela a IDE, isto é uma das vantagens das muitas que existem, em usar uma IDE, como o Android Studio.

Após corrigir este erro, o código anteriormente referido já apareceu em normal, como se pode visualizar na seguinte figura.

```
// Fazer com que a aplicação ocupe o ecrã inteiro, inclusive o "Notch" dos SmartPhones
getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

Figura 166: Importação realizada com sucesso

Após estas modificações que realizei anteriormente, a aplicação ficou da forma como se demonstra na figura:

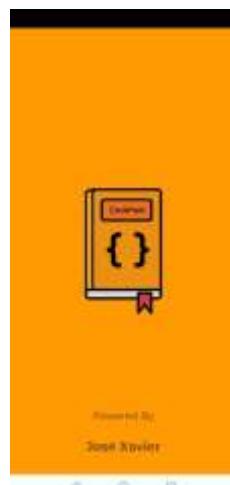


Figura 167: Visualização da app após de algumas alterações da UI

Após realizar a execução do projeto, verifiquei que existia uma barra preta no topo do ecrã e que a barra inferior estava a ser mostrada, o que supostamente não deveria acontecer, logo o código que realizei anteriormente não estava a realizar aquilo que pretendia inicialmente.

Neste caso, o código estava a correr a 100%, ou seja, sem nenhum erro, nem aviso, mas o problema era de facto o código que estava a utilizar, como referido anteriormente, não estava a completar as minhas expectativas.

Com este pequeno imprevisto, foi preciso então realizar outra pesquisa, mais uma vez, desta vez fui ao site dos desenvolvedores de Android para tentar retirar informação mais teórica e aprender com este “pequeno” erro da minha parte.

Quase de imediato, encontrei a teórica que está por trás dos layouts em ecrã inteiro, o que realmente necessitava.

Para então realizar as devidas mudanças, foi primeiro necessário, realizar outra pesquisa onde aprendi mais sobre o próprio layout de cada atividade, o site onde fui pesquisar esta informação, desta vez, foi o site dos desenvolvedores de Android, sendo o link este:

<https://developer.android.com/training/system-ui/immersive#EnableFullscreen>

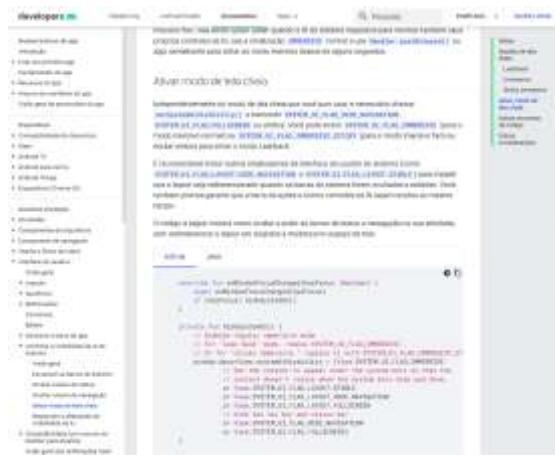


Figura 168: Téorica do modo de Visualização "FullScreen" de um layout

Depois de ter lido esta página toda e retirar a informação mais importante, percebi que não estava a realizar as coisas da forma mais correta.

Após esta análise, concluí que precisava ir buscar a view do layout e atribuir-lhe parâmetros próprios, como disponibilizado no site referido anteriormente.

```
// Fazer com que a aplicação ocupe o ecrã inteiro
View decorView = getWindow().getDecorView();
decorView.setSystemUiVisibility(
    View.SYSTEM_UI_FLAG_LAYOUT_STABLE
        | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
        | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
        | View.SYSTEM_UI_FLAG_FULLSCREEN
        | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
        | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY);
```

Figura 169: Código para colocar a aplicação em ecrã inteiro

Ao executar a aplicação obteve-se o seguinte resultado, como se pode observar na figura abaixo representada:

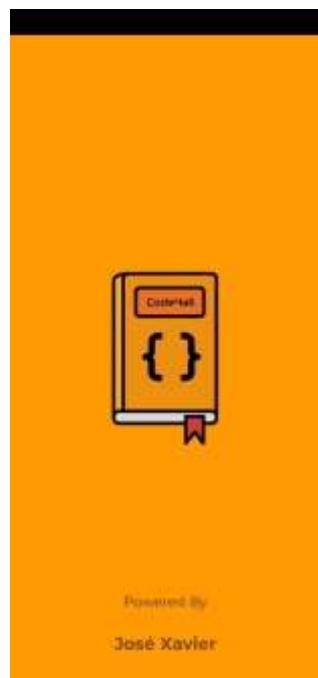


Figura 170: Resultado do Layout, após da configuração por código Java

Após de ter executado a aplicação já com as alterações ditas anteriormente, observei que de certa forma, estava uma barra preta no topo do ecrã, isto acontece devido ao facto do meu dispositivo ter um “Notch” que é bastante conhecida nos smartphones mais recentes e tornou se basicamente uma “moda” e a maior parte dos smartphones têm uma.

Este “Notch” foi criado para que de certa forma, o ecrã do telemóvel fosse ainda mais aproveitado e para manter os sensores e câmaras que necessitam de ficar na frente do telemóvel.

Por curiosidade, o primeiro telemóvel a “sair” com o Notch pela primeira vez, foi o IPhone X, desde então, a maior parte dos dispositivos tem saído com o mesmo, como referi anteriormente.

Procedi a uma pesquisa para saber como iria fazer para retirar a barra preta, visto que não fica nada bem no layout em si.

Nesta altura, também observei que quando bloqueava e desbloqueava o meu dispositivo android, algo de estranho acontecia na atividade, a barra superior, onde ficam as notificações, e a barra de navegação inferior voltavam ao seu estado normal e não se recolhiam automaticamente.

Após bloquear e desbloquear o telefone o layout da “SplashActivity” ficava da seguinte forma:



Figura 171: Bug do layout, após de bloquear e desbloquear o dispositivo Android

Comecei por pesquisar a solução do porquê do “Notch” estar a apresentar uma barra preta, em vez de estar a apresentar a cor laranja, como se encontra o resto do layout.

Mais uma vez vim parar ao StackOverflow, um dos meus fóruns favoritos e seguidamente procurei pela solução, como nesta pergunta haviam várias soluções experimentei uma a uma até que desse, até que cheguei à solução sendo ela apresentada na figura que se encontra na próxima página, sendo o link o seguinte.

<https://stackoverflow.com/questions/56353716/how-to-remove-top-status-bar-black-background>



Figura 172: Solução do bug da barra preta no "Notch" do dispositivo Android

Comecei por implementar o código do método “hideSystemUI” na minha “SplashActivity” e fazendo as alterações necessárias para colocar as variáveis, o próprio método, comentários, com nomes portugueses. Para que a sua leitura seja mais fácil, tanto para mim como para um possível leitor.

A 1^a alteração que tive de passar o método de “public static void”, para apenas “void” visto que vai ser utilizado nesta classe apenas. De seguida, declarei uma variável para entender mais facilmente o código com o nome “visualizacao”.

Por fim mudei o nome do método para “esconderInterface” e evoquei o mesmo no método “onCreate” um método que é executado uma única vez, na criação da atividade, neste caso “SplashActivity”, ficando assim este resultado:

```
void esconderInterface() {
    View visualizacao = getWindow().getDecorView();
    visualizacao.setSystemUiVisibility(
        View.SYSTEM_UI_FLAG_FULLSCREEN
        | View.SYSTEM_UI_FLAG_LOW_PROFILE
        | View.SYSTEM_UI_FLAG_LAYOUT_STABLE
        | View.SYSTEM_UI_FLAG_IMMERSIVE
        | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
        | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION);
}
```

Figura 173: Método “esconderInterface” da SplashActivity

```
    @Override
    protected void onResume() {
        super.onResume();

        // Evocar o método para que a aplicação ocupe o ecrã inteiro
        esconderInterface();
    }
```

Figura 174: Evocação do método "esconderInterface" no método "onResume"

O código acima descrito, é executado cada vez que o utilizador sai e entra na aplicação, como o próprio nome indica.

Agora falta a outra parte do código onde existe uma condição “IF”, que de certa forma verifica se a versão do android é maior ou igual à do Android Pie (Android 9.0) e se for verdade, é aplicado um código para que seja possível o próprio layout estender a sua visualização para além do Notch.

Coloquei esta condição dentro do mesmo método “esconderInterface”, para simplificar o código e por questões de organização, visto que tanto a condição como o restante código fazem a função de literalmente, esconder a interface, como indica o próprio método, ficando o mesmo da seguinte maneira:

```
void esconderInterface() {
    // Verificar se a versão do android é maior ou igual a 9 (Android Pie)
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
        getWindow().getAttributes().layoutInDisplayCutoutMode = WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_SHORT_EDGES;
    }

    View visualizacao = getWindow().getDecorView();
    visualizacao.setSystemUiVisibility(
        View.SYSTEM_UI_FLAG_FULLSCREEN
        | View.SYSTEM_UI_FLAG_LOW_PROFILE
        | View.SYSTEM_UI_FLAG_LAYOUT_STABLE
        | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY
        | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
        | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION);
}
```

Figura 175: Método “esconderInterface” da "SplashActivity"

Agora sim, com o código colocado e sem erros, aparentemente, passei à execução do código, o mesmo executou às mil maravilhas e os dois erros que ditei anteriormente foram resolvidos, como se pode visualizar, na seguinte página.

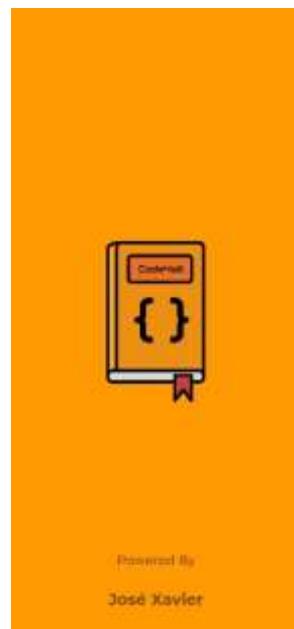


Figura 176: Layout da "SplashActivity" após da implementação do método "esconderInterface"

Passando à última das fases desta atividade, a colocação do GIF no layout, visto que se colocar um “ImageView” o GIF estará parado simplesmente, visto que o componente irá identificar o GIF, como uma imagem.

Já sabia que isto iria acontecer, visto que em projetos anteriores realizados por mim já tinha passado por este problema e neste caso não foi exceção.

Para ultrapassar este obstáculo no desenvolvimento, foi necessário realizar uma pesquisa no StackOverflow, visto que foi onde encontrei a solução que usei no outro projeto e normalmente quando pesquiso no StackOverflow a solução de um erro normalmente adiciono o mesmo link aos favoritos e neste caso não foi diferente, a solução encontra-se na figura abaixo.

<https://stackoverflow.com/questions/42898968/adding-gif-in-android-layout>

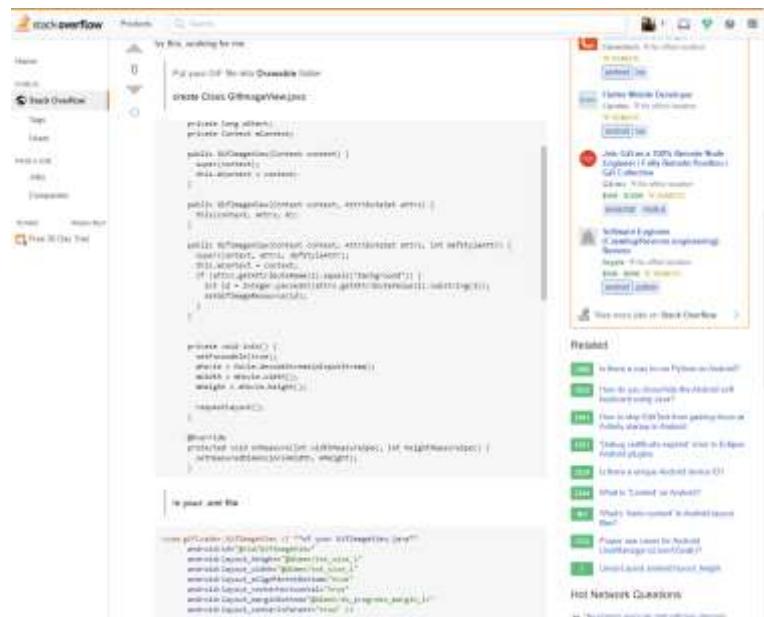


Figura 177: Solução de colocar um GIF num layout a funcionar corretamente

Com esta solução apresentada no StackOverflow consigo colocar o GIF a funcionar corretamente (supostamente), para isso, é preciso criar uma atividade de suporte onde serão efetuadas ações para que o mesmo aconteça e depois invocar no ficheiro xml do layout (activity_splash.xml), fazer a devida invocação do componente e após isso segundo o comentário, basta fazer a configuração do componente no ficheiro da atividade (SplashActivity.java) e está configurado.

Passo a explicar os procedimentos que realizei para que o GIF funcionasse corretamente:

A criação de uma nova classe de apoio em Java chamada de “GifImageView”, para que depois possa adicionar o componente dentro do layout e fazer a configuração do mesmo dentro da atividade principal.

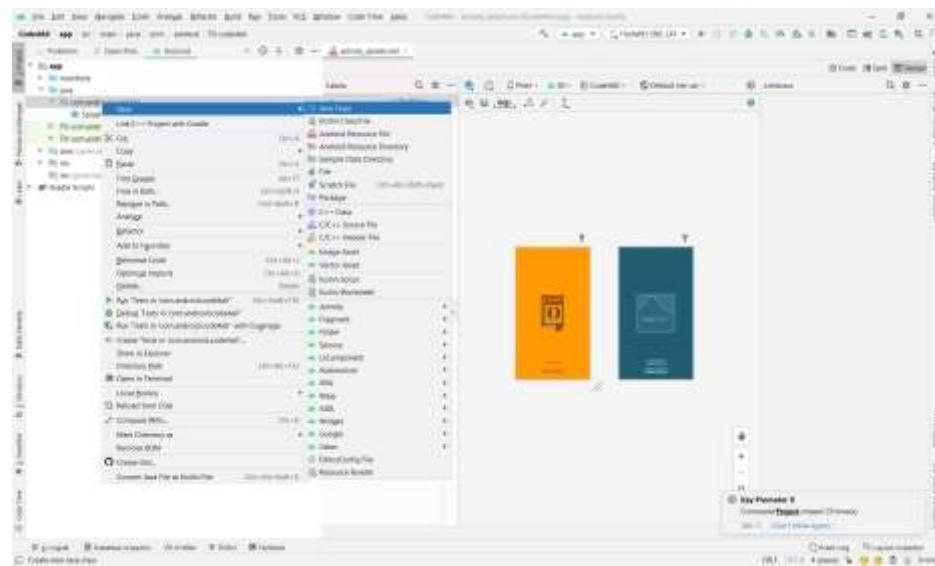


Figura 178: Criação de uma nova classe de Java

Passei depois à definição do nome da classe, neste caso “GifImageView”

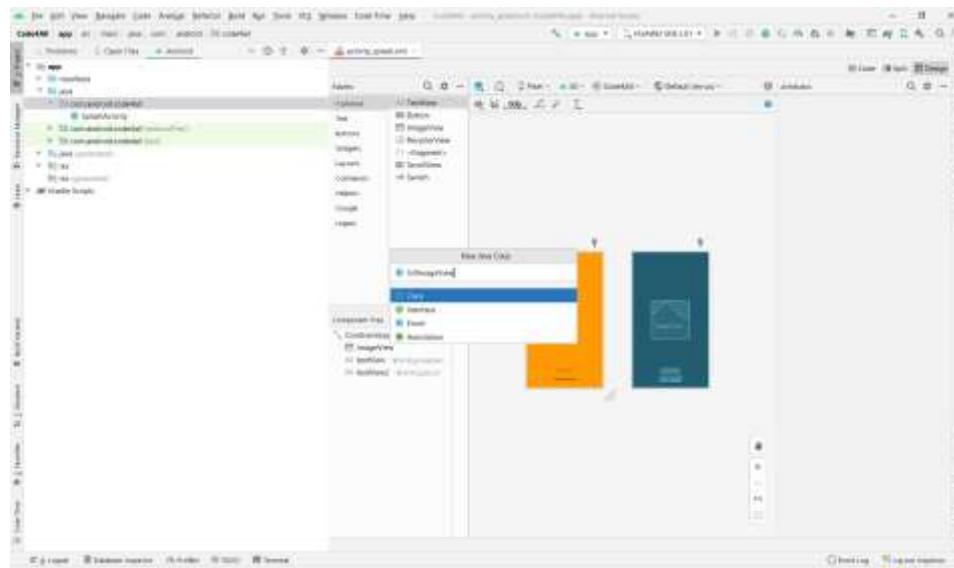


Figura 179: Definição do nome para a classe a vir a ser criada

Com na imagem que se representa em baixo, pode-se dizer que a classe foi criada com sucesso.



Figura 180: Abertura pela 1^a vez do ficheiro "GifImageView"

Após ter criado a classe, como visto anteriormente, bastou apenas copiar o código disponibilizado no StackOverflow e colar o mesmo na classe criada.

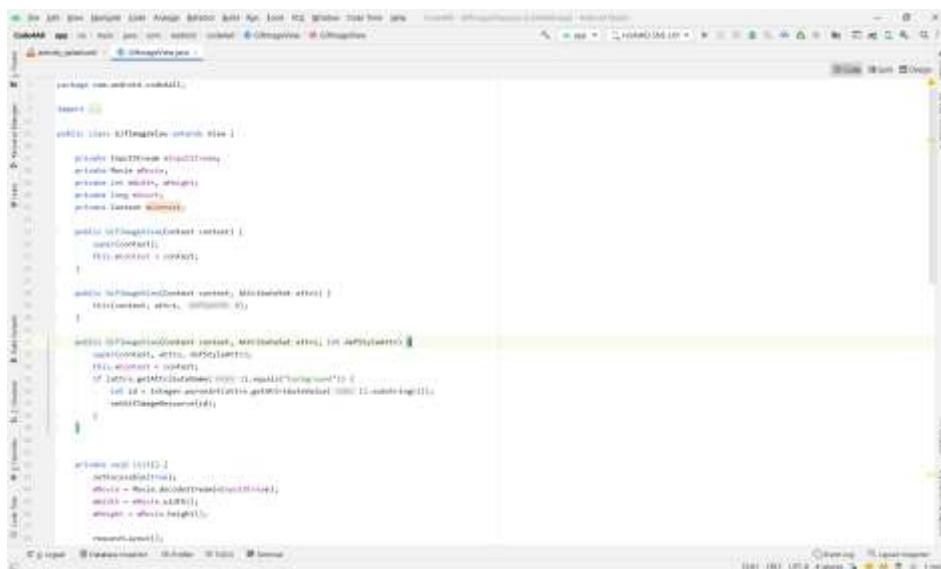


Figura 181: Colocação do código para colocar o GIF a funcionar corretamente

Após ter criado e configurado a classe de suporte para o GIF, bastou apenas realizar o passo que vinha a seguir, que neste caso era copiar o código em xml e colar no arquivo de layout desta atividade, mas logo foram apresentados alguns erros como se pode observar:

```
<com.gifLoader.GifImageView // **of your GifImageView.java**  
    android:id="@+id/GifImageView"  
    android:layout_height="@dimen/txt_size_1"  
    android:layout_width="@dimen/txt_size_1"  
    android:layout_alignParentBottom="true"  
    android:layout_centerHorizontal="true"  
    android:layout_marginBottom="@dimen/sb_progreses_margin_lr"  
    android:layout_centerInParent="true" />
```

Figura 182: Código XML retirado do StackOverFlow

Seguidamente, foi necessário fazer as devidas alterações para que os erros desaparecessem e tudo funcionasse como devido, começando com a alteração do caminho da classe que antes estava “com.gifLoader.GifImageView”, que passou para o meu pacote, como se vê na figura abaixo:

```
<com.android.code4all.GifImageView  
    android:id="@+id/gifImageView"  
    android:layout_width="100dp"  
    android:layout_height="100dp"  
    android:layout_alignParentBottom="true"  
    android:layout_centerInParent="true"  
    android:layout_centerHorizontal="true"  
    app:layout_constraintBottom_toTopOf="@+id/textView"  
    app:layout_constraintEnd_toEndOf="@+id/imageView"  
    app:layout_constraintStart_toStartOf="@+id/imageView"  
    app:layout_constraintTop_toBottomOf="@+id/imageView" />
```

Figura 183: Configuração do Componente GIF

Após realizar a configuração em xml, bastou passar à realização do alinhamento e tamanho do mesmo.

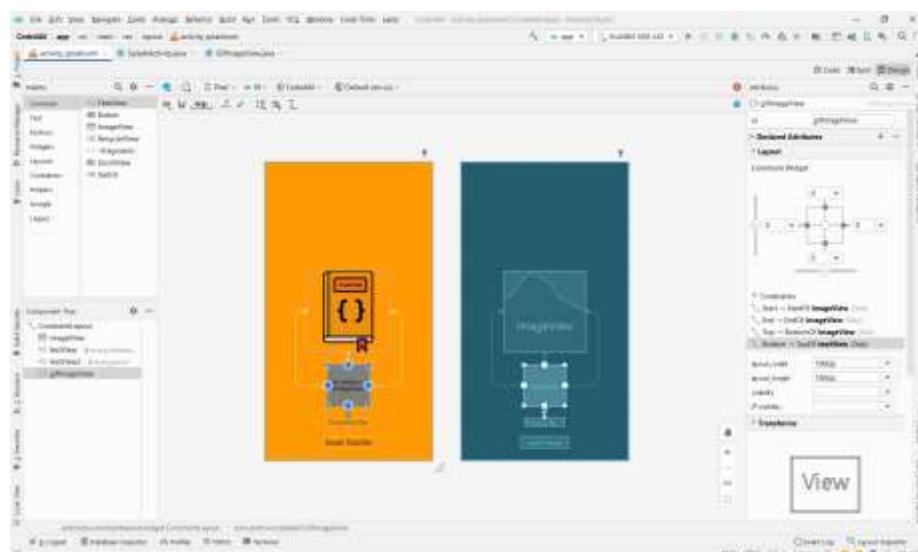


Figura 184: Alinhamento e tamanho do GIF

Depois de configurar o componente no layout, falta agora importar o ficheiro GIF, para dentro do projeto, para depois o configurar em Java e então fazer com que tudo funcione como devido.

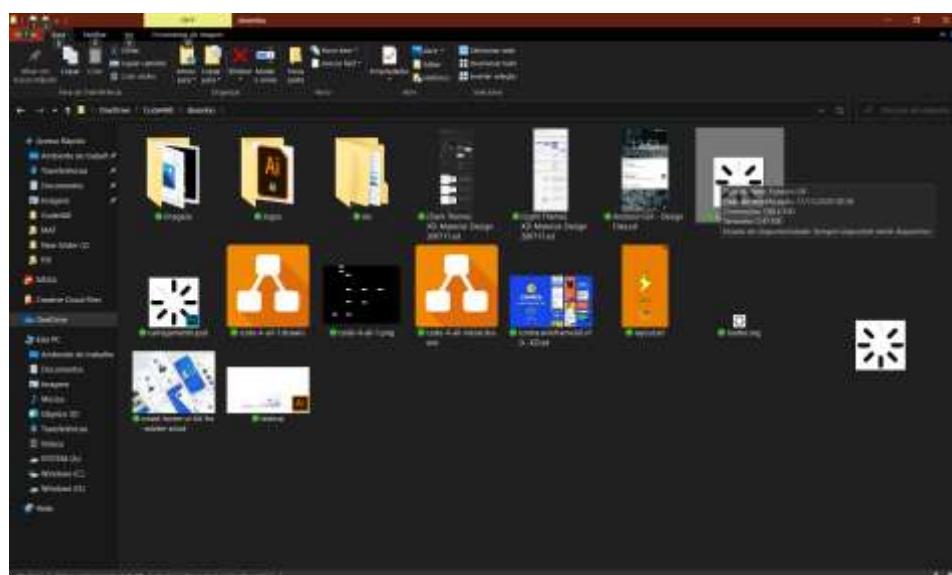


Figura 185: Ficheiro "carregamento.gif"

```
// Configurar GIF
GifImageView gifImageView = (GifImageView) findViewById(R.id.gifImageView);
gifImageView.setGifImageResource(R.drawable.carregamento);
```

Figura 186: Código necessário para definir um GIF como recurso do componente

Ficando este resultado, abaixo está representada uma captura de ecrã, que tirei quando estava nesta parte da codificação, aqui o carregamento está parado devido a ser uma foto apenas, mas o GIF estava a funcionar corretamente.

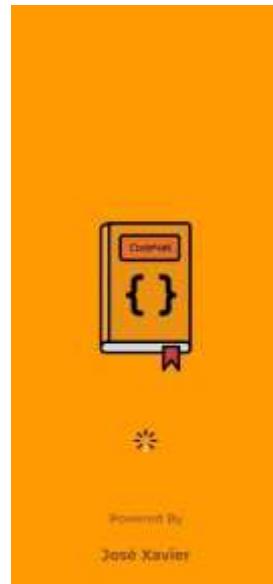


Figura 187: GIF já pronto e a funcionar corretamente

Após de colocado o GIF a funcionar, por questões de desempenho decidi colocar esta atividade apenas visível em modo vertical, ou seja, se o telefone for virado para a horizontal nada será feito, visto que é uma atividade de carregamento, a mesma deve apenas ser visualizada verticalmente.

Para fazer essa configuração basta adicionar “android:screenOrientation='portrait'” dentro da etiqueta “activity”.



Figura 188: Definição da atividade "SplashActivity" em modo “portrait”

Depois de ter realizado aquele pequeno ajuste nesta atividade, voltei para o ficheiro responsável pela mesma (SplashActivity.java) e passei à codificação do “carregamento”, para isso, realizei dois métodos.

Começando pelo “carregamento()”, onde foi criada uma nova Thread que basicamente é o mesmo que um processo, normalmente, já existe uma Thread a decorrer chamada de “UI Thread”, criei a mesma para que este processo seja realizado ao mesmo tempo mas separadamente para que a “UI Thread” não seja afetada neste no decorrer deste código, podendo ficar “encravado” ou algo do género.

Seguidamente a ter criado um processo novo, foi preciso criar um “handler”, que depois com o método “postDelayed()” e a realização de uma expressão do tipo lambda para simplificar o código, este método irá pedir dois parâmetros um do tipo runnable que está indicado como o método abaixo criado “introducao()” e de seguida o tempo desejado em miléssimas de segundo, onde coloquei o valor “3000”, ou seja, três segundos.

Após de ter a thread criada e configurada bastou apenas dar início na mesma com o código “thread.start();”.

Passando para o método “introducao()”, foi criada uma nova Intent, que serve para dar instruções ao dispositivo, neste caso redirecionar o mesmo para uma nova atividade, neste caso usa-se a classe, e para isso precisamos do contexto do pacote, sendo ele o da própria “SplashActivity” e da atividade para a qual se pretende redirecionar, neste caso a atividade “IntroActivity” que está a vermelho devido a ainda não ter sido criada.

Com a instrução criada, basta apenas utilizar o código “startActivity();”, e depois colocar dentro a instrução para que seja começada uma nova atividade.

Por fim, utilizei o código “finish()”, para finalizar a atividade “SplashActivity” e assim poupar recursos, decidi finalizar a mesma devido ao simples facto de que nunca mais vai ser utilizada até a aplicação ser fechada e aberta novamente, logo não tem sentido nenhum estar aberta em segundo plano.

O respetivo código pode ser visualizado nas imagens abaixo.

```
void carregamento() {  
    Handler handler = new Handler();  
  
    Thread thread = new Thread() {  
        @Override  
        public void run() {  
            handler.postDelayed(() -> introducao(), delayMillis: 3000);  
        }  
    };  
  
    thread.start();  
}  
  
void introducao() {  
    Intent intent = new Intent(packageName: SplashActivity.this, IntroActivity.class);  
    startActivity(intent);  
    finish();  
}
```

Figura 189: Método "carregamento()"

```
// "Atividade de Carregamento"  
carregamento();
```

Figura 190: Invocação do método "carregamento()" dentro do método "onCreate"

Para fazer a verificação se o utilizador se encontra “logado”, tive que ir pesquisar como se fazia o mesmo.

Eu sabia como se fazia, mais ou menos, visto que abordei este assunto no curso que realizei, mas mesmo assim gostava de ter 100% de a certeza, para não errar.

<https://stackoverflow.com/questions/44583834/firebase-how-to-check-if-user-is-logged-in>

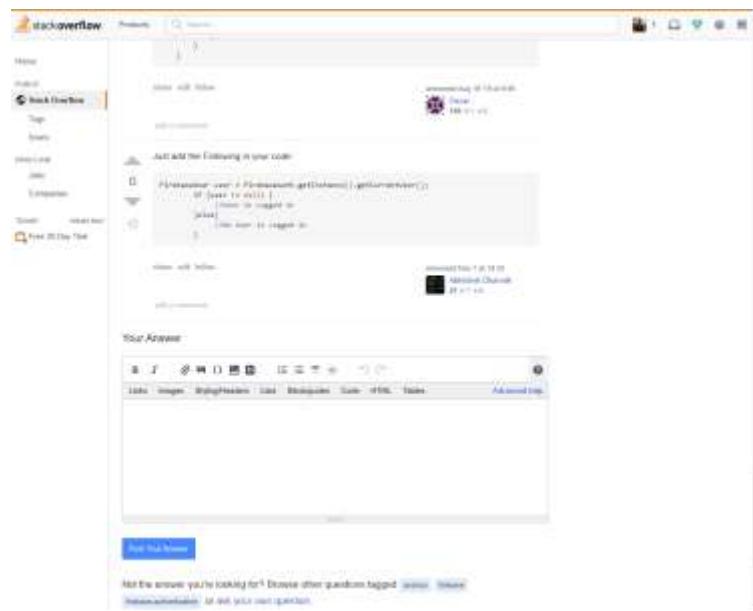


Figura 191: Verificação ao utilizador para ver se está "logado" ou não

Com a solução que encontrei na pesquisa anterior, decidi modificar o método “introducao()” visto que era esse que fazia a decisão de para onde o utilizador vai depois dos três segundos.

Comecei por fazer uma condição “se” que iria verificar se o utilizador fosse diferente de nulo e então levaria o utilizador em questão para a PrincipalActivity e se não houvesse nenhum utilizador “logado”, o mesmo seria reencaminhado para a atividade IntroActivity

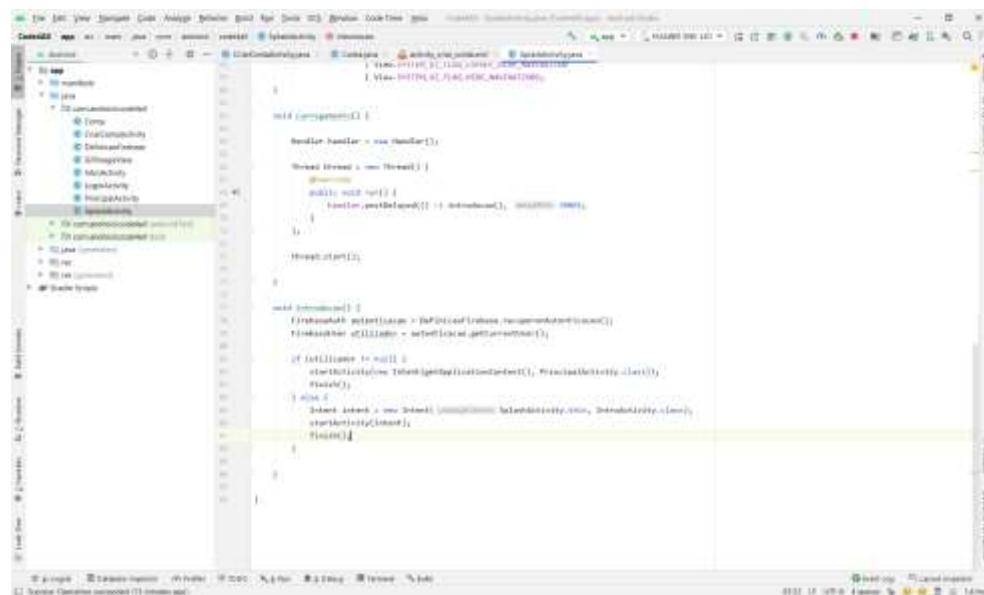


Figura 192: método introducao(), com a verificação de autenticação do utilizador

Seguidamente, criei um método “buscarUtilizador()”, dentro do ficheiro de configurações gerais, é neste ficheiro que coloquei o código que poderia ser reutilizado, evitando assim código duplicado. A função deste método era basicamente ir buscar o utilizador, dentro da base de dados e retornar uma “Intent” (instrução de atividade), para depois ser aberta a atividade consoante o seu grupo.



Figura 193: Criação do método "buscarUtilizador()"

Depois, foi instanciar esse método, dentro da função introdução().

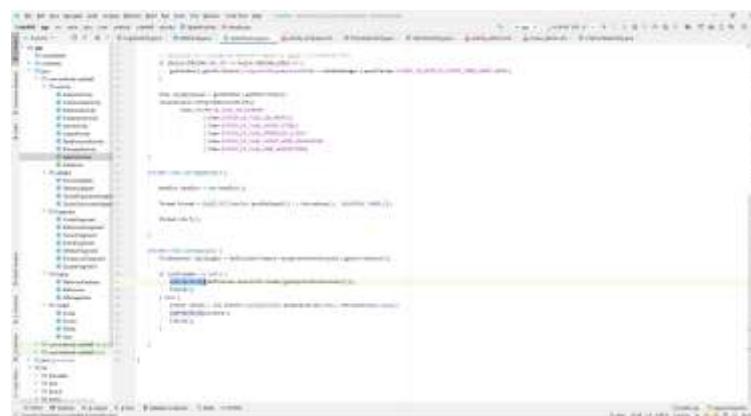


Figura 194: Método "buscarUtilizador()"

Para o método realmente funcionar sem problemas, passei à criação de uma “*Interface*”.

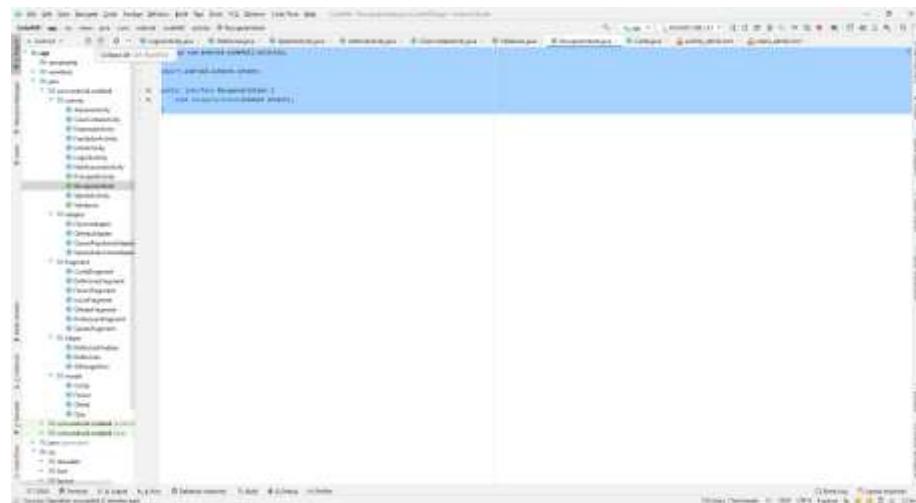


Figura 195: Criação da Interface "RecuperarIntent"

Adicionei mais um argumento no método “buscarUtilizador()” sendo ele a “Interface” anteriormente criada e atribui também a esse argumento a “Intent” que retornaria do grupo do utilizador.



Figura 196: Atribuição da "Intent" para com a Interface "RecuperarIntent"

Voltando para a atividade em si, tive de modificar o método, visto que um novo argumento foi adicionado, ficando algo do género.

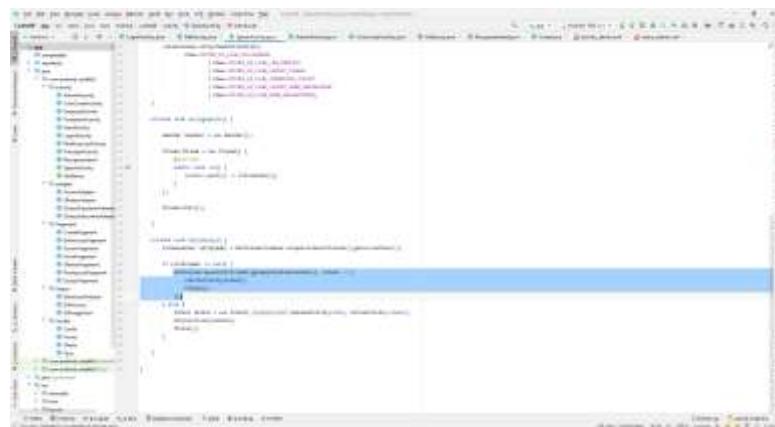


Figura 197: Configuração do novo argumento dentro do método "buscarUtilizador()"

Com isto, tive a necessidade de modificar o método “carregamento()”, visto que já não iria precisar do “falso” carregamento, pois esta operação de ir buscar o grupo do utilizador já daria um tempo interessante de carregamento da atividade.

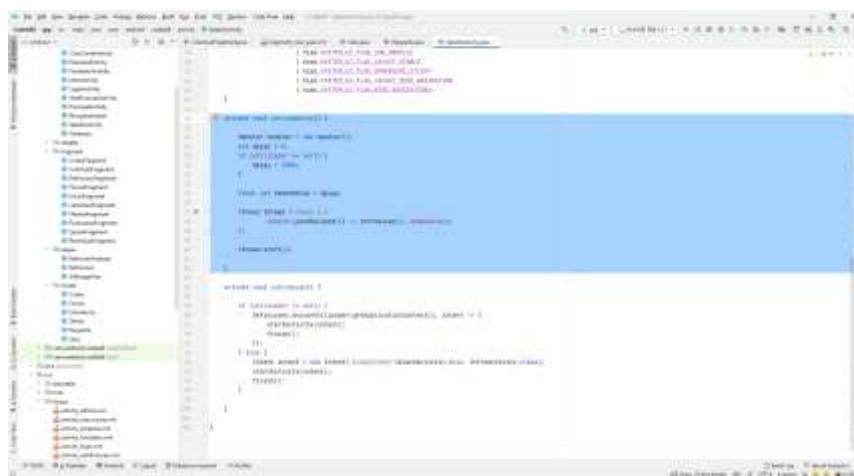


Figura 198: Alteração do método "carregamento()"

Seguidamente, encontrei uma libraria que faria a inserção do GIF de carregamento, sendo ela muito mais prática do que a criação de uma classe em Java.

<https://github.com/koral--/android-gif-drawable>

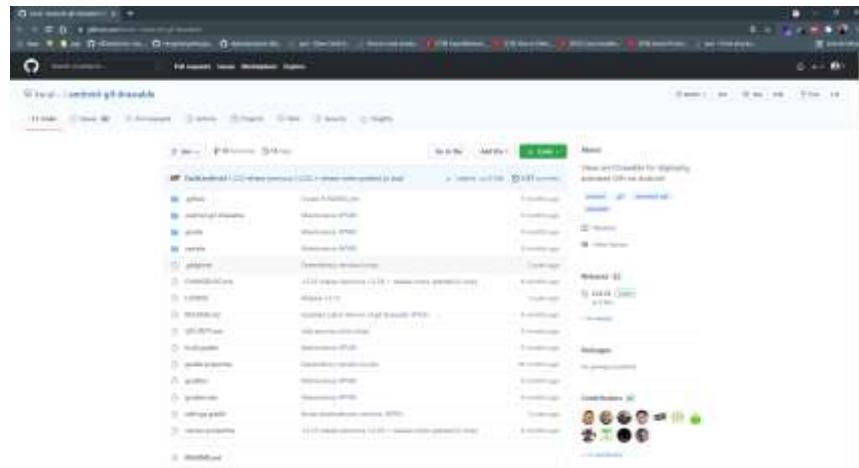


Figura 199: Android Gif Drawable (GitHub)

De imediato, adicionei a “*implementation*” desta libraria.

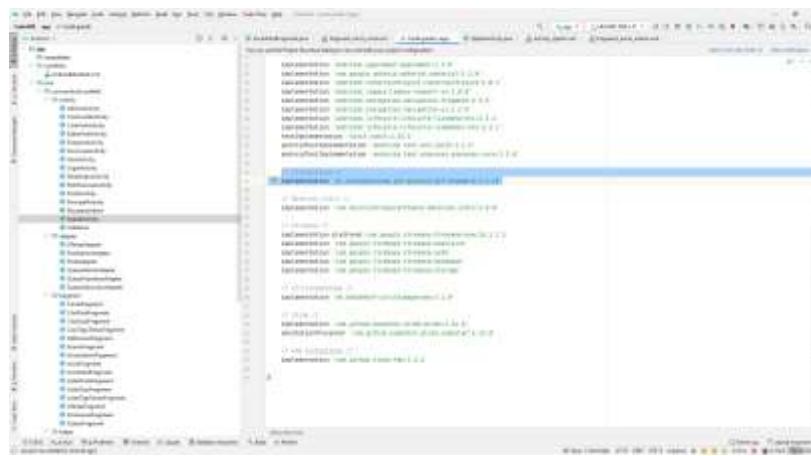


Figura 200: Implementação da libraria "GifImageView"

Com isto, também tive que remover a antiga classe java e ir ao ficheiro de layout, para trocar o elemento.



Figura 201: Alteração do elemento para “GifImageView”

Para dar uma extra funcionalidade à app, decidi também implementar um método que verificasse a conexão à internet, visto que é estritamente necessária na utilização da minha app.

<https://stackoverflow.com/questions/4238921/detect-whether-there-is-an-internet-connection-available-on-android>



Figura 202: Solução de como verificar a conexão à Internet

Copiei o método e coloquei para dentro desta atividade, fazendo também uma verificação dentro do método “carregamento()”. Se houvesse internet a app, continuava a sua execução, se não, pára por ali a sua execução.



Figura 203: Implementação do método de verificar internet

Passado esta configuração da internet e testadas as suas funções, tive ainda que realizar umas alterações dentro do método “buscarUtilizador()”.

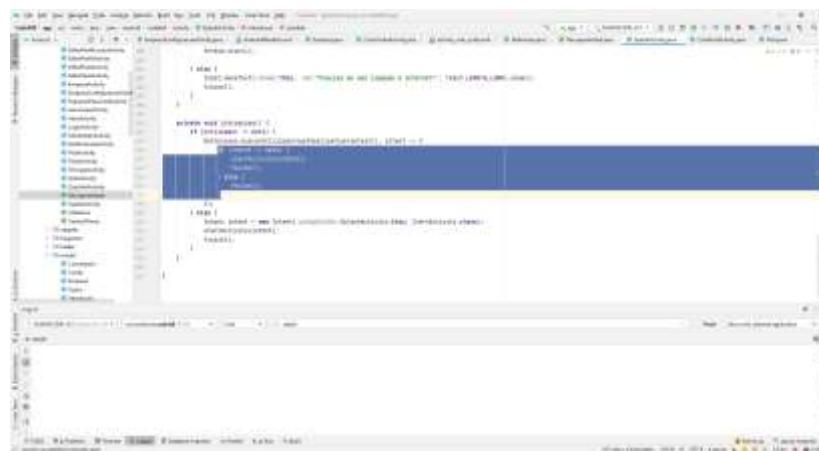


Figura 204: Se a Intent fosse "null" finalizava a atividade em si, evitando um crash da app.

Se nenhum grupo fosse returnado, a intent era definida com null.

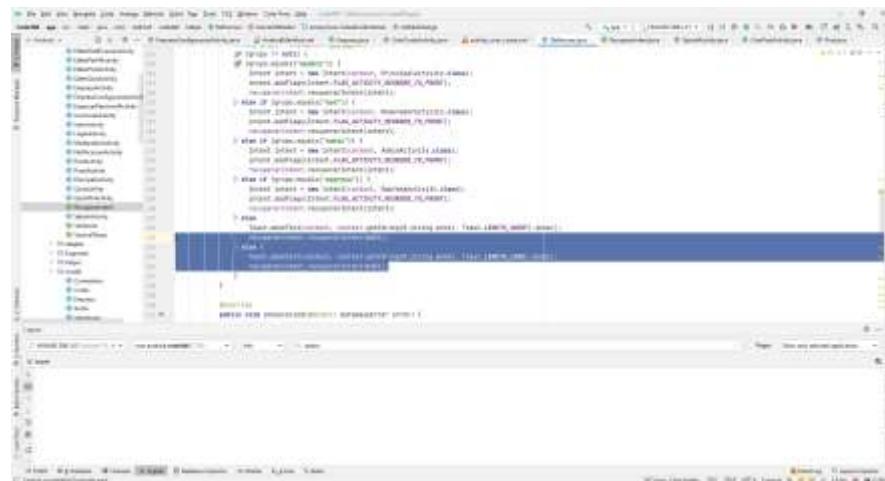


Figura 205: Definição da Intent como null, se nenhum grupo fosse encontrado

Para finalizar esta atividade, coloquei todas atividades criadas até aquele momento, com a “screenOrientation” como “portrait”, fazendo a app apenas executar na vertical.

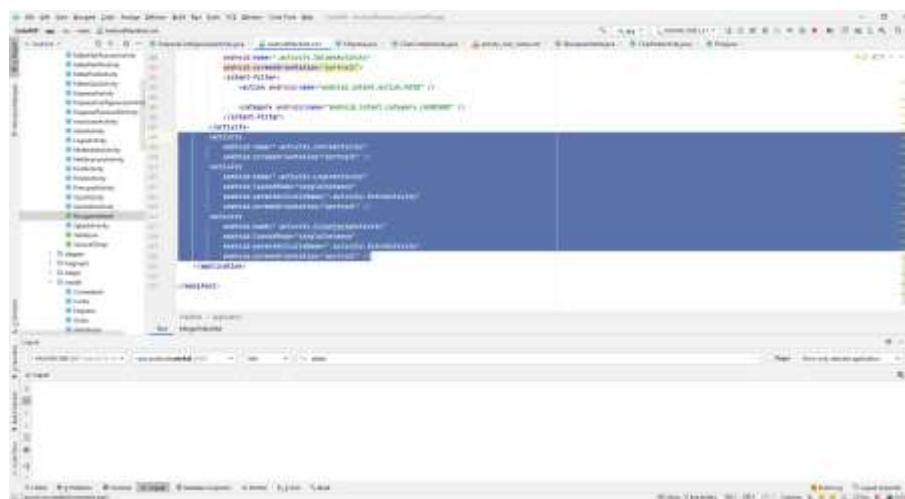


Figura 206: Definição da orientação do ecrã como vertical

IntroActivity

Nesta atividade, para facilitar o desenvolvimento da mesma, utilizei uma libraria open-source “material-intro”, que se encontra no GitHub, uma plataforma bastante conhecida por a maior parte dos programadores.

Utilizei a mesma devido a conseguir configurar vários layouts numa atividade apenas poupando assim código e melhorando o desempenho da aplicação.

E não só, mas também, devido à mesma ter uma API bastante interessante e de certa forma conseguir editar a mesma de uma forma mais simples, poupando tempo e código de uma só vez.

O link e as informações desta libraria encontram-se abaixo:

<https://github.com/heinrichreimer/material-intro/>

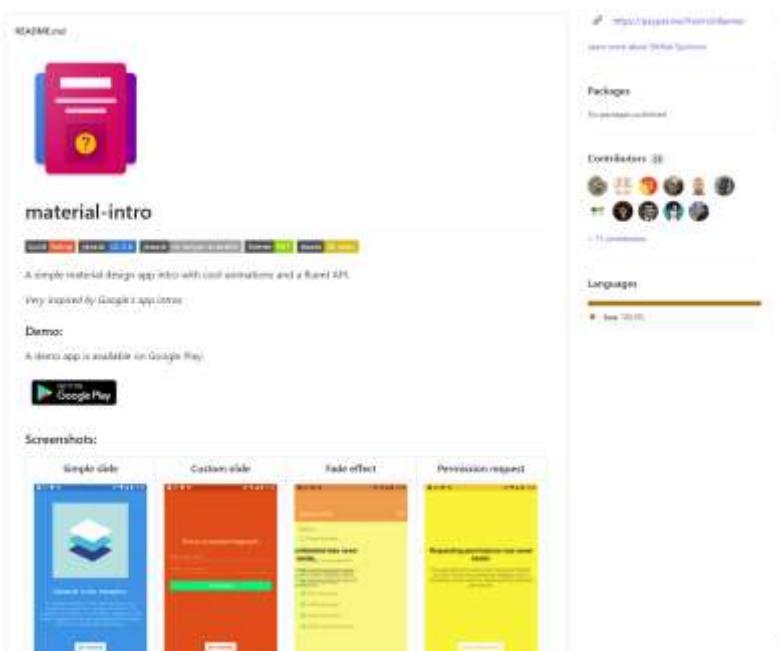


Figura 207: Libraria "material-intro"

Abaixo da introdução da libraria, encontram-se os requisitos para colocar esta libraria no meu projeto a funcionar corretamente, pois para a usar, tenho de importar a mesma para dentro do meu projeto.



Figura 208: Requisitos da libraria "material-intro"

Após ter visto os requisitos, percebi que tinha de abrir o ficheiro "build.gradle" do módulo e então adicionar a libraria com o código "implementation 'com.heinrichreimersoftware:material-intro:x.y.z'", troquei o "x.y.z" para a versão mais estável naquele momento, sendo ela a "2.0.0", adicionei também um comentário relativamente à importação da libraria para futuramente lembrar-me do que faz cada uma das importações que realizei ao longo do projeto, como se pode visualizar na captura de ecrã.



Figura 209: Configuração da libraria "material-intro" no ficheiro "build.gradle" relacionado ao módulo

Após ter realizado a configuração no “build.grade” do módulo, falta agora fazer a configuração dentro do mesmo ficheiro, só que ao nível do projeto, com o seguinte código “maven { url 'https://jitpack.io' }”. Como pode ser visualizado na imagem abaixo:

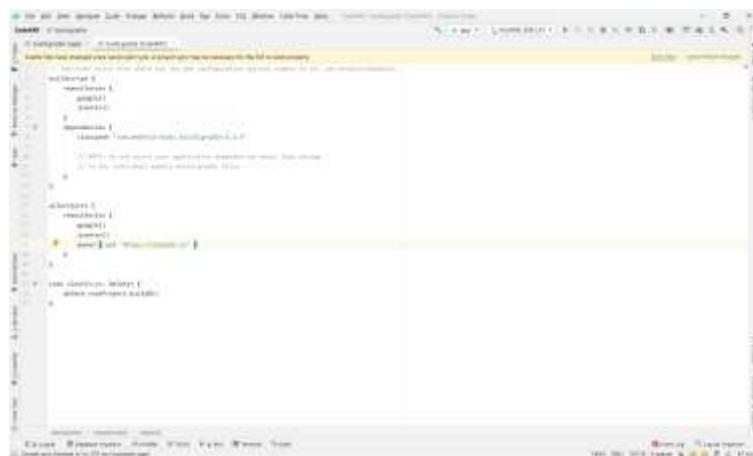


Figura 210: Configuração da libraria "material-intro" no ficheiro "build.gradle" do projeto

O código que se encontra abaixo, é aquele que está disponibilizado na página da libraria do GitHub, copiei o código e coleei dentro da atividade em questão, após realizar isso foram apresentados alguns erros, visto que não tinha alguns recursos que eram necessários, ou seja, seriam necessárias algumas mudanças para que tudo funcionasse, visto que primeiramente o meu objetivo era fazer um teste para ver se a libraria esta corretamente instalada.

```
addSlide(new SimpleSlide.Builder()
    .title(R.string.title_1)
    .description(R.string.description_1)
    .image(R.drawable.image_1)
    .background(R.color.background_1)
    .backgroundDark(R.color.background_dark_1)
    .scrollable(false)
    .permission(Manifest.permission.CAMERA)
    .build());
```

Figura 211: Código disponibilizado na página do "material-intro"

Após ter colado o código acima descrito, decidi fazer umas alterações no mesmo para que ele corresse sem problemas e assim ter mais ou menos uma visão do que iria acontecer.

```
addSlide(new SimpleSlide.Builder()
    .title("Título")
    .description("Descrição")
    .image(R.drawable.Logo)
    .background(R.color.corDiferente)
    .scrollable(false)
    .build());
```

Figura 212: Código anteriormente referido após de algumas modificações

Após fazer as modificações no código, decidi executar o código e instalar a aplicação no meu telemóvel, mas logo obtive um erro. Primeiramente, fui à página oficial da libraria no GitHub para entender onde tinha errado, logo aí deparei-me que tinha me esquecido de modificar o atributo “android:theme='@style/Theme.Code4All” para “android:theme='@style/Theme.Intro”, visto que era um dos requisitos pedidos para que a libraria funcionasse corretamente.

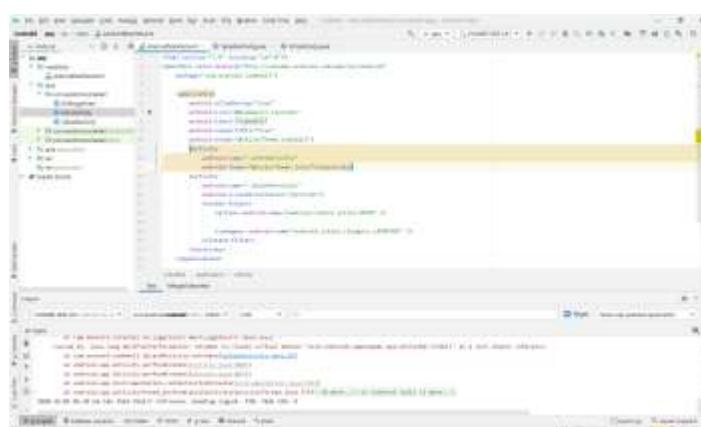


Figura 213: Correção da propriedade "android:theme" dentro da atividade

Ao fazer esta pequena alteração, decidi executar a aplicação, ao que desta vez não obtive nenhum erro e apresentou o seguinte comportamento:

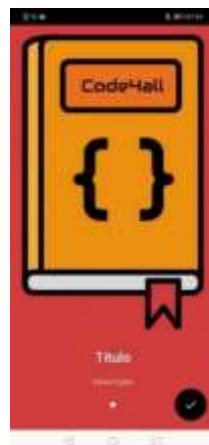


Figura 214: IntroActivity teste de execução

Com a fase anterior realizada passei à criação do layout da primeira interface.

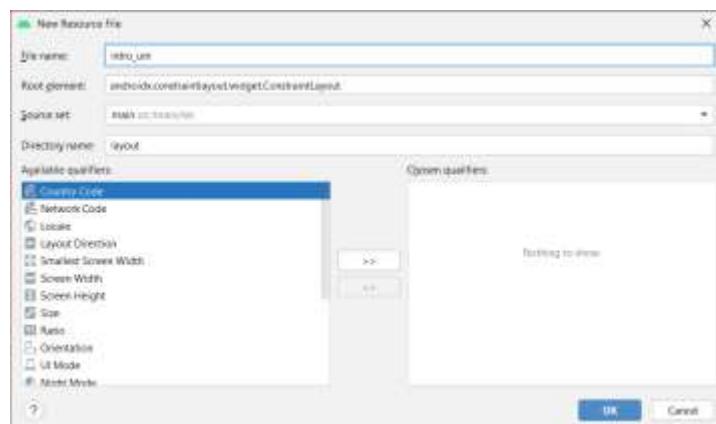


Figura 215: Criação do primeiro layout de introdução ao app

Seguidamente, passei à configuração e criação do primeiro slide, com o código que se encontra em baixo, passando agora a ter um ficheiro de layout próprio para cada slide, como era o meu objetivo inicial.

```

@Override
protected void onCreate(Bundle savedInstanceState) {

    setFullscreen(true);

    super.onCreate(savedInstanceState);

    addSlide(new SimpleSlide.Builder()
        .layout(R.layout.intro_um)
        .build());

    setButtonNextVisible(true);

}

```

Figura 216: Configuração do primeiro slide de introdução à app

Comecei por editar o ficheiro de layout “intro_um.xml”, para realizar um teste e verificar se estava tudo a funcionar como devido, adicionei um TextView ao layout e coloquei o texto “INTRO 1”, para quando executar ter de aparecer este texto.

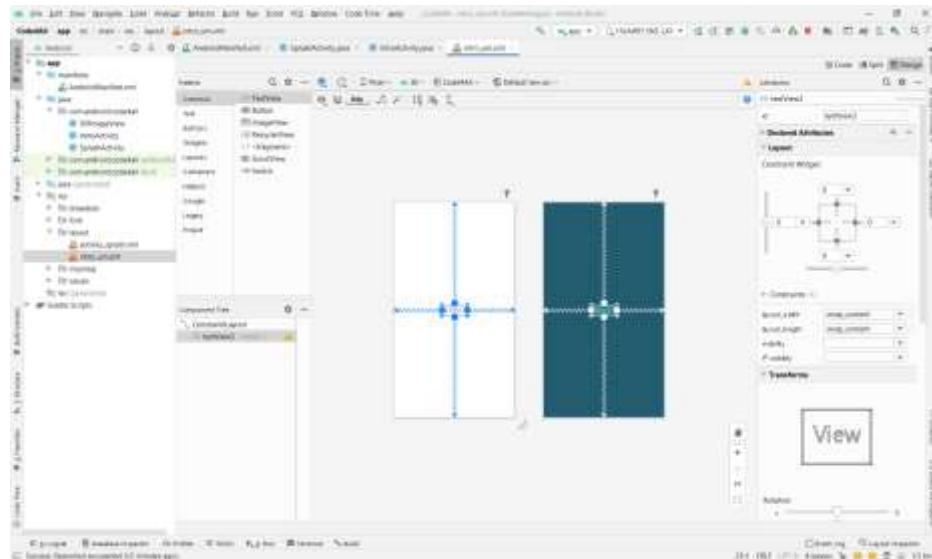


Figura 217: Colocação de um TextView no layout "intro_um.xml"

Passei então a configurar o ficheiro “colors.xml”, colocando três cores básicas, como se pode observar na figura da página seguinte.

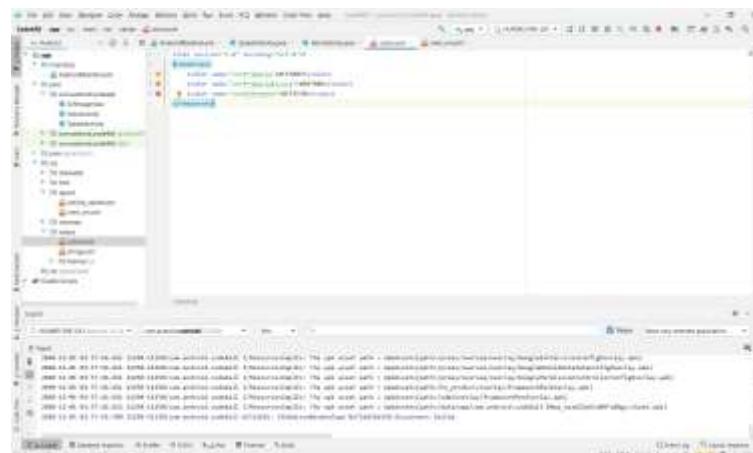


Figura 218: Configuração das três cores básicas no "colors.xml"

Depois de ter colocado uma “marcação” no layout e de configurar as cores, decidi executar a aplicação, mas logo surgiu um erro a dizer que faltava o atributo “background”, voltei à atividade e fui configurar o mesmo, com a cor primária, definida no ficheiro “colors.xml”.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    setFullscreen(true);

    super.onCreate(savedInstanceState);

    addSlide(new SimpleSlide.Builder()
        .layout(R.layout.intro_wm)
        .background(R.color.corPrimaria)
        .build());

    setButtonNextVisible(true);
}
```

Figura 219: Configuração final do primeiro slide na atividade

Ao instalar e executar a aplicação, obtive este resultado com o código, a partir daqui concluí que o layout estava a ser carregado corretamente e que havia mais uns problemas com o notch, visto que o mesmo estava “invisível”, muito provavelmente havia de ser da propriedade “setFullscreen(true);”, mas para agora já estar a executar, já era um grande avanço.



Figura 220: Demonstração do primeiro layout

Após ter o layout minimamente configurado e a funcionar, era hora de abrir o meu layout, que realizei no Adobe XD, e configurar de forma igual/semelhante no layout da aplicação, para isso comecei por ir buscar a foto que usei no primeiro layout introdutório da app.

Após adicionar a imagem que pretendia para dentro do projeto, arrastei o componente “ImageView” para dentro do layout, como se pode verificar na próxima imagem (página seguinte).

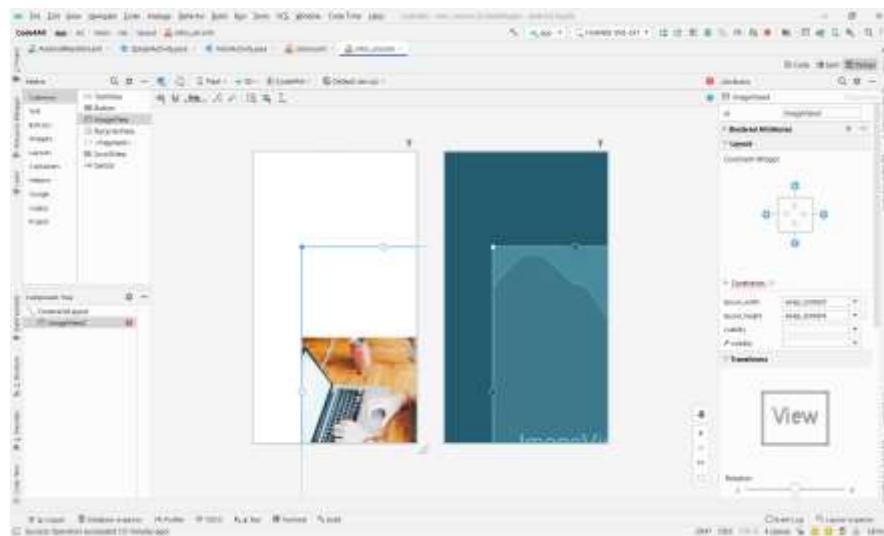


Figura 221: Imagem descritiva (intro_um.xml)

Após inserir a imagem no layout, coloquei a mesma com 200dp, tanto na largura como na altura,

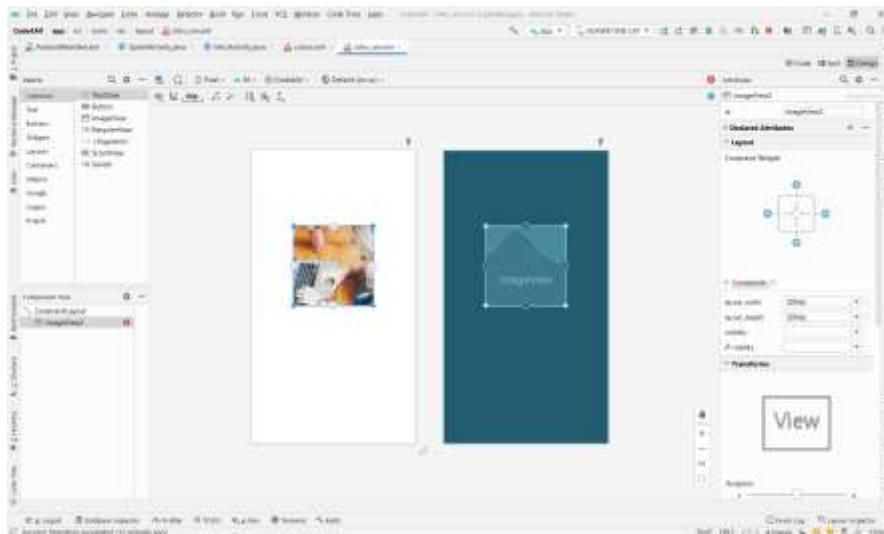


Figura 222: Definição do tamanho e largura da imagem

Seguidamente, decidi colocar a imagem centrada ao centro, por questões de aspetto.

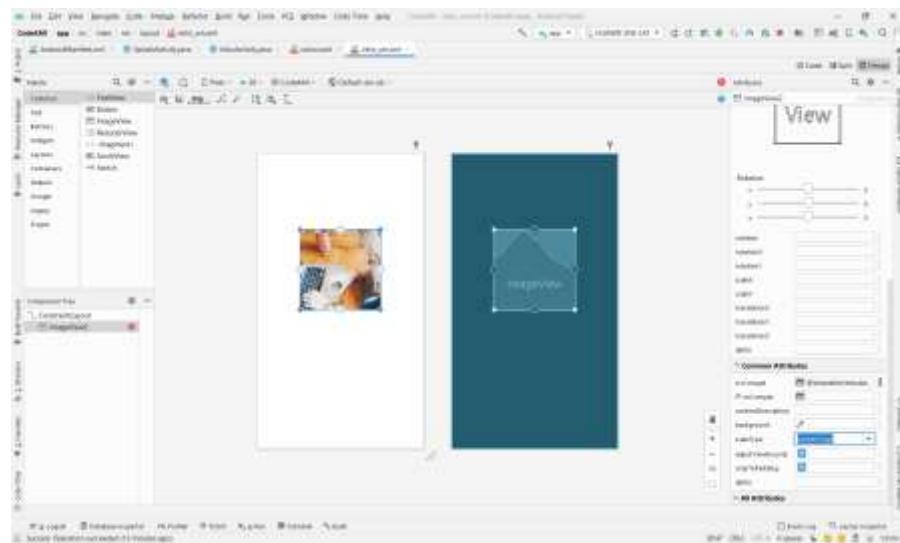


Figura 223: Centralização da imagem

Com a imagem devidamente configurada, passei à colocação de um TextView para assim colocar um texto agregado ao mesmo, como fiz na fase de desenho ao planejar este layout. Fiz um recurso string para o mesmo, para ficar mais dinâmico e ter a possibilidade de alterar tudo o que quiser num só lugar (strings.xml). Mudei a fonte, mudei o tamanho de letra e coloquei alinhado ao meio.

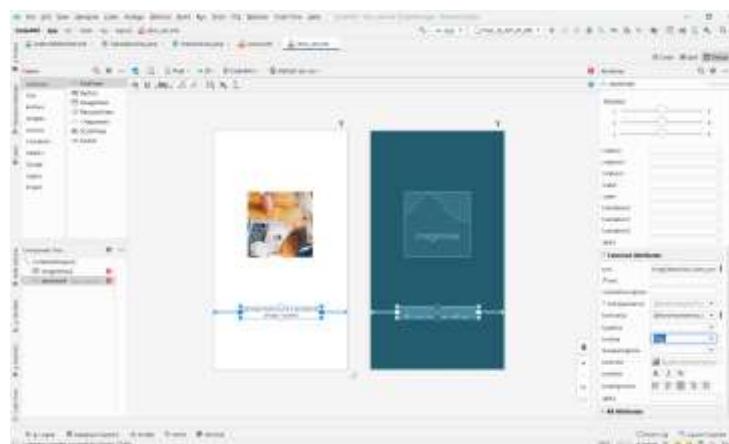


Figura 224: Configuração do TextView, fonte, tamanho, texto...

Visto que também queria colocar as bordas arredondadas na imagem, fui procurar pela devida solução visto que não sabia como fazer. O primeiro sítio onde fui pesquisar, foi ao stackoverflow onde estão (quase) sempre as respostas para as minhas dúvidas. E logo encontrei uma suposta solução.

<https://stackoverflow.com/questions/2459916/how-to-make-an-imageview-with-rounded-corners>

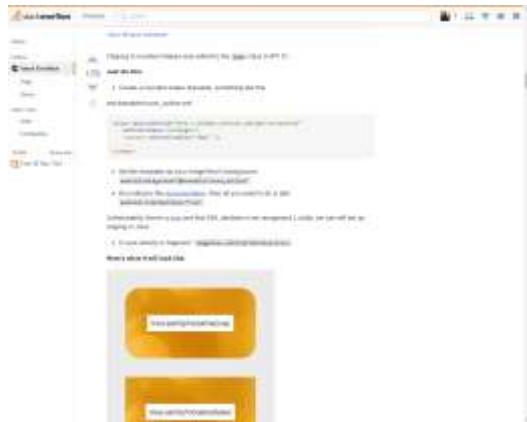


Figura 225: Solução de colocar a imagem com bordas redondas

Com uma suposta solução arranjada, comecei logo por aplicar o que era dito por um utilizador da plataforma, que consistia em criar as definições para um efeito num ficheiro XML, dentro da pasta “drawable”.

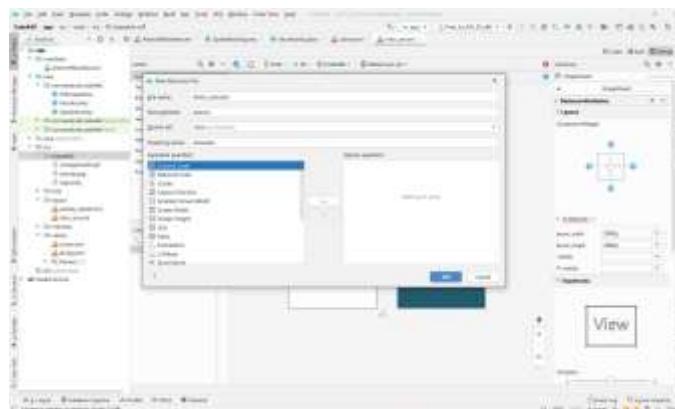


Figura 226: Criação do ficheiro xml “efeito_redondo”

Com o ficheiro criado tive de trocar a etiqueta “selector” para “shape”, visto que iria fazer configurações à nível da estrutura, neste caso da imagem. Tive ainda que colocar o “shape” como retângulo e para arredondar os cantos coloquei a etiqueta “corners”, em português “cantos”, com um raio de 16dp para assim dar um efeito redondo.



Figura 227: Modificação da etiqueta "selector" para "shape"

Com a configuração do efeito redondo finalizada, bastou apenas colocar o atributo “android:background”, seguidamente do caminho do efeito, neste caso “@drawable/efeito_redondo”. Um a parte, tanto o ImageView, como o TextView, estão a vermelho visto que ainda não estão alinhados verticalmente e horizontalmente.

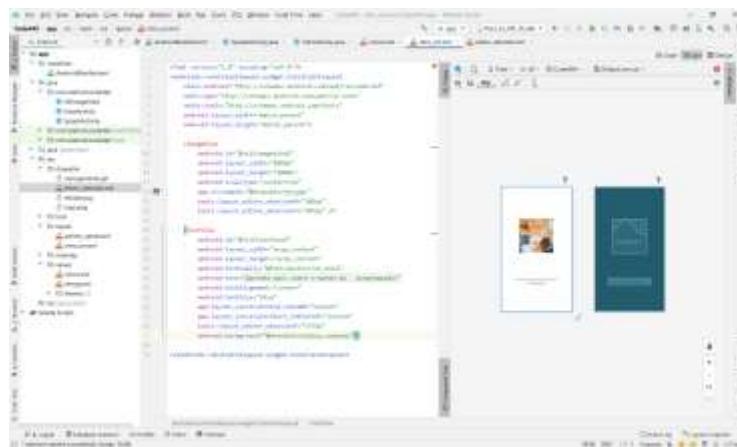


Figura 228: Definição do background com o efeito redondo

Após definir o atributo “background” com o efeito redondo, realizado anteriormente, passei ao alinhamento da imagem ao centro, visto que era como tinha planeado fazer

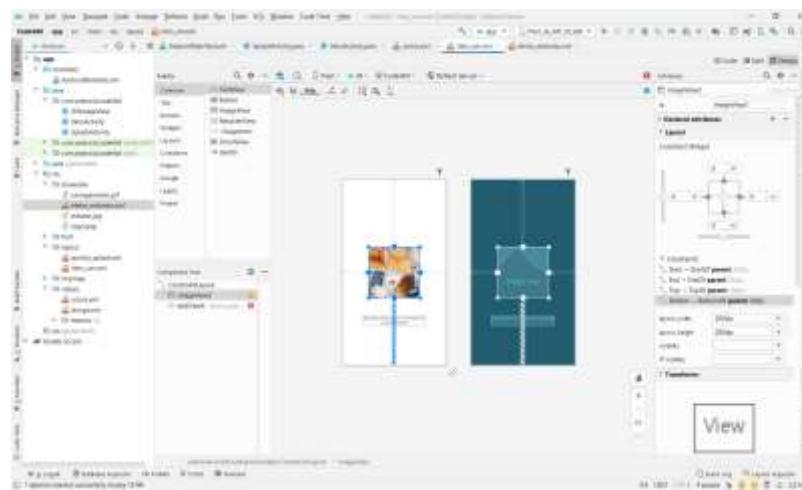


Figura 229: Alinhamento da imagem ao centro

Agora faltava realizar o alinhamento para o texto, ficando o seguinte resultado.

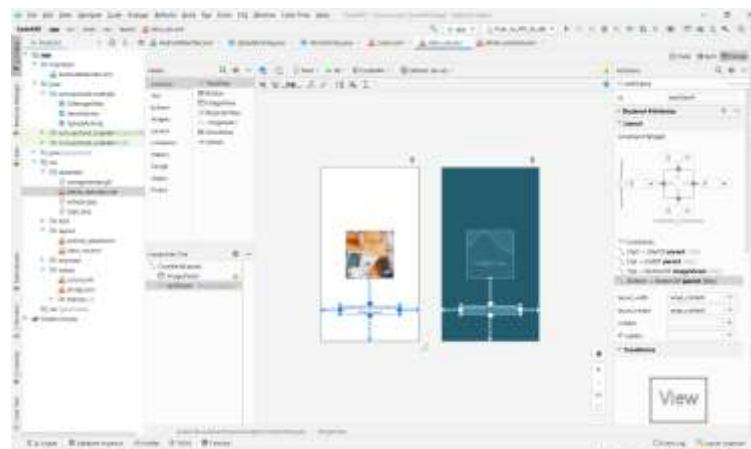


Figura 230: Alinhamento do texto ao centro

Com estas configurações finalizadas propriamente passei à execução mais uma vez da aplicação, o que me deu o resultado abaixo apresentado. Uma observação, é que o notch está branco, ou seja, algo não está certo, o que poderá ser um possível bug. Com esta execução também concluí, que o código que realizei anteriormente não estaria a funcionar, visto que a imagem continua quadrada, não sendo esse o meu objetivo

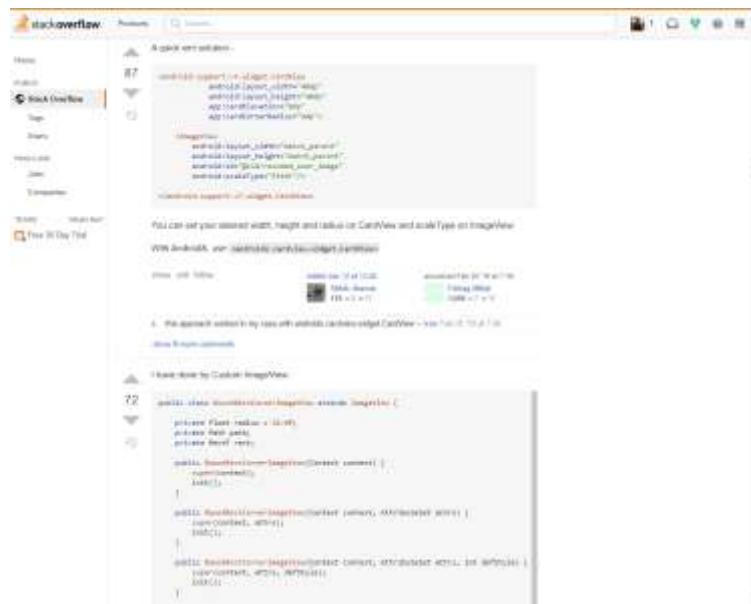


Figura 231: Resultado do layout com a imagem e o texto devidamente configurados

Visto que ainda não tinha chegado ao meu objetivo de colocar a imagem redonda e como sou bastante persistente, fui de imediato realizar uma nova pesquisa, com mais calma, sobre as minhas alternativas de colocar a imagem com os cantos redondos, uma delas foi a apresentada abaixo, em que consistia eu colocar a imagem dentro de um CardView e aplicar as propriedades redondas ao mesmo, sendo essas propriedades aplicadas também na imagem, conseguindo assim atingir o meu objetivo.

Por curiosidade, o site onde fui buscar esta solução é o mesmo, só que é outra das soluções propostas pelos vários utilizadores que responderam a esta questão.

<https://stackoverflow.com/questions/2459916/how-to-make-an-imageview-with-rounded-corners>



Para começar a realizar as modificações apresentadas pela solução anteriormente descrita, precisava antes de remover tudo o que fiz, para não haver arquivos sem qualquer função dentro do projeto, comecei por remover o atributo background do arquivo “efeito_redondo.xml”.

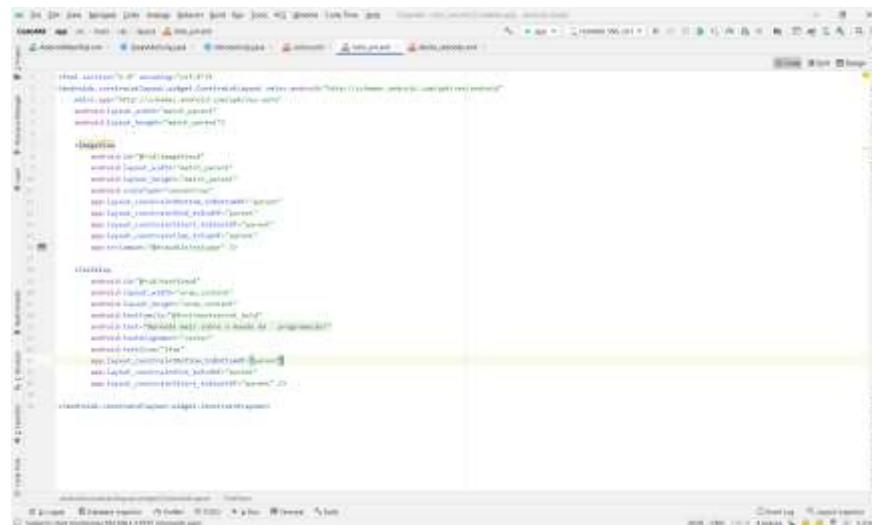


Figura 232: Remoção do atributo "background" da ImageView

Após ter realizado a remoção do atributo anteriormente dito, passei à colocação de um CardView que tem por dentro a ImageView, fiz também as devidas configurações de alinhamento e tamanho, e coloquei o atributo “cardCornerRadius” com 24dp, este atributo vai dar o tão esperado efeito redondo à minha imagem, como se pode visualizar na figura.

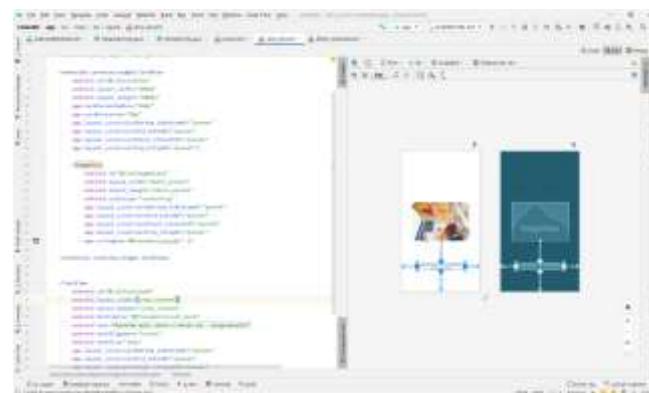


Figura 233: Colocação e configuração do CardView

Seguidamente, mudei a cor de fundo do layout, sendo ela inicialmente a cor primária, definida no ficheiro “colors.xml”, decidi ir buscar a cor ao layout realizado no Adobe XD (#D97B06) e criar um recurso novo de cor chamada de “corIntroUm”, sendo este de fácil interpretação.

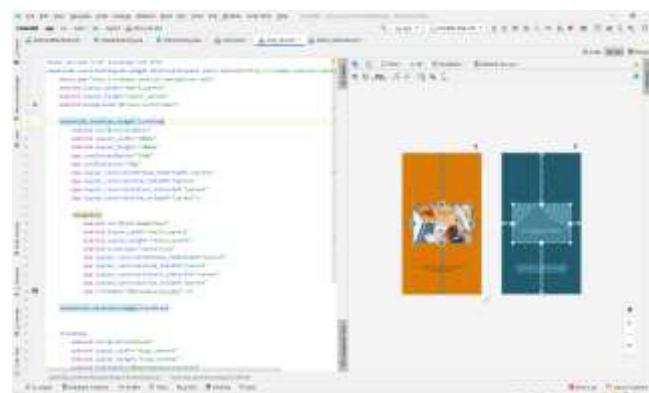


Figura 234: Modificação da cor de fundo do layout "intro_um"

Após ter feito esta pequena modificação no fundo do layout, passei a resolver o aviso que o layout me estava a dar, sendo ele, a tal descrição da imagem, para os cegos terem mais ou menos uma noção do que poderá ser apresentado, neste caso, utilizei como descrição da imagem, a mesma frase apresentada em baixo.

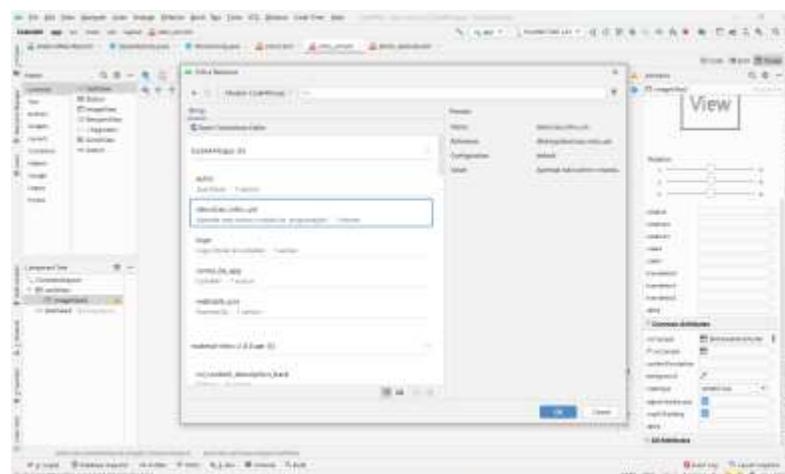


Figura 235: Definição de uma descrição para a ImageView

Aqui dou como finalizado o layout da primeira página de introdução. Como se pode observar na figura abaixo.

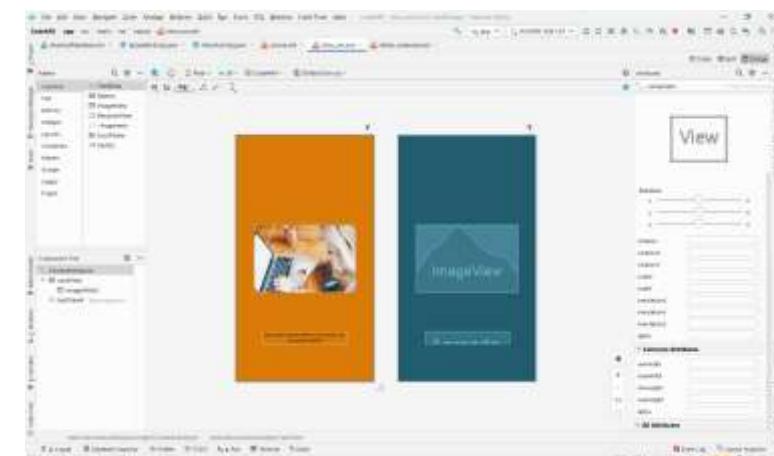


Figura 236: Primeiro layout de introdução finalizado com sucesso

Com o primeiro layout acabado, continuei o desenvolvimento e passei para o segundo.

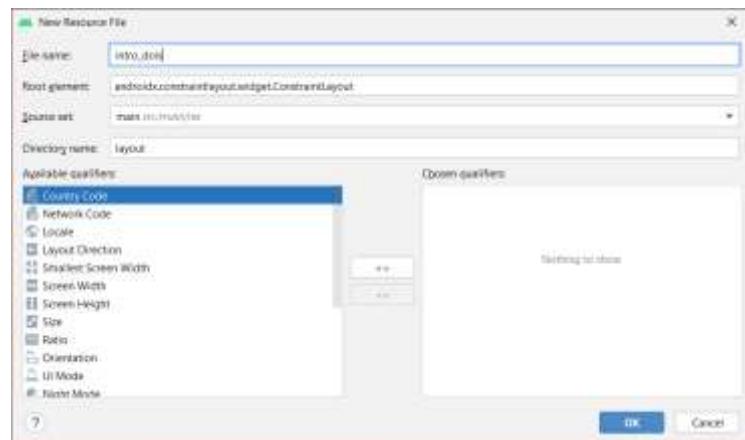


Figura 237: Criação do segundo layout de introdução

Em baixo, encontra-se o segundo layout introdutório, como é óbvio ainda se encontra sem nada visto que acabou de ser criado.

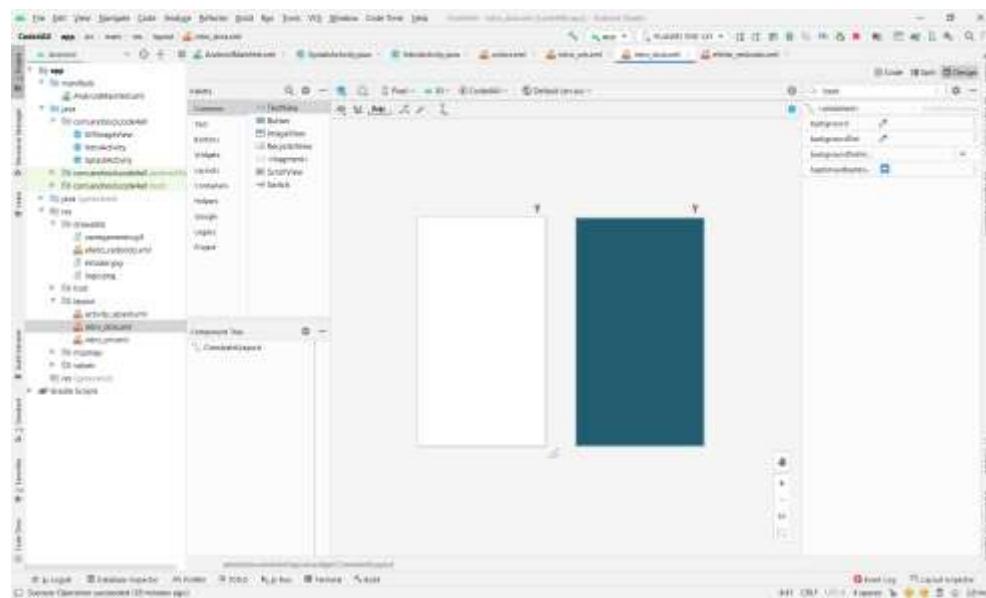


Figura 238: Primeira abertura do ficheiro "intro_dois.xml"

Após ter criado o layout, passei à criação da cor do segundo layout, como representado no layout, sendo ela #B45C09

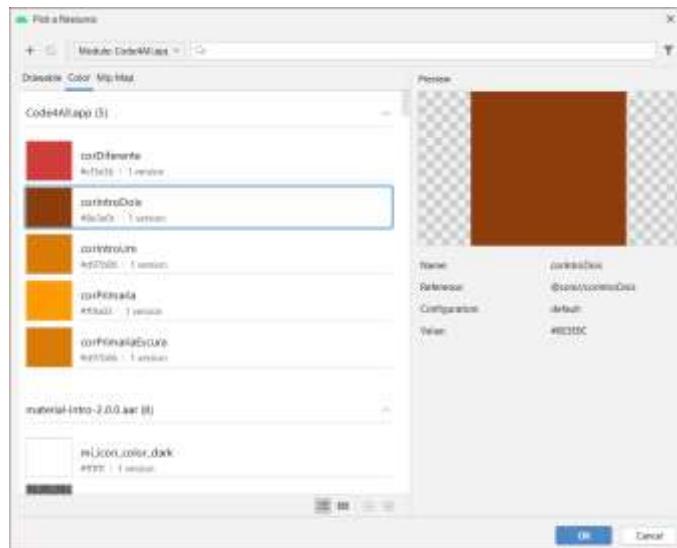


Figura 239: Criação da cor de fundo do segundo layout de introdução

Aqui apresento o ficheiro “colors.xml” que se encontra até este ponto do desenvolvimento, com as três cores iniciais do tema normal e também as duas cores dos layouts de introdução. Totalizando assim, um total de cinco recursos de cor.



Figura 240: Ficheiro "colors.xml" com as cores iniciais e as duas cores dos layouts

Após ter configurado a cor para o layout, fiz a importação da imagem, tal como no layout anterior.

Com a importação da imagem para o projeto, bastou apenas adicionar um ImageView ao layout e selecionar a imagem anteriormente importada, como se pode visualizar na seguinte captura de ecrã.

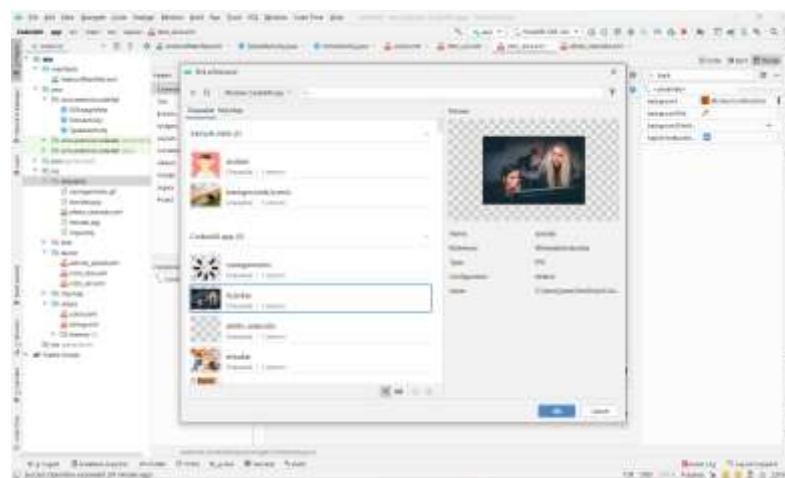


Figura 241: Inserção de uma ImageView, com a imagem anteriormente importada

Com a ImageView dentro do layout, basta agora ir ao layout do primeiro, e ir buscar o código do CardView, como se pode visualizar na figura abaixo.

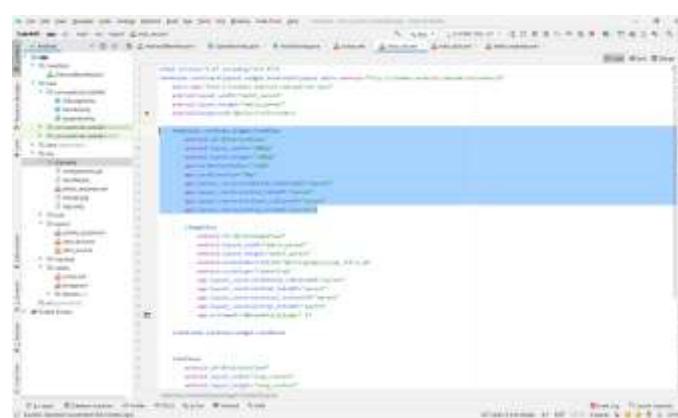


Figura 242: Seleção do código do CardView do layout anterior

Com o CardView do layout anterior copiado, bastou colar no layout que está a ser trabalhado, como representado na figura abaixo.

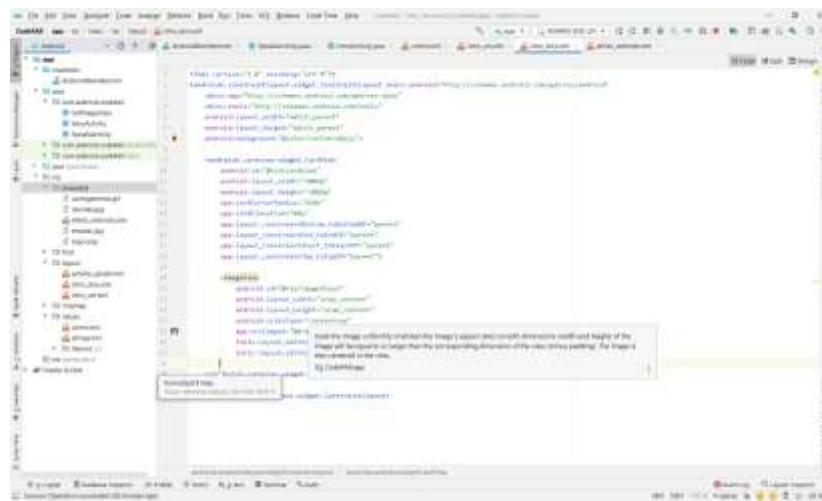


Figura 243: Colocação da ImageView dentro do CardView

Com as configurações realizadas anteriormente obtive o resultado apresentado em baixo.

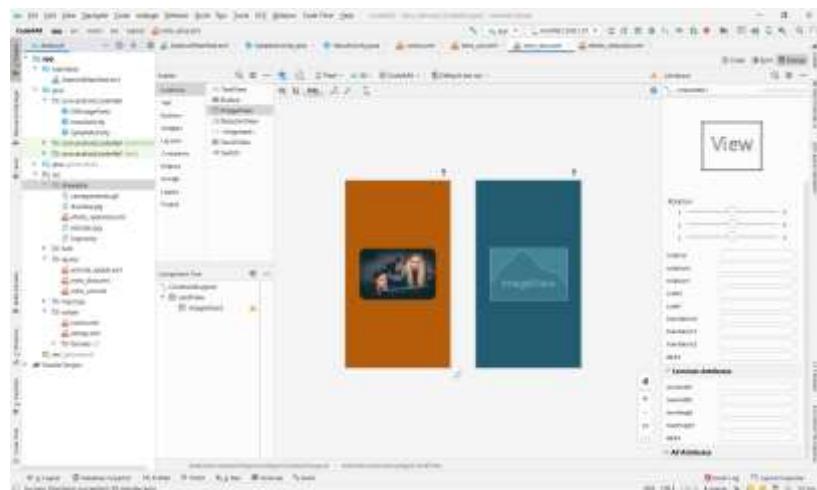


Figura 244: Resultado da ImageView, após de ser colocado o CardView

Após ter o ImageView corretamente configurado falta adicionar o respetivo TextView para colocar também o texto descritivo, como realizado no layout anterior.

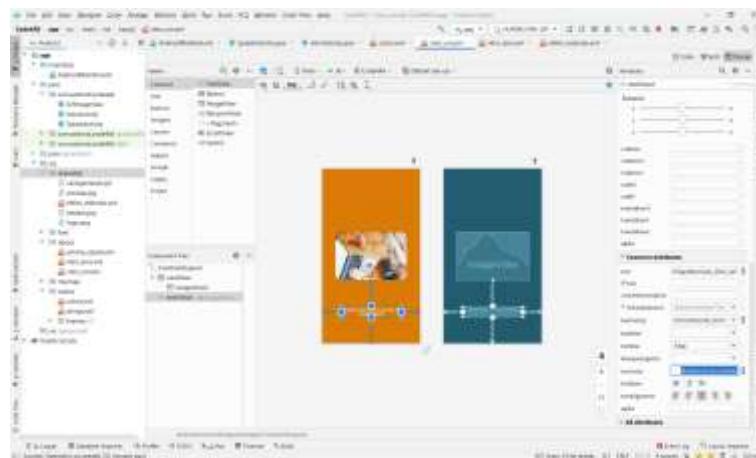


Figura 245: Colocação do TextView dentro do segundo layout

Com o TextView inserido, passei a fazer um recurso String, onde foi colocado uma frase descritiva, mais uma vez, como realizado na fase de desenho, no Adobe XD.

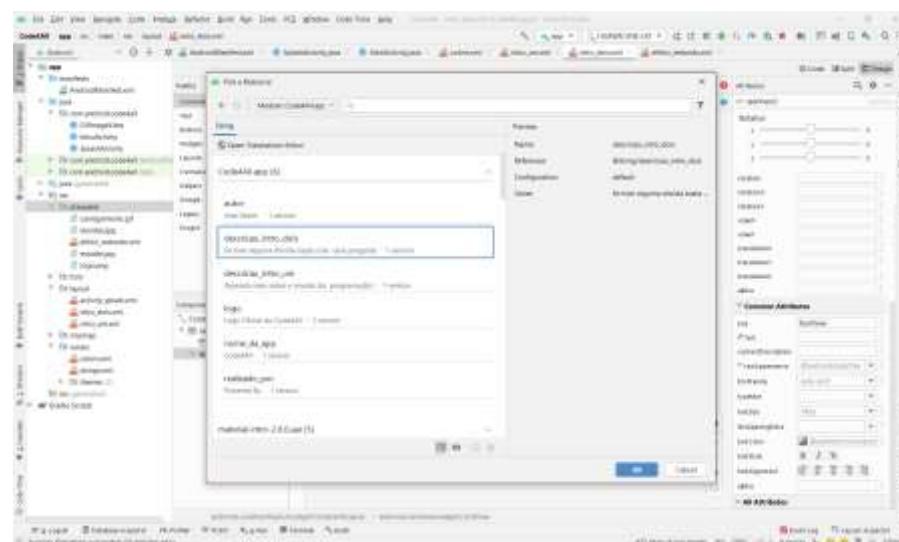


Figura 246: Criação e definição de um recurso string "descricao_intro_dois"

Após ter adicionado e feito as mesmas configurações do layout anterior, faltava apenas colocar a descrição da imagem, mais uma vez como foi feito no layout anterior.

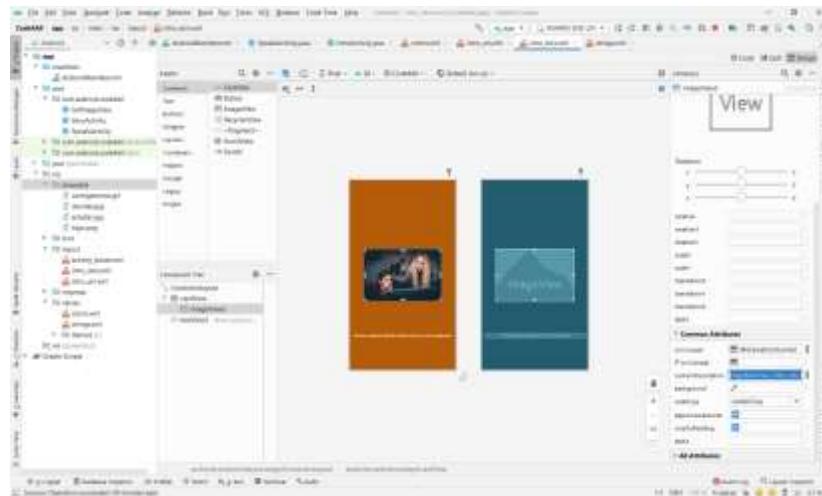


Figura 247: Configuração do segundo layout finalizada

Com o layout devidamente preparado e configurado, faltava adicionar um novo slide com o mesmo e foi isso que realizei.

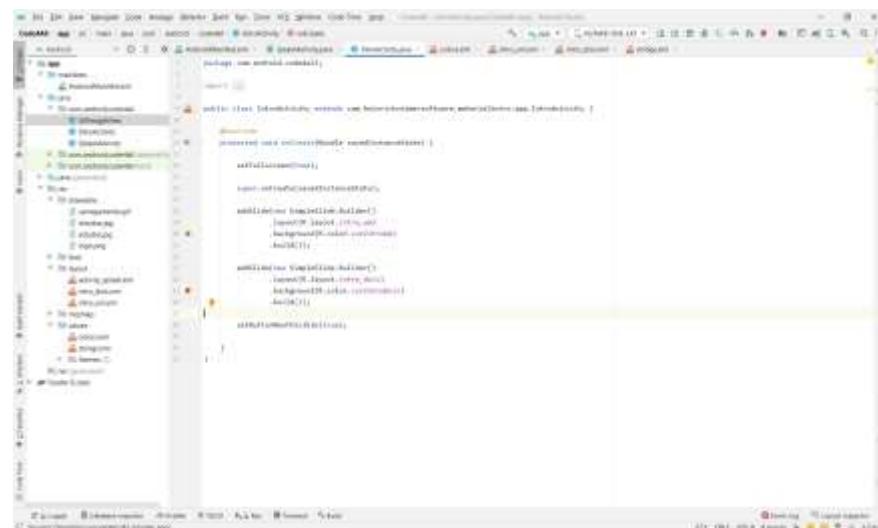


Figura 248: Inserção de um novo slide, com o segundo layout

Com isto, decidi executar a aplicação para ver se ocorria algum erro e para ver se estava tudo a funcionar.



Figura 249: Slide um da parte introdutória

Comecei por fazer a transição e reparei logo, que estava algo de errado para além do notch invisível, que a parte de baixo estava com uma cor diferente.

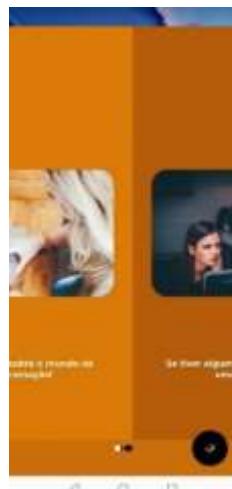


Figura 250: Transição entre o layout um e dois

No segundo layout, estava tudo como devia estar, sem ser a parte do notch...



Figura 251: Segundo layout de introdução

Passei para a resolução da barra inferior de cor diferente, na transição entre os dois layouts, comecei por apagar o atributo “background” dentro do “ConstraintLayout”, visto que a cor de fundo também era definida na atividade, em Java. Fiz esta modificação tanto no primeiro layout, como no segundo layout.

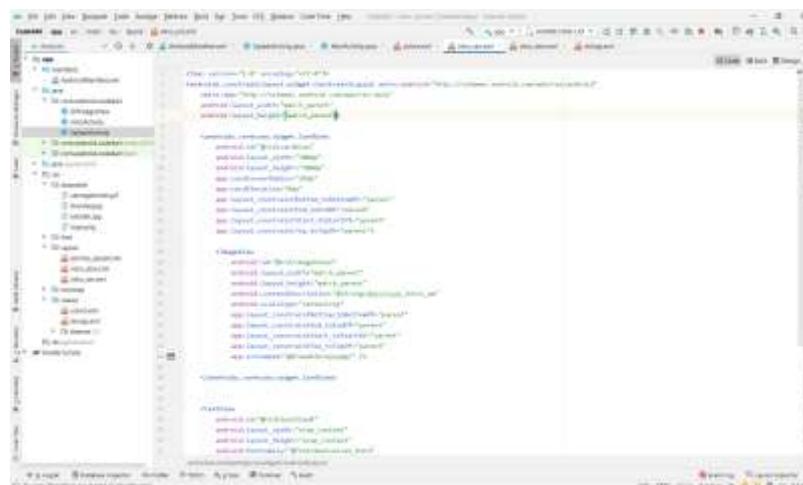


Figura 252: Remoção do atributo "background" do primeiro layout

Depois, fiz basicamente a mesma operação, remover o atributo “background” dentro do “ConstraintLayout”, e com esta pequena alteração dou como concluídos os dois layouts de introdução, faltando agora o terceiro e último.

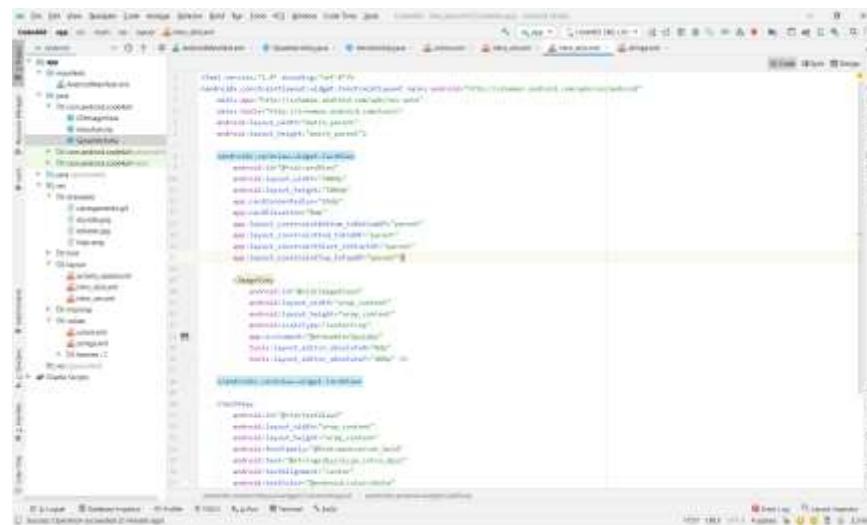


Figura 253: Remoção do atributo "background" do segundo layout

Passei à criação do terceiro e último layout de introdução.

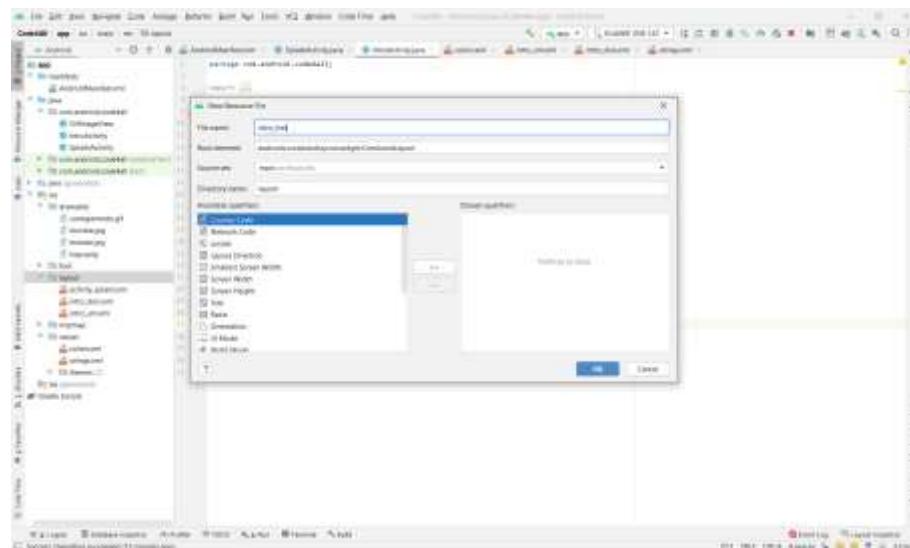


Figura 254: Criação do terceiro e último layout

Após ser criado o layout, o mesmo será aberto automaticamente, para a edição ser logo imediata.

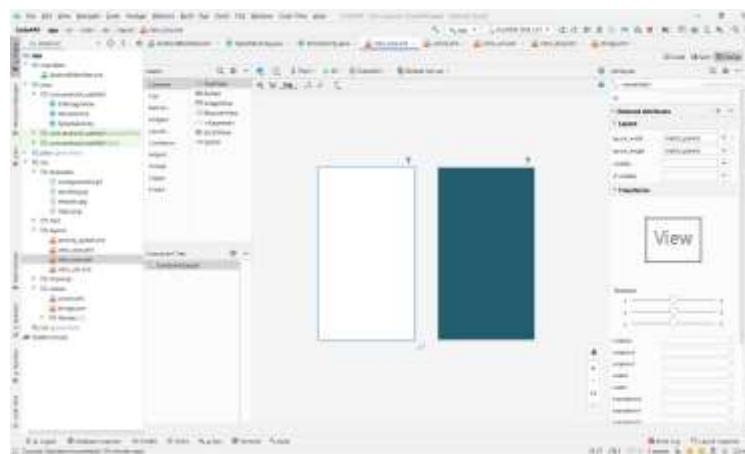


Figura 255: Abertura do layout "intro_tres.xml" pela a primeira vez

Após ter o layout criado, dirigi-me mais uma vez à atividade e configurei logo um novo slide, como se pode ver na figura abaixo. Mudei o layout, mas não mudei a cor visto que ainda não tinha vindo a ser criada.

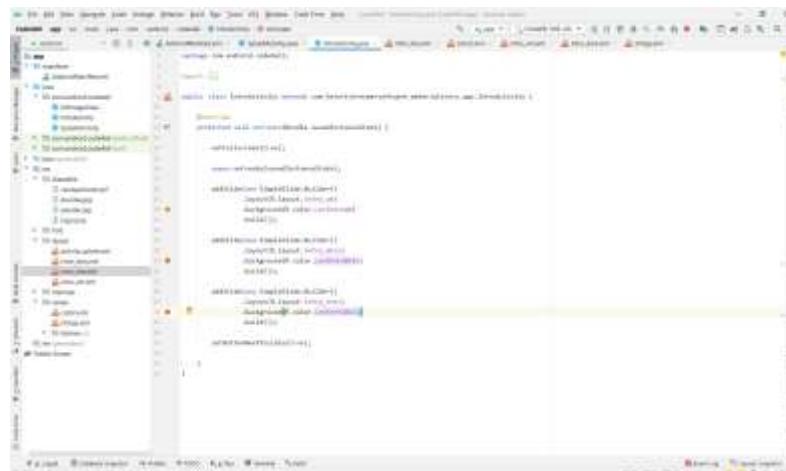


Figura 256: Criação do terceiro slide

Como dito anteriormente, a cor de fundo ainda não tinha sido criada, passei à criação da mesma.

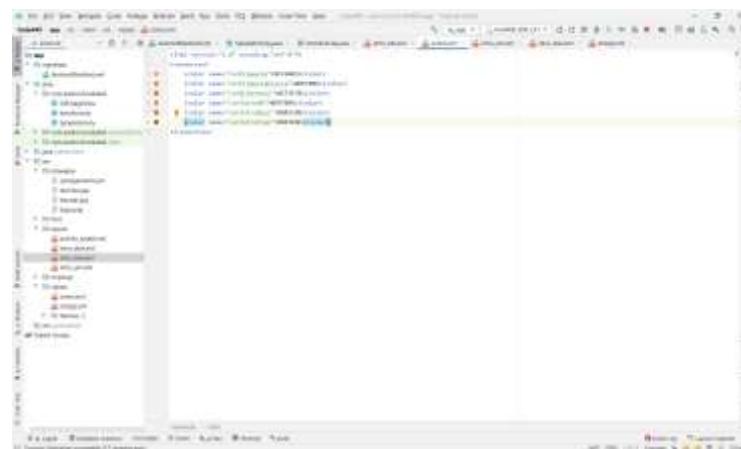


Figura 257: Criação da cor "corIntroTres" para uso no terceiro layout

Seguidamente, passei à alteração da cor no terceiro três, de “corIntroDois” para “corIntroTres”, como demonstrado na figura...



Figura 258: Modificação da cor de fundo do terceiro slide

Após ter configurado o terceiro slide, faltava agora dar início ao desenvolvimento da estrutura do layout, começando por importar a imagem.

Com a importação da imagem, para dentro da pasta “drawable”, bastou arrastar uma ImageView para o layout.

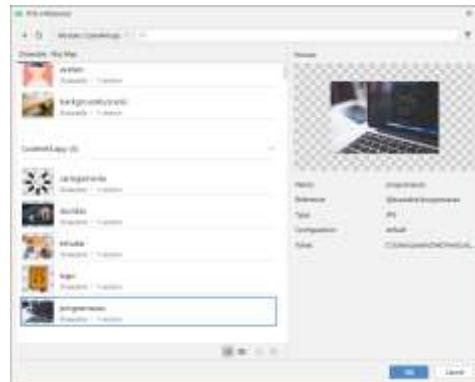


Figura 259: Seleção da imagem para a ImageView

Na imagem abaixo, está representado o comportamento da ImageView logo após de ser selecionada uma nova imagem.

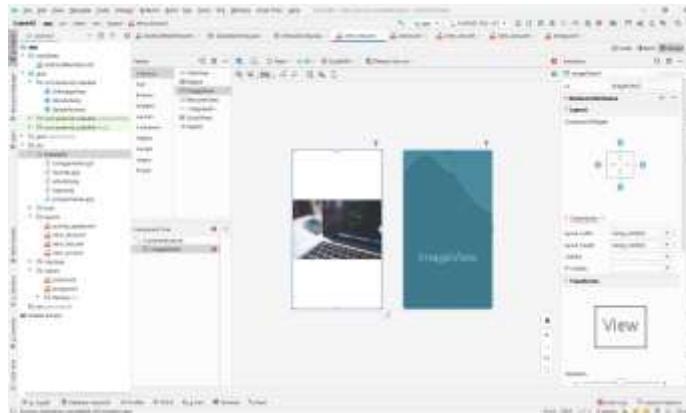


Figura 260: Estado do layout após a inserção de uma ImageView

Seguidamente, fui copiar o código do CardView, por questões de facilidade e rapidez, tal como realizei no layout anterior.

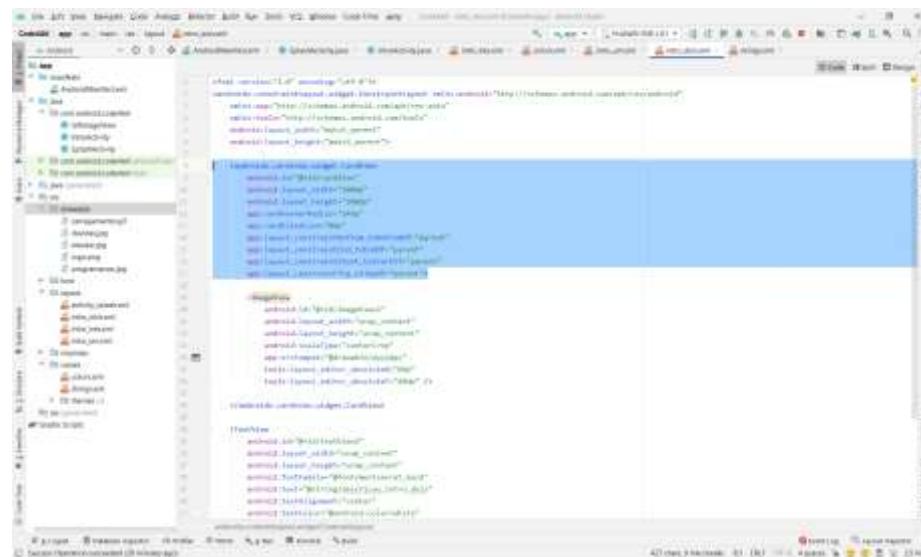


Figura 261: Seleção do CardView colocado no layout "intro_dois.xml"

Com o CardView copiado, bastou colar o mesmo e dentro dele colocar a ImageView, como se pode visualizar na figura abaixo descrita.

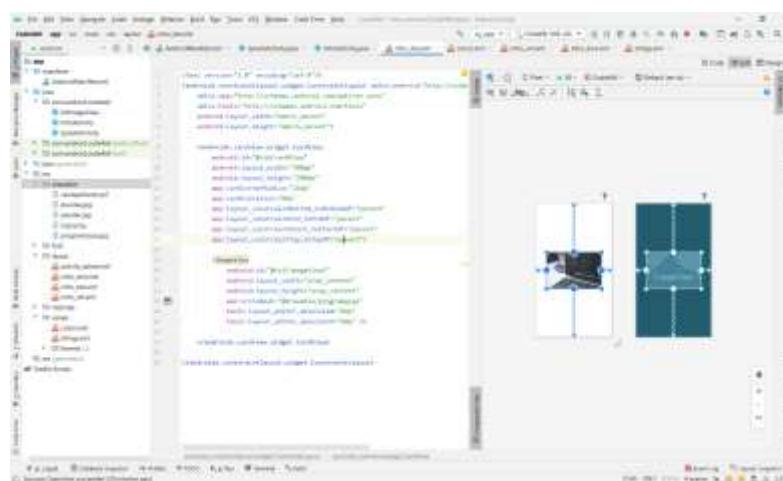


Figura 262: Inserção e configuração do CardView e ImageView

Logo depois, inseri um TextView e fiz a criação do recurso String, com a descrição do terceiro layout introdutório, mais uma vez, como realizei no Adobe XD.

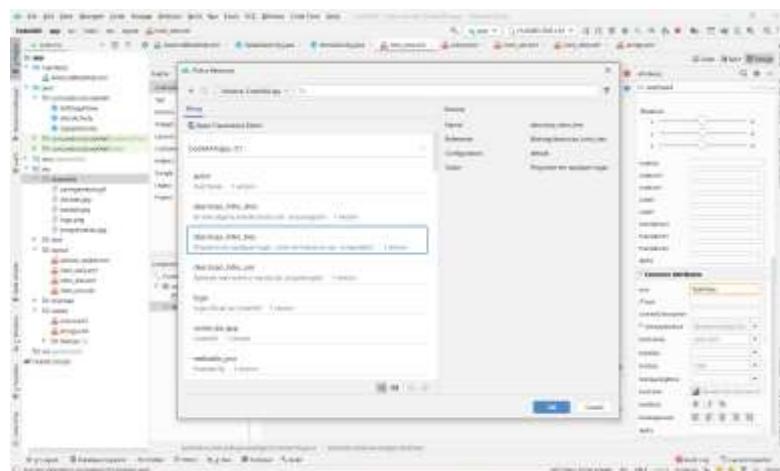


Figura 263: Criação do recurso String com a descrição para o TextView

Com o TextView inserido e com o texto da descrição, realizei então o alinhamento, tal como fiz nos dois layouts anteriores.

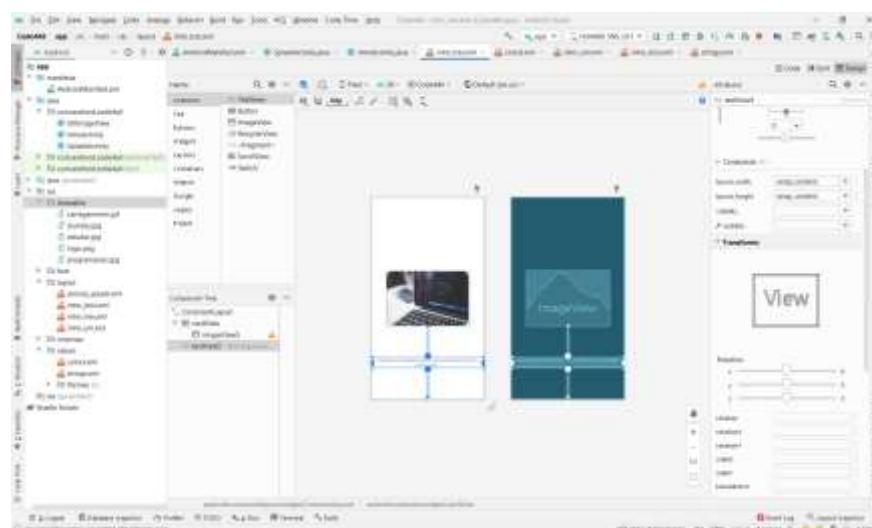


Figura 264: Alinhamento do TextView do terceiro layout

Seguidamente de ter o TextView corretamente alinhado, alterei a sua fonte e a sua cor para branco, mais uma vez de forma igual aos outros layouts.

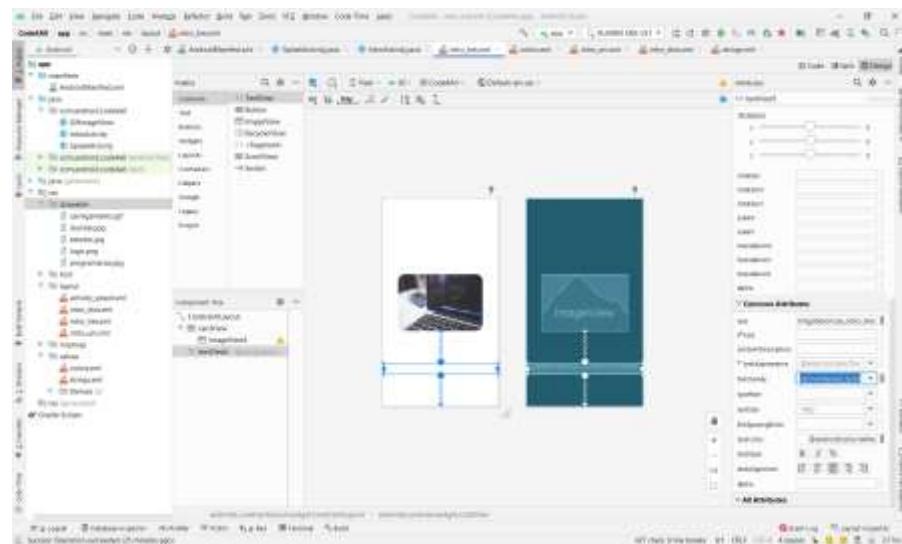


Figura 265: Configuração e ajustes no TextView

Com o TextView devidamente configurado, fiz a última mudança na ImageView, colocando uma descrição na mesma, para desaparecer o aviso e assim ter o layout corretamente configurado e desenhado

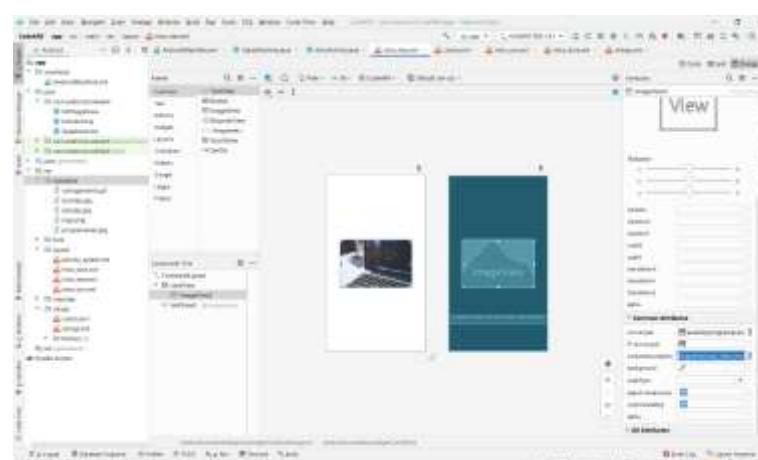


Figura 266: Colocação da descrição na ImageView

Com o terceiro layout totalmente configurado, fiz a execução da aplicação, obtive o seguinte resultado. Um pequeno realce é que o notch desta vez está branco.



Figura 267: Comportamento do terceiro layout após da sua execução

Como solução do notch, falei com o meu coordenador de curso e o mesmo disse que poderia deixar a aplicação com a barra superior e inferior no seu estado normal, ou seja, sem estar em ecrã inteiro. Para fazer isso removi o código “`setFullscreen(true);`”



Figura 268: Remoção do código para que a `IntroActivity` deixe de atuar em ecrã inteiro

Com apenas a remoção daquele pequeno código, já fez uma grande diferença no layout em si e para ser sincero gostei muito mais do resultado assim.



Figura 269: Resultado do layout, com a remoção do código "setFullscreen(true);"

Com os três slides configurados e adaptados corretamente, falta agora adicionar o último slide, aquele que irá permitir fazer o login (Email/Facebook/Twitter) e criar conta, desta vez vou criar uma atividade chamada de EntrarActivity.

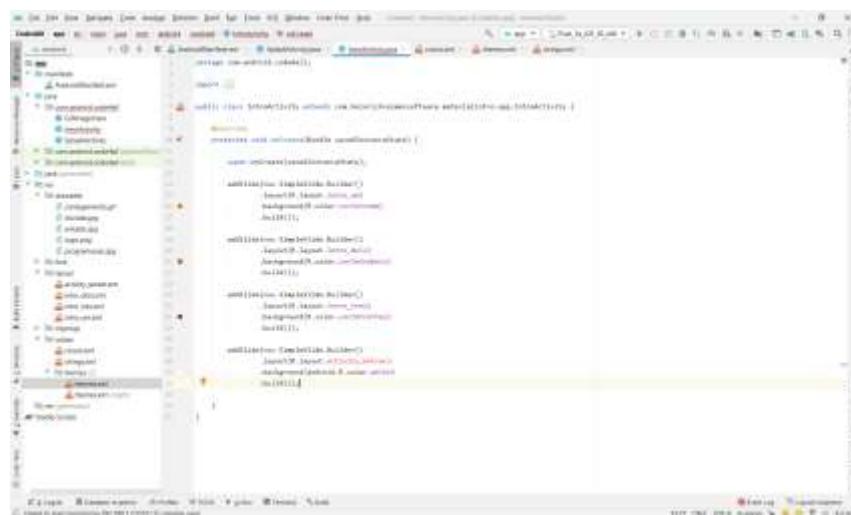


Figura 270: Inserção de um novo slide para a “EntrarActivity”

Agora sim, a IntroActivity pode ser dada como finalizada, na imagem abaixo foram adicionadas quatro linhas de código:

Duas no slide da “EntrarActivity” para que o utilizador quando chegar à mesma, não consiga nem andar para trás nem para a frente.

```
.canGoBackward(false)
.canGoForward(false)
```

Figura 271: Configurações para impedir o utilizador de andar para trás e para a frente

E também duas linhas de código, no próprio método onCreate(), para que o botão de andar para a frente e andar para trás desapareçam, obrigando assim o utilizador a deslizar o dedo para avançar nos slides.

```
setButtonBackVisible(false);
setButtonNextVisible(false);
```

Figura 272: Código para esconder os botões de andar para a frente e para trás

Aqui, apresento o código dos três slides e as respetivas configurações da libreria “material-intro”

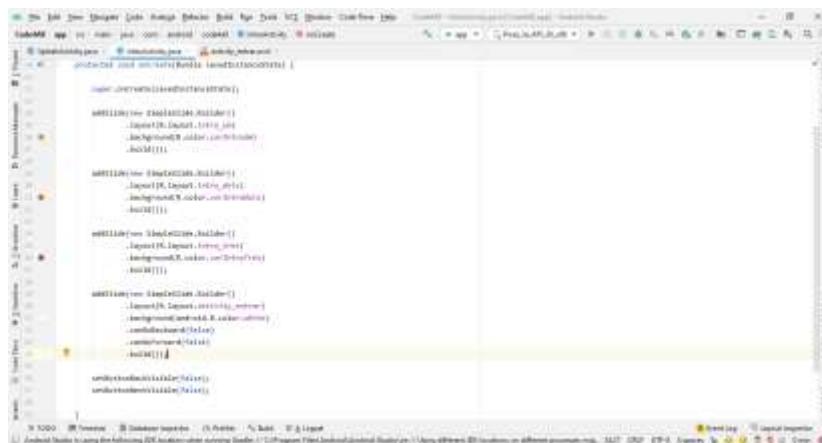


Figura 273: Código geral na IntroActivity

Para finalizar esta atividade, procedi à criação de uma nova classe “Permissao” que iria verificar se todas as permissões haviam sido autorizadas ou não.

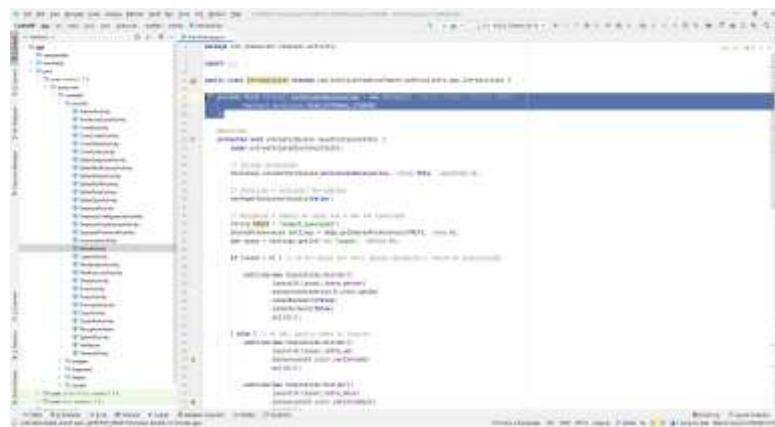


Figura 274: Permissão "READ_EXTERNAL_STORAGE" para permitir a visualização das imagens da galeria

Método “validarPermissoes()”, dentro da classe “Permissao”. Foi graças ao curso que tirei, que aprendi a realizar este método, para que todas as permissões fossem aceites e não houvesse nenhum problema com a execução da app.



Figura 275: Método "validarPermissoes()" da classe "Permissao"

“EntrarActivity” (Layout da IntroActivity)

Passando à “EntrarActivity” e com os ficheiros da mesma criados, efetuei a importação dos elementos que iria necessitar para a realização do layout desta atividade.



Figura 276: Importação do logo do code4all (livro aberto)

Seguidamente, coloquei uma ImageView no layout desta atividade com a imagem anteriormente importada.

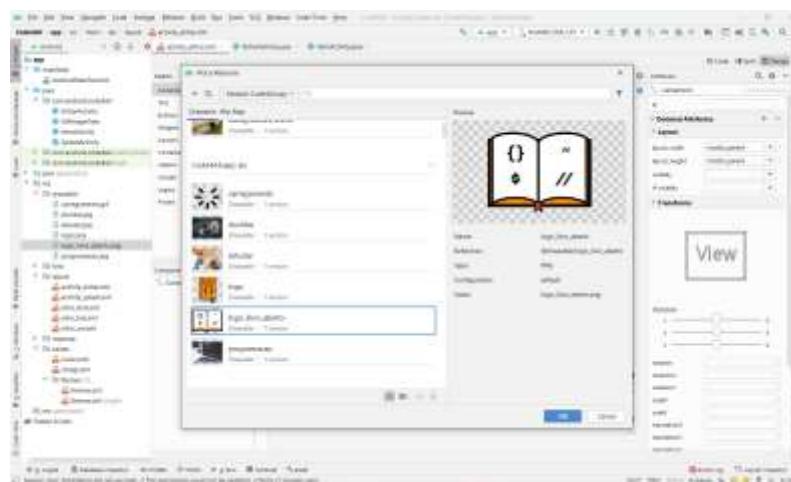


Figura 277: Inserção de um ImageView para colocar o logo (livro aberto)

Logo de seguida, comecei a configuração dessa mesma ImageView, colocando uma largura e altura de 100dp.

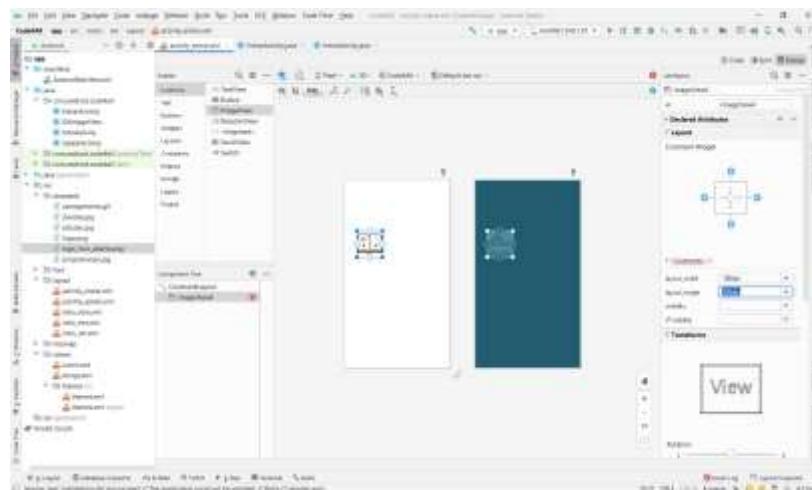


Figura 278: Configuração do tamanho da ImageView

Depois, realizei o alinhamento do logo (livro aberto), colocando o mesmo alinhado horizontalmente no meio e verticalmente ao topo do layout com um espaçamento superior de 16dp.

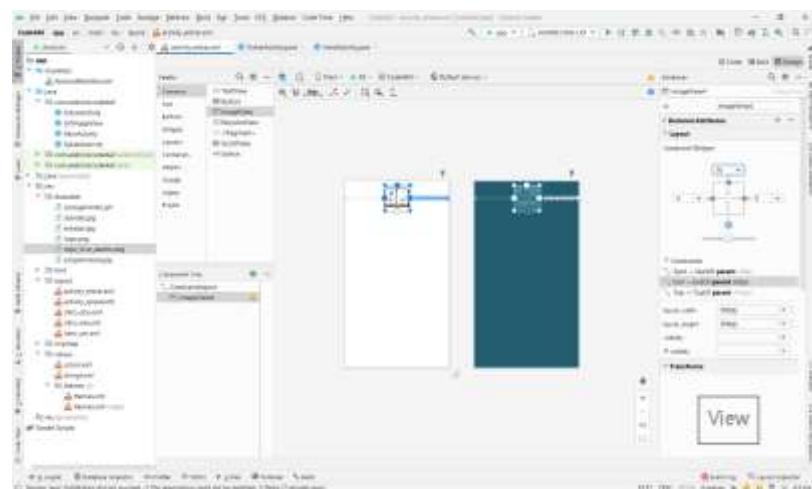


Figura 279: Alinhamento da ImageView

Com a imagem devidamente alinhada, atribui a descrição da mesma com um recurso String, o mesmo que atribui ao logo (livro fechado).

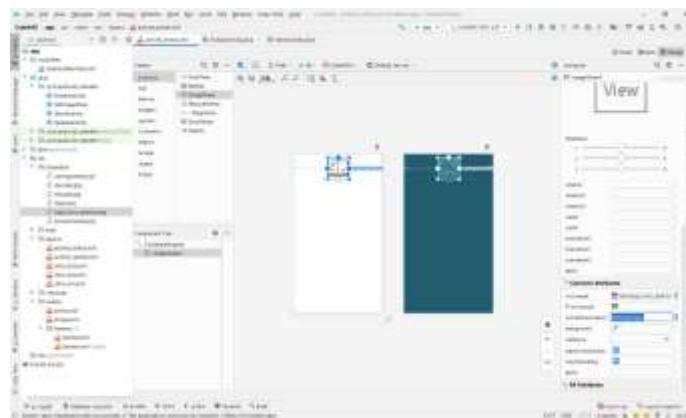


Figura 280: Definição da descrição da imagem

Seguidamente, passei à pesquisa de como iria colocar a borda preta que planeei na fase de desenho e com alguma pesquisa encontrei o seguinte site, que me deu uma possível solução para o que queria atingir.

<https://stackoverflow.com/questions/35369691/how-to-add-colored-border-on-cardview>

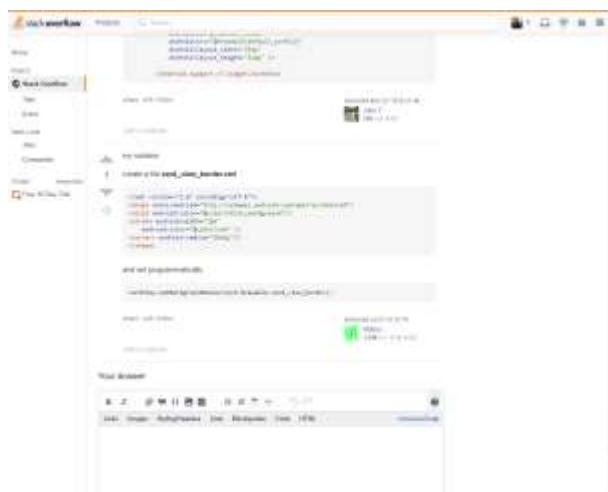


Figura 281: Possível resposta para colocar barras pretas num CardView

Primeiramente, comecei por selecionar o CardView anteriormente realizado, por ser mais rápido e fácil, utilizei o novamente pois achei que seria mais fácil utilizar o mesmo componente, pois já tinha um pouco de experiência com o mesmo, como visto na realização dos layouts anteriores

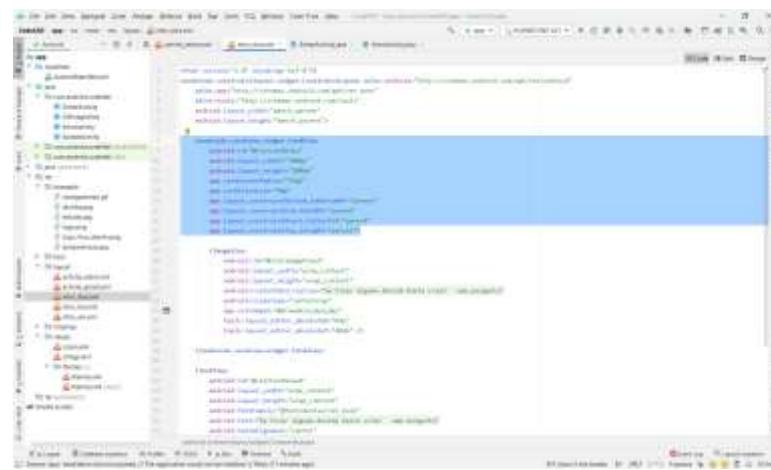


Figura 282: Seleção do CardView no layout "intro_dois.xml"

Após de ter selecionado o respetivo código do CardView colei o mesmo no layout desta atividade para depois começar o desenvolvimento do mesmo.

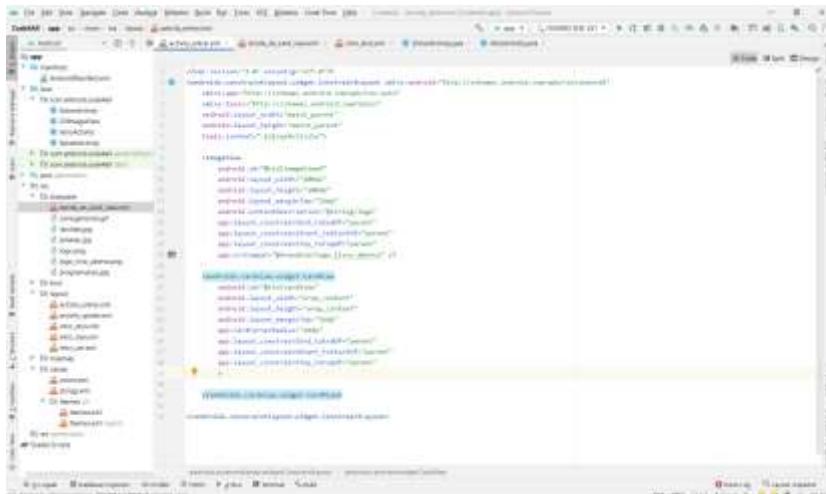


Figura 283: Colocação de um CardView no layout da atividade

A partir daqui, comecei a aplicar a solução que anteriormente descobri, mais uma vez no stackoverflow, comecei então por criar um ficheiro xml para que depois seja usado para alterar o aspeto do CardView, visto que era a minha prioridade colocar as barras pretas de volta do mesmo.

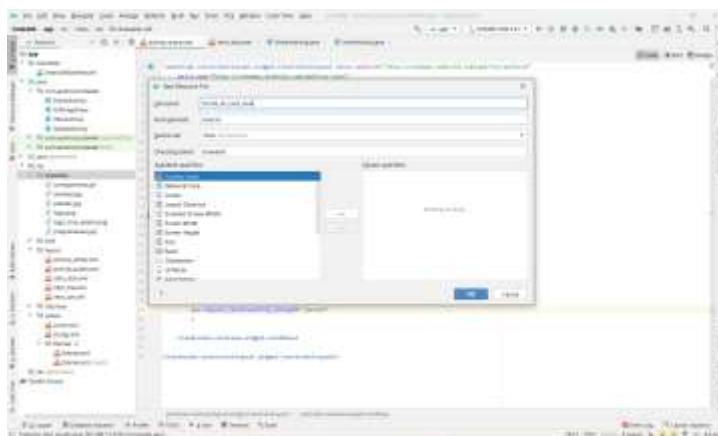


Figura 284: Criação do ficheiro "borda_do_card_view.xml"

Com o ficheiro de “efeito” criado, comecei logo a editar o mesmo, colocando logo uma etiqueta “shape” para de logo modificar o seu aspetto, neste caso a sua forma. Coloquei uma linha (stroke) de 2dp e arredondei os cantos com um raio de 20dp.

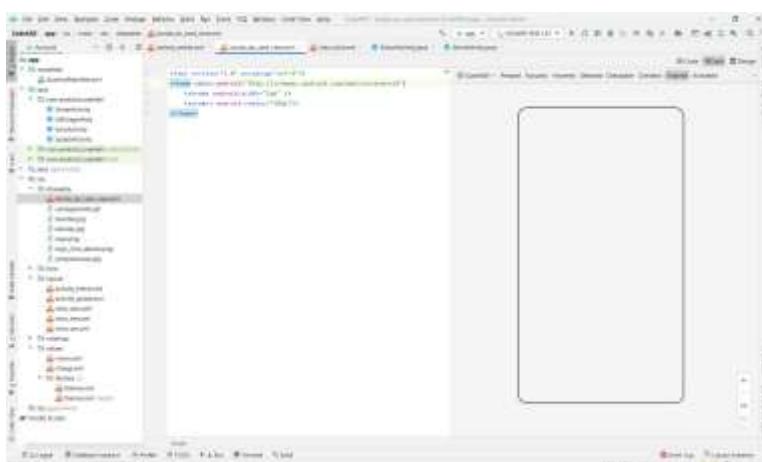


Figura 285: Configuração e edição do ficheiro "borda_do_card_view.xml"

Após ter o ficheiro “borda_do_card_view.xml” devidamente configurado e finalizado, o próximo passo era bastante simples, bastava apenas definir o atributo “background” como o ficheiro anteriormente criado ficando da seguinte forma “`android:background='@drawable/borda_do_card_view'`”, de imediato o efeito foi aplicado, até aqui estava tudo a funcionar conforme planeado.

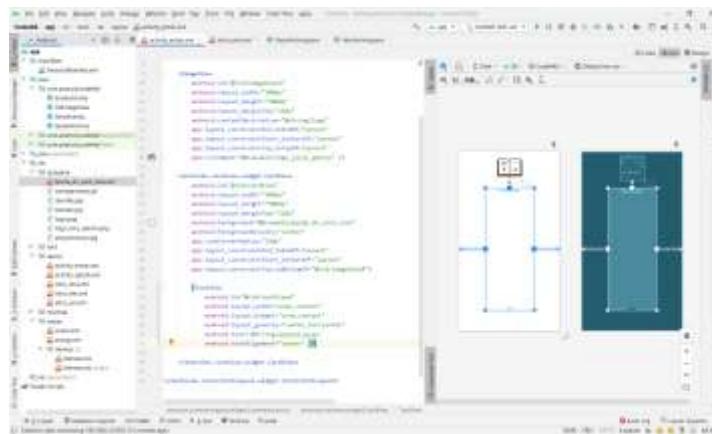


Figura 286: Definição do atributo "background" dentro do CardView

Seguidamente, passei à colocação de um TextView, para tentar espelhar o que fiz na fase de desenho. Para realizar esse feito, precisava primeiro de importar uma nova fonte da família “Montserrat”, sendo ela do tipo “ExtraBold”.

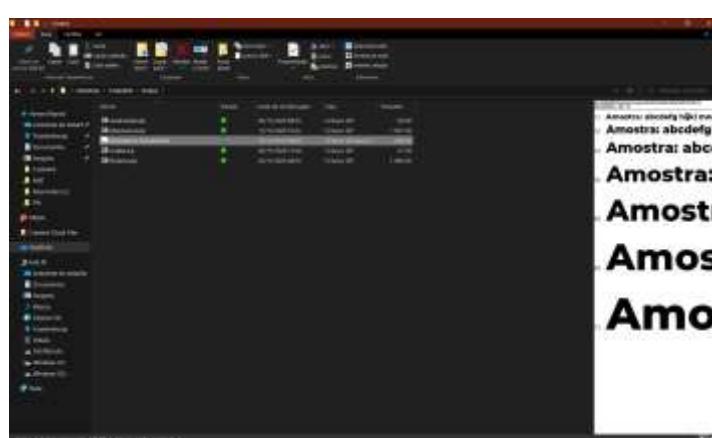


Figura 287: Importação da fonte "Montserrat ExtraBold"

Com a importação desta fonte, é obrigatoriamente necessário mudar o seu nome, visto que tem de ser simples e não pode conter qualquer caracter especial. Tal como realizei na figura abaixo, colocando o nome da fonte como “montserrat_extra_bold”.

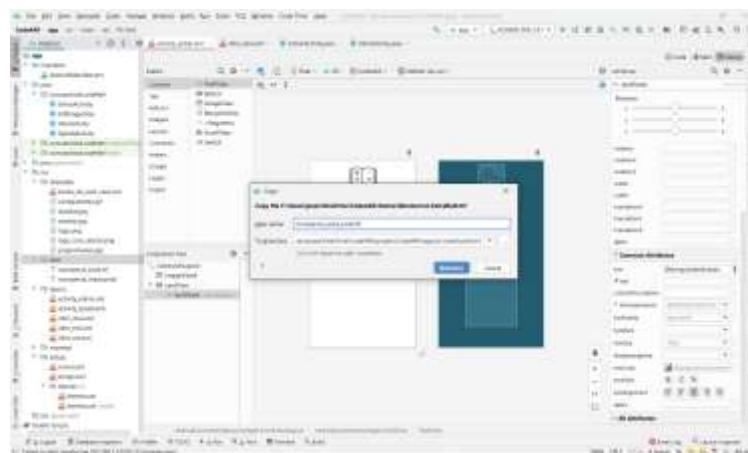


Figura 288: Importação da fonte para o projeto com a modificação do seu nome

Com a importação da fonte realizada com sucesso, bastou definir o atributo “fontFamily” com a fonte anteriormente importada, criei ainda um recurso String com o nome “autenticacao” que contém a palavra “Autenticação” e mudei a cor do mesmo para preto, como se representa na figura abaixo.

Por fim, alterei o CardView para um LinearLayout, visto que me apercebi que ambos têm as mesmas funções e como já utilizei em outros projetos o mesmo, nesta parte decidi utilizar o mesmo.

Um *LinearLayout*, é basicamente um componente que permite a organização de outros componentes, quando esses componentes se encontram dentro do mesmo.

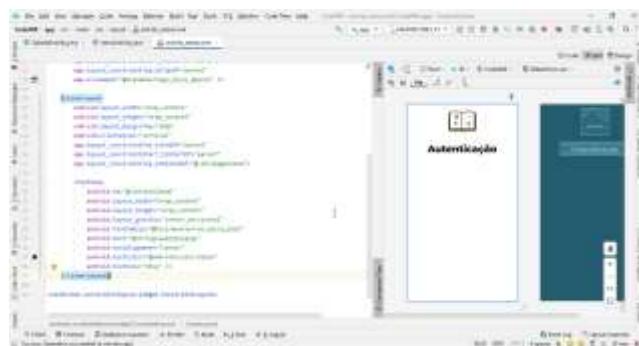


Figura 289: Definição do texto TextView e Modificação do CardView para LinearLayout

Nesta altura observei que o nome “*borda_do_card_view*” não era o mais correto, e decidi mudar o nome do mesmo para “*borda_preta*” com o atalho “SHIFT+F6” e obviamente com o ficheiro selecionado, como se representa na figura abaixo.

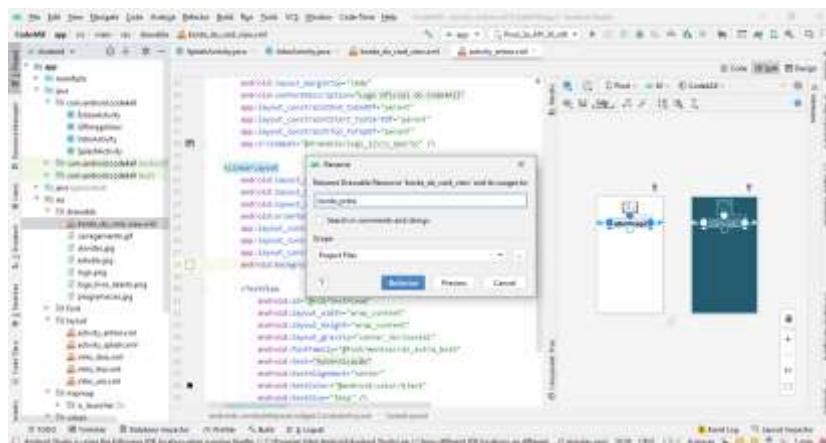


Figura 290: Modificação do nome do ficheiro "borda_do_card_view" para "borda_preta"

Após realizar esta pequena alteração no ficheiro, continuei o resto do desenvolvimento do layout, desta vez, criando um recurso String sendo este o slogan da minha aplicação. O slogan está representado na figura da próxima página.

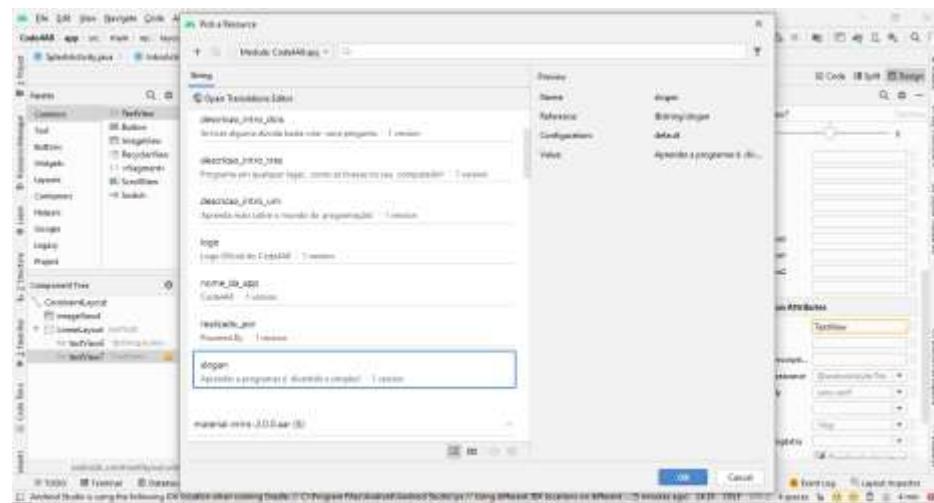


Figura 291: Criação do recurso String "slogan"

Com o recurso String criado, bastou associar o mesmo à TextView para esta ficar com o texto do recurso, seguidamente mudei o tamanho da letra para 18sp e a fonte da letra para “montserrat_medium”, ficando da seguinte forma:

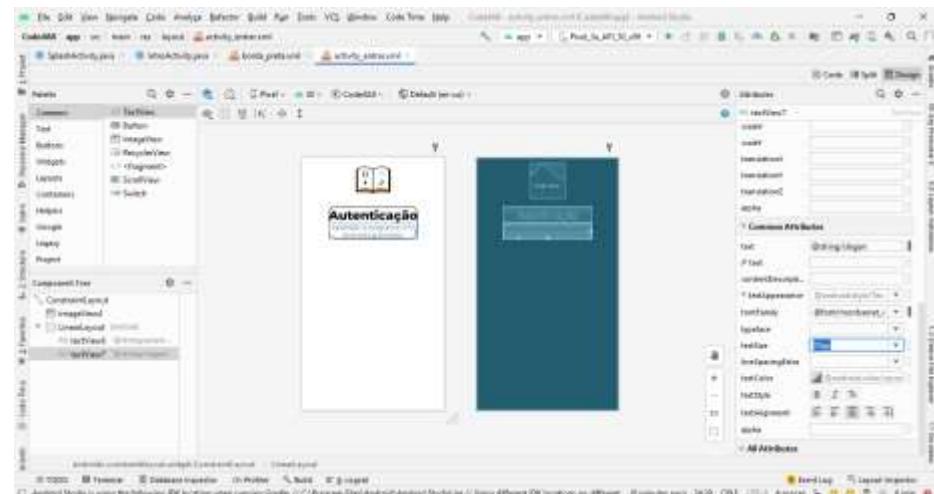


Figura 292: Resultado do TextView (slogan)

Passei à colocação de um “Button”, e coloquei como atributo “text”, a String criada e também fiz a definição do atributo “background” com o ficheiro (@drawable/borda_preta) xml anteriormente criado, que coloca um efeito redondo no botão, como se pode visualizar na figura abaixo.

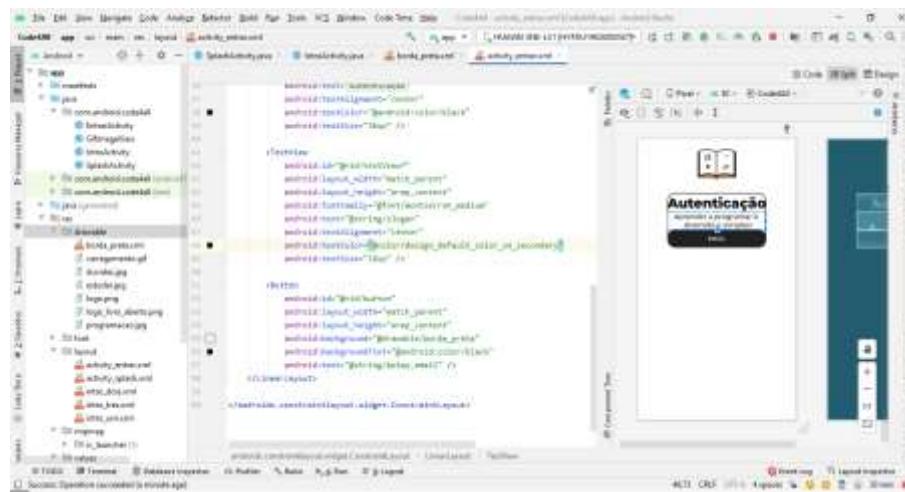


Figura 293: Configuração do botão "Email"

Seguidamente, passei para a modificação da cor dos botões anteriormente adicionados, começando pelo botão do Facebook (#1947E5).

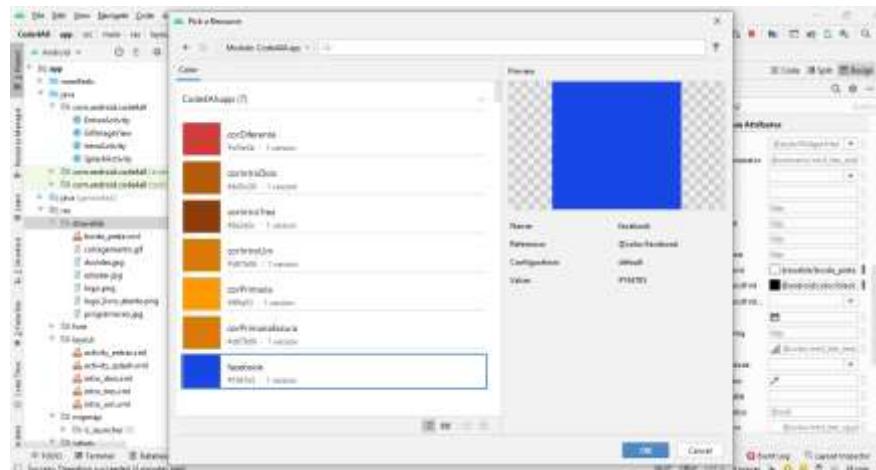


Figura 294: Criação da cor hexadecIMAL "facebook"

E realizei o mesmo processo, para o botão do “Twitter” (#00C6AE), como se pode observar na figura abaixo.

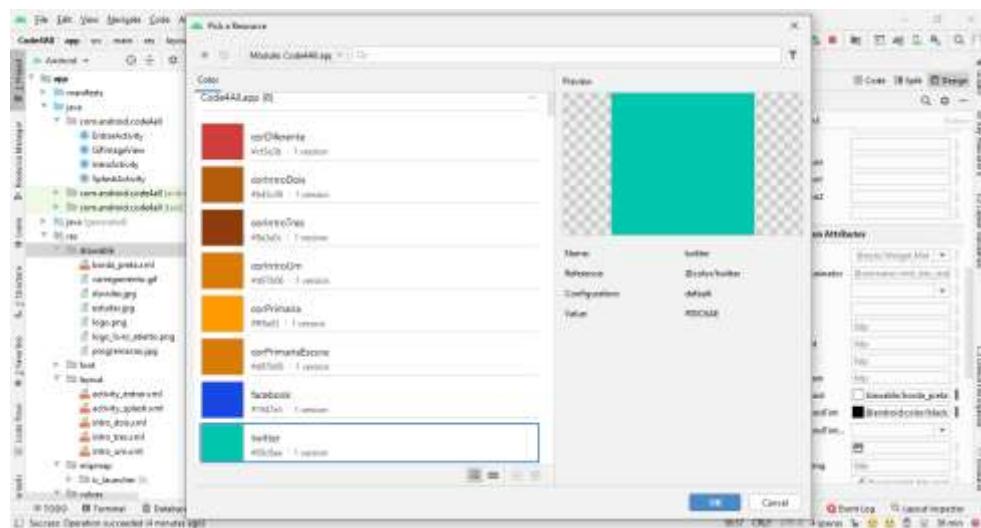


Figura 295: Criação da cor hexadecimal "twitter"

Seguidamente, fiz a definição do atributo “backgroundTint”, usa-se este atributo para definir a cor de fundo de um botão e usa-se o “background” para atribuir efeitos/aspetos ao mesmo. Começando pelo botão Email, que defini com o valor “@colors/corPrimariaEscura”, logo depois o Facebook, com o valor “@color/facebook” e finalmente o botão Twitter com o valor “@colors/twitter”.

Seguidamente, passei ao ajuste do texto “Autenticação” colocando as margens do mesmo para 16dp, dando um aspetto mais interessante ao mesmo. Também se observa na figura abaixo que, de certa forma, os botões continuam pretos, mesmo após ter definido o background.

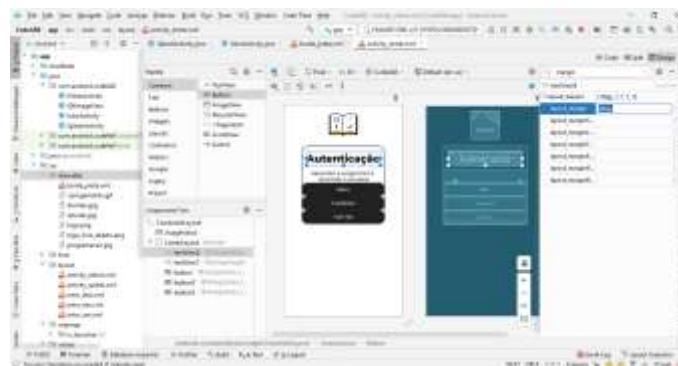


Figura 296: Configuração das cores de fundo dos botões e pequeno ajuste no texto

Após ter realizado a mudança de cor dos botões e um pequeno ajuste no texto “Autenticação”, realizei uma execução da aplicação, para ver se estava tudo a funcionar corretamente, o resultado da mesma apresenta-se em baixo.



Figura 297: Execução da atividade com as cores dos botões e com o texto ajustado

Seguidamente, como estive a mexer com espaçamentos, decidi realizar o mesmo, no LinearLayout, para que ficasse mais apresentável. O resultado, com um padding de 8dp ficou como se observa na próxima figura.

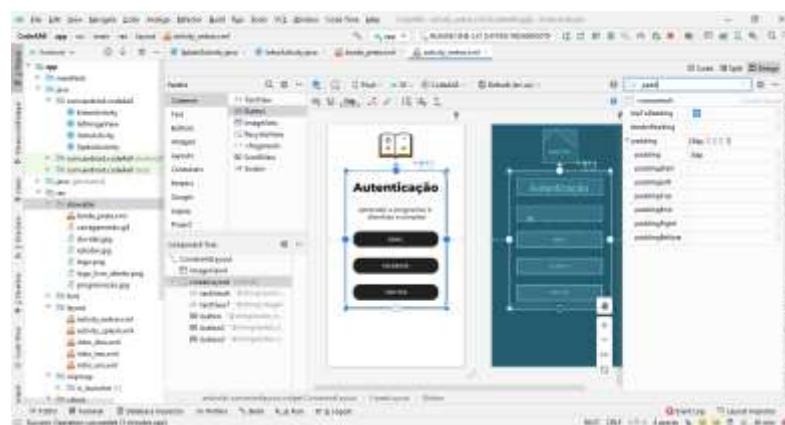


Figura 298: Ajuste de espaçamento no LinearLayout

Com o recurso String criado, passei à configuração de mais um novo TextView, com esse mesmo recurso. Decidi também ainda, alinhar este mesmo componente ao centro horizontalmente, e no meio do LinearLayout e na parte inferior do layout.

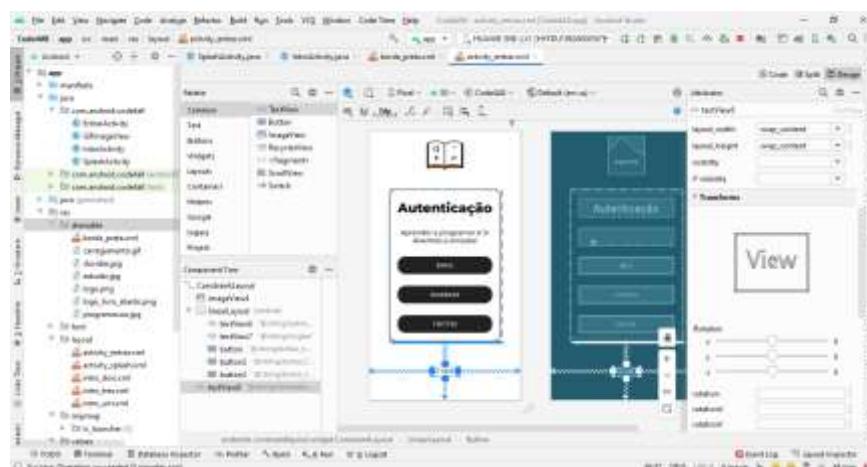


Figura 299: Colocação, configuração e alinhamento do TextView

Depois, passei à criação de mais um recurso String, para que seja colocado num TextView que será mais à frente criado.

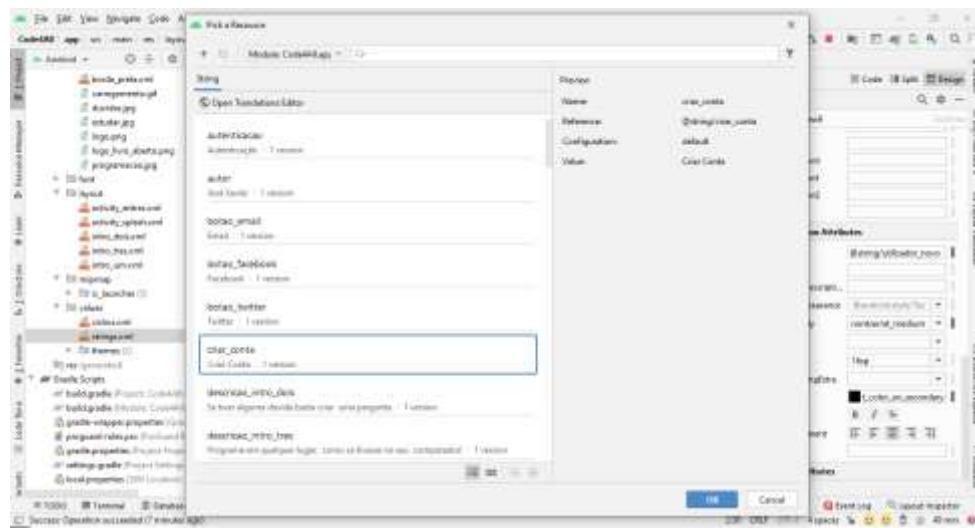


Figura 300: Criação do recurso String "criar_conta"

Com o recurso String criado, visto que teria de modificar a cor do TextView, passei à criação da sua cor (#F95A2C).

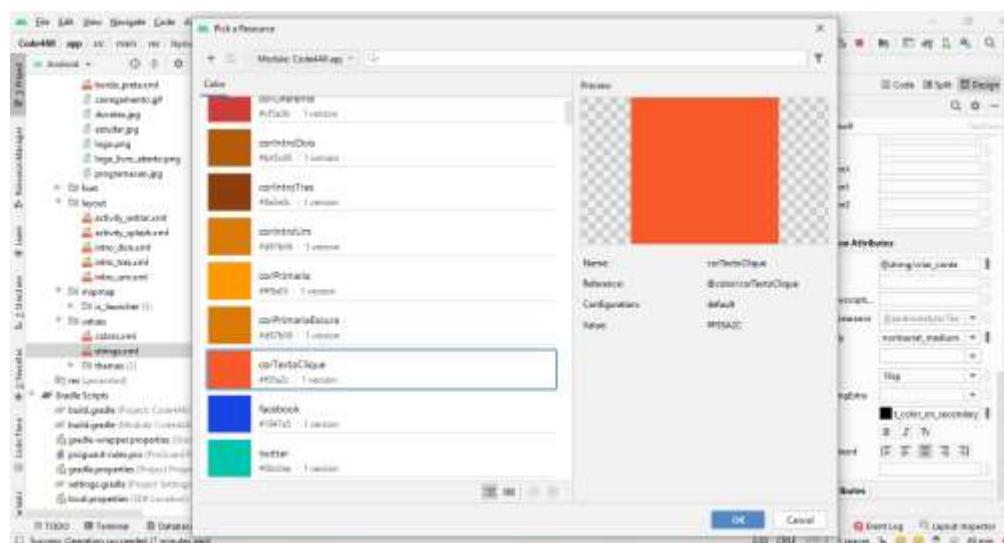


Figura 301: Criação de uma cor "corTextoClique"

Seguidamente, coloquei um TextView, configurei o seu texto, a cor, o tamanho do texto (16dp) e por fim, o seu alinhamento, como se pode observar na figura abaixo.

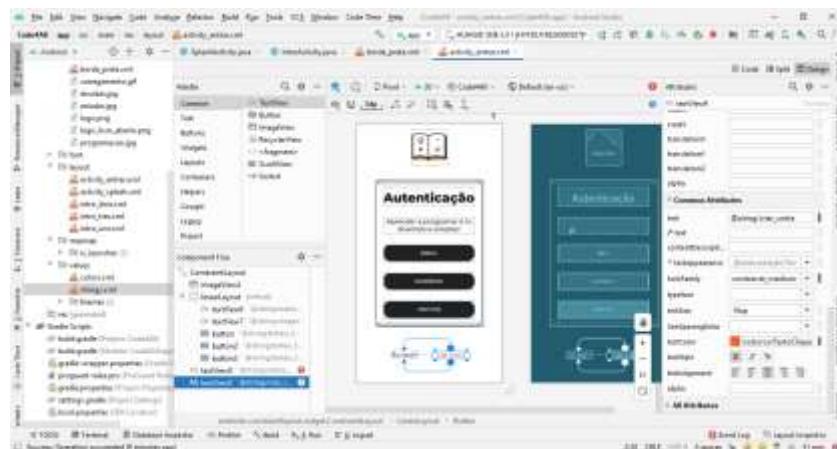


Figura 302: Configuração e alinhamento do TextView

Passei ao alinhamento dos dois TextView's localizados em baixo, o “És novo?” alinhei horizontalmente ao LinearLayout com um espaçamento de 24dp e o “Criar Conta” foi a mesma coisa, só que para o lado direito do mesmo. Seguidamente, alinhei o LinearLayout no meio e verticalmente alinhei o mesmo à imagem do livro, com um espaçamento de 24dp, visto que ficava mais interessante dessa maneira.

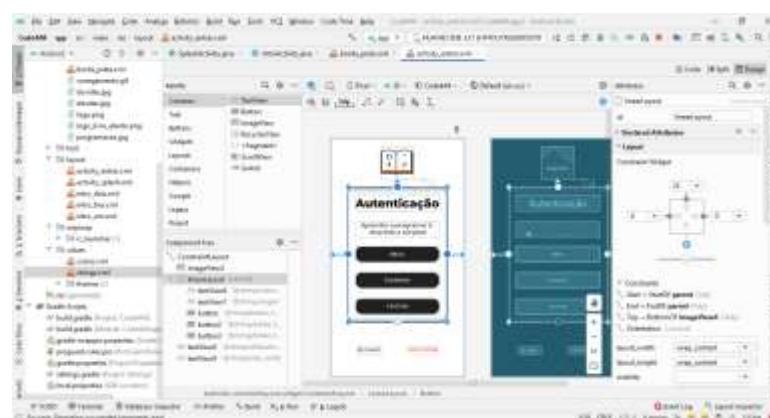


Figura 303: Alinhamento do LinearLayout e dos dois TextView's previamente colocados

Como alinhamento final, alinhei o logo (livro aberto) no meio horizontalmente e com um espaçamento de 24dp do topo do layout, como se verifica na figura abaixo representada.

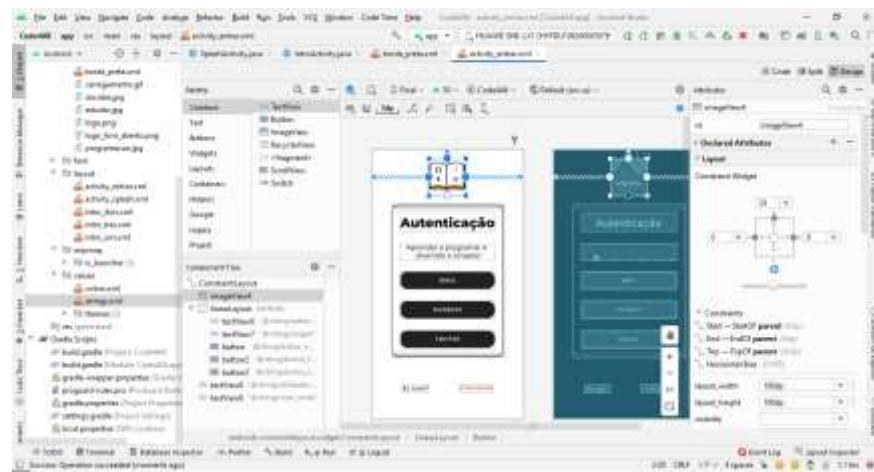


Figura 304: Alinhamento do logo (livro aberto)

Com o layout minimamente pronto faltava agora colocar os ícones de email, Facebook e Twitter, nos respetivos botões, para isso fui ao site <https://feathericons.com/> para que arranjasse uns ícones open-source, ou seja, uns ícones que pudesse utilizar sem problemas de direitos de autor. Fui logo para este site, devido a já anteriormente ter utilizado o mesmo, na construção dos layouts, na fase de desenho.

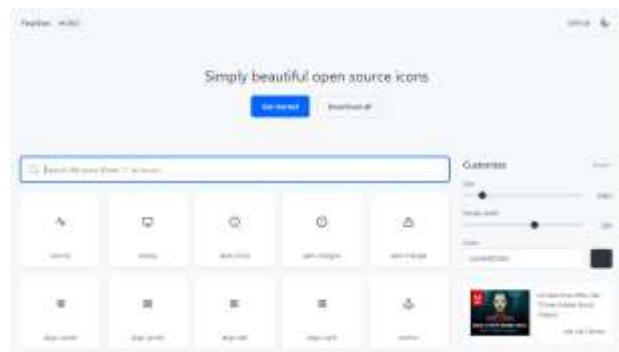


Figura 305: Site dos ícones open-source (feather-icons)

Então basicamente o que realizei, fui procurar na barra de pesquisa os ícones que precisava, fazendo o download deles para o meu computador, ficando um total de três ícones.

Uma informação ainda que pretendo realçar é que estes três ficheiros tem a extensão .svg. Esta é um tipo de extensão bastante semelhante ao .png, bastante famoso, uma das principais diferenças é que um ficheiro .svg, contém todo o tipo de informação que a imagem pode ter, tornando possível editar a foto com mais facilidade. Na figura representada, apresento os ícones que transferi para o meu computador.

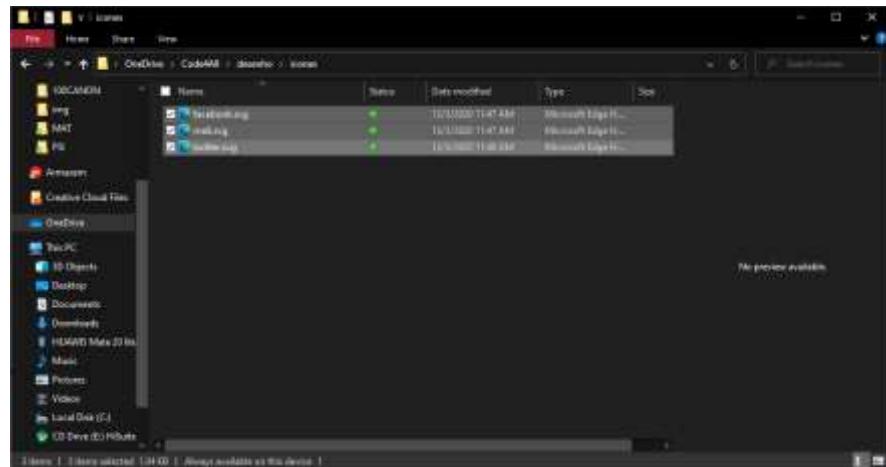


Figura 306: Pasta com os 3 ícones, email, facebook e Twitter

Com os ficheiros no meu computador, o procedimento a seguir era fácil, era copiar os mesmos e colar na pasta “res/drawable” do meu projeto.

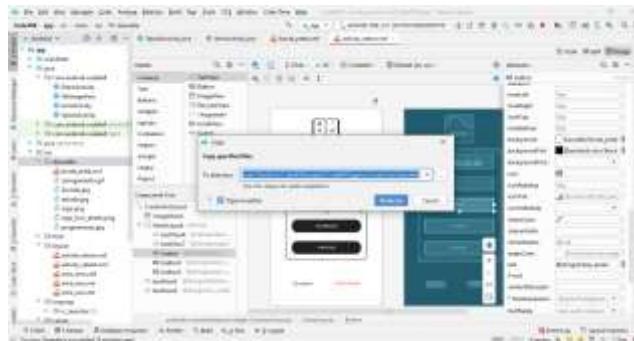


Figura 307: Importação dos três ficheiros .svg para dentro do meu projeto

Na respetiva figura, mostro que os três ficheiros .svg realmente foram importados.

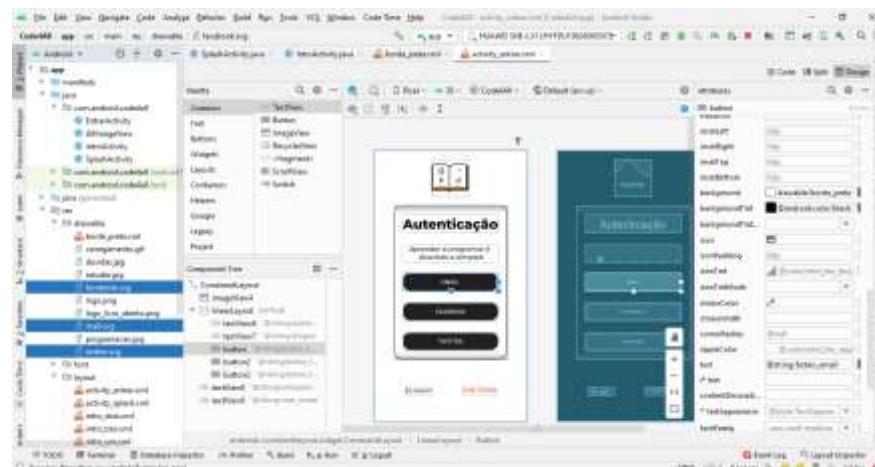


Figura 308: Visualização da correta importação dos três ficheiros .svg

Mas logo deparei-me com um problema, visto que o ImageView não aceita ficheiros .svg e o mesmo não estava a reconhecer os ícones.

Decidi fazer uma mini-pesquisa, para comprovar isso e de facto era verdade, a ImageView não aceitava o ficheiro .svg naquele formato. Visto que a mesma não aceitava, fui obrigado a apagar os três ícones que tinha importado.

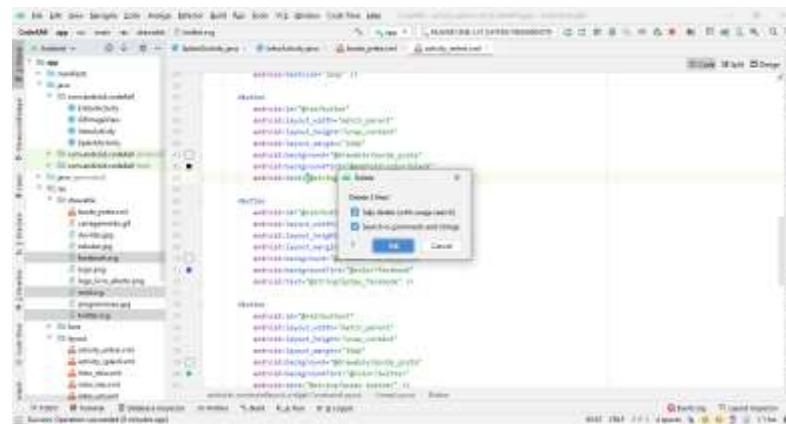


Figura 309: Remoção dos três ícones anteriormente importados

Como primeira solução, veio-me à cabeça converter o ficheiro svg que tinha anteriormente transferido e passar o mesmo para png. E foi o que fiz, escrevi no google “svg to png” e de imediato encontrei este site <https://svgtopng.com/pt/>. Seguidamente, dei “upload” dos meus ficheiros em svg e de logo foram convertidos para png.



Figura 310: Site usado para converter SVG para PNG

Após ter transferido as imagens convertidas, coloquei as mesmas dentro da pasta do meu projeto, juntamente com os outros ficheiros SVG.

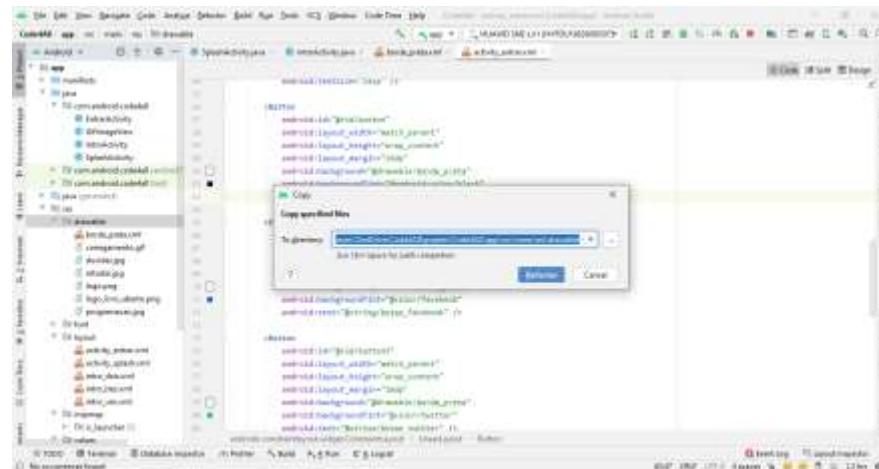


Figura 311: Importação das três imagens já convertidas para PNG

Logo ap s ter importado as imagens em PNG, decidi aumentar o padding dos bot es para 16dp, como se observa na imagem, da p gina seguinte.

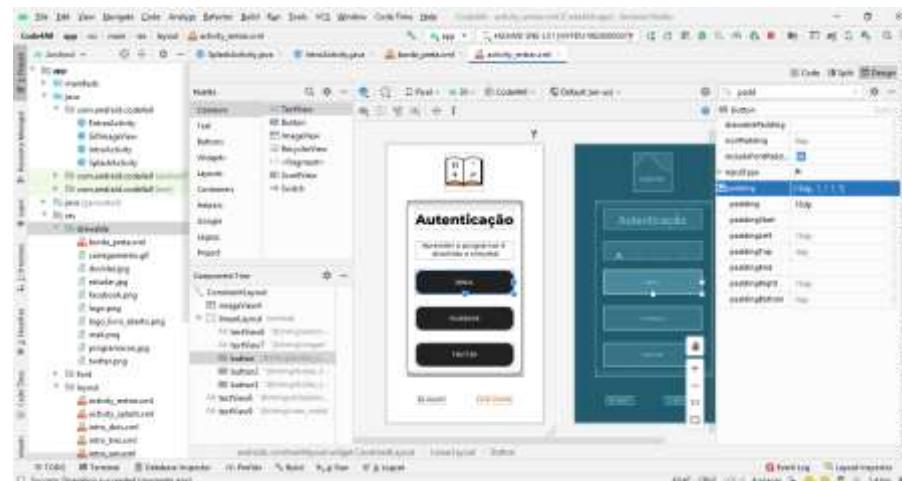


Figura 312: Aumento no "padding" dos três botões

Seguidamente, passei à inclusão dos “ícones” dentro dos botões, comecei por pesquisar pelo o atributo “drawable” visto que é o atributo que me permite colocar ícones dentro de um botão. Com esse atributo, bastou apenas atribuir à respetiva imagem, começando pelo o email, seguindo então para o Facebook e por fim o botão do Twitter, configurando assim todos da mesma forma, com ícones diferentes.

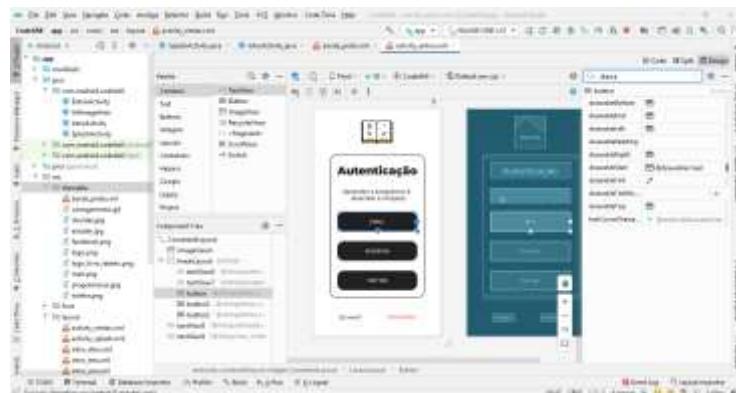


Figura 313: Configuração dos ícones dentro dos botões

Após esta configuração dos ícones, decidi executar a app, para verificar o seu comportamento. Logo verifiquei que a qualidade dos ícones estava péssima e dava para se notar os pixéis de cada um, pelo menos para mim, e como quero fazer um projeto perfeito, decidi investigar o que poderia fazer para melhorar a qualidade dos mesmos.



Figura 314: Execução da atividade, para visualizar os botões e os seus ícones

Eu, deparando-me com esta má qualidade dos ícones, sem saber o que fazer, fiquei em pânico, dirigi-me logo ao meu coordenador de curso para lhe perguntar o que poderia fazer para fazer com que os ícones não perdessem nenhuma qualidade.

O meu coordenador de curso respondeu dizendo que poderia colocar a imagem “PNG” dentro do Photoshop e aumentar o seu dpi, para mais ou menos 120, eu para segurar a qualidade, decidi colocar 150dpi.

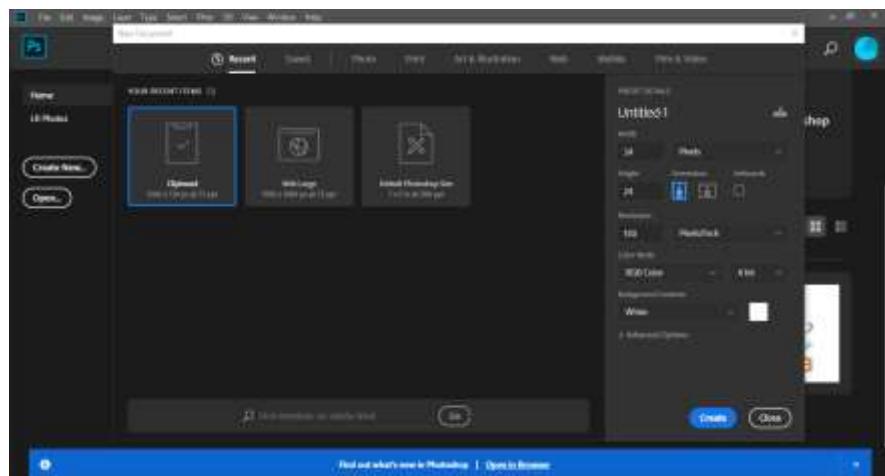


Figura 315: Abertura do Photoshop para aumentar o dpi dos ícones

Com o Photoshop previamente aberto e com as configurações devidamente realizadas, com 24 pixéis de largura e altura, visto que são as medidas que normalmente um ícone possui e ainda os 150dpi fundamentais para melhorar a qualidade do ícone, o objetivo principal da realização desta atividade no Photoshop. Com a abertura do projeto, bastava copiar e colar o ficheiro PNG anteriormente transferido, como se observa na figura da página seguinte.

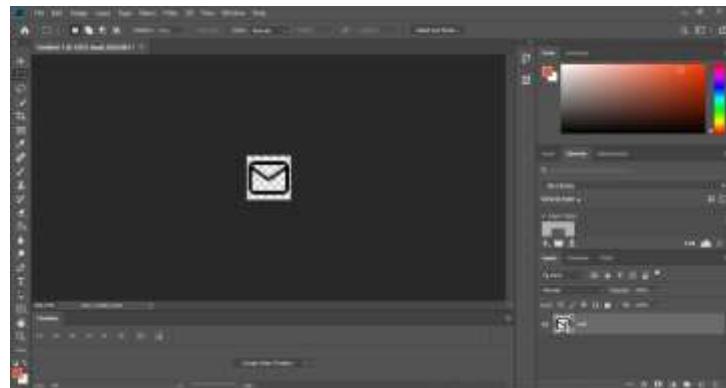


Figura 316: Abertura de um projeto com umas determinadas configurações

Com o ícone em PNG inserido na camada principal, passei à exportação do ícone, novamente com a mesma extensão (PNG). Basicamente, ao fazer esta operação, o Photoshop irá alterar os dpi's do ícone para 150, melhorando assim a qualidade do mesmo visto que quanto maior for melhor será a sua visualização

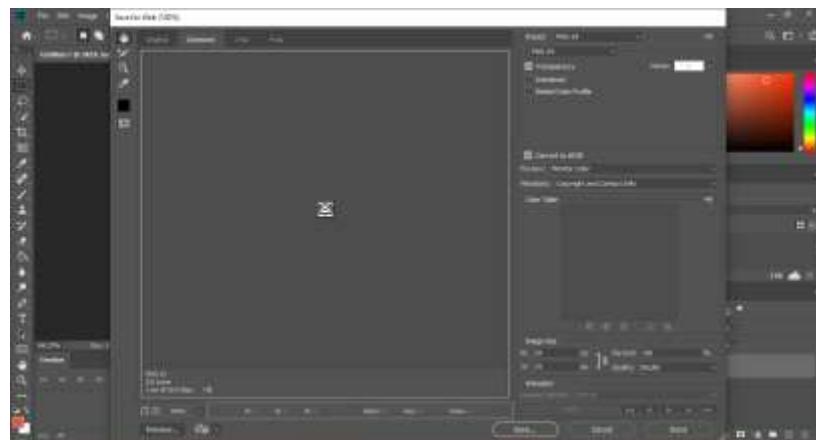


Figura 317: Exportação do ícone com um tamanho de 24x24 e 150dpi

Com a qualidade do ícone do email melhorada para 150dpi, o que tive de fazer, foi aplicar os mesmos passos realizados anteriormente no Photoshop, para melhorar a qualidade dos outros dois ícones (Facebook e Twitter).

Com essas, melhorias feitas e os ícones exportados novamente para PNG, coloquei os mesmos de novo para a pasta do meu projeto “desenho/icones” substituindo assim os “antigos” que lá estavam.



Figura 318: Substituição dos ícones de má qualidade (PNG)

Com a substituição dos ícones em PNG, dentro da pasta “desenho/icones”, como realizado e visto anteriormente, decidi selecionar os mesmos e importar para dentro do meu projeto no Android Studio, como se verifica na imagem de baixo, uma informação a dizer que o ficheiro que pretendo importar já existe e para todos os três ícones cliquei em “Overwrite”, para substituir os anteriores que lá estavam.

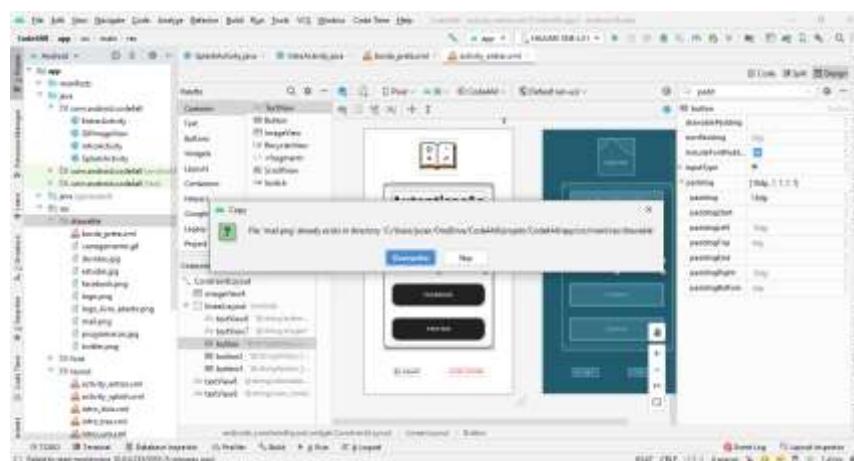


Figura 319: Substituição dos ícones com má qualidade, pelos os ícones com 150dpi

Seguidamente, executei a aplicação no meu dispositivo Android para observar se houve alguma diferença, com este melhoramento nas imagens PNG para 150dpi. Eu realmente notei uma diferença na qualidade, mas para mim ainda não estava suficientemente bom e tive de procurar outra solução.



Figura 320: Captura de ecrã com os novos ícones melhorados a 150dpi

Com este “problema” da má qualidade nas imagens, decidi então explorar mais um pouco, porque eu sabia que daria para utilizar imagens .svg, visto que o próprio Android Studio disponibiliza um “pack” de ícones e esses usam a extensão SVG.

Realizei uma pequena pesquisa no meu fórum favorito e acabei por encontrar um site que me deu a devida solução.

Para colocar ficheiros .svg para dentro do meu projeto, eu só não estava a fazer a correta abordagem para importar os mesmos, como se verifica na imagem abaixo.

<https://stackoverflow.com/questions/34990236/how-to-use-svg-image-in-imageview/>



Figura 321: Site com a explicação de como importar ficheiros .svg para um projeto no Android Studio

Com informação que li anteriormente, fui de imediato apagar os outros ficheiros PNG que não precisava, dentro da pasta “desenho/icones”.

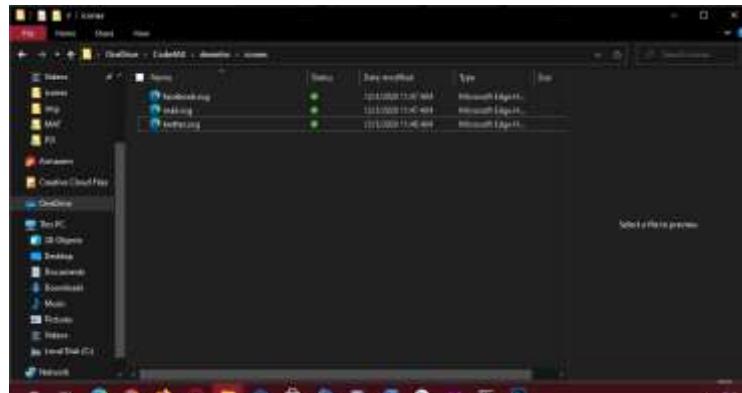


Figura 322: Remoção dos ficheiros PNG dentro da pasta "desenho/icones"

Com os ficheiros PNG removidos do meu “dossier”, dossier este, onde guardo tudo sobre a minha Prova de Aptidão Profissional, desde imagens a simples blocos de notas com pequenas anotações.

De certa forma, o meu próximo passo, foi aplicar o que a solução me apresentava, que era clicar com o botão direito com a pasta “drawable” selecionada, seguidamente ir em “New” e por fim clicar em “Vector Asset”, como se pode visualizar na figura abaixo.

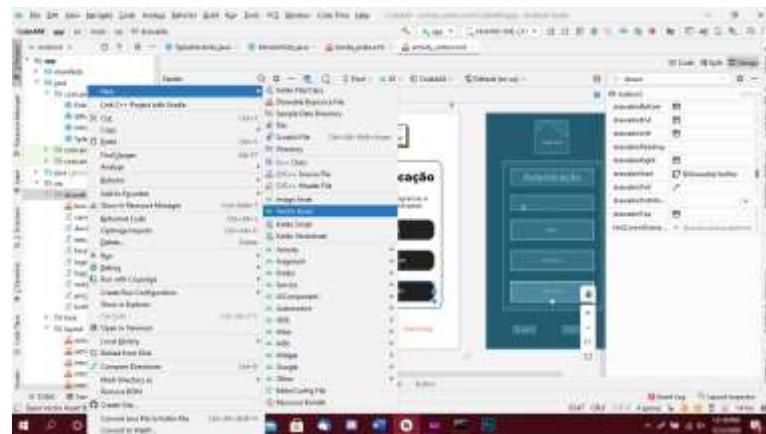


Figura 323: Criação de um novo "Vector Asset" para o projeto no Android Studio

Logo de imediato, foi-me aberta a seguinte janela. Janela esta, que já falei anteriormente, é aqui onde poderei eventualmente adicionar uns ícones que o próprio Android Studio me proporciona, segundo a licença “Apache License Version 2.0”.

<http://www.apache.org/licenses/LICENSE-2.0.txt>

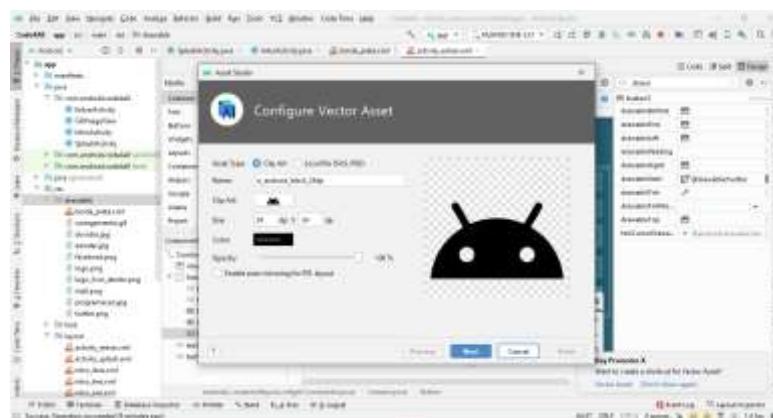


Figura 324: Configuração de um novo "Vector Asset"

Mas, segundo a solução apresentada no stackoverflow, poderia adicionar um novo “Vector Asset”, com uma imagem SVG ou PSD, tudo aquilo que eu precisava... Primeiramente para testar, fiz este processo com o ícone do email, bastando apenas colocar um nome (ic_email), o caminho onde está o arquivo SVG anteriormente transferido para o meu computador e foi só clicar em “Next” e está feito.



Figura 325: Importação do ficheiro SVG para dentro do projeto no Android Studio

Prontamente, configurei novamente o atributo “drawable” do botão email, para “ic_email” o ficheiro anteriormente criado, com base no ficheiro SVG.

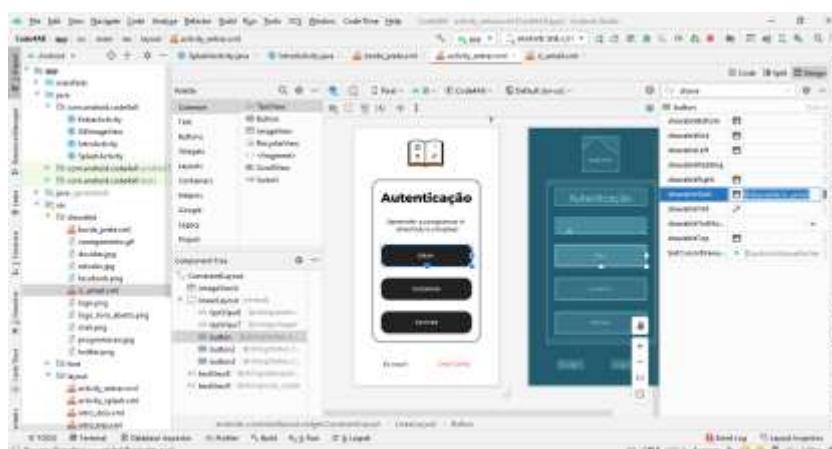


Figura 326: Configuração do atributo "drawable" do botão email

Eu todo contente, de ter importado o ficheiro SVG corretamente e estar tudo a dar sem erros (aparentemente), mas esta felicidade durou pouco tempo, visto que ao executar a aplicação, ocorreu-me o seguinte erro:

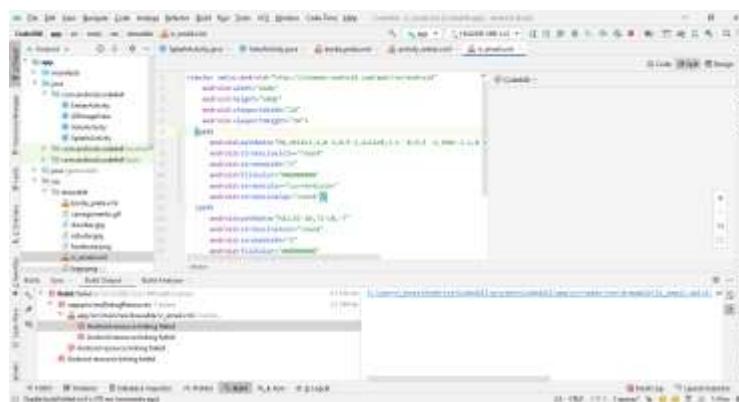


Figura 327: Captura do ecrã do erro que ocorreu ao tentar executar a aplicação

Normalmente, quando tenho um erro em algum ficheiro de XML, a primeira reação que tenho é fazer uma reconstrução do projeto, pois poderá eventualmente ser um ficheiro que ainda não tenha sido reconhecido. Este “Rebuild Project” faz uma “varredura” nos ficheiros todos, para que se algo que não esteja de facto presente, seja com esta “varredura” encontrado e posicionado. Para reconstruir o projeto, basta ir à barra superior onde diz “Build” e seguidamente clicar “Rebuild Project”.

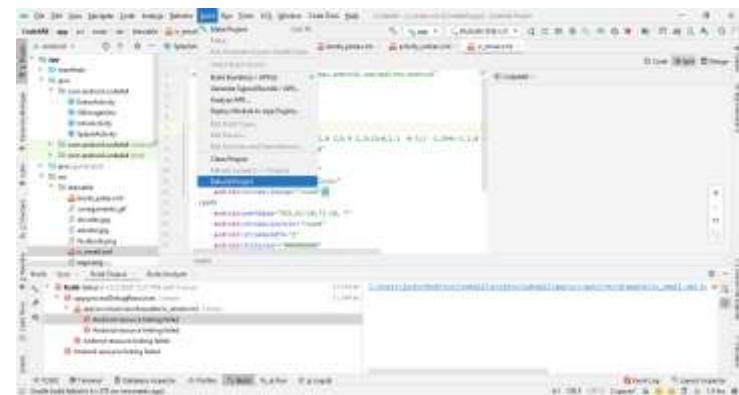


Figura 328: Realização de uma reconstrução do projeto

Após ter realizado o “Rebuild Project”, executei mais uma vez a aplicação e a mesma deu erro novamente, ou seja, o problema não estava em nenhum ficheiro que o Android Studio não tivesse listado corretamente.

Decidi então prestar mais atenção ao erro e realmente vi que o problema estava no “ic_email.xml”, o ficheiro que tinha sido anteriormente criado com base no SVG do email, já transferido anteriormente.

Para solucionar este problema, tinha que, modificar o atributo “android:strokeColor”, para uma cor já conhecida ou interpretada pelo o Android Studio, visto que a minha aplicação não estava a executar pois não encontrava nenhuma cor chamada “currentColor”, visto que não tinha feito essa configuração. Decidi então, definir o atributo anteriormente falado com o valor “@android:color/white”, tornando assim o ícone email totalmente branco.

Este valor currentColor, veio de quando realizei download dos ícones, visto que como se representa na figura, teria a hipótese de mudar ou deixar a cor definida por padrão, como não realizei nenhuma alteração, a cor com que eu transferi os ícones permaneceu a “currentColor”. E foi devido a este pequeno erro meu que a aplicação não executava, isto é um dos exemplos pelos quais me fui apercebendo que o meu projeto será bastante difícil, porque ao existir o mínimo erro que seja, a aplicação simplesmente não executa e terei de ir atrás do “erro” e procurar a sua solução.



Figura 329: Modificação do valor "currentColor" para "@android:color/white"

Com esta pequena modificação, dentro do ficheiro “ic_email.xml”, passei à execução da aplicação e desta vez a mesma ocorreu sem erros. E, como se vê na figura abaixo o ícone foi totalmente mudado, não tive perda de qualidade nenhuma e como fiz anteriormente, a cor do mesmo está branca.



Figura 330: Visualização do ícone dentro do botão de email

Após esta execução da aplicação, eu olhei para o layout em si e vi que o TextView “És novo?”, não estaria bem adequado, visto que a linguagem utilizada não estava a ser da mais “educada”, sendo assim, decidi perguntar ao meu coordenador de curso, por um termo mais “correto” e “educado” o meu coordenador disse para eu colocar “Entrou agora?” e foi o que de imediato realizei, como se observa na figura abaixo.



Figura 331: Modificação do recurso String "utilizador_novo"

Com a modificação do recurso String, para um nome mais “correto”, passei ao alinhamento dos dois TextViews, com uma “Horizontal Chain”, podendo esta ser feita com apenas 2 ou mais componentes e que me permite colocar os componentes alinhados ao centro (horizontalmente), neste caso, no LinearLayout. Verticalmente, realizei o alinhamento, da mesma forma em ambos, alinhando superiormente na parte de baixo do LinearLayout e alinhando inferiormente os mesmos na parte de baixo do layout em si.

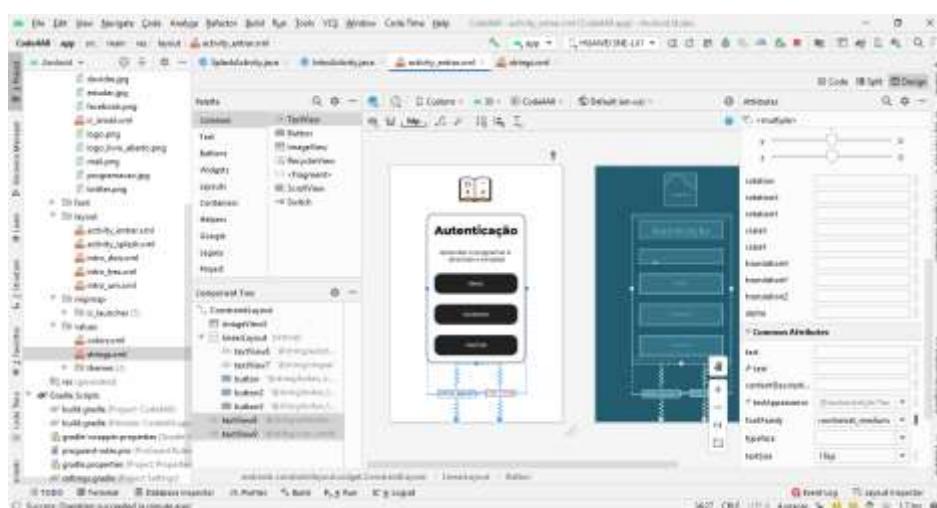


Figura 332: Alinhamento dos TextView's localizados na parte inferior do layout

Com este alinhamento, por questões de segurança, decidi executar a aplicação para ver se estava tudo ordem até ao momento, o resultado encontra-se na figura abaixo. Nesta parte, o layout já estava a ganhar a sua forma.



Figura 333: Captura de ecrã do layout nesta parte do desenvolvimento

De certa forma, esta execução da aplicação fez me ver que ainda tinha que realizar a importação de dois ficheiros SVG, sendo eles do Facebook e Twitter. Começando pelo ficheiro do Facebook, configurando o mesmo da seguinte forma.

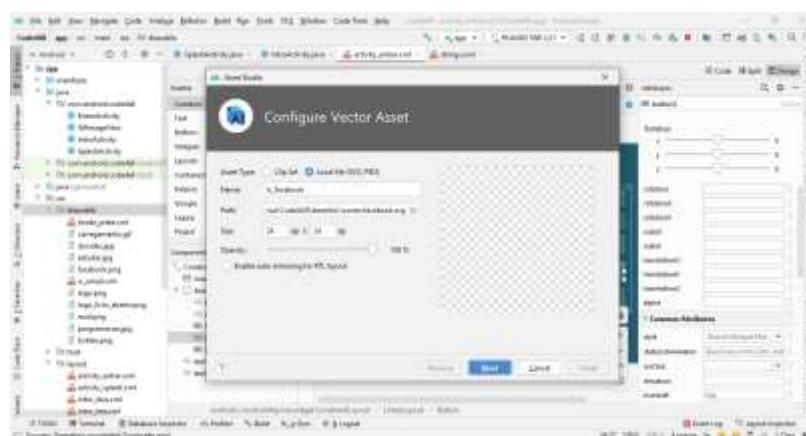


Figura 334: Configuração de um "Vector Asset" para o Facebook

Com a importação de mais um ficheiro SVG, para dentro do projeto, foi necessário refazer o que fiz anteriormente, mudar a cor “currentColor” para a “@android:color/white”, para mudar o ícone, neste caso do Facebook para branco.

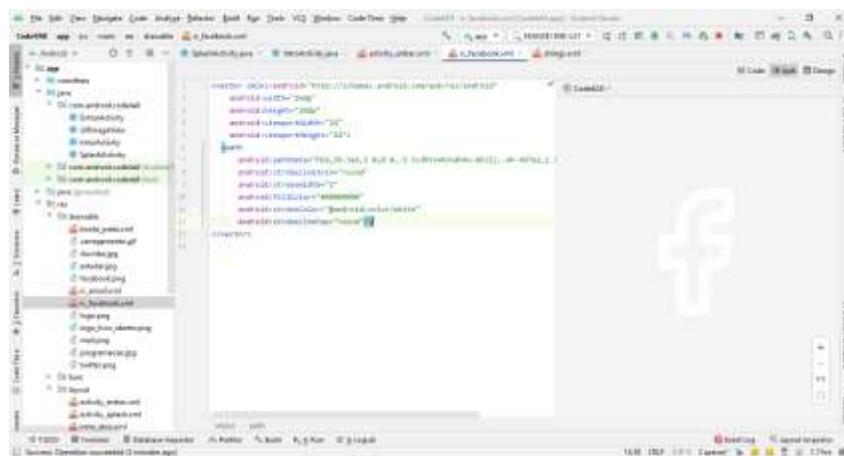


Figura 335: Correção na cor do ícone do Facebook

Visto que já estava com as “mãos na massa”, decidi logo criar também importar o ficheiro SVG do Twitter, para facilitar mais o meu trabalho e colocar mais uma coisa de parte, as suas respetivas configurações encontram-se na figura abaixo.

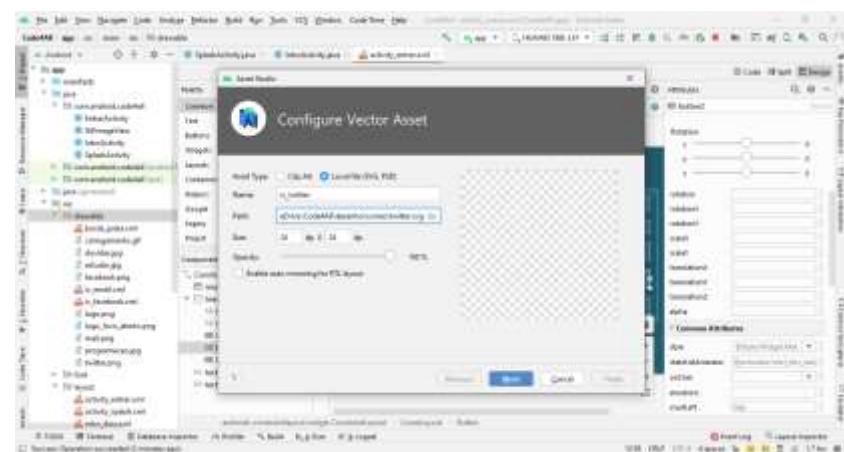


Figura 336: Configuração de um "Vector Asset" para o Twitter

Mais uma vez, passei à modificação da cor do próprio ícone, alterando o mesmo para branco, como já explicado anteriormente.



Figura 337: Correção na cor do ícone do Twitter

Como próximo passo, passei à configuração do atributo “drawable” no botão do Facebook, e no mesmo atribui o ficheiro anteriormente criado e corrigido “ic_facebook”, como se apresenta na figura abaixo.

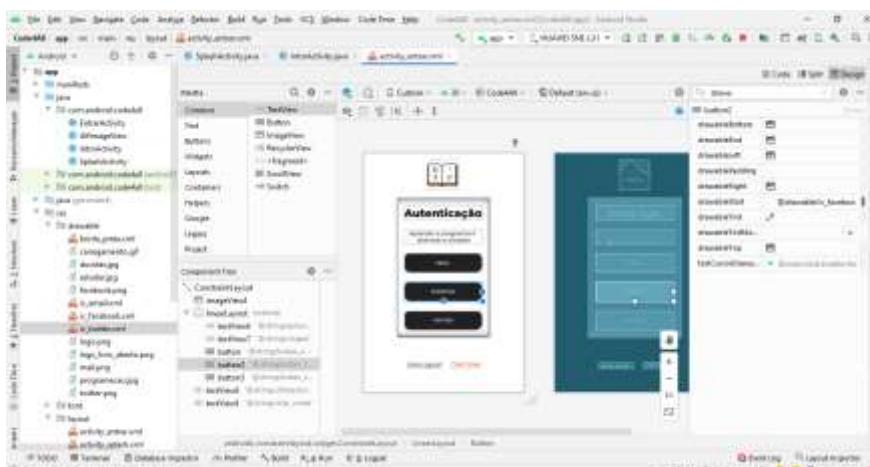


Figura 338: Atribuição do ícone "ic_facebook"

Como realizei no passo anterior, neste não foi exceção, desta vez configurei então o ícone (ic_twitter) para o botão do Twitter, como se pode visualizar na imagem.

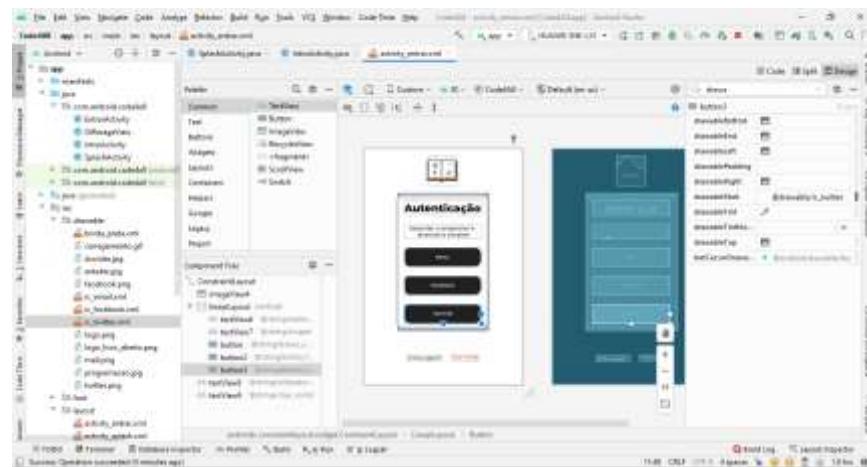


Figura 339: Atribuição do ícone "ic_twitter"

Com a configuração dos botões e dos ícones acabada, passei a eliminar ficheiros que não iria utilizar mais, três deles foram os ícones PNG que importei antes e que tinham uma má qualidade.

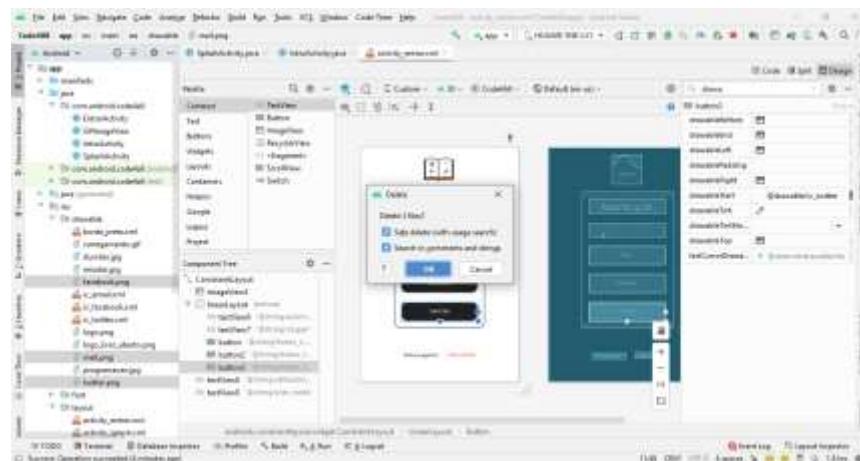


Figura 340: Remoção dos três ícones (PNG)

Para ajustar o layout, decidi colocar um “padding” de 8dp no LinearLayout, visto que ficava mais interessante a colocação do mesmo.

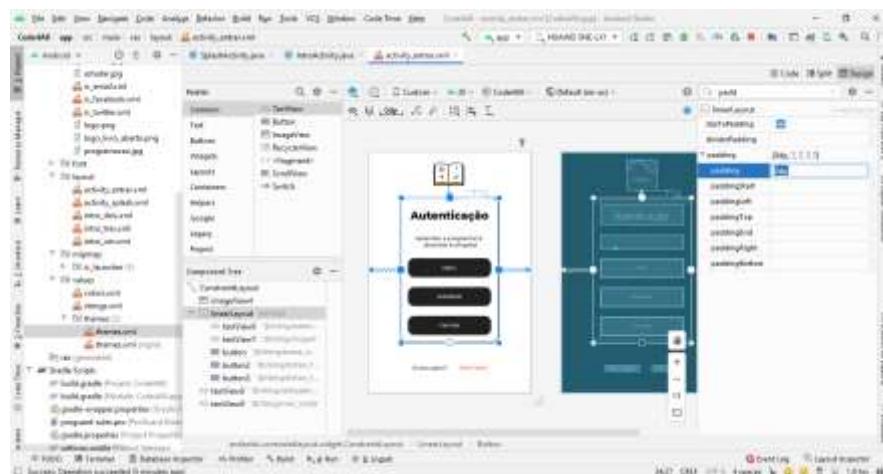


Figura 341: Colocação de um "padding" de 8dp no LinearLayout

Visto que estava a realizar configurações de aspeto no LinearLayout, aproveitei e alterei o “padding” de 16dp para 8dp, dentro do TextView onde diz “Autenticação”.

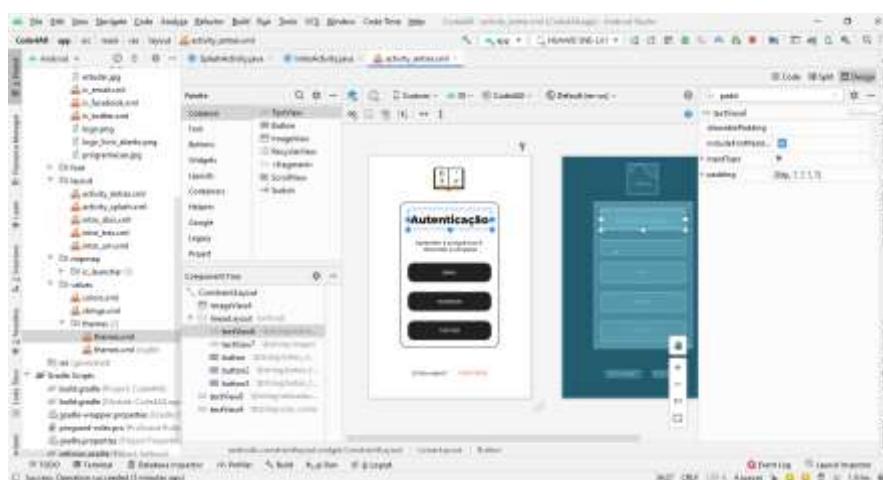


Figura 342: Alteração do "padding" dentro da "Autenticação"

Com as anteriores configurações, alinhamentos, “padding’s” decidi executar a atividade para ver se estava tudo a funcionar como devia e assim de certa forma passar para a próxima fase do desenvolvimento.

Neste caso, estava tudo a funcionar nas suas perfeitas condições, como se observa na figura.



Figura 343: Captura do ecrã de como se encontrava o layout

Seguidamente, mostrei o resultado do layout ao meu coordenador de curso, nesta parte do desenvolvimento, o mesmo disse que aqueles “três pontos” ficavam bem nos três layouts introdutórios, mas não fariam muito sentido no último layout, visto que não seria permitido ao utilizador nem andar para trás nem para frente, logo ele sugeriu me que os mesmos desaparecessem.

Para isso, foi fácil, apenas tive de colocar um ouvinte “addOnPageChangeListener” para quando a página for mudada e a sua posição for igual a três, os “três pontos” desapareçam com o código “setPagerIndicatorVisible(false)”. Coloquei o número três, visto que em programação começasse a contar do 0, logo o número três seria o último layout.



Figura 344: Colocação de um ouvinte na "IntroActivity"

Com a colocação deste ouvinte e fazendo desaparecer com os “três pontos”, teria de verificar se o código estava a funcionar conforme o que queria e neste caso, como se verifica na figura, o código anteriormente escrito funcionou às mil maravilhas.



Figura 345: Execução da aplicação para verificar o código escrito

Por motivos de organização, decidi mudar o ID do TextView, onde diz “Criar Conta”, para “textoCriarConta”, decidi fazer esta alteração pois iria utilizar o componente e quando utilizo um componente é sempre bom dar-lhe um nome simples e fácil.

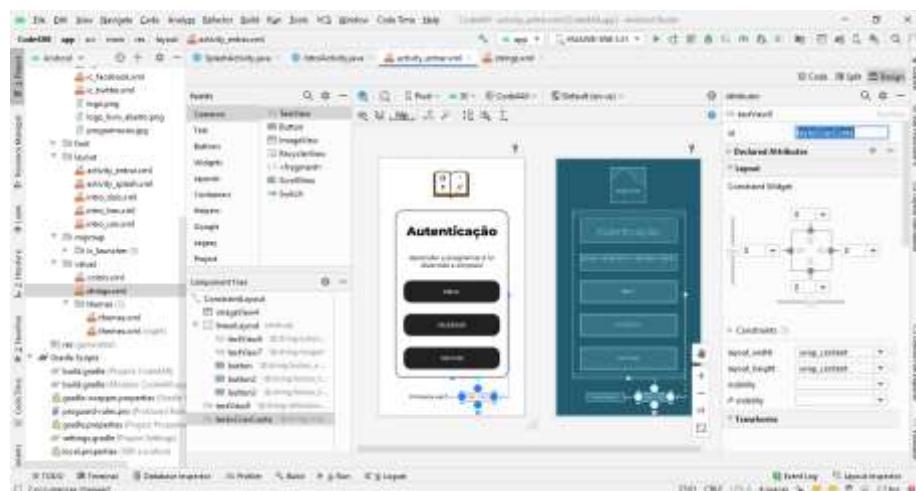


Figura 346: Modificação do ID do TextView para "textoCriarConta"

Seguidamente, tive de procurar uma solução, pois nunca tinha estado nesta situação antes, onde teria uma atividade que teria outros layouts, pois normalmente, uma atividade tem um respetivo layout e apenas isso. Nessa atividade é feita depois toda a configuração dos componentes que se encontram no próprio layout.

Mas neste caso, é diferente, pois tenho uma atividade com vários layouts. A meu ver tinha duas soluções, uma de colocar um atributo “onClick” dentro do componente e no mesmo colocar um método onde será executado código quando o mesmo for pressionado ou ainda tenho a opção de criar uma variável e ligar à mesma com o ID do componente, neste caso poderia me surgir alguns problemas, pois teria de saber quando é que o layout era carregado.

Achei mais fácil e rápido então criar logo um método em que a única coisa que tinha de fazer é adicionar um atributo “onClick” dentro do textCriarConta e colocar o método onde está o código a ser executado pela ação de clique no texto.

Continuando, criei um método público na “IntroActivity” chamado de “criarConta”, este método tem ainda que colocar como parâmetro uma View, pois se não for passada o método não será encontrado pelo o componente e ocorrerá um “Crash” na aplicação.

Dentro deste método, criei uma “Intent” que me levará para a “LoginActivity” que será a próxima atividade a ser criada.

Obs: a parte do código onde diz “LoginActivity.class”, não me está a dar erro, pois criei a atividade mais cedo para realizar uns testes no projeto, mas normalmente iria ficar vermelha visto que ainda não teria sido criada.

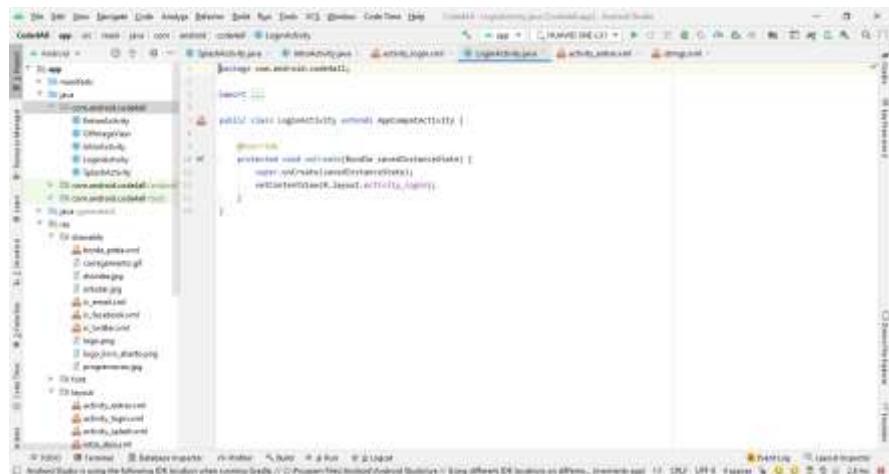


Figura 347: Criação do método "criarConta" dentro da "IntroActivity"

Já com o método criado e sem “erros”, basta adicionar o atributo “onClick” dentro do componente, neste caso do botão, dentro desse atributo é colocado um método que será executado quando o mesmo for pressionado, coloquei então “entrarConta”.

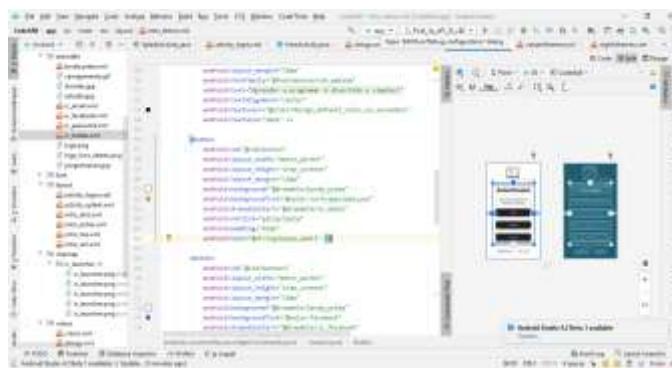


Figura 348: Colocação do atributo "onClick" com o método "entrarConta"

Após pensar bem no assunto, vi que não era preciso eu ter criado uma atividade de propósito, bastava apenas um layout, pois conseguia criar os métodos e tudo o que precisava dentro da IntroActivity, por isso é que coloquei nesta atividade (Interligado com a IntroActivity), vendo as coisas desta maneira decidi então eliminar a atividade devido a não estar a ser utilizada.

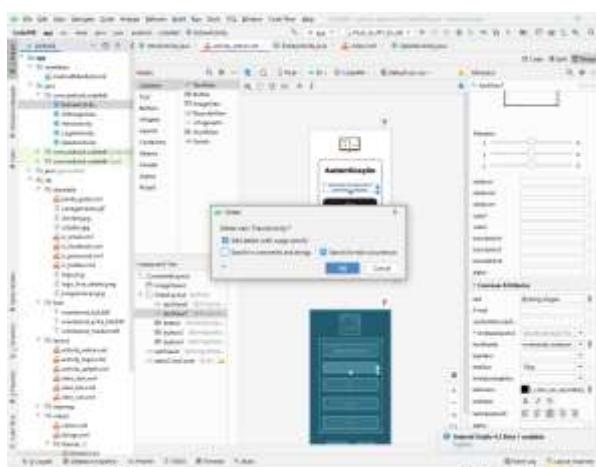


Figura 349: Remoção da atividade "EntrarActivity"

Por fim, decidi mudar o nome do ficheiro de “activity_entrar” para “intro_entrar”, devido ao simples facto de não ter sentido nenhum continuar a ter “activity” visto que a mesma foi anteriormente apagada e fazer sim sentido ter o “intro”, pois a mesma faz parte da introdução à aplicação.

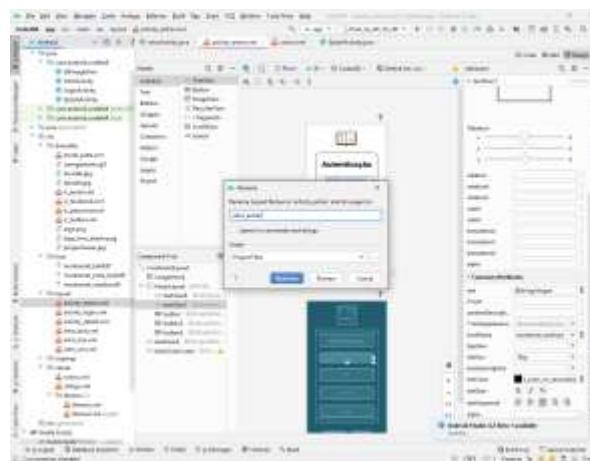


Figura 350: Modificação do nome do respetivo layout

Com o método “entrarConta()” finalizado, tive de criar também outro método para executar a função de levar o utilizador para a CriarContaActivity.

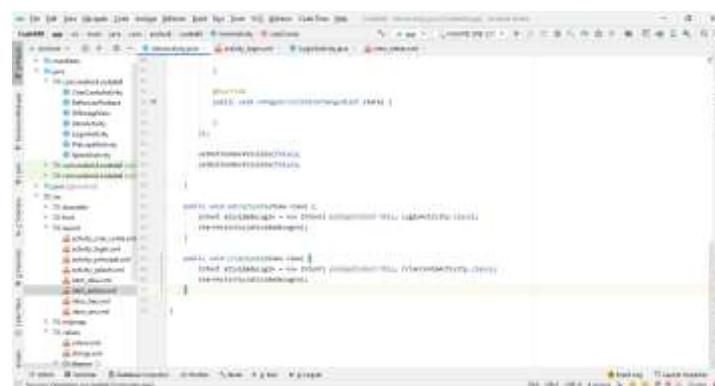


Figura 351: Criação do método criarConta()

De seguida, foi preciso adicionar o atributo “onClick” dentro do TextView “Criar conta”, com o método anteriormente realizado em Java.

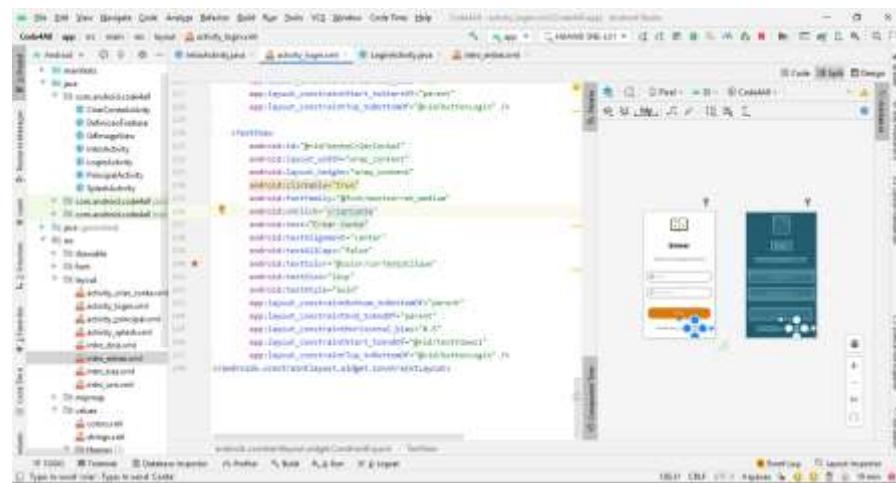


Figura 352: Atribuição do método "criarConta()" dentro do TextView "Criar conta"

Devido ao facto de ter sentido um “ligeiro” atraso ao esconder o indicador de páginas de introdução, decidi desativar o mesmo, tal como se observa na captura de ecrã abaixo. Com o código “setPageIndicatorVisible(false)” isso foi possível.

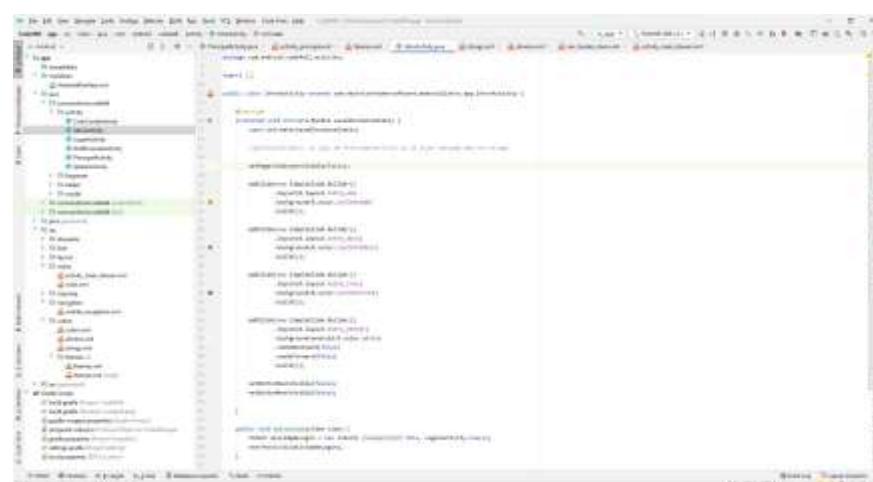


Figura 353: Desativação total do indicador de páginas de introdução

Para dar um efeito aos botões de início de login, decidi pesquisar sobre o “*ripple effect*”, onde encontrei um vídeo que me explicava como o configurar num ficheiro *drawable*.

<https://www.youtube.com/watch?v=Pm0A6FnFzaA/>

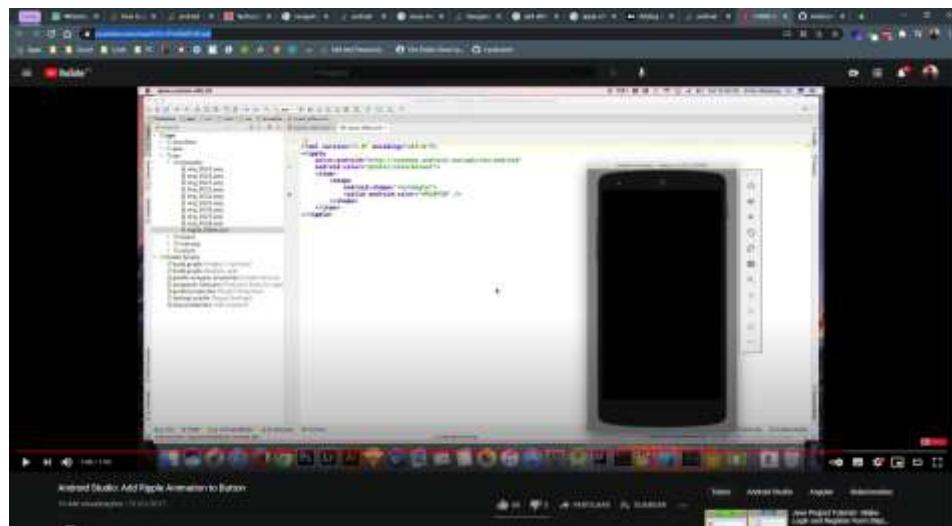


Figura 354: Print do vídeo que visualizei para configurar o "ripple effect"

Criei o ficheiro “*borda_preta.xml*” e configurei-o da seguinte forma.

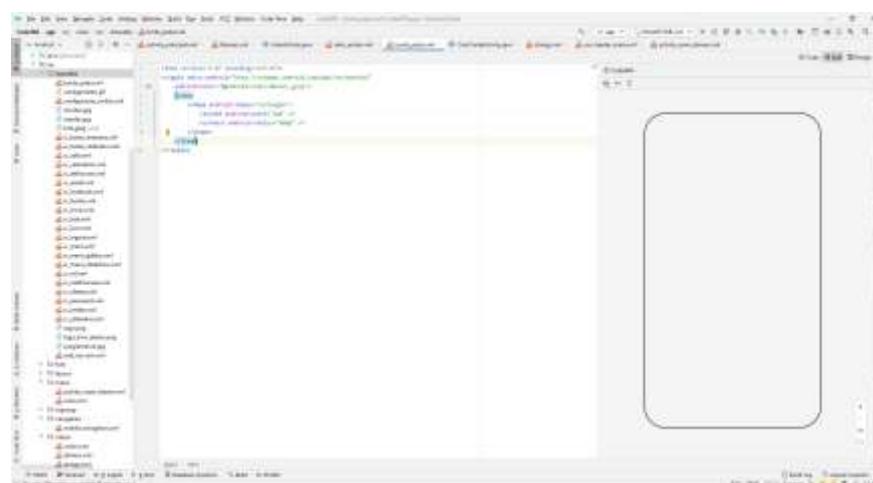


Figura 355: Criação do ficheiro "borda_preta.xml"

Com o ficheiro anteriormente criado, bastou apenas definir o atributo “background” com esse.

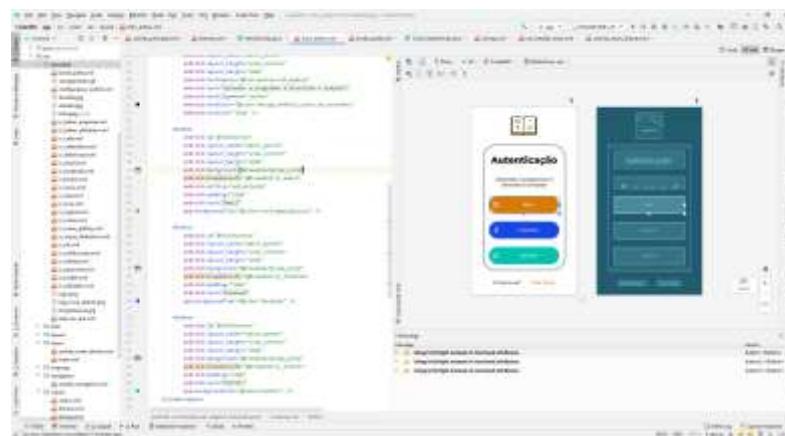


Figura 356: Definição do atributo "background" com o ficheiro "borda_preta"

Com isto, percebi que tinha um problema na atividade, visto que quando clicava muitas vezes num botão, várias funções eram executadas, ou seja, várias atividades eram lançadas, perdendo bastante desempenho na app. Como solução encontrei mais um post, no *stackoverflow*.

<https://stackoverflow.com/questions/8077728/how-to-prevent-the-activity-from-loading-twice-on-pressing-the-button/>



Figura 357: Solução de prevenir uma atividade de lançar mais que uma vez

Adicionei em cada atividade o código “`android:launchMode`” com a “`singleInstance`”, que em português quer dizer “única instância”.

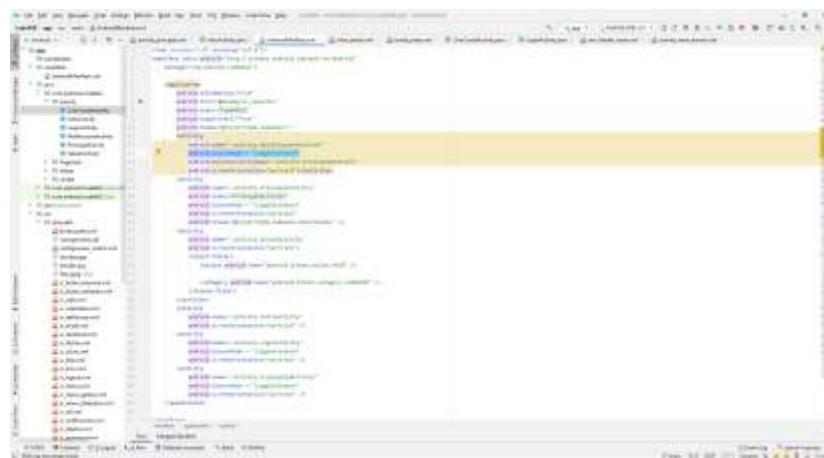


Figura 358: Definição do atributo "launchMode" com "singleInstance"

Mesmo assim, este código não me solucionou o meu problema, logo tive que pesquisar mais um pouco, onde encontrei a solução que incluía uma tag no antes de lançar a atividade.

<https://stackoverflow.com/questions/8906456/prevent-opening-activity-for-multiple-times/>

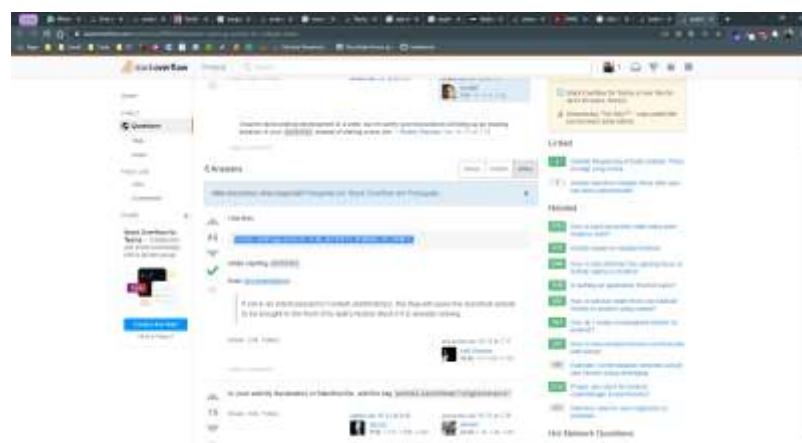


Figura 359: Solução de lançar apenas uma atividade

No método “onClick”, decidi adicionar a tal flag, que me executaria apenas uma atividade, tal como se observa na figura abaixo.



Figura 360: Adição da flag, "REORDER_TO_FRONT" dentro de cada uma das Intent's

Depois de ter testado a solução anterior, concluí que de facto era este o meu objetivo e com esta “flag” tudo estaria a funcionar corretamente e a atividade era lançada exclusivamente uma vez. Seguidamente, com o avanço do projeto, voltei atrás para adicionar uma funcionalidade importante, que me indicaria quantas vezes a app já foi aberta. Com essas vezes, conseguia então perceber se mostrava os layouts iniciais ou não, tal como se observa na condição abaixo.



Figura 361: Se executou a app mais que uma vez, apresenta apenas o layout de entrar, se não, apresenta todos.

Após essa configuração, reformulei mais um pouco o layout, ficando assim.

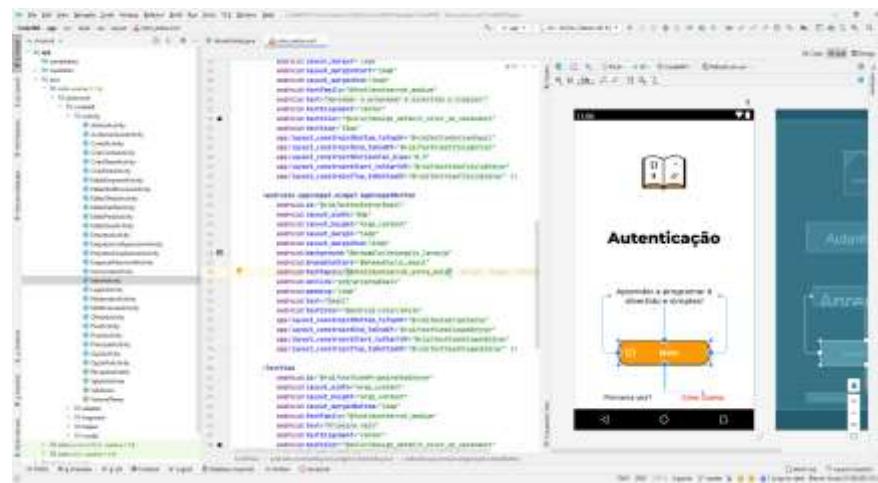


Figura 362: Reformulação no layout.

LoginActivity

Esta atividade, irá conter os componentes necessários para que seja possível a entrada de dois campos, sendo eles o email e a password.

Comecei logo por adicionar um ImageView, como fiz no layout da “atividade” anterior. As configurações da mesma, são as mesmas utilizadas anteriormente, 100dp de largura e altura, horizontalmente alinhado no meio e verticalmente alinhado ao topo do layout com um espaçamento de 24dp.

Nesta parte, pensei em criar um ficheiro xml à parte com as configurações do logo (livro aberto), pois vou usar o mesmo em algumas atividades, mas decidi deixar manualmente e quando tiver a realizar a manutenção do código, fazer sim essa alteração.

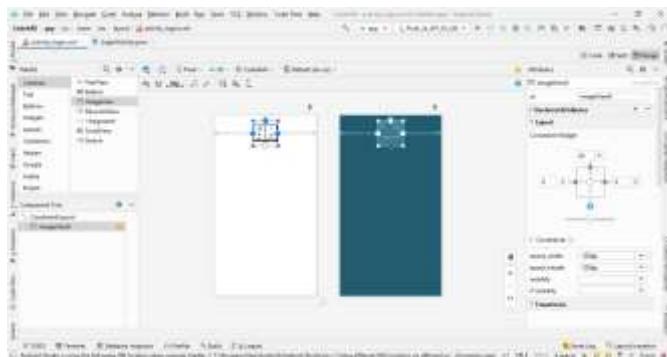


Figura 363: Colocação de uma ImageView com o logo (livro aberto)

Como visto anteriormente, não sabia como mudar a cor do botão “Entrar” visto que quando aplicava o atributo “background” para efetuar uma estilização no botão da mesma forma que apliquei nos “Plain Text’s” os mesmos trocavam logo a sua cor para preto. Mas neste caso, não é o que eu quero, queria trocar a cor de preto para a cor laranja.

Para resolver esse pequeno obstáculo, decidi realizar uma pesquisa para encontrar a solução do mesmo. A mesma foi encontrada no link que se encontra abaixo.

<https://stackoverflow.com/questions/55087014/cant-change-background-color-on-materialbutton-without-change-coloraccent>



Figura 364: Pesquisa da solução para mudar a cor do botão para laranja

Pela a pesquisa que realizei, concluí que tinha duas soluções a primeira que era adicionar um atributo do tipo “app”, basicamente a única diferença que existe entre “app” e “android” é que a app suporta as versões Android mais antigas e dá para fazer mais customizações. Eu testei com os dois, mas neste caso só com o tipo “app” é que funciona.

Esse atributo, teria ainda um “backgroundTint” que seria equivalente ao “android:background”, que estou a utilizar para aplicar um efeito arredondado, e depois bastou apenas colocar a cor de escolha, sendo ela “@color/corPrimariaEscura”. E como se pode observar na figura, a cor do botão foi mesmo mudada para laranja.

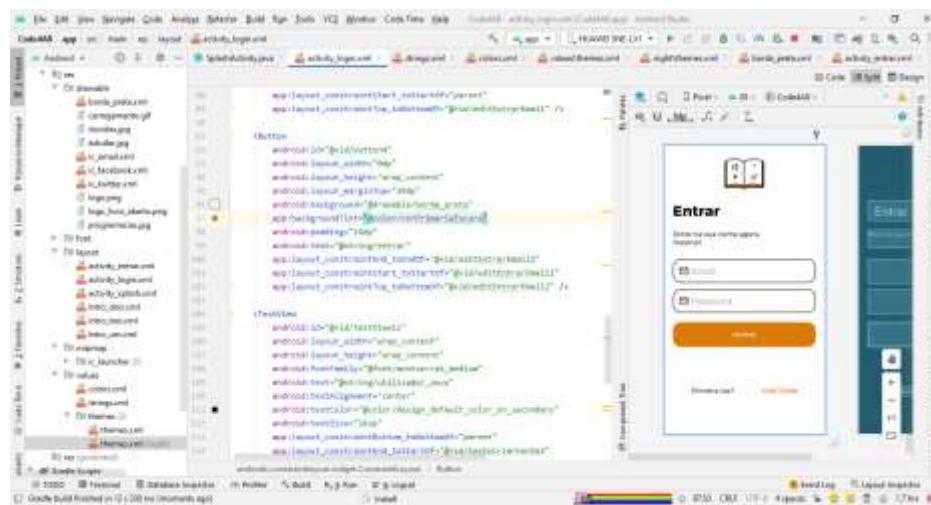


Figura 365: Definição do atributo "app:backgroundTint" com a cor primária escura

Com o ícone transferido e colocado dentro do meu dossier do projeto, realizei a importação do mesmo para dentro do meu projeto no Android Studio. E configurei da mesma forma que se representa na figura abaixo.



Figura 366: Importação do ícone (ic_password)

Como aconteceu nos outros ícones, é preciso uma configuração adicionar para que não exista nenhum problema. Para isso, é necessário mudar a cor do ícone de “currentColor”, para neste caso “@android:color/white”.

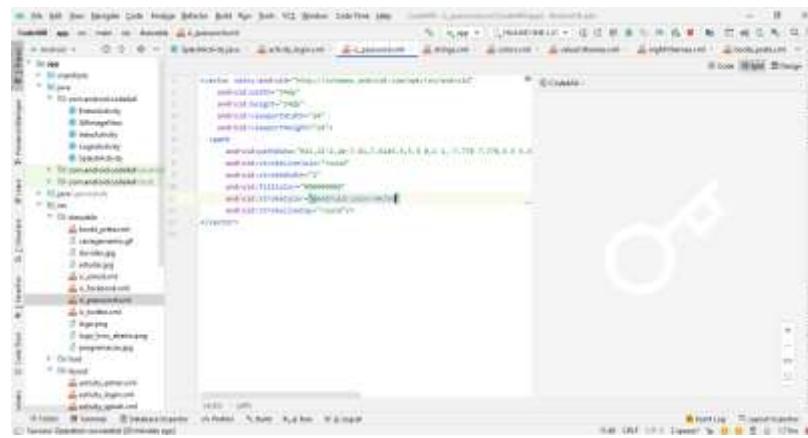


Figura 367: Modificação da cor do ícone password para branco

Com a finalização da configuração do campo de password, decidi executar a aplicação para ver o resultado até ao momento, o mesmo encontra-se na captura de ecrã indicada abaixo.



Figura 368: Captura de ecrã com a atividade em execução

Com a execução da aplicação, decidi mostrar a mesma ao meu coordenador de curso, para saber a sua opinião. Este disse que, poderiam ser feitas algumas alterações a nível de alinhamentos dos componentes e disse que existiam algumas diferenças entre o layout anterior (`intro_entrar.xml`) e o (`activity_login.xml`). Vendo assim as coisas de outra maneira, com a observação do meu professor, decidi reformular o meu layout inteiro.

Comecei por alinhar o texto “Entrar” ao centro (horizontalmente) e o texto “Entre na sua conta agora mesmo!” tendo assim os dois a mesma configuração.

Seguidamente, passei à inclusão de um `LinearLayout`, visto que iria necessitar do mesmo para depois ser mais fácil de aplicar uma “Vertical Chain” que me permite alinhar automaticamente os componentes que eu selecionar. E com o `LinearLayout` conseguia fazer essa gestão sem que os campos (`email` e `password`) e o seu espaçamento fosse afetado.

Por fim, configurei os 2 `TextView's` que se encontram na parte inferior do layout, alinhei tanto um como o outro ao botão “Entrar”, fiz este alinhamento verticalmente. Horizontalmente, apliquei uma “Horizontal Chain” para os alinhar no meio dando um efeito muito mais interessante.

Estas alterações podem ser visualizadas na captura de ecrã abaixo.

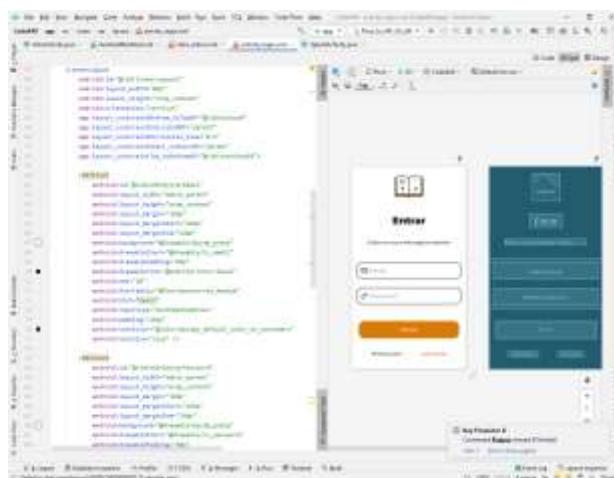


Figura 369: PrintScreen do código XML e do Layout

Com o alinhamento totalmente reformulado, faltava ainda configurar o tipo de dados que cada campo (email e password) iria levar. Para realizar essa configuração, tive que editar o próprio código em XML e adicionar o atributo “inputType” e depois escolher o tipo de dados a introduzir. No campo de email, coloquei “textEmailAddress” e no campo de password, coloquei “textPassword”. Essa configuração pode ser observada nas duas figuras abaixo.

Alterei ainda o ID dos campos para “editLoginEmail” e “editLoginPassword” visto que faria mais sentido pois estamos na atividade de Login.

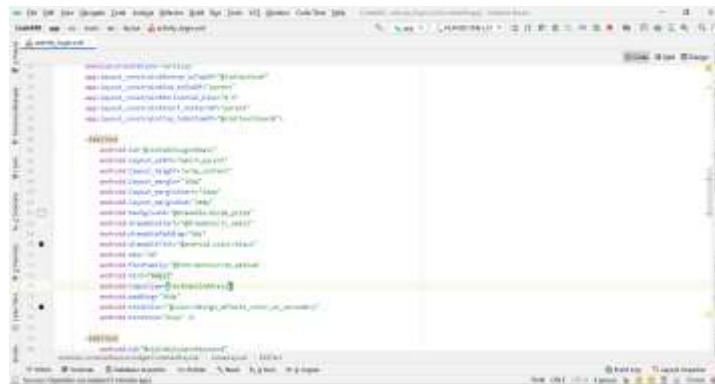


Figura 370: Configuração do tipo de dados do campo de Email

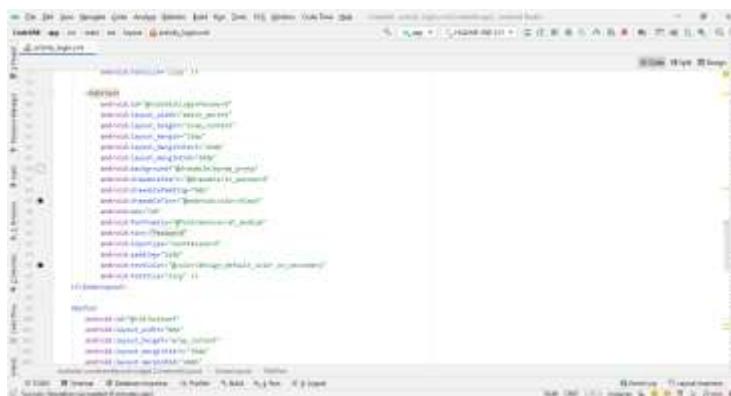


Figura 371: Configuração do tipo de dados do campo da Password

Visto que esta atividade vai ser apenas usada em modo vertical, tenho de indicar essa informação no ficheiro “AndroidManifest.xml”, pois a mesma não está configurada, nem tem sentido correr no modo horizontal também.

Para mim só tem sentido correr a aplicação no modo horizontal, quando esta tem vídeos ou de certa forma dá mais jeito ao utilizador a visualização no mesmo modo.

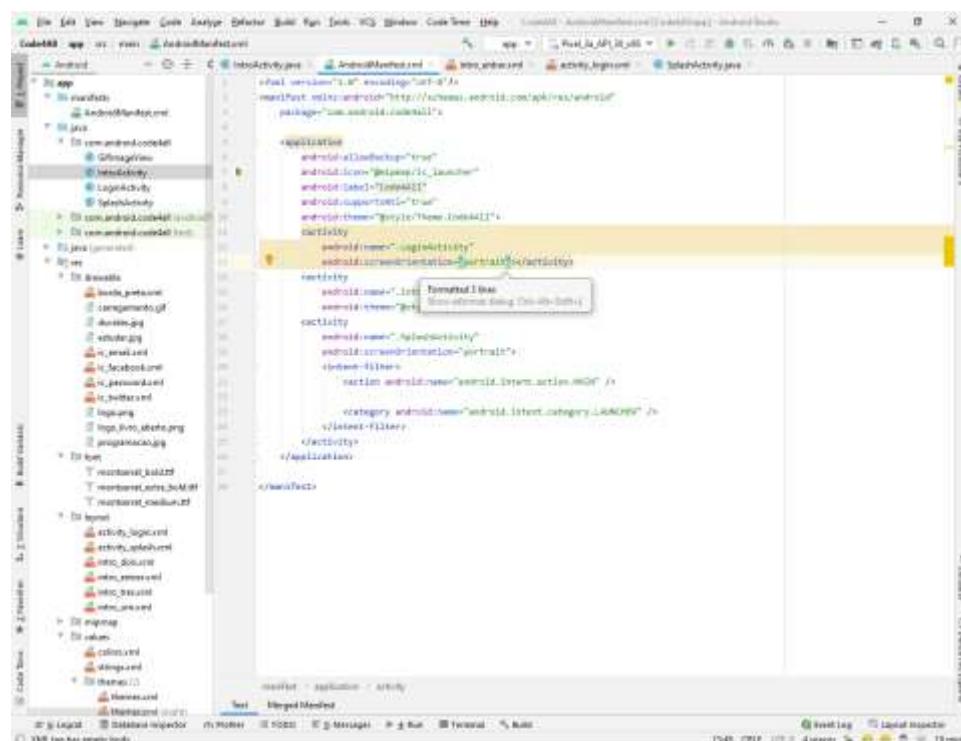


Figura 372: Configuração da atividade para atuar apenas em modo vertical

Nesta atividade, foi preciso ativar a “Firebase Authentication” para ser possível registar utilizadores com o email e password, como se observa nas figuras abaixo.

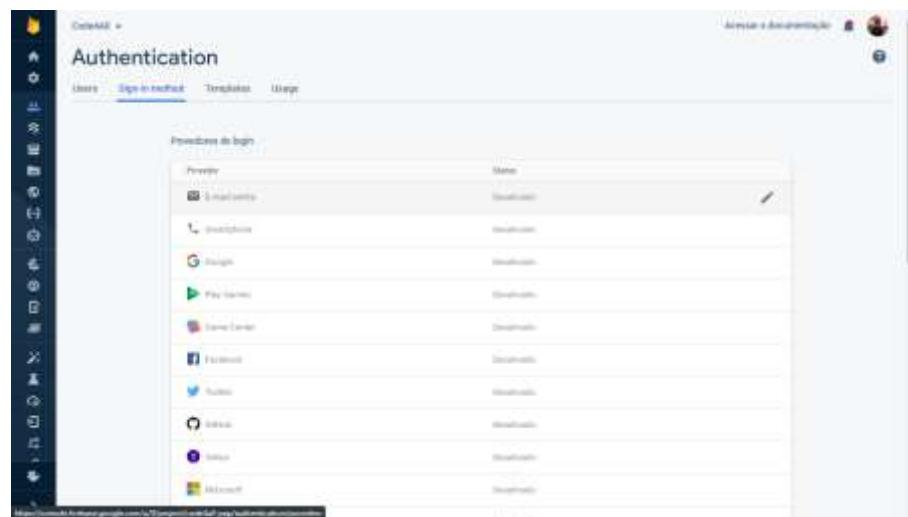


Figura 373: "Firebase Authentication" provedores de login

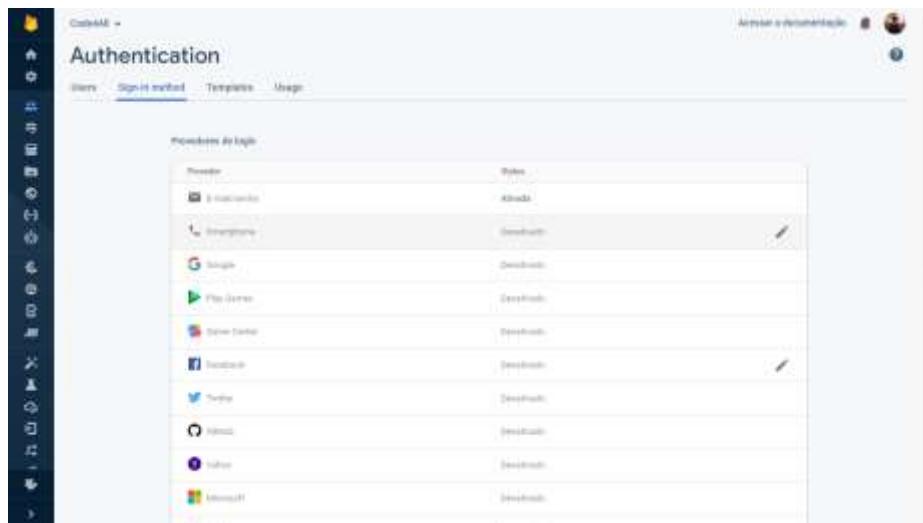


Figura 374: Ativação do "Firebase Authentication" relacionado com "Email/senha"

Seguidamente, fui à documentação do Firebase à procura da libreria necessária para utilizar a “Firebase Authentication” e cliquei em “Primeiros passos para Android”.

Encontrei a documentação do firebase neste site, <https://firebase.google.com/docs>



Figura 375: Página inicial da documentação do Firebase

De seguida, dirigi-me à parte do “Firebase Auth” e cliquei onde dizia “Autenticação com senha” que me daria então a implementação que teria de fazer no meu projeto para colocar o mesmo a funcionar.

<https://firebase.google.com/docs/auth/android/password-auth>



Figura 376: Código de implementação do "Firebase Authentication"

Com o código de implementação que precisava para colocar o “Firebase Auth” a funcionar no meu projeto, bastou apenas abrir o ficheiro “build.gradle” ao nível da app e colocar o código que retirei da documentação no mesmo, como se verifica na captura de ecrã.

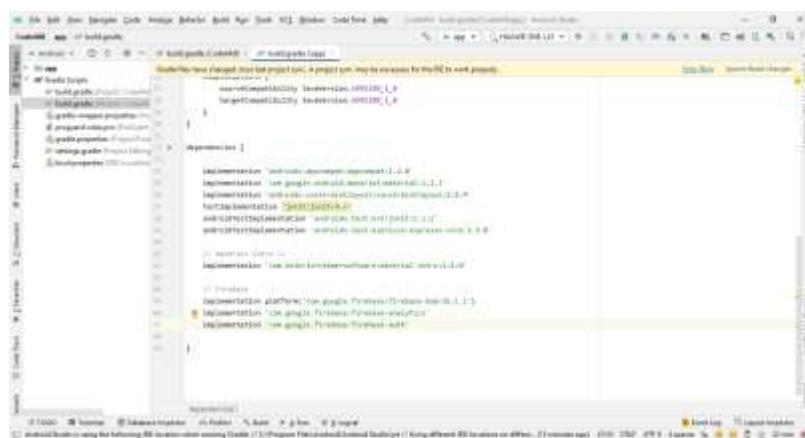


Figura 377: Implementação da "Firebase Auth"

Com a FirebaseAuth, importada para dentro do meu projeto, passei à invocação dos campos, pois para fazer operações com os mesmos tenho de atribuir ID's aos mesmos, lá está, como falei anteriormente estes ID's devem ser únicos para não gerar nenhuma confusão.

Como os componentes que criei para os campos de email e password são do tipo de EditText, foi preciso criar variáveis para os mesmos e atribuir os ID's, como se observa na figura da página seguinte.

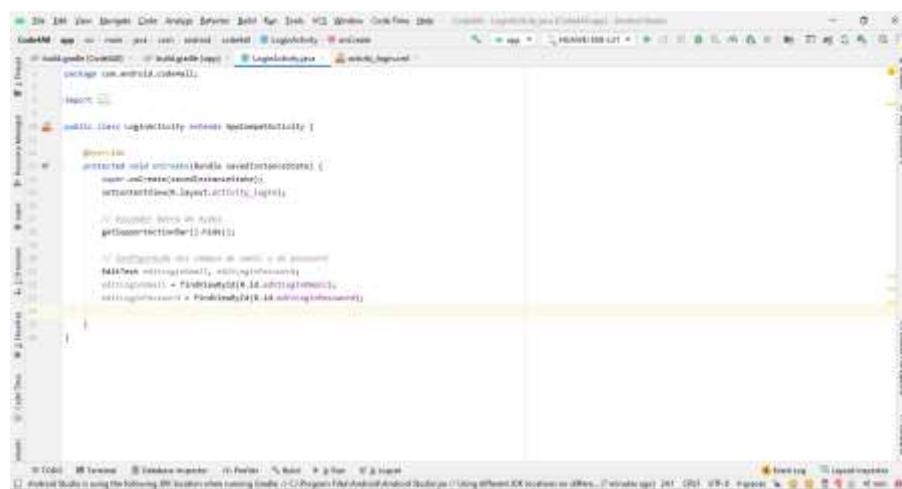


Figura 378: Criação e atribuição dos EditText (email e password)

Nesta parte, defini então um id para o botão “Entrar”, visto que o vou utilizar o mesmo para realizar o login do utilizador.

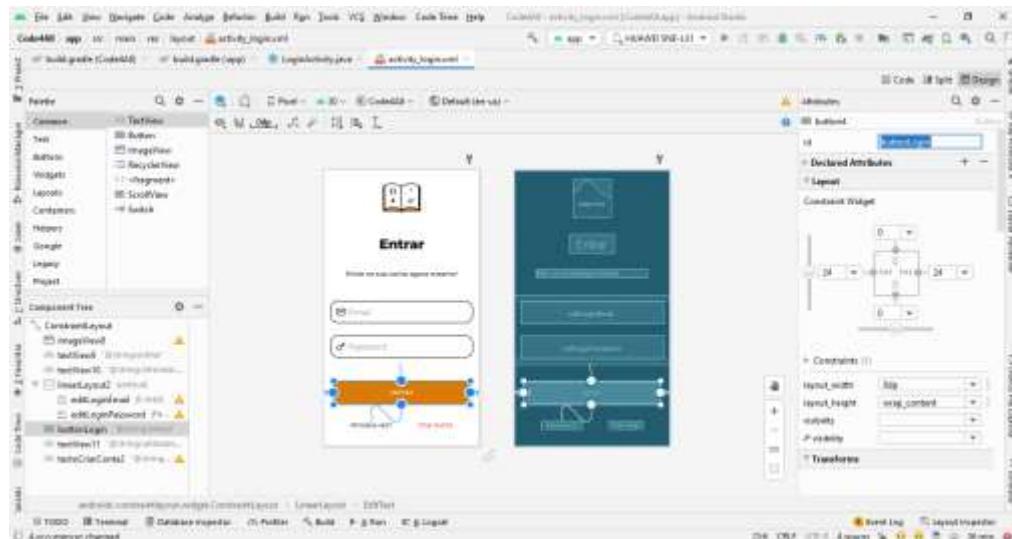


Figura 379: Modificação do ID do botão "Entrar" para "buttonLogin"

Com o ID do botão definido, fui novamente para o ficheiro da atividade e passei a configuração e a declaração do mesmo.

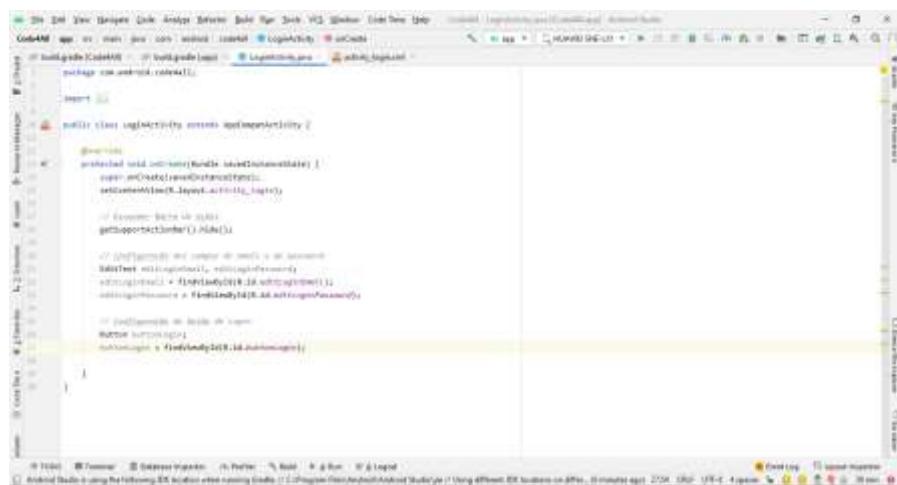


Figura 380: Configuração e atribuição do botão "Entrar"

Com o botão devidamente configurado, passei à colocação de ouvinte no clique do mesmo, ou seja, quando ocorrer um clique no botão, o ouvinte será chamado e o código dentro dele será executado.

Neste caso apenas será atribuído na variável email e password, o texto que se encontra nos campos, na altura que o clique foi efetuado.

É possível recuperar o texto do campo, com a função “getText()” coloquei ainda o “toString()” para que seja feita uma conversão do texto para string e que de certa forma não haja erros.

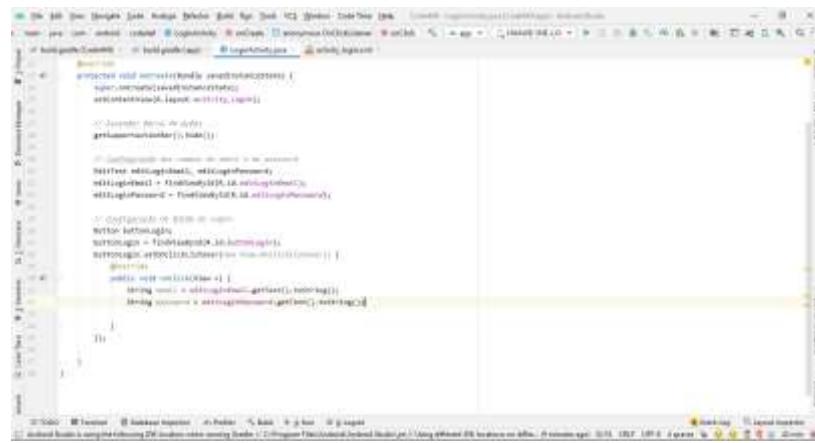


Figura 381: Configuração do ouvinte de cliques do botão e recuperação dos textos dos campos

Coloquei uma estrutura com duas condições, onde são verificados os campos de email e password e se os mesmos estão preenchidos, se estiverem, por enquanto nada acontece, mas se não estiverem, irá ser exibida de imediato uma informação que o determinado campo está vazio. Isto é possível com o “Toast.makeText()”, que me permite enviar informações para a tela do utilizador



Figura 382: Criação da estrutura "if" e colocação dos Toast's de informação

Logo a seguir, criei uma nova Java Class, com o nome “DefinicaoFirebase” que será utilizada para retornar, por enquanto, a FirebaseAuth e para assim realizar o login. Mais à frente no projeto, esta classe será bastante útil para retornar outras funções da Firebase.

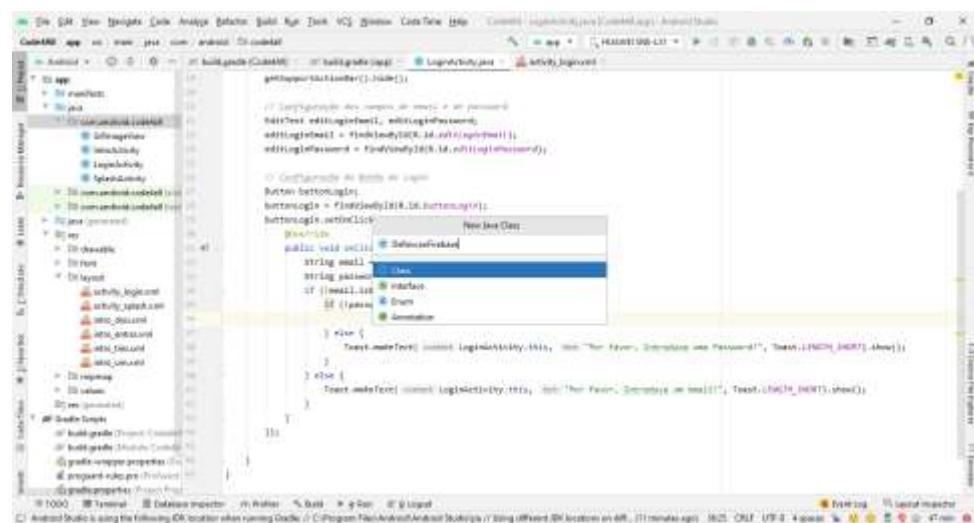


Figura 383: Criação da classe "DefinicaoFirebase"

Na class criada, criei logo um método “recuperarAutenticacao()”, para realizar a recuperação da “Firebase Authentication” para dentro de uma variável “autenticacao”.

Fiz ainda uma verificação, se a variável for nula, a mesma será igualada à instância do “Firebase Auth” retornando no final a “autenticacao”, impedindo assim que se no caso executar mais de que uma vez o método, quando for feita novamente a verificação será pulada a mesma, visto que a variável já não estará nula, a partir da primeira execução



Figura 384: Criação do método "recuperarAutenticacao()" dentro da classe "DefinicaoFirebase"

Com a autenticação da firebase recuperada num método “public static void” dentro da classe, esse método estático permite me usar o método que criei em qualquer parte do projeto, bastou criar uma variável onde será definido o mesmo.

Fiz ainda a verificação que se a tarefa correr bem “task”, ou seja, o utilizador existe e as credenciais foram bem introduzidas e de logo será começada a PrincipalActivity e finalizada esta atividade.

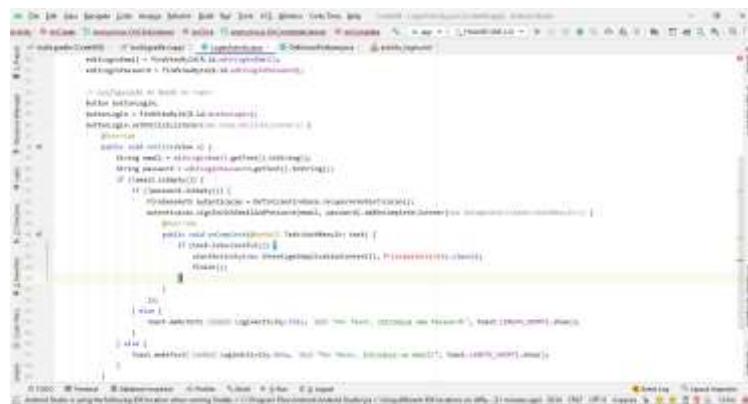


Figura 385: Condição "if" para verificar se a "task" executou sem problemas

Com isto, pretendia agora fazer o tratamento das exceções, para que se surgisse um erro ao realizar login na aplicação, o utilizador fosse avisado do mesmo e para também não ocorrerem “crashes” na app.

<https://stackoverflow.com/questions/37859582/how-to-catch-a-firebase-auth-specific-exceptions>



Figura 386: Try Catch para as exceções do Firebase Auth (login)

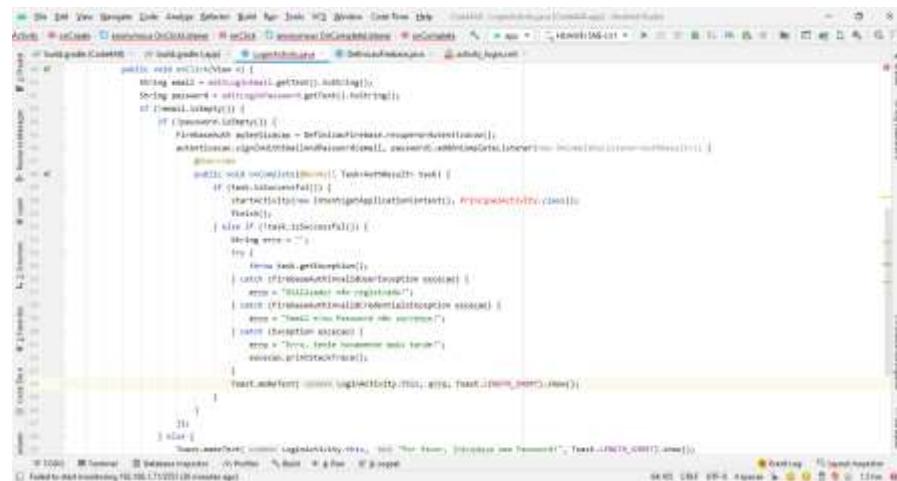
Com base na solução anterior, decidi pegar e alterar a mesma, conforme as minhas necessidades. Como esta atividade, era de Login, ou seja, teria que haver uma entrada de credenciais, tais como, o email e a password. Foi preciso realizar verificações próprias para que não ocorressem erros inesperados no processo de autenticação do utilizador à Firebase Database (base de dados em tempo real).

Realizei então três verificações, começando com a verificação de email e depois então a de password. Estas verificações realizadas aos campos, servem apenas para ver se os mesmos se encontram vazios ou não.

Após isso, recuperei uma referência de autenticação para a Firebase, e configurei um ouvinte para a autenticação de um utilizador com email e password “signInWithEmailAndPassword”, pois era o tipo de autenticação que vinha a ser feita, neste caso.

Depois, adicionei um ouvinte “addOnCompleteListener”, para quando fosse completa a autenticação um método fosse executado, neste caso, o “onComplete”.

Por fim, realizei uma verificação, para saber se a tarefa tinha sido executado com sucesso ou não “task.isSuccessful()”, retornando ela “true” se sim e “false” se não. Se sim, levava o utilizador para a atividade Principal. Se não, iria ser executado um “try catch”, para então descobrir onde está o erro e apresentar o mesmo para o utilizador.



```

public void iniciar(View v) {
    String email = editTextEmail.getText().toString();
    String password = editTextPassword.getText().toString();
    if (email.isEmpty()) {
        Toast.makeText(getApplicationContext(), "Por favor, insira o seu email!", Toast.LENGTH_SHORT).show();
    } else if (password.isEmpty()) {
        Toast.makeText(getApplicationContext(), "Por favor, insira a sua senha!", Toast.LENGTH_SHORT).show();
    } else {
        mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                    startActivity(intent);
                } else {
                    String error = task.getException().getMessage();
                    if (error.equals("User exists")) {
                        error = "Utilizador já existente!";
                    } else if (error.equals("Email address is invalid")) {
                        error = "Email inválido!";
                    } else if (error.equals("Wrong password")) {
                        error = "Senha incorreta!";
                    }
                    Toast.makeText(getApplicationContext(), error, Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
    
```

Figura 387: Autenticação do utilizador e tratamento de erros

Seguidamente, para resolver o erro a vermelho foi preciso criar a atividade em falta, sendo ela a “PrincipalActivity”; para onde serão redirecionados os utilizadores, que já estiverem autenticados na aplicação.



Figura 388: Criação da atividade "PrincipalActivity"

Seguidamente, realizei a definição do método “onClick”, do textView (CriarConta), para que este quando fosse clicado executasse um método definido na atividade (LoginActivity).

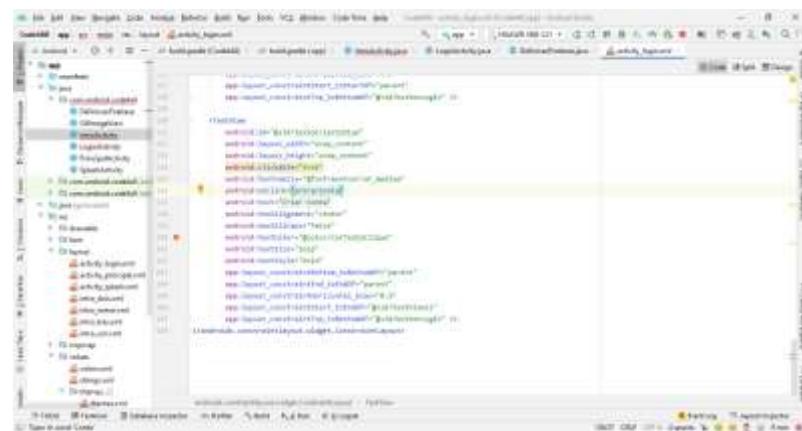


Figura 389: Definição do método onClick "entrarConta"

Nesta parte, observei que poderia realizar testes no código anteriormente escrito, para verificar se o código que escrevi, estava a funcionar e a correr sem problemas. Para verificar isso, como estou na LoginActivity, foi preciso passar à criação de um utilizador na Firebase, visto que a CriarContaActivity ainda não estava criada.

Criei um utilizador com o email “josexavier46@outlook.pt” e com a password “provadeaptidaoprofissional”. E como dá para observar na figura, o utilizador foi criado com sucesso e de imediato, foi criado também um UID para o mesmo. Este UID (Unic Identification) vai ser bastante importante no desenvolvimento de aplicação, permitindo a maior parte das operações na mesma. E graças ao Firebase, este UID é criado automaticamente, sem a necessidade de o programador criar um manualmente.

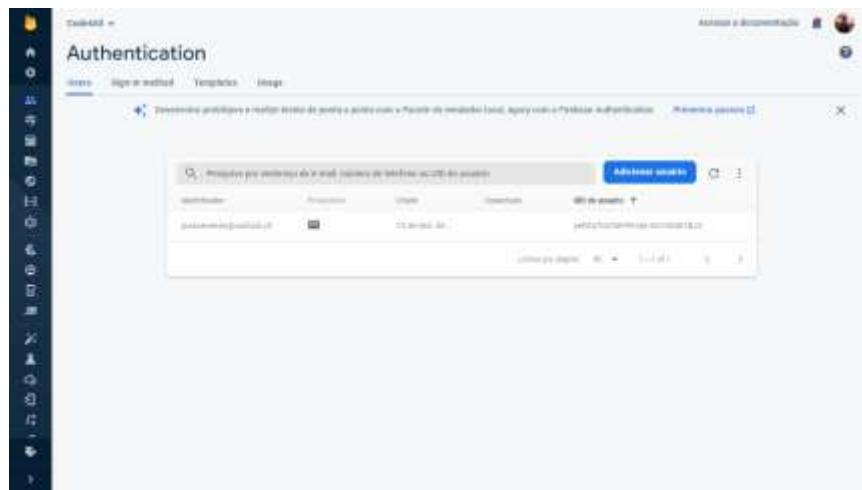


Figura 390: Criação de um utilizador na FireBase

Com o utilizador criado, abri a aplicação no meu dispositivo Android e fui testar o código da LoginActivity.



Figura 391: Teste de utilizador não criado

Continuando os testes ao código, decidi nesta parte testei não introduzir nenhum dado nos campos.



Figura 392: Teste de campos vazios

Com os anteriores testes realizados, conclui que todo o código anteriormente escrito havia sido bem escrito e estava a funcionar e a correr sem problemas. Decidi fazer login com a conta que criei anteriormente, para verificar se o login e a ligação à base de dados foram bem realizados.



Figura 393: Teste de login com email e password

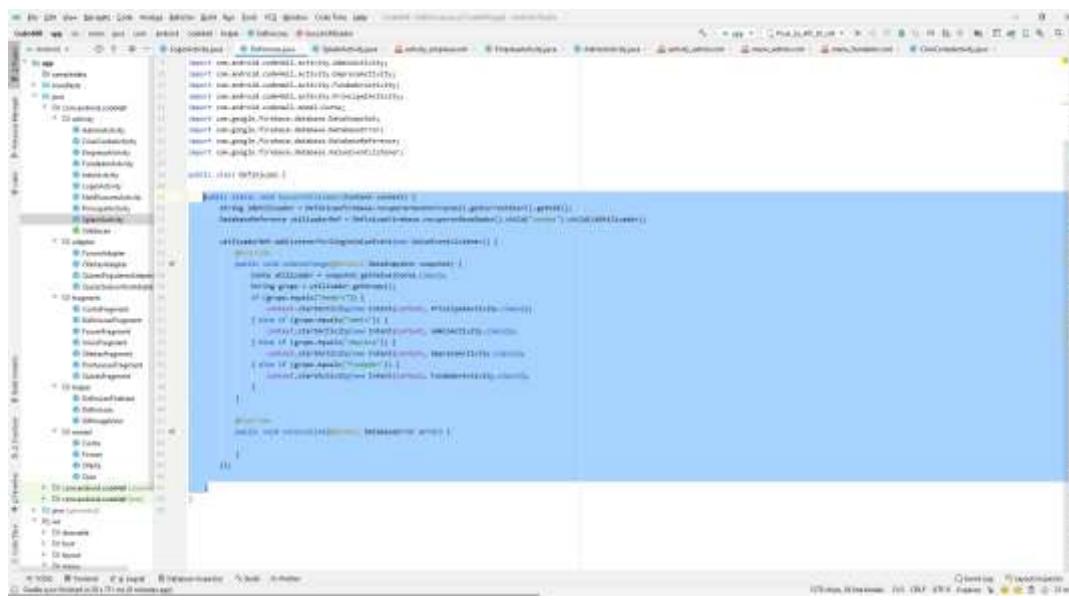
Como se observa na figura da página seguinte, o utilizador (eu) foi redirecionado para a PrincipalActivity (que neste caso já estava criada para efetuar o teste de Login).

Como se pode ver na figura abaixo, a autenticação das credenciais, do telemóvel e com a base de dados (Firebase) foram realizadas com sucesso e o utilizador josexavier46@outlook.pt entrou com sucesso na sua conta.

PRINCIPAL ACTIVITY

Figura 394: Autenticação do utilizador para a PrincipalActivity

Como já dito anteriormente, foi criado este método, para retornar a “*Intent*” que levaria o utilizador para atividade correta, consoante o seu grupo.



```

private void buscarUtilizador() {
    String idUtilizador = FirebaseAuth.getInstance().getCurrentUser().getUid();
    DatabaseReference referenciaUtilizador = FirebaseDatabase.getInstance().getReference("utilizadores");
    referenciaUtilizador.child(idUtilizador).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            Utilizador utilizador = dataSnapshot.getValue(Utilizador.class);
            if (utilizador != null) {
                String grupo = utilizador.getGrupo();
                Intent intent;
                if (grupo.equals("Administrador")) {
                    intent = new Intent(getApplicationContext(), AdministradorActivity.class);
                } else if (grupo.equals("Funcionario")) {
                    intent = new Intent(getApplicationContext(), FuncionarioActivity.class);
                } else if (grupo.equals("Familiar")) {
                    intent = new Intent(getApplicationContext(), FamiliarActivity.class);
                } else if (grupo.equals("Voluntario")) {
                    intent = new Intent(getApplicationContext(), VoluntarioActivity.class);
                } else if (grupo.equals("ApoioSocial")) {
                    intent = new Intent(getApplicationContext(), ApoioSocialActivity.class);
                } else if (grupo.equals("Educador")) {
                    intent = new Intent(getApplicationContext(), EducadorActivity.class);
                } else if (grupo.equals("Estudante")) {
                    intent = new Intent(getApplicationContext(), EstudanteActivity.class);
                } else if (grupo.equals("Pai")) {
                    intent = new Intent(getApplicationContext(), PaiActivity.class);
                } else if (grupo.equals("Mae")) {
                    intent = new Intent(getApplicationContext(), MaeActivity.class);
                } else if (grupo.equals("Outro")) {
                    intent = new Intent(getApplicationContext(), OutroActivity.class);
                }
                startActivity(intent);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
        }
    });
}

```

Figura 395: Método "buscarUtilizador()"

Com o método criado e se o login fosse bem efetuado, o utilizador era redirecionado para a “sua” atividade.



Figura 396: Uso do método "buscarUtilizador()"

Visto que o botão continuava sem mudar de cor, decidi explorar mais o porquê disso acontecer e cheguei à conclusão que o problema estava em eu usar o elemento “Button” enquanto devia estar a usar o “AppCompatButton”.

<https://stackoverflow.com/questions/31858374/android-button-background-color-not-changing/>

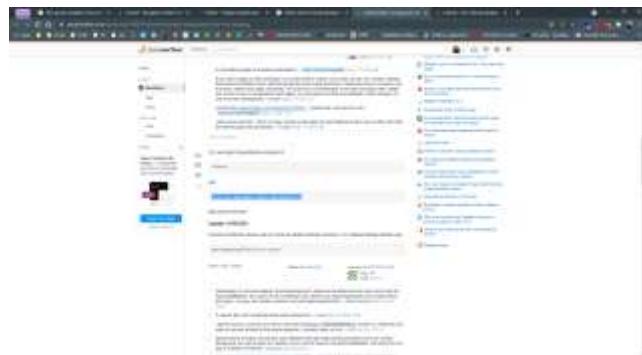


Figura 397: Solução para trocar a cor de fundo de um botão

Aplicando a solução em cima explicada, o meu layout ficou agora muito mais interessante.

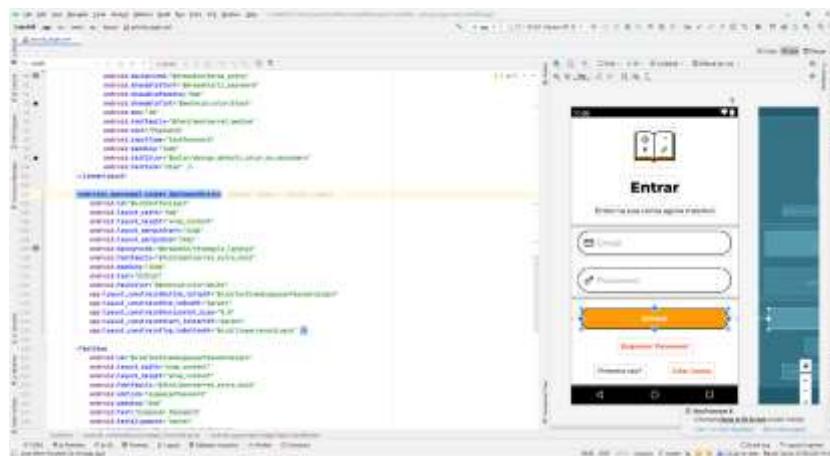


Figura 398: Troca do elemento "Button" para o "AppCompatActivity"

Seguidamente, configurei o *TextView* do “EsquecerPassword” no seu método “*onClick*”, com a abertura da atividade “*EsquecerPasswordActivity*”.



Figura 399: Configuração do método "onClick" do TextView EsquecerPasswor"

EsquecerPasswordActivity

Na imagem abaixo, encontra-se a configuração realizada ao nível de layout.

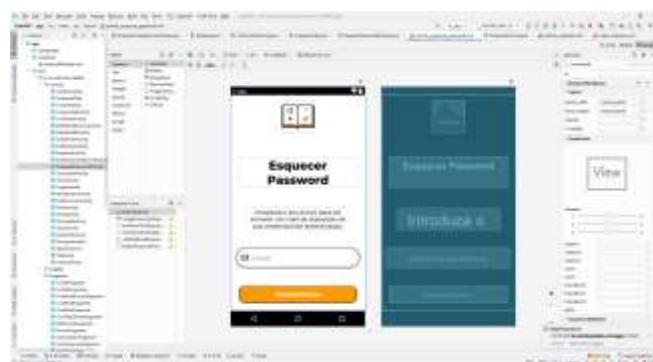


Figura 400: Configuração do layout

Depois configurei o método “onClick” do botão, que por si mesmo, utiliza o método “sendPasswordResetEmail()”, que depois pede como argumento um email, email esse, que seria introduzido pelo o seu utilizador e se tudo corresse bem o utilizador desse email, receberia um email, para repor a sua password.



Figura 401: Configuração do método "onClick" para repor a password

CriarContaActivity

Para iniciar o desenvolvimento nesta atividade, é necessária a criação da mesma. De imediato, fui ao ficheiro da atividade e adicionei a linha de código “getSupportActionBar().hide()”, para esconder a barra superior de ações e assim dar um “look” mais moderno à app.



Figura 402: Adição da linha de código para esconder a barra de ações

Depois, pensei na melhor solução de como colocar os campos do formulário de registro no layout e cheguei à conclusão que, a melhor opção que tinha era “embrasar” todos os EditText’s dentro de um LinearLayout (vertical), para depois o alinhamento dos campos seja mais fácil de fazer.

Seguidamente, passei à construção da ideia anterior, colocando um LinearLayout e dentro dele um EditText. Pretendo ainda realçar que, a configuração do EditText é sempre a mesma logo não vou estar sempre a explicar a configuração do mesmo, o único atributo que muda é a referência ao recurso String, que por si mesmo leva à alteração do seu texto de “hint” (ajuda).

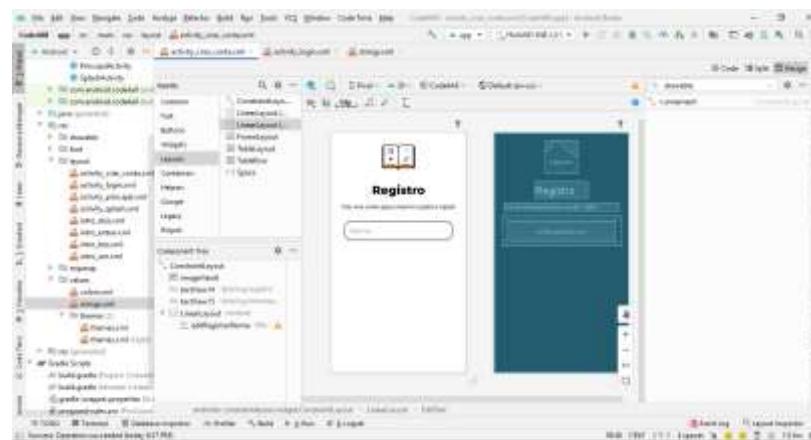


Figura 403: Configuração do LinearLayout e EditText

O campo nome, já estava praticamente configurado, mas faltava um pequeno detalhe e todos sabemos que os pequenos detalhes é que fazem a diferença. Neste caso, era adicionar um pequeno ícone no lado direito do EditText, visto que este faz uma grande diferença no campo e ficava assim mais apelativo e fácil de entender para o utilizador.

Fui buscar esse ícone descritivo, mais uma vez ao Feather Icons, como se observa na figura abaixo.

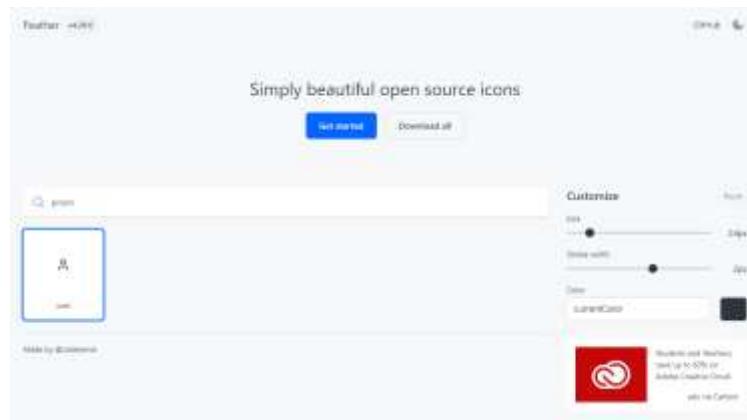


Figura 404: FeatherIcons - Ícone Utilizador

Com a importação de mais um ícone concluída, foi preciso, tal como os outros, mudar a cor do mesmo para branco, visto que a cor que já vinha incluída não era reconhecida pelo o Android Studio e a app não iria executar com esse erro.

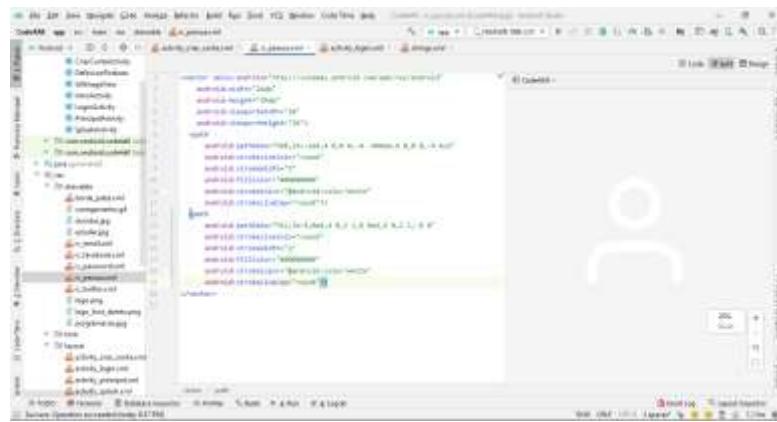
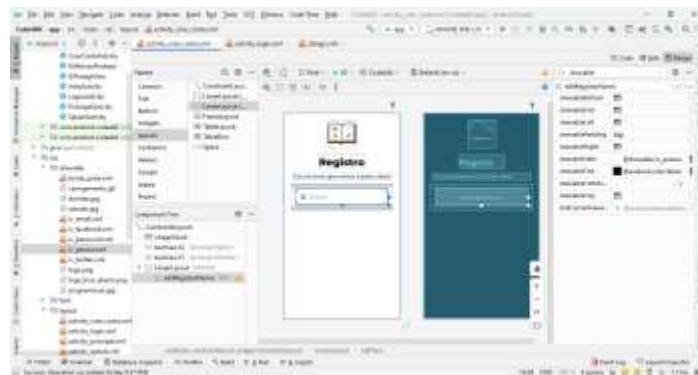


Figura 405: Modificação da cor do ícone

Já com o ícone devidamente importado e configurado, voltei novamente para o layout desta atividade e dirigi-me ao componente EditText (nome), para proceder à inserção da imagem da “pessoa” conquistando assim um aspecto visual muito mais apelativo.

Para isso, pesquisei pelo o atributo “drawable” no EditText e na propriedade “drawableStart” coloquei o caminho do ícone e para “condizer” com o resto do layout defini a cor do ícone como “black” (preta).



Coloquei também neste componente, uma margem de 8dp em todos os lados do mesmo (layout_margin).

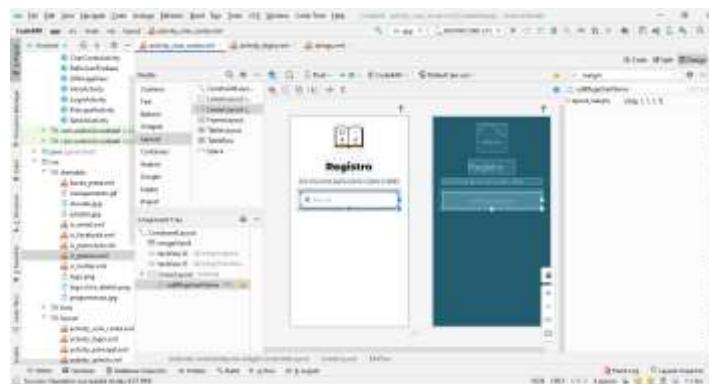


Figura 406: Definição de 8dp para as margens do EditText

Por fim, no EditText, coloquei ainda um “padding” de 16dp para dar um aspeto mais interessante ao próprio.

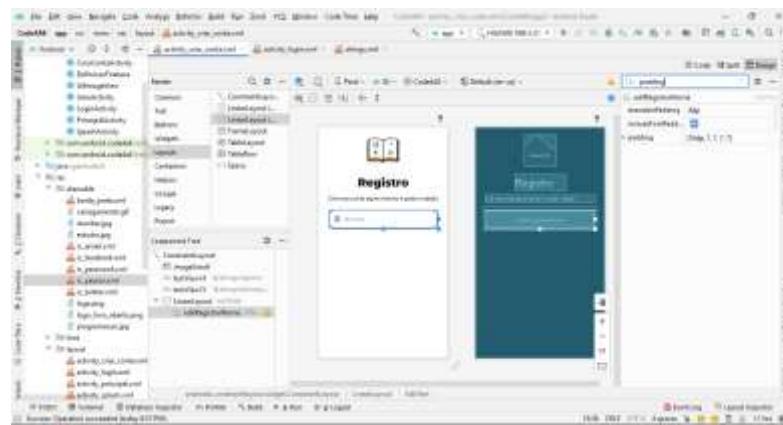


Figura 407: Definição de 16dp para o "padding" do EditText

Seguidamente, dirigi-me mais uma vez, ao meu site favorito de ícones (open-source), desta vez para localizar e realizar “download” do ícone de um calendário, para ficar de certa forma alusivo ao campo de data de nascimento, para mim era o que faria mais sentido no campo em questão.

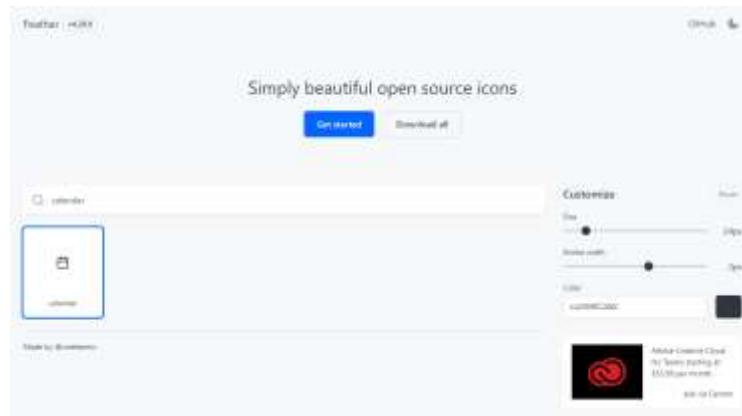


Figura 408: Ícone de Calendário (Feather Icons)

E como realizado nas importações de ícones anteriores, é necessário mudar a cor do ícone, visto que a cor que vem por predefinição no ícone, não é reconhecida pelo o Android Studio.



Figura 409: Modificação do atributo "strokeColor" dentro do ícone (calendário)

Com o último ícone desta atividade importado e configurado com sucesso, passei à finalização do layout, visto que já tinha todos os conteúdos para o realizar. Como o primeiro EditText (nome), já estava totalmente configurado, então o que fiz, foi copiar o código do primeiro e replicar o mesmo mais quatro vezes, distribuindo consoante o que tinha planeado na fase de desenho e alterando os atributos dos mesmos. Ficando então, como se observa na figura abaixo.

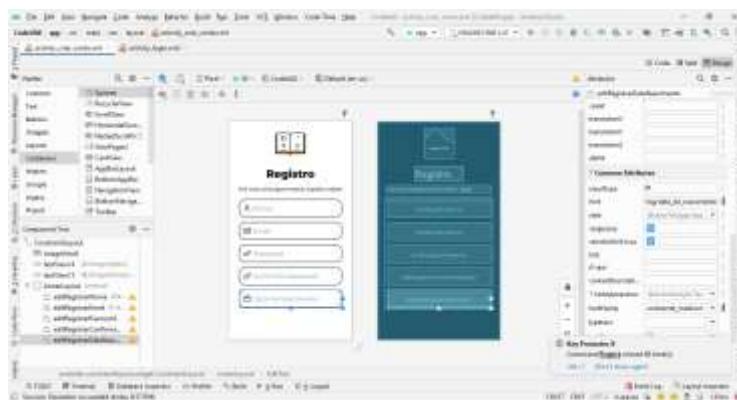


Figura 410: Resultado do layout após da configuração dos EditText's

Com a parte dos EditText's minimamente tratada, decidi logo tratar da configuração do EditText (data de nascimento), visto que gostaria de colocar no mesmo uma interface de escolha de uma data apelativa, como se observa nas grandes apps.

Para isso, foi necessário realizar uma pesquisa, para saber o que o componente EditText em si, conseguia fazer. Com uma breve e resumida pesquisa conclui que, este tem sim a propriedade de servir de calendário, mas essa propriedade não alterava o aspecto estético do mesmo e sim o formato dos dados que eram introduzidos. Mas o meu objetivo ainda não estava totalmente concluído, visto que queria tanto o formato dos dados correto (data) como também, uma interface gráfica apelativa, como referida anteriormente, para que o utilizador pode-se escolher consoante a sua data de nascimento.

E após uma breve pesquisa na procura da solução do meu problema, a mesma foi encontrada, mais uma vez no fórum do StackOverflow.

A proposta da suposta solução para fazer o EditText apresentada verifica-se na figura abaixo e resumidamente ao aplicar esta solução tive de modificar o ficheiro de layout (XML) e o ficheiro da atividade (Java), sendo a página esta:

<https://stackoverflow.com/questions/14933330/dialogue-how-to-popup-datepicker-when-click-on-edittext/>



Figura 411: Solução do Calendário - StackOverflow

Com a suposta solução localizada, verifiquei ao ver o código que vinha a ser sobescrito o método “onClick” do EditText, logo teria de tornar o mesmo não editável, visto que o utilizador viria a selecionar a sua data de nascimento através de uma interface gráfica e não a digitar a mesma no campo.

Visto que não sabia como tornar o campo não editável, tive de realizar outra pesquisa e para “variar” vim parar novamente ao StackOverflow, onde encontrei várias propostas de solução, como se observa na figura abaixo.

O link da pergunta no StackOverflow é o seguinte:

<https://stackoverflow.com/questions/9470171/edittext-non-editable/>



Figura 412: Solução StackOverflow - EditText não editável

Com as soluções dos meus dois problemas planeadas, passei a à escrita e modificação do código, consoante as minhas necessidades. Comecei por colocar, todo o código que estava disponibilizado no StackOverflow dentro da minha atividade, mais precisamente dentro do método “onCreate()”.

Criei também um atributo privado “calendario” do tipo “Calendar” e inicializei o mesmo com o código “Calendar.getInstance();” e logo depois, recuperei o componente EditText do calendário.

Continuando, foi criado também um “DatePickerDialog” com um ouvinte “OnDateSetListener”, para que fosse disputada a ação que queria realizar, de abrir um calendário já e o utilizador escolher uma data. E foram passados quatro argumentos, sendo eles: “view”, “ano”, “mes” e “dia”. Cada vez, que este método fosse executado, na variável “calendario”, eram armazenados três valores, o ano, o mês e por fim o dia.

De seguida, criei um método com o nome “atualizarTexto()” para que o texto do campo “Data de Nascimento”, seja atualizado com a seleção da data do utilizador.

Figura 413: Alterações no código de abertura do calendário do EditText – Parte 1

Continuando a explicação do código, na linha 37, é subscrito o método "onClickListener", do "editDataNascimento", para que seja mostrado um "DatePickerDialog" ao simples clique no campo da data de nascimento.

Para isso, é preciso instanciar um novo “DatePickerDialog” e passar cinco parâmetros, sendo eles:

- context -> CriarContaActivity.this
 - date -> OnDateSetListener()
 - calendario.get(Calendar.Year) -> int (ano)
 - calendario.get(Calendar.Month) -> int (mês)
 - calendario.get(Calendar.DAY_OF_MONTH) -> int (dia do mês)

Por fim, é necessário colocar o “.show()”, para que este dialog seja mostrado como uma interface para o layout desta atividade, como se vai ver mais à frente neste relatório espetacular.

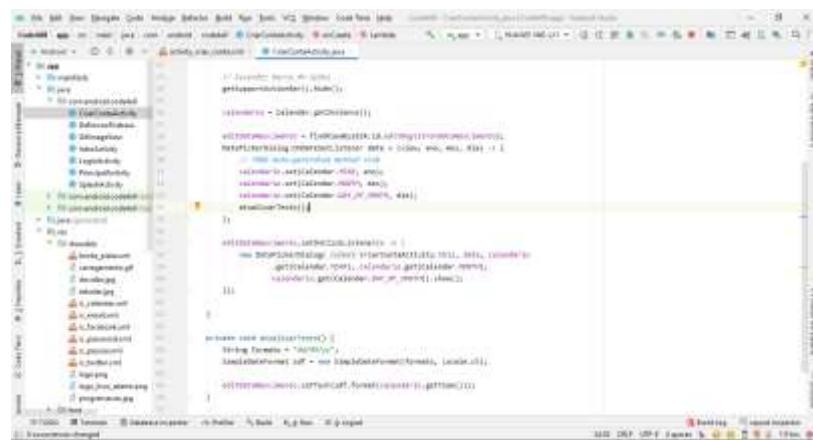


Figura 414: Alterações no código de abertura do calendário do EditText – Parte 2

Nesta parte, executei a aplicação desde o seu início e quando cheguei a esta atividade, o meu coordenador de curso (Profº Michael Teixeira), disse-me que deveria retirar os espaços em branco para que de certa forma as quebras de linha “\n”, estivessem bem alinhadas.

De certa forma, também decidi mostrar este arquivo “strings.xml” visto que é este que guarda todos os recursos “String” criados e usados nos layouts das atividades ou fragmentos, permitindo assim a edição de um recurso apenas e apenas só num único lugar, sendo ele este.

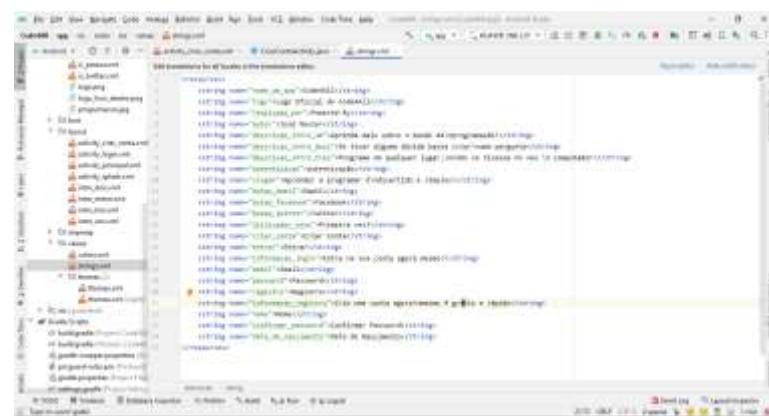


Figura 415: Eliminação dos espaços "vazios" dentro do ficheiro "strings.xml"

No print que disponibilizo, está representado o layout desta atividade, depois de sofrer uma correção a nível de texto e dos espaços vazios (strings.xml).

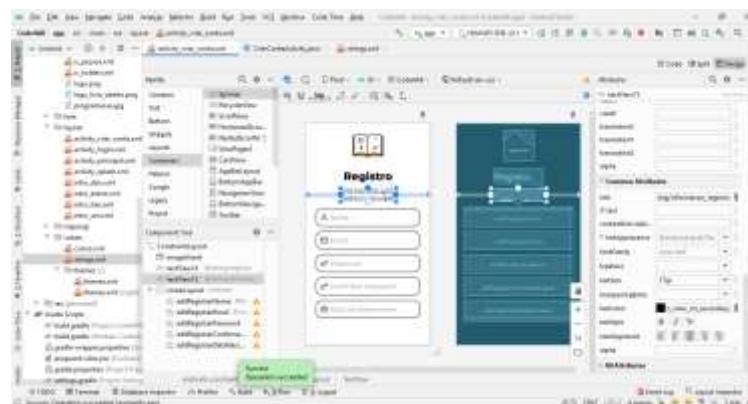


Figura 416: Alterações do layout consoante o ficheiro "strings.xml"

Como foi referido anteriormente, o método “onClickListener” foi sobreposto, mas mesmo assim quando o utilizador pressionasse o campo data de nascimento, o teclado vinha a ser aberto e para corrigir esse aspeto, foi preciso realizar uma pesquisa, pesquisa essa que já havia sido calculada por mim, nas páginas anteriores. Essa solução passava por desativar o próprio EditText, para impedir este de “chamar” o teclado e apenas o código em “Java” ser executado lançando assim apenas o calendário, com a definição do atributo “setEnabled” como “false”, como se visualiza na figura.



Figura 417: Desativação do EditText (data de nascimento)

Após de ter colocado a linha de código que referi anteriormente, executei a app no meu Android e percebi que nada tinha feito, o EditText continua a “chamar” o teclado, ação que eu não queria que acontecesse.

Voltei novamente para a página do StackOverflow, já referida anteriormente, pois esta me dava várias alternativas de solução.

Seguidamente, voltei a IDE e decidi experimentar novamente só que com uma linha de código diferente, neste caso a “setKeyListener(null)”, que traduzido significa algo semelhante como, “definir ouvinte de letra (vazio)” e supostamente iria definir o ouvinte do chamar do “teclado” como nulo, logo nada seria executado quando este fosse pressionado. Na imagem, está a ser mostrada a atividade com a adição desta linha de código.

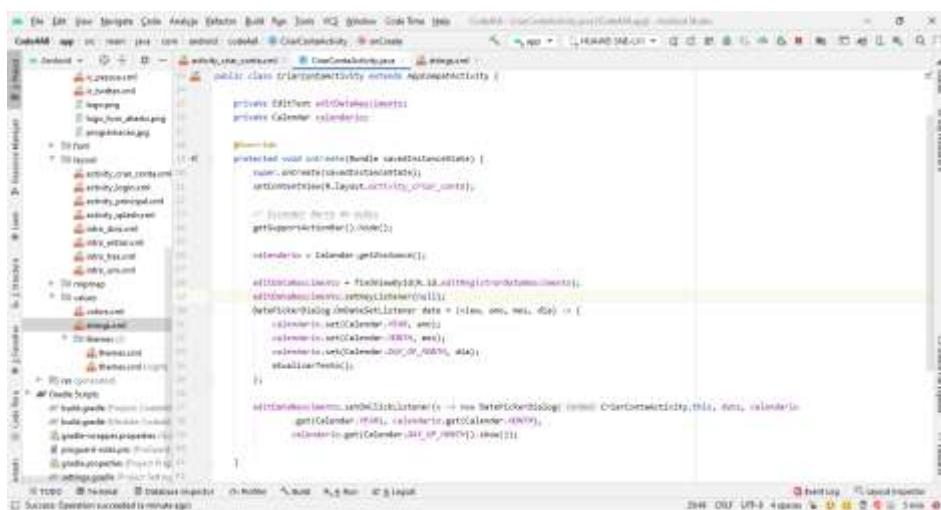


Figura 418: Definição do ouvinte "setKeyListener" como vazio "null" do EditText

Com então a alteração feita, ao nível do EditText da data de nascimento, decidi executar a aplicação para ver se finalmente estava tudo a funcionar como eu desejava.

E na verdade, sim estava, cliquei no EditText e o mesmo abriu o calendário sem primeiro me abrir o teclado, tal como eu o codifiquei para realizar.



Figura 419: Captura de ecrã do Calendário

Depois disso, selecionei uma data aleatória e cliquei em “OK” para ver se a data era corretamente retirada do “Dialog” e colocada dentro do EditText e como se observa na captura de ecrã, essa operação esta a ser realizada com sucesso.



Figura 420: Captura do ecrã após de ter selecionado a data de nascimento

Com os campos completamente configurados, havia chegado a altura de configurar um botão “Switch”, de ligar e desligar, para proceder à configuração de uma conta de “Utilizador” ou de “Empresa”, passei então à colocação do componente “Switch” dentro do layout desta atividade.

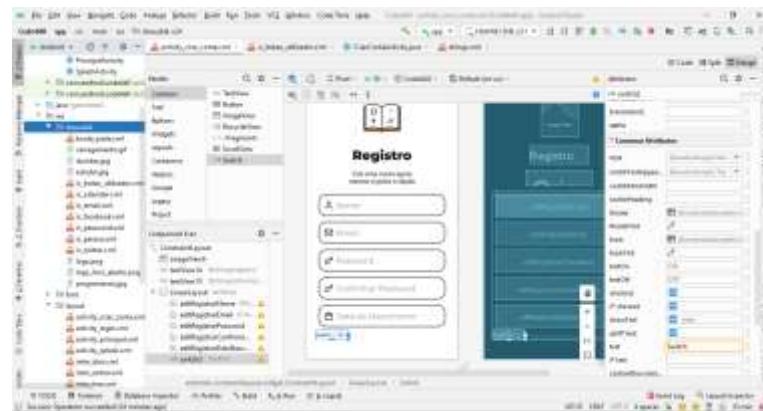


Figura 421: Colocação do Componente "Switch" dentro do layout

Com o componente adicionado na interface, voltei ao adobe XD, e entrei no planeamento do layout da minha app. Sem querer, descobri que este dava para copiar o código SVG de cada elemento presente, o que me facilitava bastante as coisas, não perdendo nenhuma qualidade na transferência do componente do adobe XD para o Android Studio.



Figura 422: Cópia do componente (Switch Utilizador) do Adobe XD

Com o ficheiro colocado dentro do portfólio, o passo a seguir realizado foi importar o respetivo ficheiro para dentro do Android Studio, colocando o nome de “ic_botaao_utilizador”, mais à frente irei abordar o porquê deste nome.



Figura 423: Importação do ícone "ic_botao_utilizador"

Com o botão de utilizador criado, faltava então realizar o mesmo para o outro lado do botão, o lado da “empresa”. Para a criação e importação do mesmo para o meu projeto no Android Studio, foi necessário realizar novamente todos os passos anteriormente descritos.

Com os passos realizados, tive que importar novamente o ficheiro SVG, mas desta vez, com a “bola” direcionada para o lado inverso.



Figura 424: Importação do botão para a seleção da empresa

Seguidamente, já importados os dois vetores, com sucesso, passei então ao planeamento de como iria colocar um componente que tivesse apenas dois estados, um de utilizador e outro de empresa, podendo ser um Switch (On/Off), pois já tinha utilizado este componente no curso que realizei.

A utilização desse componente “*Switch*”, foi a primeira que me veio à cabeça naquele momento de desenvolvimento e de logo fui procurar pela a solução de como modificar a “imagem” do estado “on” e “off” e alterar as mesmas para os vetores que importei anteriormente.

Para ver se essa funcionalidade era permitida e a ideia aplicável, decidi realizar uma pesquisa, para comprovar o mesmo. De logo, fui deparado com esta página do *StackOverflow*, que me dizia, para realizar a criação de um novo arquivo “XML” de configuração para definir uma cor no estado “on” e “off” do *Switch*.

Se conseguisse definir uma cor, muito facilmente iria conseguir colocar uma imagem, visto que estaria a alterar da mesma forma o comportamento inicial do componente *Switch*.

Para realizar essa alteração, era apenas adicionar um atributo “background” no *Switch*, dentro do layout da atividade, neste caso “CriarContaActivity”, para que os estados e as imagens com a sua referência (utilizador ou empresa) fossem alterados. A seguinte solução foi encontrada na seguinte página e print que se encontram abaixo.

<https://stackoverflow.com/questions/29231749/how-to-display-switch-widget-with-image-in-android/>



Figura 425: Solução de colocar uma cor consoante o estado do botão switch

Com a solução mais ou menos planeada, como visto no fórum “StackOverflow”, realizei a criação de um novo ficheiro “configuracao_switch”, como se pode observar na figura abaixo.

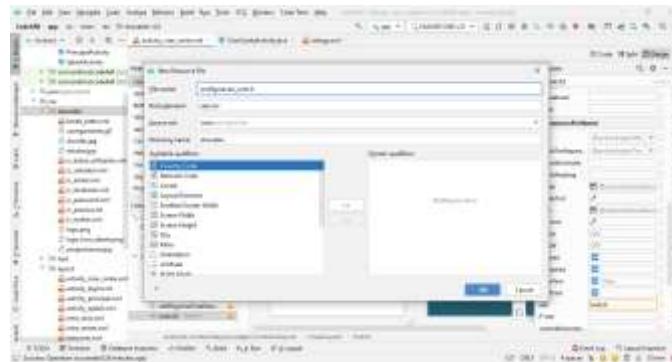


Figura 426: Criação do ficheiro "configuracao_switch"

Com o ficheiro criado, foi só escrever o código consoante o que estava dito na opção de solução dada. Começando pela colocação “selector”, para a criação de regras no componente Switch.

Por fim, foi preciso colocar duas tags “item”, com o atributo “android:state_checked”, sendo este “true” ou “false”, para que se o “Switch” esteja ligado apareça uma imagem (SVG) e outra se o mesmo estiver desligado.

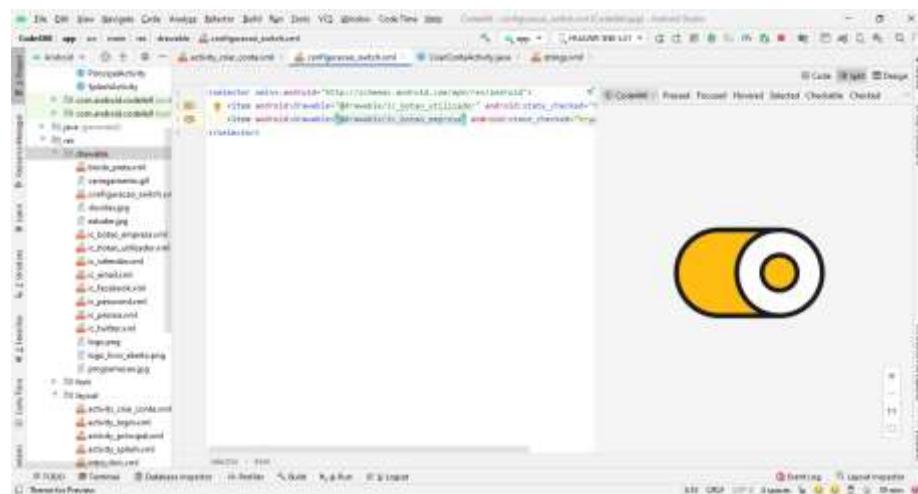


Figura 427: Definição de dois estados do Switch "checked=true" e "checked=false"

Se o “Switch” estiver selecionado, o “ic_botao_empresa” será mostrado, se não, o “ic_botao_utilizador” será aquele que será visualizado no layout, conforme codificado no ficheiro anteriormente falado (“configuracao_switch”).

Seguidamente, voltei para o desenvolvimento do layout, de logo deparei-me que existiria um botão mais prático de aplicar o código que realizei anteriormente, chamado de “toggleButton” e decidi então trocar o “Switch” pelo o mesmo.

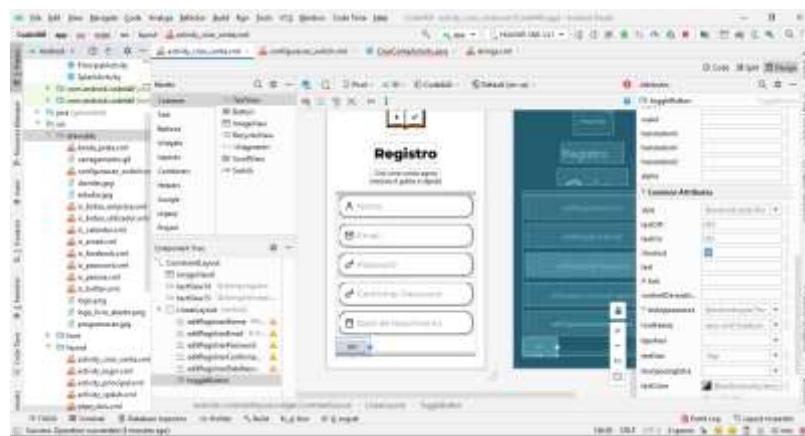


Figura 428: Troca do "Switch" pelo o "ToggleButton"

Depois de fazer a troca de componentes, observei que teria um possível problema, relacionado com o texto do componente, como se observa na figura acima.

O texto era relacionado ao estado do botão, ou seja, se tivesse desligado aparecia “OFF” e se tivesse ligado aparecia “ON” e o meu objetivo era retirar esse texto de informação, para depois aplicar como atributo “background”, com as definições que criei anteriormente no ficheiro “configuracao_switch.xml”.

Com o obstáculo do texto a sobrepor o botão, tive de pesquisar uma solução para esconder o texto, como previsível e mais uma vez para variar, vim parar ao StackOverflow. O site e a captura de ecrã da suposta solução, apresenta-se mais abaixo.

<https://stackoverflow.com/questions/11604476/it-is-possible-to-create-a-togglebutton-without-text/>



Figura 429: Recomendação de solução para esconder o texto de estado do "ToggleButton"

Com a pesquisa que realizei, conclui, que de certa forma, teria três opções, sendo elas o adicionamento de atributos, podendo eles ser “textOff='@null””, “textOn='@null” ou “textSize='0dp”.

No código do layout, em XML, fiz a configuração dos mesmos atributos e o único que funcionou para mim foi o “textSize=0”, como se pode observar na figura da próxima página.

Como dito anteriormente, coloquei o atributo “textSize=0” dentro do componente “ToggleButton”, como se observa na imagem. E ainda coloquei, o atributo “background” com o ficheiro “configuracao_switch”, para oToggleButton ser definido, consoante as regras criadas naquele ficheiro de configuração.

Ao lado direito da imagem, consegue-se observar uma pequena pré-visualização, do layout em si e também o componente “ToggleButton” e aqui conclui que este componente havia sido bem configurado, pois estava a mostrar a imagem que tinha anteriormente importado, o que faltava testar era apenas o clique, para saber se ao clicar a imagem “trocava” de posição entre o “utilizador” e a “empresa”.

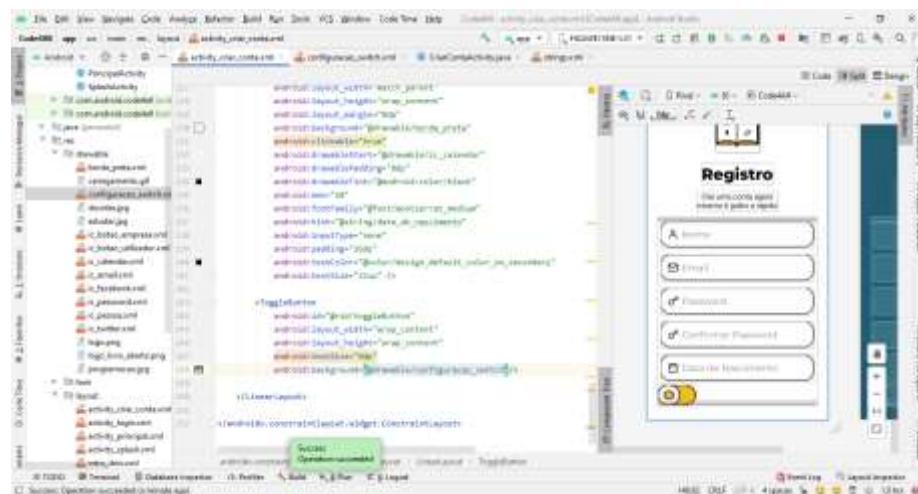


Figura 430: Configuração do "ToggleButton"

Seguidamente, passei à execução, visto que já não a executava à algum tempo e queria perceber como a mesma estava e se existiriam erros no código novo que introduzi, desde a última execução e também para testar oToggleButton e a sua configuração, para ver se estava tudo correto com a mesma.

Executei a aplicação, como se observa na figura abaixo e observei que o botão “Switch” já aparecia, logo à partida estava tudo bem e a correr como previsto.



Figura 431: Execução da aplicação para observar novas atualizações

Logo após de ter criado os recursos String, inseri no layout duas TextView's, uma com o recurso “utilizador” e outra com a “empresa”, alinhei os mesmos verticalmente entre o último EditText (Data de Nascimento) e o fim do layout, e horizontalmente alinhei ao “Switch” com um espaçamento de 16dp.

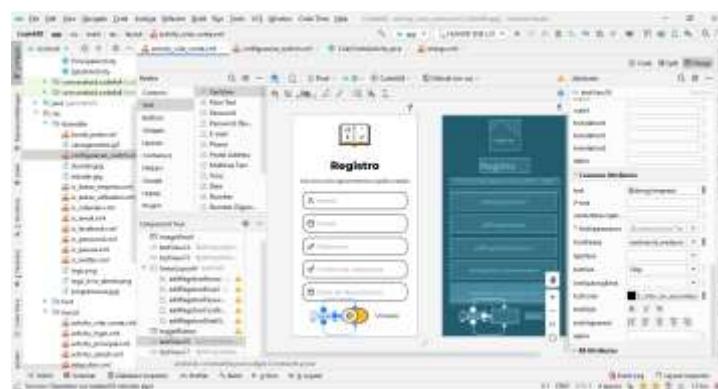


Figura 432: Inserção de duas TextView's e alinhamento das mesmas com o "Switch"

Veio então a necessidade de colocar um botão para o utilizador quando acabasse de preencher os campos todos necessários, pressionasse o mesmo. Mais uma vez, a configuração do layout do botão em si é a mesma dos que foram criados anteriormente, com a exceção da cor, que no layout se apresenta como “preta”, mas na realidade é laranja.

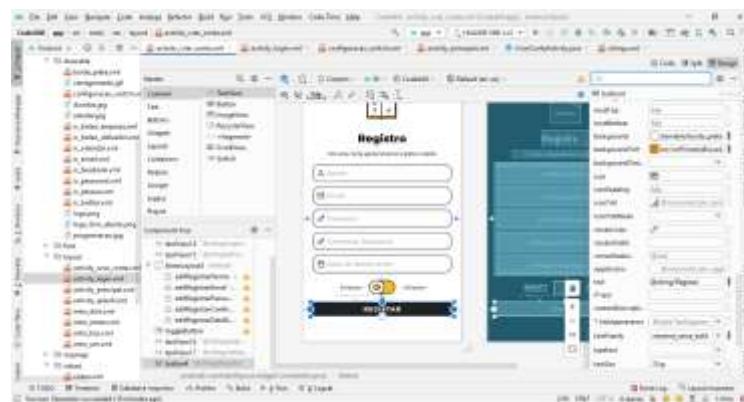


Figura 433: Configuração do botão "Registrar"

Para colocar um aspetto mais interessante no botão, optei por colocar um “padding” geral do botão como “16dp”.

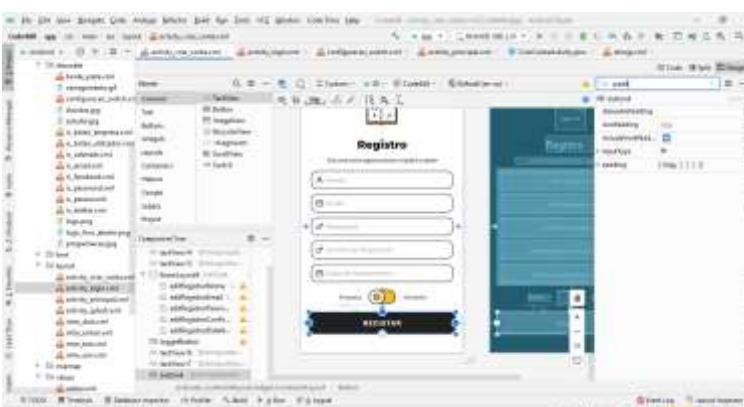


Figura 434: Colocação do padding do botão registar como "16dp"

Com a criação dos dois recursos String, passei à colocação de, mais uma vez, dois TextView's, sendo eles alinhados horizontalmente na ponta do botão (cada um) e verticalmente na parte inferior do botão de registo da conta. E para ficar tudo bem alinhado, apliquei uma “Horizontal Chain” nesses TextView's.

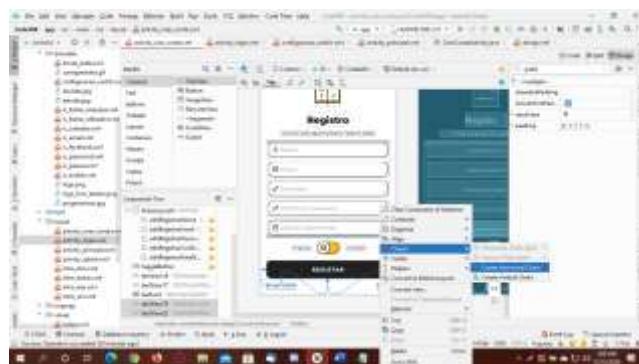


Figura 435: Alinhamento dos TextView's e da criação de uma "Horizontal Chain"

Com a criação e definição da “Horizontal Chain” o layout desta atividade ficou da seguinte forma.

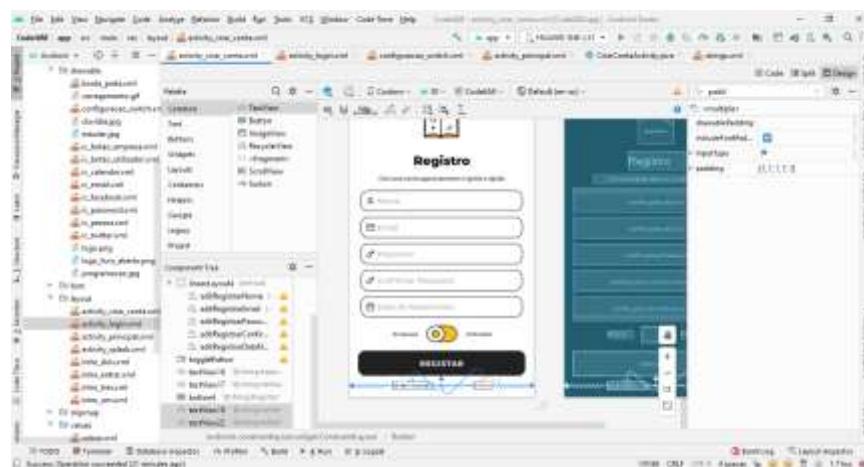


Figura 436: Efeito da "Horizontal Chain" no “rodapé”

Para dar melhor aspeto ao botão “Switch”, coloquei uma margem superior de 16dp, para lhe dar ele ficar, de certa forma, maior e ser mais fácil de clicar.

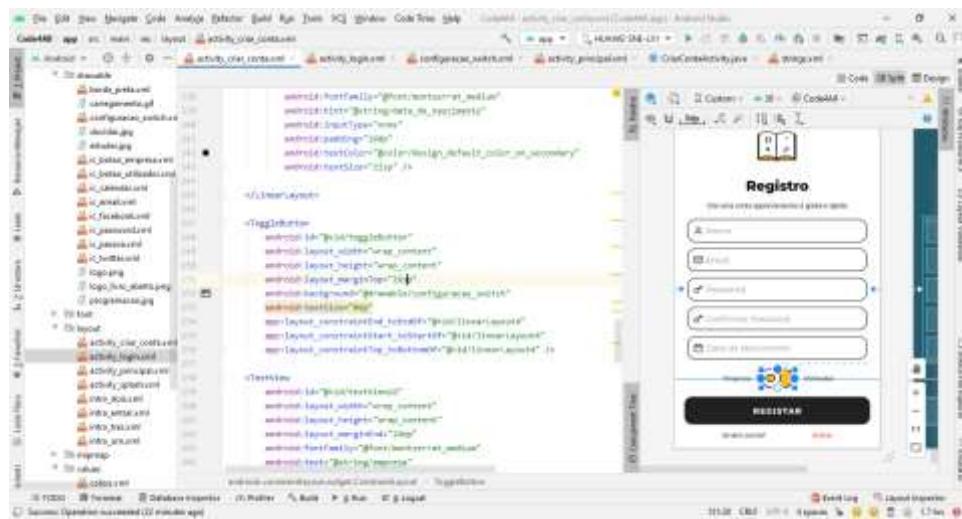


Figura 437: Definição do atributo "layout_marginTop" como "16dp"

Com esta última alteração, dei a realização da 1^a parte do layout desta atividade como finalizada.

Observei no Adobe XD que o layout desta atividade iria ser um pouco longo, logo teria de levar com um componente que permitisse o layout ser “scrollable” ou seja rolável.

O nome desse componente é ScrollView” e este deve ser o componente “raiz” da atividade, visto que assim irá tornar todos os outros componentes roláveis. Para tornar a sua inserção mais fácil, decidi editar diretamente pelo o XML, em vez da interface gráfica porque para mim é mais confuso.

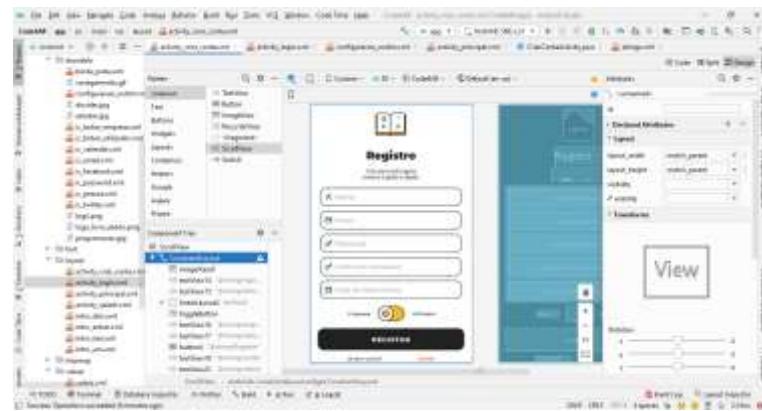


Figura 438: Colocação de um "ScrollView" dentro do layout

Com a introdução e inserção de mais um componente, mais uma vez procedi à execução da aplicação e observei na mesma, que o scroll já estaria a funcionar como programado e configurado anteriormente.

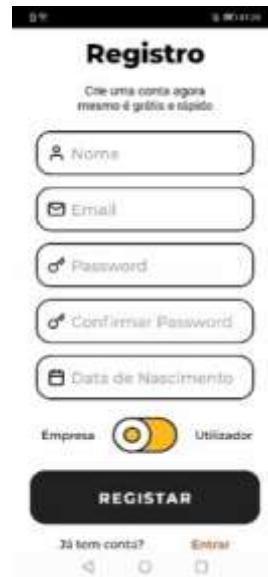


Figura 439: Execução da aplicação para testar o "Scroll"

Para dar ainda um efeito mais interessante, decidi colocar o atributo “padding” com 8dp, dentro dos dois TextView’s da zona do rodapé, sendo eles o “Já tem conta?” e o “Entrar”.

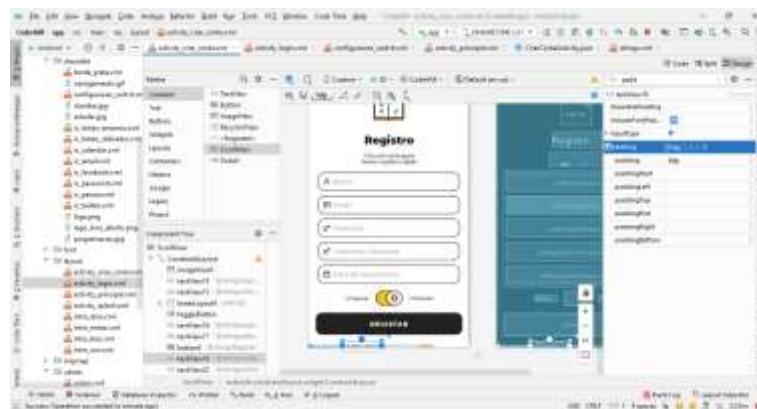


Figura 440: Definição do atributo "padding" de 8dp

Logo após, de ter feito aquela pequena alteração no “rodapé”, observei que a cor do botão estava preta e não era essa a finalidade, precisava então de mudar a cor do mesmo para a “corPrimariaEscura” (já criada no ficheiro “colors.xml”). Mudei a cor do botão com o atributo “app:backgroundTint”

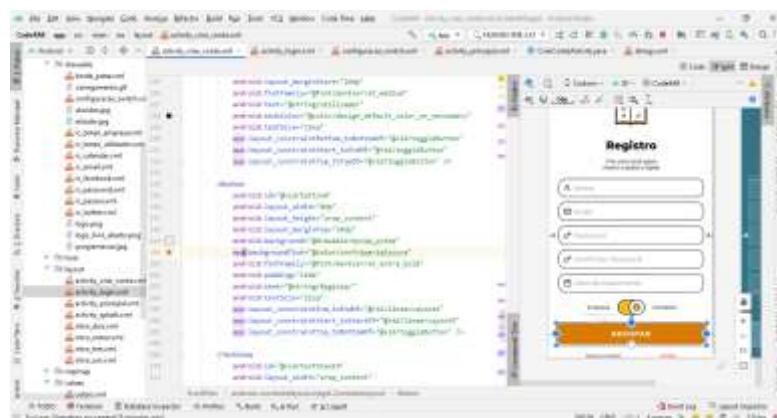


Figura 441: Modificação da cor do botão "Registrar"

Com as alterações que fiz anteriormente, decidi agora então ver se a app estava a executar conforme o que tinha programado, tanto na minha cabeça, como em código.



Figura 442: Execução da app, no meu SmartPhone

De imediato, voltei para a IDE e observei que tinha um aviso no componente “ScrollView”. Passei logo então à solução do mesmo, a mesma era bastante fácil, era só clicar no botão “Fix”. O que esta solução fazia era trocar o conteúdo do atributo “size” por “wrap_content”

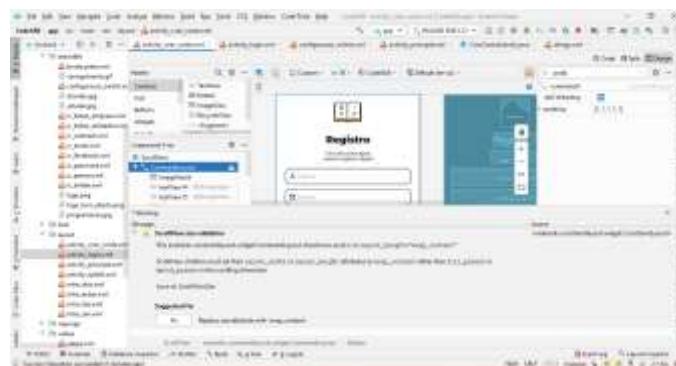


Figura 443: Aviso "size" do ScrollView

Com o layout minimamente configurado, passei à organização do código desta atividade, com a criação de um método “configuracoesIniciais” onde são colocadas, como o próprio diz, as definições básicas dos componentes, para então dar início à atividade. Adicionei 5 variáveis privadas (apenas usáveis nesta atividade) do tipo “EditText”, referentes aos campos presentes nesta atividade,

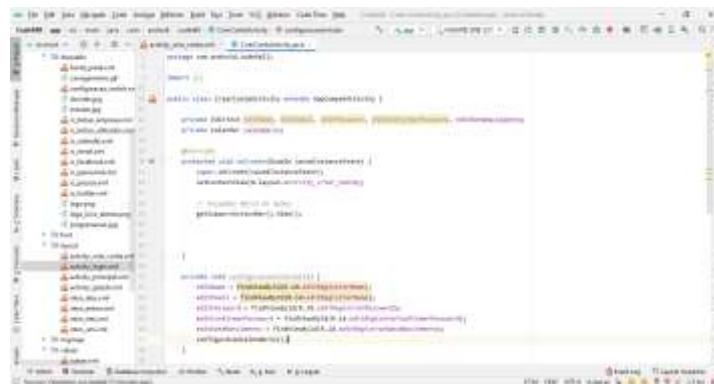


Figura 444: Organização do código com a criação do método "configuracoesIniciais"

Estas variáveis privadas, são bastante importantes, pois são com elas que irei obter de volta o texto que foi escrito pelo o utilizador nesta atividade. Na imagem abaixo, representa-se ainda o restante código, já anteriormente explicado.



Figura 445: Restante código presente na atividade "CriarConta"

Com o código organizado, consoante a minha organização, o passo seguinte que realizei, foi criar um novo método que iria tratar o clique do botão “Registar” (do layout desta atividade). O método ficou com o nome de “registarUtilizador”, pois era essa a sua função principal, para além de validar os campos, nesse método tem de ser passado ainda um parâmetro do tipo “View”, visto que iria atribuir este método diretamente pelo o código em XML.



Figura 446: Criação do método "registarUtilizador"

Completando o que disse anteriormente, fui ao layout, cliquei no botão “Registar”, dirigi-me ao atributo “onClick” e escrevi o nome do método criado anteriormente (registarUtilizador), como se observa na figura.

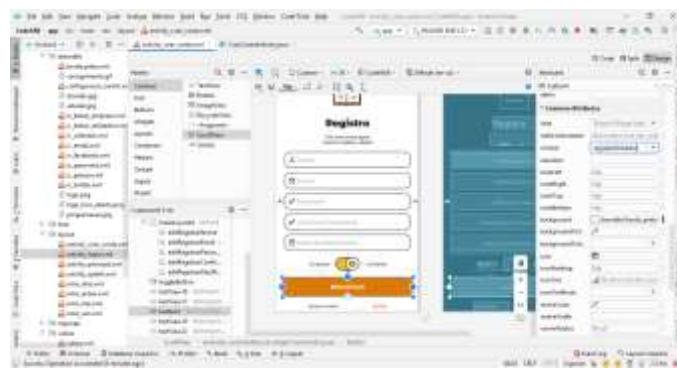


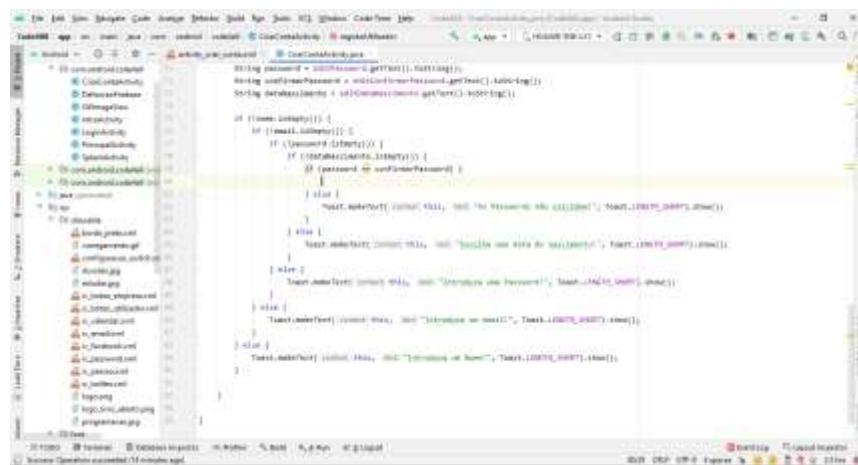
Figura 447: Definição do atributo "onClick" do botão Registar, pelo o método "registrarUtilizador"

Com o método “onClick” devidamente configurado, pude então dar início à programação da verificação dos campos, comecei então por obter o texto “String” do momento em que o utilizador clicasse no botão “Registar”, pois este método é chamado pelo o clique.



Figura 448: Obtenção dos dados de registro

Seguidamente, passei à verificação dos campos, se eles tivessem “vazios” iria surgir uma mensagem ao utilizador para preencher os mesmos, se não passava para a verificação seguinte, um à parte, esta lógica também foi usada na “LoginActivity”.



```

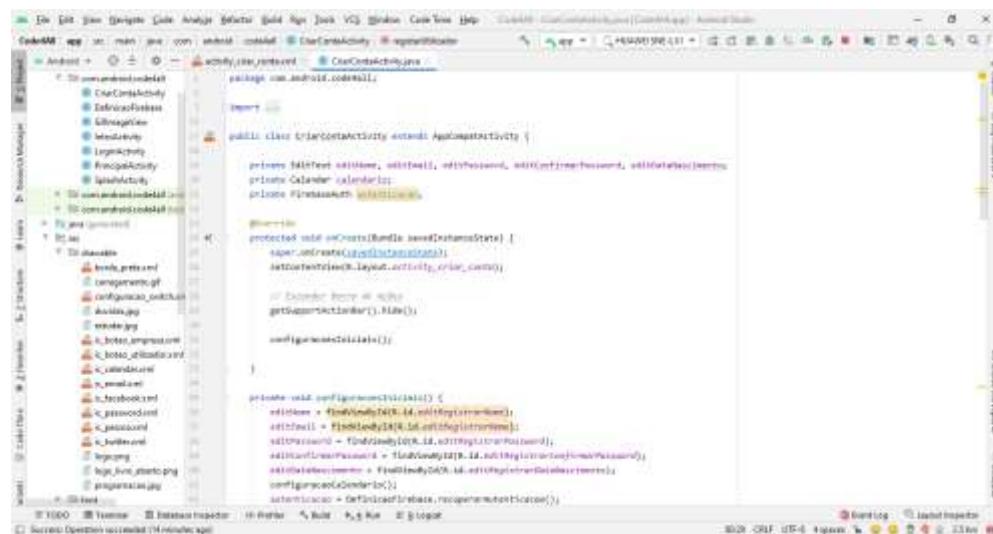
public class MainActivity extends AppCompatActivity {
    ...
    private EditText editTextEmail;
    private EditText editTextPassword;
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ...
        editTextEmail = findViewById(R.id.editTextEmail);
        editTextPassword = findViewById(R.id.editTextPassword);
        ...
        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ...
                String email = editTextEmail.getText().toString();
                String password = editTextPassword.getText().toString();
                ...
                if (email.isEmpty() || password.isEmpty()) {
                    Toast.makeText(getApplicationContext(), "Por favor, insira o seu e-mail e a sua palavra-passe.", Toast.LENGTH_SHORT).show();
                } else {
                    ...
                }
            }
        });
    }
}

```

Figura 449: Verificação dos campos de registro, se estão vazios ou não

Com o tratamento dos dados que realizei anteriormente, foi preciso adicionar uma nova variável privada, chamada de “autenticacao” e do tipo “FirebaseAuth” e com esta é que irei realizar o registo de todos os novos utilizadores. Mais abaixo, no método “configuracoesIniciais”, defini o mesmo, com a classe “DefinicaoFirebase”, onde ficam os métodos que dizem respeito aos componentes da base de dados (Firebase), sendo o método neste caso “recuperarAutenticacao”.

Coloquei também, o método “configuracoesIniciais”, no método “onCreate”, para o mesmo ser executado, pois antes não lá estava e se não lá estiver, nenhum código é executado, porque a função não esta a ser chamada.



```

public class CreateContactActivity extends AppCompatActivity {
    ...
    private EditText editTextNome, editTextEmail, editTextSenha, editTextConfirmarSenha;
    private Calendar calendar;
    private EditText editTextDataNascimento;
    ...
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create_contact);
        ...
        editTextSenha = findViewById(R.id.editTextSenha);
        editTextConfirmarSenha = findViewById(R.id.editTextConfirmarSenha);
        ...
        ...
    }
    ...
    private void configurarEditText() {
        editTextNome = findViewById(R.id.editTextNome);
        editTextEmail = findViewById(R.id.editTextEmail);
        editTextSenha = findViewById(R.id.editTextSenha);
        editTextConfirmarSenha = findViewById(R.id.editTextConfirmarSenha);
        ...
        ...
    }
    ...
    private void configurarDataNascimento() {
        editTextDataNascimento = findViewById(R.id.editTextDataNascimento);
        ...
    }
}
    
```

Figura 450: Criação da variável privada "autenticacao" e definição da mesma

Com esta pequena configuração da variável “autenticacao”, poderia dar início ao processo do registro do utilizador.

A condição “password == confirmPassword”, verifica se o utilizador introduziu duas vezes a mesma palavra-passe.

Este método é utilizado pela a maior parte dos sistemas de autenticação, porque permite ao utilizador reenscrever a password duas vezes, eliminando grande parte das chances de se enganar ao escrever a mesma.

Logo abaixo, encontra-se uma linha de código “gigantesca”, mas tudo o que está escrito lá, tem o seu sentido. Começando então pela a variável “autenticacao” e o seu método “createUserWithEmailAndPassword”, que como o próprio nome indica, cria um utilizador com Email e Password, este método requer dois parâmetros, sendo eles o email e password do utilizador em questão.

Logo após, foi adicionado um ouvinte de finalização, o tal de “addOnCompleteListener”, que indica que o processo de criação de utilizador foi terminado, podendo ele ter dado certo ou errado. No método “onComplete”, tenho acesso à variável “task” que terá então o resultado da operação realizada.

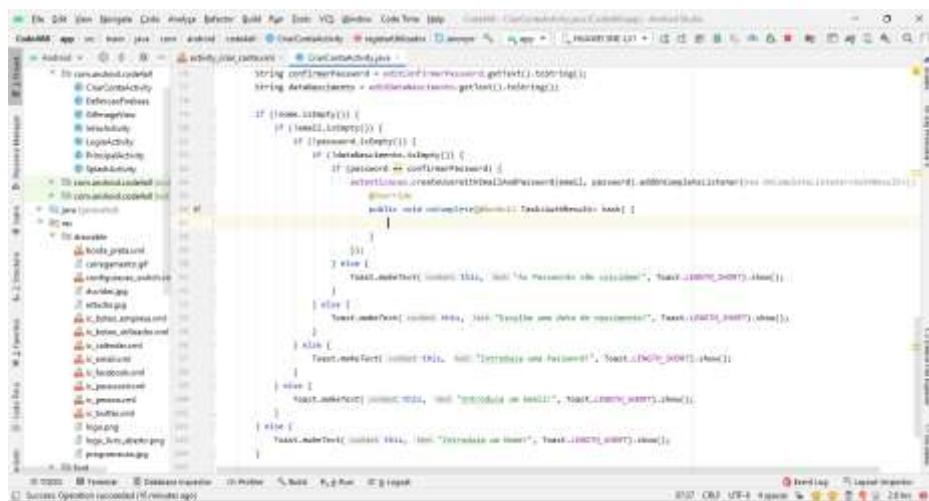


Figura 451: Condição "password=confirmarPassword" e código para criar utilizador

Com o código devidamente avançado, faltava agora os dados “cumprirem” os requisitos mínimos da base de dados (FireBase). Para isso fui procurar mais informações sobre o assunto e encontrei um “post” de um utilizador no StackOverflow com a mesma dúvida que a minha, o link e a captura de ecrã da mesma, encontram-se abaixo.

<https://stackoverflow.com/questions/37859582/how-to-catch-a-firebase-auth-specific-exceptions/>



Figura 452: Exceções para a criação de um utilizador (Email e Password)

Para realizar essa verificação, foi preciso realizar uma pequena pesquisa, pois são precisos erros específicos para a criação a conta, como por exemplo o incumprimento de caracteres mínimos para a password. Podem haver até três possíveis erros, sendo eles:

- FirebaseAuthWeakPasswordException -> Palavra-passe curta
- FirebaseAuthInvalidCredentialsException -> Email/Pasword inválidos
- FirebaseAuthUserCollisionException -> Duplicação do mesmo utilizador

Com a pesquisa dada com concluída, era só então aplicar os conhecimentos retirados da mesma e aplicar no código da atividade.

Dei início com a criação de mais uma estrutura de decisão “IF/SE”, verificando se o processo tinha sido bem executado “task.isSuccessful()” e se esse fosse o caso, o utilizador iria ser redirecionado para a “PrincipalActivity”.

Se não, a condição “!task.isSuccessful()” que é o mesmo que anterior, só com a diferença de ter um ponto de exclamação, negando a expressão em si, o mesmo de, não ter sucesso. Seguidamente, criei uma variável do tipo String chamada de “erro”, que viria a conter o texto do “erro” que viria ser apresentado ao utilizador.

Coloquei ainda, uma estrutura já conhecida neste relatório, “Try Catch”, para verificar se algum dos erros presentes foi ocorrido e se algum for, será guardado na variável “erro”, a sua devida mensagem, para mais tarde ser mostrada, tal e qual como se verifica na captura de ecrã abaixo este tipo de abordagem é realizado para ser dado, de certa forma, algum tipo de feedback ao utilizador do que ele realizou de errado.

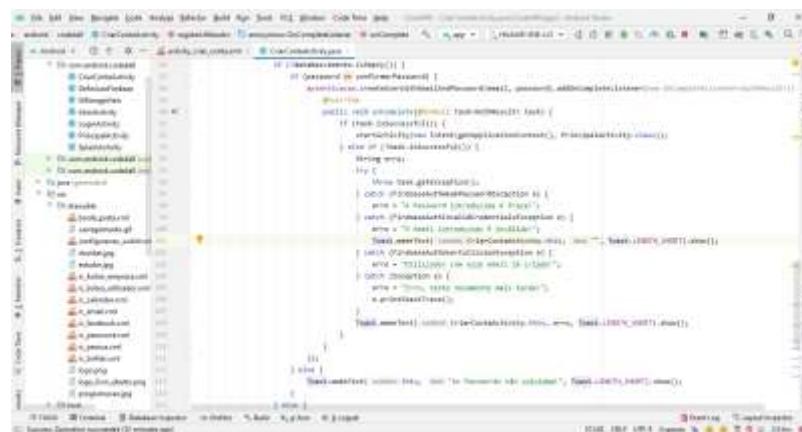


Figura 453: Configuração das verificações da Firebase

Agora sim havia configurado tudo bem e tudo estaria a funcionar conforme o que tinha planeado. Revi o código feito e realizei apenas duas alterações:

1. Na condição (`password == confirmPassword`), passei para `"password.equals(confirmPassword)"`, visto que estaria a comparar duas "String's" este é o método mais adequado para o efeito (`equals`).
2. Remoção do Toast (mensagem para a interface), na linha 101, visto que era inútil, porque mais abaixo é realizado já um aviso, mais precisamente na linha 107.

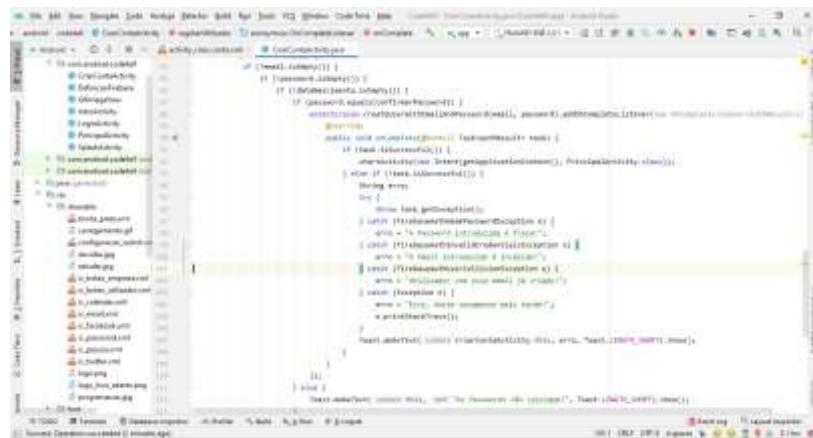


Figura 454: Revisão do código realizado

Depois de ter dado uma revisão no código e realizado as alterações necessárias para o bom funcionamento da app, decidi executar a aplicação no seu estado atual, para verificar se o código estava a correr como previsto.

Com a aplicação em execução, como se observa na figura, preenchi o formulário, apenas para testar, para ver se realmente iria para a “PrincipalActivity”.



Figura 455: Registro de um novo utilizador para testar o código anteriormente escrito

Cliquei no botão “Registrar” e fui levado para a “PrincipalActivity”, como estava planeado para ser feito, visto que todos os dados que introduzi no formulário estavam corretos e passariam em todas as verificações que fiz.



Figura 456: Redirecionamento do utilizador da CriarContaActivity para a PrincipalActivity

Em baixo, está presente uma captura de ecrã do painel de controlo da base de dados da app (Firebase), que apresenta todos os utilizadores criados até ao momento, ou seja, dá para concluir que o código que foi realizado está a ser de certa forma bem executado.

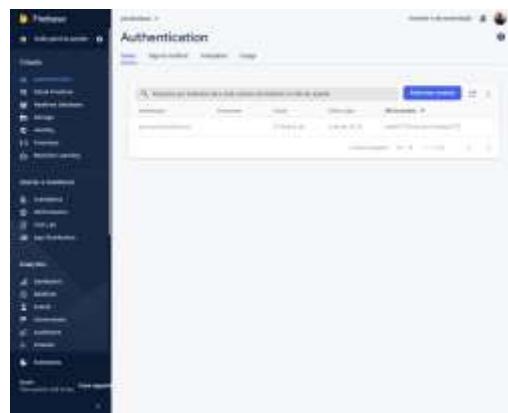


Figura 457: Listagem dos utilizadores do FireBase (naquele instante)

Com tudo isto, faltava agora, incorporar na mesma interface (para poupar recursos), as duas variantes de utilizador (Utilizador Normal ou Empresa). Para isso, precisaria de colocar um ouvinte no ToggleButton (botão switch amarelo do layout), para que consoante o seu estado, o layout havia de ser mudado, de acordo o tipo de utilizador.

<https://stackoverflow.com/questions/11776423/android-togglebutton-listener/>



Figura 458: Ouvinte para o "ToogleButton"

Com a pesquisa que realizei anteriormente, consegui uma suposta solução que consistia em criar uma variável do o ToogleButton e fazer a respetiva localização no layout com o código “findViewById(R.id.toggleButton)” e depois sim, definir o ouvinte “setOnCheckedChangeListener”, que em português significa definir um ouvinte de alteração de estado (mais coisa menos coisa). E basicamente, apliquei essa solução conforme explicado no StackOverflow, dentro desta atividade, como se observa na imagem.



Figura 459: Definição do ToogleButton e do seu ouvinte na atividade

Dirigi-me ao “feather icons” (site de ícones), e procurei pelo o que mais achava interessante e acabei por selecionar aquele que se encontra na seguinte imagem.

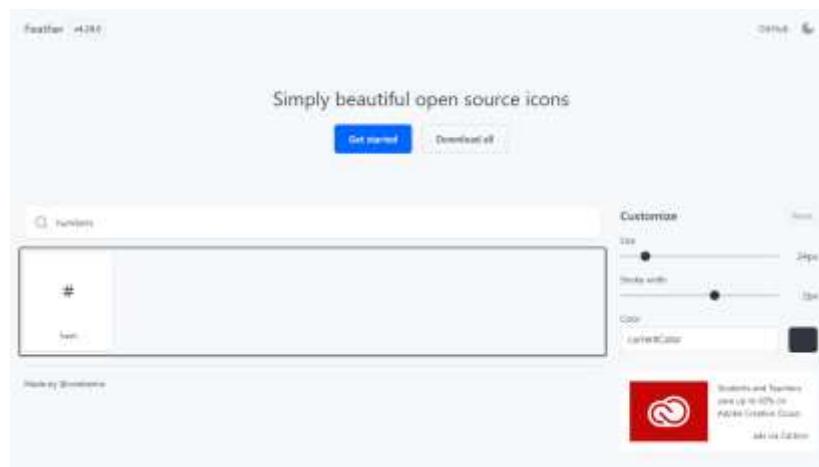


Figura 460: Pesquisa do ícone para o NIF

Com ícone já importado para dentro do projeto no Android Studio, faltava colocar uma cor que o próprio Android Studio reconhecesse, como nos outros ícones já importados este não é diferente e ficou com a cor “@android:color/white”.

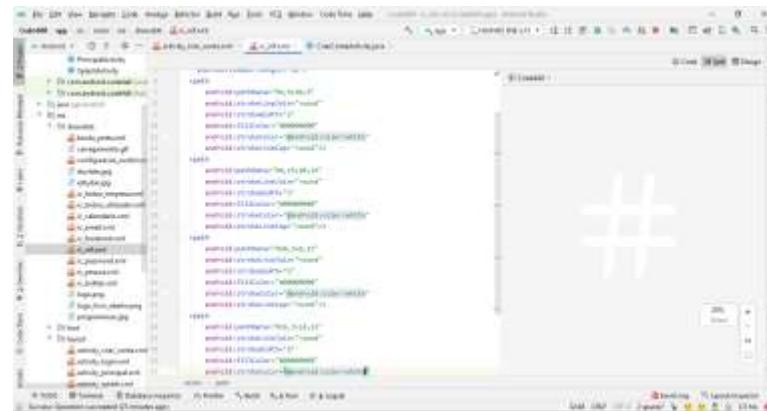


Figura 461: Definição da cor do ícone NIF (branco)

Com o ícone importado e configurado com sucesso, voltei então ao layout da atividade e dupliquei um campo (EditText), e alterei a sua posição para último e o atributo “hint” com o recurso string “nif” já anteriormente criado e depois associar o ícone NIF ao atributo “drawableStart” e para dar um efeito mais interessante coloquei no “drawablePadding” 8 dp.

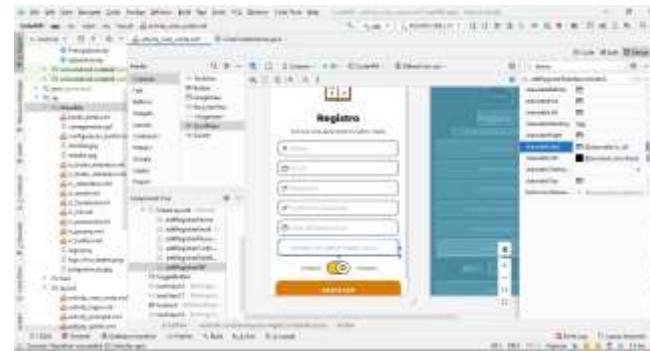


Figura 462: Configuração Campo do NIF (Empresas)

Nesta altura, apercebi-me que tinha feito um erro na configuração da cor do ícone “nif”, pois tinha configurado a cor do mesmo como branca (inicialmente), o que era errado, deveria ter configurado com a cor preta “@android:color/black”.

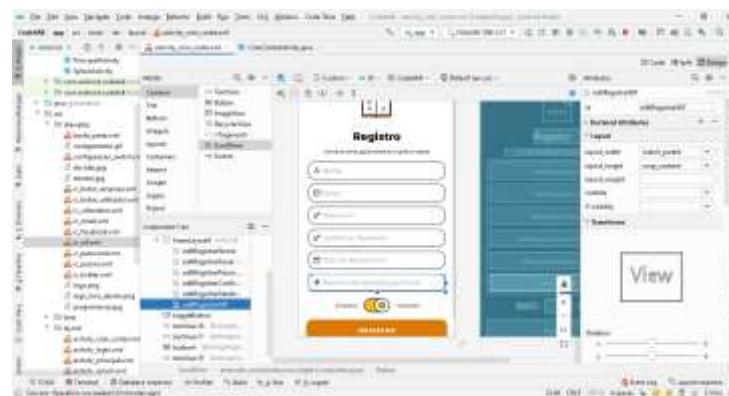


Figura 463: Configuração da cor do ícone "nif" como preto

Com a alteração da cor do ícone efetuada, voltei novamente para o arquivo Java desta atividade, para configurar o estado do botão. Essa configuração foi feita de certa forma, com o atributo que já era dado “isChecked” que significa, está selecionado.

Bastou então realizar uma condição SE/IF, que se estivesse selecionado, mostraria a “Data de Nascimento” e escondia o “NIF” se não, aconteceria exatamente ao contrário.

Na imagem abaixo, verifica-se a condição que tive de realizar para, de certa forma, o clique no botão (“switch” amarelo), fizesse algo no layout e esse algo era no caso da Empresa mostrar um campo NIF e esconder a data de nascimento e no caso de um utilizador (normal) efetuar o inverso.

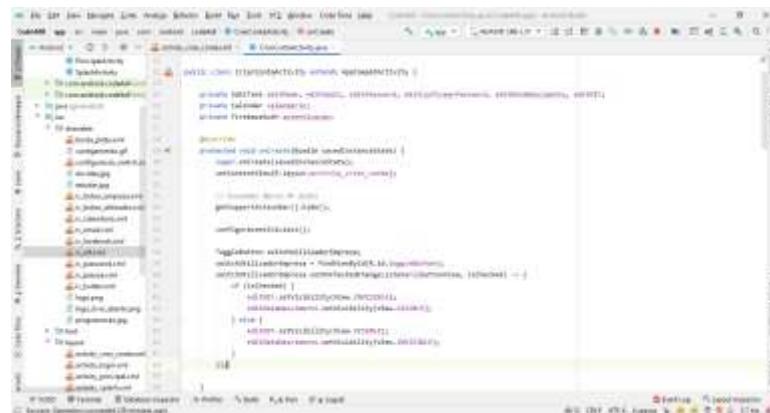


Figura 464: Condição IF/SE do ToogleButton (clique)

Na captura de ecrã que se encontra abaixo, dá para observar que o código em si está trocado, ou seja o utilizador está selecionado e o NIF está a ser mostrado, o que não deveria de acontecer e também a posição do NIF deveria subir, para substituir o espaço que não existe do campo da data de nascimento.



A registration form with fields for Name, Email, Password, Confirm Password, and Número de Identificação Fiscal. Below the fields is a switch button labeled 'Empresa' (selected) and 'Utilizador'. A large orange 'REGISTAR' button is at the bottom.

Figura 465: Switch – Utilizador selecionado

Na imagem abaixo, dá para observar o que acontece quando a empresa é selecionada e esta captura de ecrã verifica mais uma vez a minha opinião de que o código está a executar inversamente, ou seja, o que tenho de fazer é trocar o código de sítio e mudar de “INVISIBLE” para “GONE”.



A registration form with fields for Name, Email, Password, Confirm Password, and Data de Nascimento. Below the fields is a switch button labeled 'Empresa' (selected) and 'Utilizador'. A large orange 'REGISTAR' button is at the bottom.

Figura 466: Switch - Empresa selecionada

Então foi o que realizei, troquei o código de lugar na condição “if (isChecked)”, conforme disse anteriormente e adicionei o atributo “GONE” nos campos que deveriam estar escondidos.

A imagem abaixo demonstra então, o que foi alterado.

Figura 467: Ajuste do código no ouvinte do ToggleButton

Com a alteração no código que fiz anteriormente, decidi novamente executar a aplicação no meu dispositivo Android e de logo vi que aparentemente agora sim estaria a funcionar corretamente, visto que a opção “Utilizador” estava selecionada e apenas estava a ser exibida o campo de “Data de Nascimento”.



Figura 468: Switch - Utilizador (corrigido)

Na imagem abaixo, existe a vertente da seleção da opção “Empresa” e de facto o NIF estaria a ser visível e o campo “data de nascimento” e o seu espaço tinha sido “apagado”.



Figura 469: Switch - Empresa (corrigido)

Seguidamente, coloquei um aviso “Toast”, para que o utilizador soubesse que a criação de um(a) “Empresa/Utilizador” foi selecionado(a).

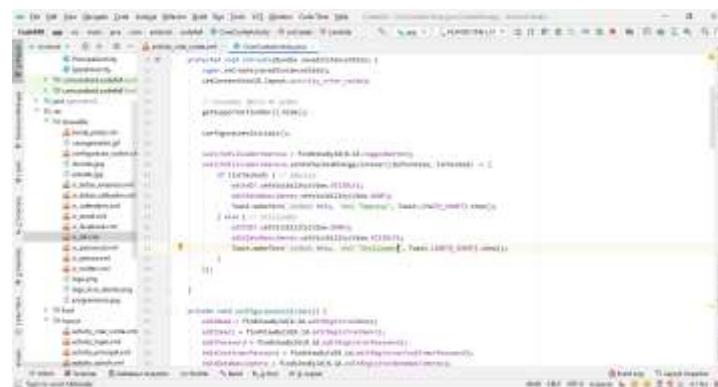


Figura 470: Avisos na tela com o tipo de utilizador a vir a ser criado

Depois, para reforçar a ideia do campo “NIF” não aparecer no início da atividade, decidi então colocar a visibilidade dela como “GONE” no método “configuracoesIniciais”.



Figura 471: Definição da visibilidade do campo NIF, como "GONE"

Coloquei também o campo do número de identificação fiscal, com o atributo “visibility” dado como “GONE”, para evitar que este seja mostrado no início da atividade e do seu layout agregado, tal como fiz anteriormente na atividade, mas por código em Java.

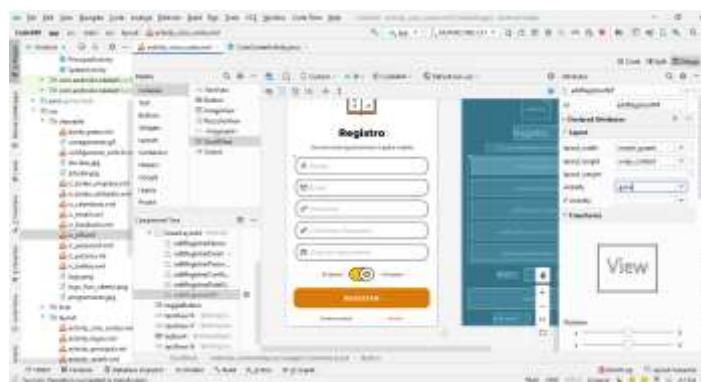


Figura 472: Definição da visibilidade do campo "NIF" como "GONE" (layout da atividade)

Aproveitei ainda e configurei o atributo “inputType” do campo NIF, pois os valores que serão introduzidos e fazem parte de um número de identificação fiscal, são apenas números. Para confirmar isso, coloquei então o atributo “inputType” com a opção “number” definida.

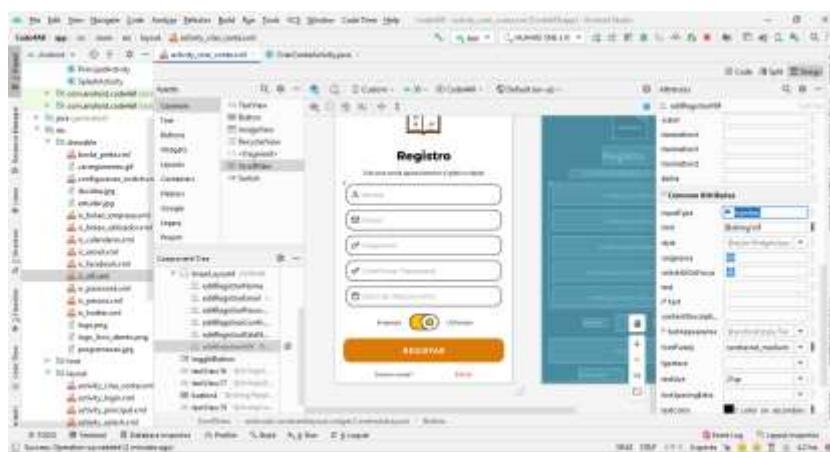


Figura 473: Definição do atributo "inputType" como "number"

Voltando para o ficheiro “CriarContaActivity.java”, faltava agora ajustar os dados consoante o tipo de utilizador (Normal/Empresa), comecei então por fazer a verificação no caso de o utilizador normal ter sido selecionado, como se observa no comentário do código.

Se fosse utilizador normal iria obter o valor do campo da “Data de Nascimento” e depois passaria pela uma nova condição IF, para testar se o campo havia sido preenchido, se não tivesse preenchido uma mensagem iria ser apresentada para o utilizador, para o mesmo escolher uma data de nascimento. Seguindo a lógica, um aviso “Toast” iria ser mostrado com o tipo de “Utilizador”. Por fim, o registo em si do utilizador era feito e o mesmo direcionado para a “PrincipalActivity”.

Se fosse empresa, a abordagem era a mesma de um utilizador normal, com a diferença do campo NIF em vez do campo “Data de Nascimento”.

Coloquei também no código uns pequenos comentários na condição IF criada, para de certa forma ajudar na visualização do código, tanto a minha, como a dos seus visualizadores.

```
public void registerUser(User user) {
    String name = user.getName();
    String email = user.getEmail();
    String password = user.getPassword();
    String confirmationEmail = user.getConfirmationEmail();
    String confirmationPassword = user.getConfirmationPassword();

    if (name.isEmpty()) {
        if (email.isEmpty()) {
            if (password.isEmpty()) {
                if (confirmationEmail.isEmpty() || !confirmationEmail.equals(email)) {
                    throw new IllegalStateException("All fields must be filled");
                }
                String errorMessage = "Email and confirmation email must match";
                throw new IllegalStateException(errorMessage);
            }
            if (confirmationPassword.isEmpty() || !confirmationPassword.equals(password)) {
                throw new IllegalStateException("Passwords must match");
            }
        }
        throw new IllegalStateException("Name and email must be filled");
    }
    if (!password.matches("[a-zA-Z0-9]{8,}")) {
        throw new IllegalStateException("Password must be at least 8 characters long");
    }
    User userExist = userRepository.findByEmail(name);
    if (userExist != null) {
        throw new IllegalStateException("User already exists");
    }
    User userCreated = userRepository.save(user);
    String confirmationLink = confirmationEmailService.createConfirmationLink(userCreated);
    confirmationEmailService.sendEmail(userCreated.getEmail(), confirmationLink);
}
```

Figura 474: Verificação do Tipo de Utilizador (Empresa/Utilizador Normal)

Depois, lembrei-me que faltava ainda colocar um atributo “onClick” dentro do texto “Entrar” no rodapé, para que ao ser clicado levasse o utilizador para a “LoginActivity”. E selecionei então o método “entrarConta”, como se observa na figura da próxima página.

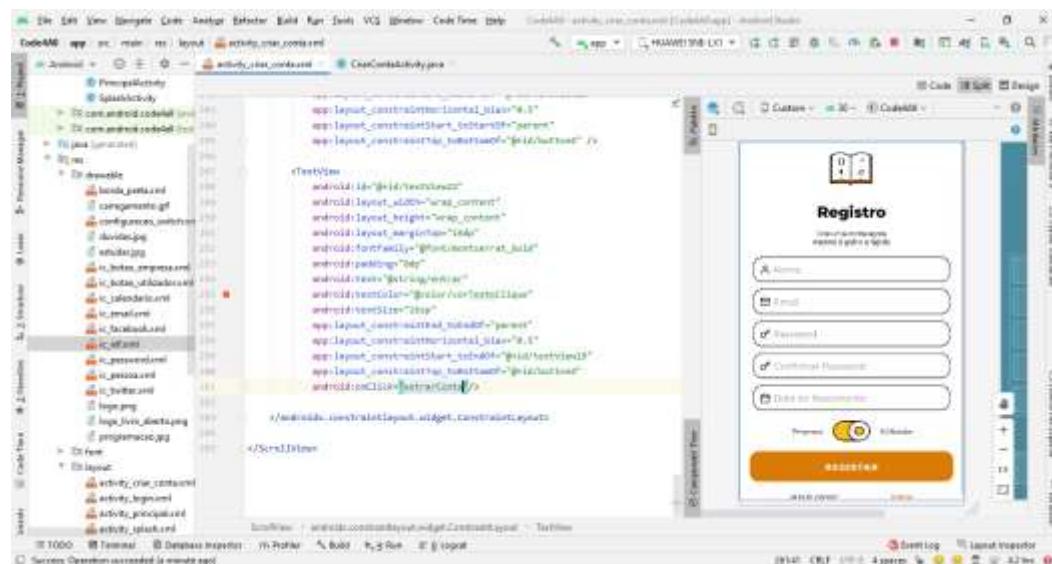


Figura 475: Definição do atributo "onClick" com o método "entrarConta" no TextView

Seguidamente, passei à criação do método “entrarConta” dentro do ficheiro java da atividade, que iria começar a atividade “LoginActivity” com o código “startActivity”. Desta vez, para ter uma abordagem diferente das outras realizadas anteriormente, decidi fazer a criação de uma “Intent” (pode se dizer que é um objeto onde ficam guardadas várias funções internas que dizem respeito ao Sistema em si), passando o contexto do tipo “Context” e a classe da atividade em questão “LoginActivity.class”.

Optei por realizar um novo tipo de abordagem, para tanto testar os meus conhecimentos aprendidos no curso que realizei, tanto para variar a escrita de código e também insinuar que existem diversas formas na programação de chegar à mesma solução

Totalizando assim o código “new Intent(getApplicationContext(), LoginActivity.class)”. É ainda usado o “new” para realizar a instanciação de um novo objeto. Houve também a necessidade de adicionar dois parâmetros dentro do método “registroUtilizador”, sendo eles email e password.

```
private void registerUser(String email, String password) {
    authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(email, password));
    User user = new User();
    user.setEmail(email);
    user.setActive(true);
    user.setRole("ROLE_USER");
    user.setSalt(UUID.randomUUID().toString());
    user.setPassword(passwordEncoder.encode(password));
    user.setActivationCode(UUID.randomUUID().toString());
    user.setActivationCodeExpireTime(LocalDateTime.now().plusMinutes(10));
    userRepository.save(user);
}

private void checkEmailAvailability(String email) {
    User user = userRepository.findByEmail(email);
    if (user != null) {
        throw new IllegalStateException("Email already exists!");
    }
}

private void checkActivationCode(String activationCode) {
    User user = userRepository.findByActivationCode(activationCode);
    if (user == null) {
        throw new IllegalStateException("Activation code is invalid!");
    }
}

private void checkUserIsActivated(String activationCode) {
    User user = userRepository.findByActivationCode(activationCode);
    if (user == null || !user.isActivated()) {
        throw new IllegalStateException("User is not activated yet!");
    }
}

private void checkUserIsNotActivated(String activationCode) {
    User user = userRepository.findByActivationCode(activationCode);
    if (user != null && user.isActivated()) {
        throw new IllegalStateException("User is already activated!");
    }
}

private void checkUserIsNotBlocked(String activationCode) {
    User user = userRepository.findByActivationCode(activationCode);
    if (user != null && user.isBlocked()) {
        throw new IllegalStateException("User is blocked!");
    }
}

private void checkUserIsBlocked(String activationCode) {
    User user = userRepository.findByActivationCode(activationCode);
    if (user != null && !user.isBlocked()) {
        throw new IllegalStateException("User is not blocked!");
    }
}
```

Figura 476: Criação do método "entrarConta" e criação de dois parâmetros no "registroUtilizador"

Depois de ter configurado os parâmetros devidamente, pude retirar as linhas comentadas, visto que se não estivessem estariam a ser executadas e como estava a fazer testes na app e no código escrito, não era boa ideia estar de facto a criar utilizadores sem sentido.

```
    public void test() {
        String name = "John Doe";
        String address = "123 Main Street";
        String city = "Anytown";
        String state = "CA";
        String zip = "90210";
        String phone = "(555) 123-4567";
        String email = "john.doe@example.com";
        String website = "http://www.johndoe.com";
        String bio = "A well-known entrepreneur and philanthropist.";
```

```
        User user = new User(name, address, city, state, zip, phone, email, website, bio);
        assertEquals("John Doe", user.getName());
        assertEquals("123 Main Street", user.getAddress());
        assertEquals("Anytown", user.getCity());
        assertEquals("CA", user.getState());
        assertEquals("90210", user.getZip());
        assertEquals("(555) 123-4567", user.getPhone());
        assertEquals("john.doe@example.com", user.getEmail());
        assertEquals("http://www.johndoe.com", user.getWebsite());
        assertEquals("A well-known entrepreneur and philanthropist.", user.getBio());
```

Figura 477: Remoção dos comentários nas linhas de execução do método "registrarUtilizador"

De certa forma, passei nesta altura, à criação de um objeto “Utilizador” que seria usado para OOP (Object-Oriented Programming ou Programação Orientada a Objetos). Decidi efetuar a criação da mesma, para facilitar o processo de envio de dados para a FireBase.

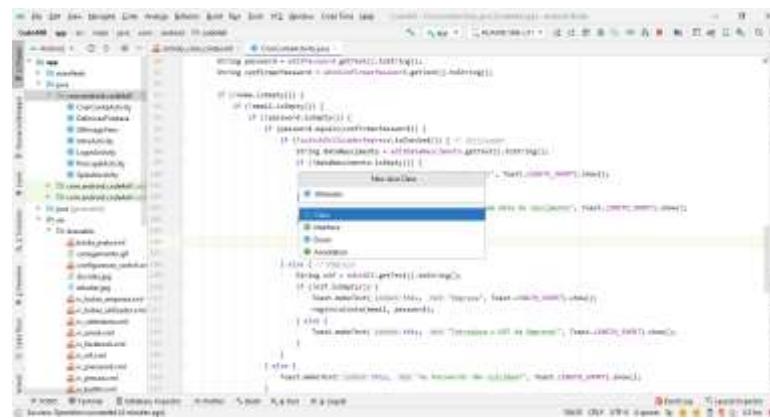


Figura 478: Criação da classe "Utilizador"

Depois, já com a classe criada, fiz a criação dos quatro dados que possivelmente iria precisar para enviar à base de dados desta app, todos relacionados com o utilizador, sendo normal ou empresa, até como dá para observar na figura, tenho dois parâmetros dos dois tipos de conta, a “dataNascimento” e o “nif”.

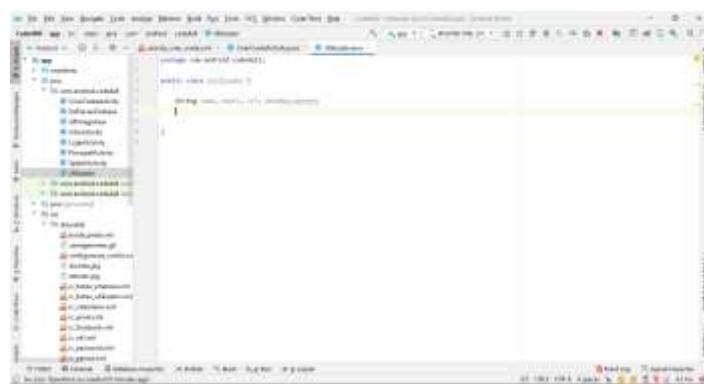


Figura 479: Configuração das variáveis da classe Utilizador

Chegou à hora de explicar o que são “Getter and Setter”, em programação, são métodos específicos para a OOP, os métodos “Getter” são para obter o valor da variável em questão e os métodos “Setter” são para “setar” (definir) o valor da variável. Este menu presente na print, pode ser acedido pelo o atalho “ALT + INSERT”.

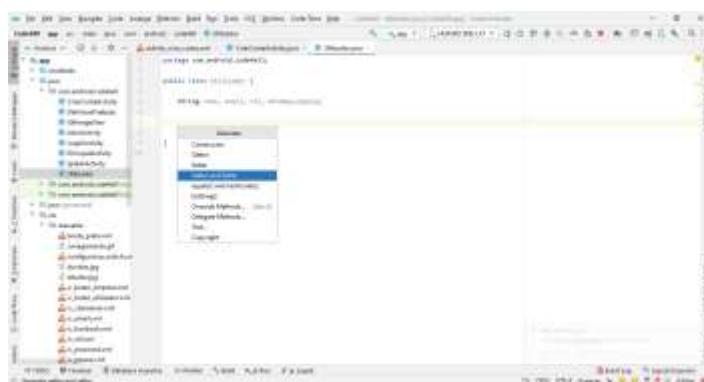


Figura 480: Menu de gerar métodos e outros

Seguidamente, após de ter selecionado a opção “Getter and Setter” do menu anterior, foi me aberta a seguinte janela (que se representa abaixo), pedindo as variáveis que queria para criar então os métodos, como viria a usar todas as variáveis que criei, selecionei todos.

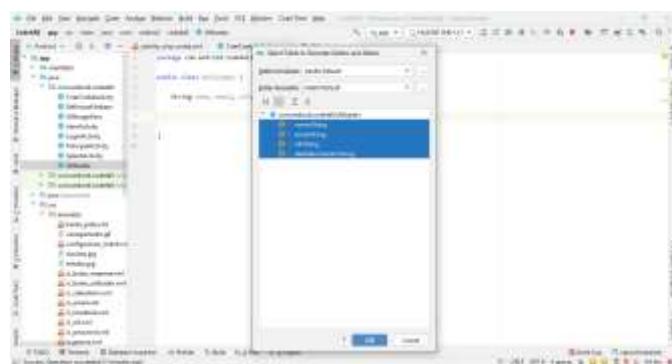


Figura 481: Seleção das variáveis para a criação dos "Getter's and Setter's"

Com o clique no botão “OK”, os métodos foram instantaneamente criados, como se pode observar na figura abaixo e todos nomeados consoante as variáveis que selecionei, como já referi anteriormente.



```

public class Utilizador {
    String nome;
    int idade;
    double peso;
    double altura;
    String email;
    String telefone;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }

    public double getPeso() {
        return peso;
    }

    public void setPeso(double peso) {
        this.peso = peso;
    }

    public double getAltura() {
        return altura;
    }

    public void setAltura(double altura) {
        this.altura = altura;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }
}

```

Figura 482: "Getter's and Setter's" da classe Utilizador

Criados os métodos “Getter and Setter” na classe “Utilizador” e pensando no nome desta classe, não fazia muito sentido na minha cabeça, com isso, decidi mudar o nome da mesma para a classe “Conta”, o que faria muito mais sentido, pelo menos na minha cabeça, pois eu associava o nome “utilizador” ao tipo de conta desta app, podendo ser utilizador ou empresa.

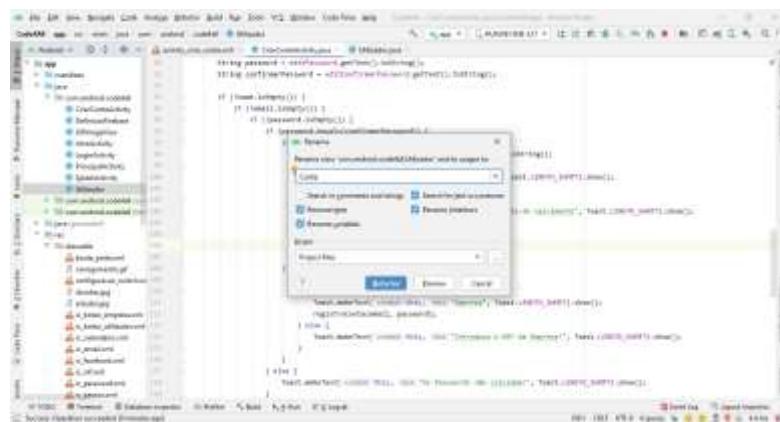


Figura 483: Renomeação da classe "Utilizador" para "Conta"

Com a classe “Conta”, antiga “Utilizador”, configurada com os métodos e as variáveis, tal como visto anteriormente, podia então voltar para o arquivo Java da atividade e realizar a criação de uma variável desse mesmo tipo e definir os seus “setters” como se pode visualizar abaixo.

Comecei por “setar” as três variáveis da conta “Utilizador”, sendo elas, nome, email e data de nascimento. Cada uma com o texto, que provém de cada um dos campos no layout, como já visto e explicado. Por fim, adicionei ainda um método “guardar()” que seria utilizado para enviar os dados para a FireBase (Realtime Database ou base de dados em tempo real), esta vertente do FireBase vai ser bastante usada no desenvolvimento desta app, visto que irá guardar todos os quizzes, informações do utilizador, pontuações, todo o tipo de dados irão ser guardados na mesma.

Esta Firebase Database, utiliza ainda NoSQL, que basicamente é um tipo inverso à base de dados relacionais (MySQL), pois não contém nenhuma relação e são usados apenas “nós” para identificar os dados, mais à frente neste relatório explicarei melhor.

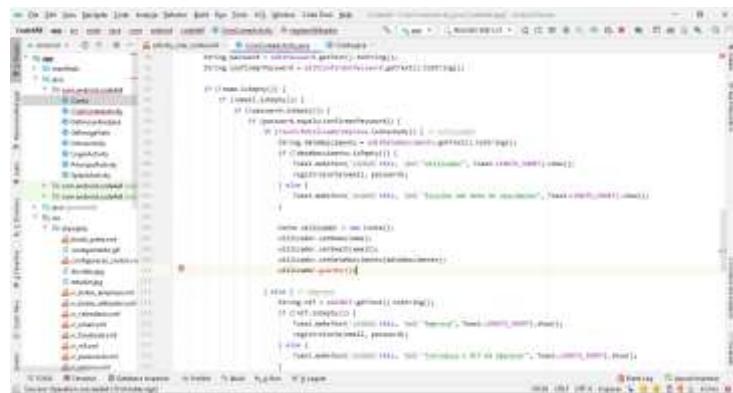


Figura 484: Criação de uma variável do tipo "Conta" e definição dos métodos

Antes de fazer o envio de dados para a Firebase Database, tive de importar a mesma, dentro do ficheiro “build.gradle” ao nível da “app” com o código “implementation ‘com.google.firebaseio:firebase-database’” e depois guardar as alterações para a IDE transferir todos os ficheiros que preciso para desenvolver com esta vertente do Firebase.

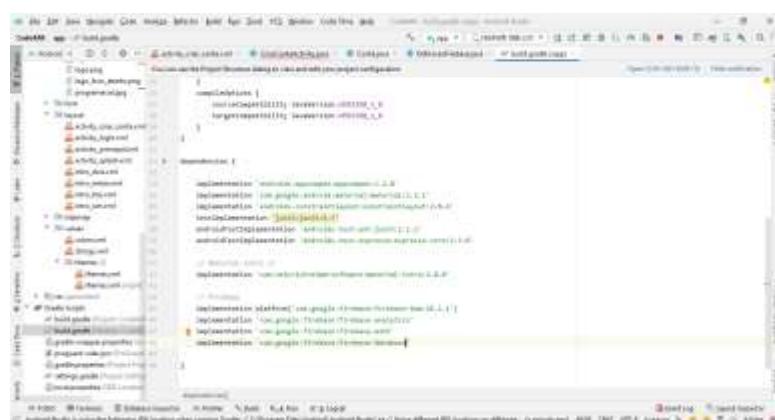


Figura 485: Importação da Firebase Database para dentro do projeto

Com a Firebase Database importada, dirigi-me ao ficheiro “DefinicaoFirebase.java”, para então realizar a criação de um novo método que retornaria a referência da base de dados para então realizar o “upload” dos dados para a FireBase. A criação deste método “recuperarBaseDados” segue a mesma lógica do que o método de “recuperarAutenticacao()”, o que varia é o nome e tipo da variável e consequentemente o que retorna o método.

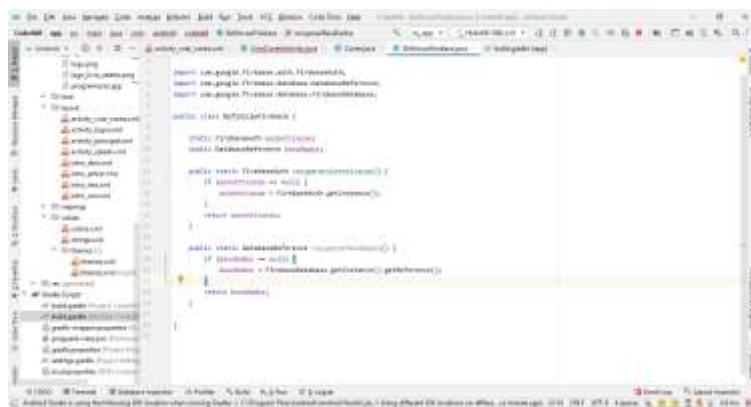


Figura 486: Criação do método "recuperarBaseDados"

No ficheiro “Conta.java”, pode então criar o método “guardar()”. No mesmo, coloquei três linhas de código essenciais para o envio de dados para a Firebase Database. Tive então de criar a variável “auth” do tipo “FirebaseAuth” e “conta” do tipo “DatabaseReference”, ambas a ir buscar dados do ficheiro “DefinicaoFirebase.java”.

Depois, bastou apenas “pegar” na variável conta, que já me retornaria o caminho da base de dados.

Seguidamente, colocar o código “.child()” para avançar na criação de um nó, sendo eles “contas”, mais uma vez usei o mesmo código, desta vez para criar um nó com o UID (identificação única do utilizador) do utilizador a realizar o registo.

Uma observação, que pretendo salientar, é que primeiramente é criado o utilizador no Firebase Auth é preciso ser realizado primeiro este passo, para depois sim retirar o UID que já é gerado automaticamente pelo o Firebase e aí então enviar os dados para a Firebase Database (em tempo real).

E por fim, o código “`setValue(this)`” para enviar todos os dados não “nulos/vazios” que estão preenchidos no objeto em questão. Estes dados serão mais tarde configurados no próprio ficheiro Java desta atividade.

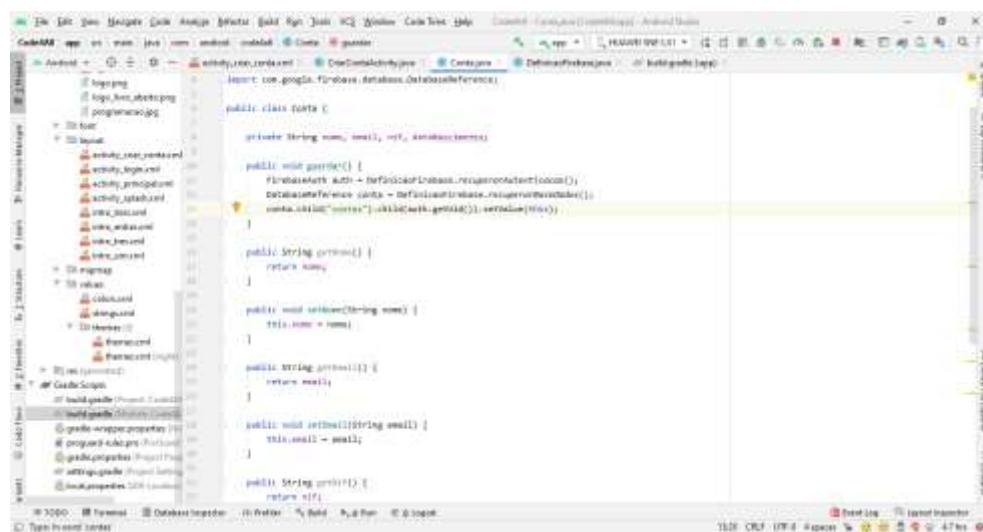


Figura 487: Criação do método "guardar" dentro da classe "Conta"

Com o código mais ou menos orientado, faltava realizar a criação da base de dados em tempo real, para que pudesse usar essa funcionalidade da Firebase.

Para isso, dirigi-me então para o “painel de controlo” do projeto (esta abordagem é feita pelo o site oficial da firebase).

E fui no menu lateral e cliquei na opção “Realtime Database” que me levou à página que se demonstra abaixo e depois então cliquei em “Criar Banco de dados”.

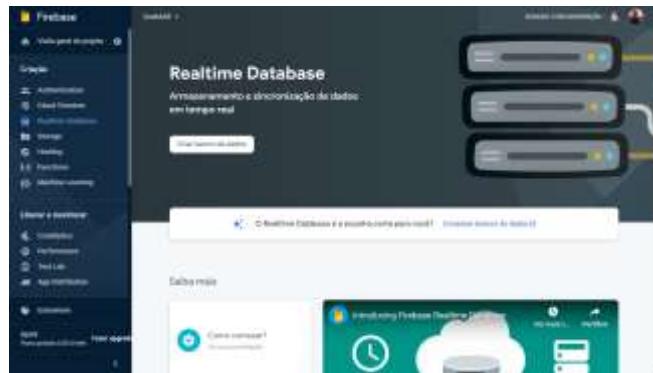


Figura 488: Página "Realtime Database" no painel de controlo do projeto na Firebase

De logo, apareceu-me esta janela, onde selecionei o modo de teste, pois é o que estou a fazer atualmente na app. Uma observação é que na localização da base de dados selecionei “Europe/West”, ou seja, a localização mais próxima a Portugal.

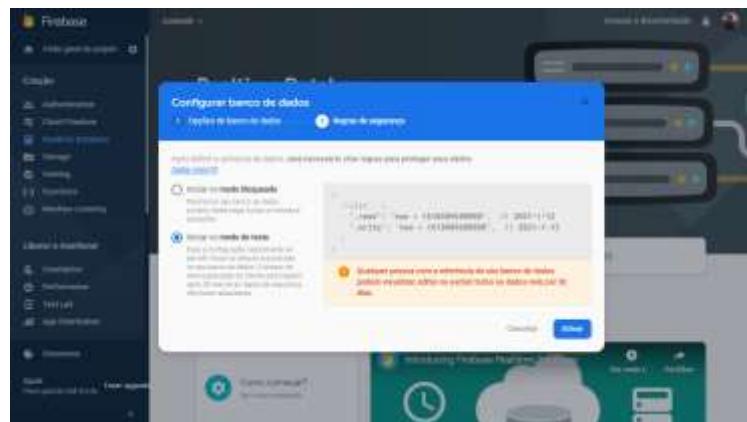


Figura 489: Escolha das Regras de segurança

Depois fui redirecionado para esta página e é nela que irão se encontrar os dados que forem enviados da app (Android) e também é possível os apagar por esta interface via web.

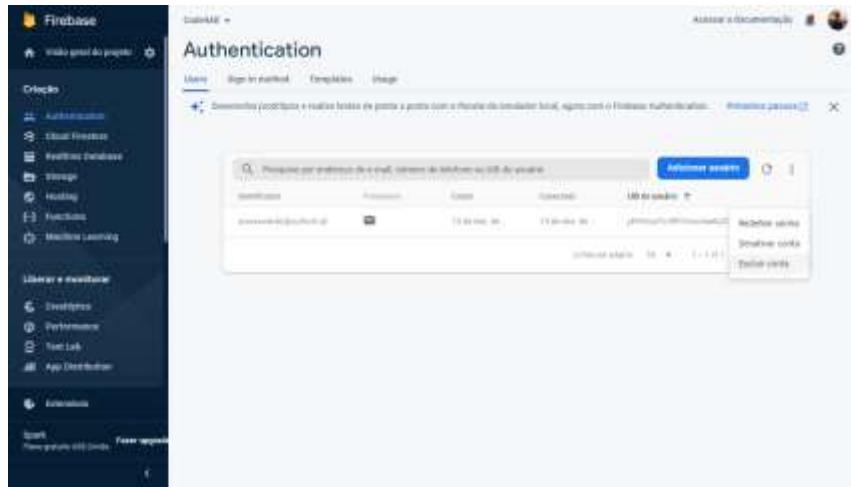
O local do banco de dados mais especificamente encontra-se na Bélgica.



The screenshot shows the Firebase Realtime Database interface. On the left, there's a sidebar with project settings like 'Visão geral do projeto', 'Realtime Database', 'Storage', 'Hosting', etc. The main area is titled 'Realtime Database' with tabs for 'Rules', 'Regras', 'Início', and 'Logs'. It displays a hierarchical tree structure of data under 'Realtime Database' with a single node named 'ETPC-Belgium-Hub'. Below the tree, there's a section for 'Documentação' (Documentation) and a 'Logs' tab.

Figura 490: Página "Realtime Database"

Depois da configuração da “Realtime Database” decidi então eliminar o único utilizador que tinha registrado como teste na app, para então voltar a testar tudo de novo para me verificar a 99.99% que está tudo a executar como esperado.



The screenshot shows the Firebase Authentication interface. The left sidebar includes 'Realtime Database' and 'Storage'. The main area is titled 'Authentication' with tabs for 'Users', 'Sign-in methods', 'Email/Password', and 'Image'. Under 'Users', there's a table listing a single user: 'josexavier46@outlook.pt'. The table columns include 'Identificador', 'Email', 'Nome', 'Último login', and 'Último acesso'. There are also buttons for 'Adicionar usuário' (Add user), 'Remover usuário' (Delete user), and 'Recuperar senha' (Reset password).

Figura 491: Eliminação do utilizador "josexavier46@outlook.pt"

Passei à execução da app, e procedi à criação de um utilizador normal, os dados que introduzi estão visíveis na captura de ecrã que realizei no meu Huawei Mate 20 Lite.



Figura 492: Captura de ecrã com o registo de um novo utilizador

Para ver se o utilizador foi registrado com sucesso, fui mais uma vez ao painel de controlo web deste projeto (Firebase) e verifiquei que na Firebase Auth, estava tudo a ser registado como devia de ser visto que tinha apagado o outro utilizador e de novo apareceu outro com o mesmo email.

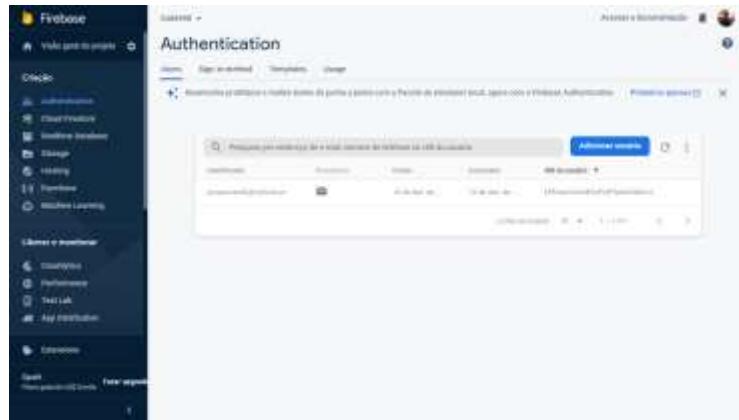


Figura 493: Verificação do utilizador na Firebase Auth

Faltava agora ver, se algo foi adicionado à Firebase Database, pois como codifiquei anteriormente, estava planeado os dados do utilizador irem para a mesma, mas quando abri a página, observei que de certa forma, não estava nenhum dado lá, ou seja, tinha algo de errado no código ou a base de dados não estava a receber os dados bem.

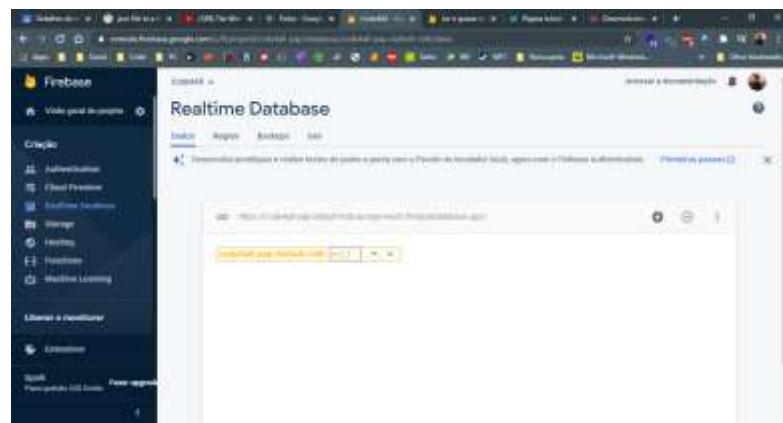


Figura 494: Listagem dos "nós" vazia

Decidi criar mais uma variável na classe “Conta”, desta vez chamada de “id” (sendo todas elas do tipo “String”), à semelhança das outras variáveis tive também de configurar um “Getter and Setter” para esta.

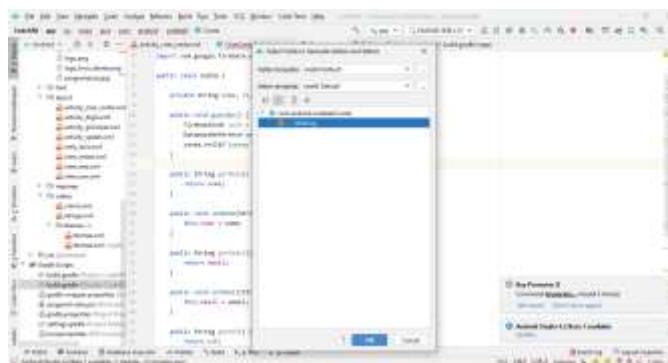


Figura 495: Configuração de uma nova variável "id" e do seu "Getter and Setter"

Depois de ter criado o “Getter and Setter” para a variável “id”, a classe “Conta” ficou da seguinte forma, como mostrado na imagem. Acabei também por remover a variável “autenticacao” pois não estava lá a fazer absolutamente nada.

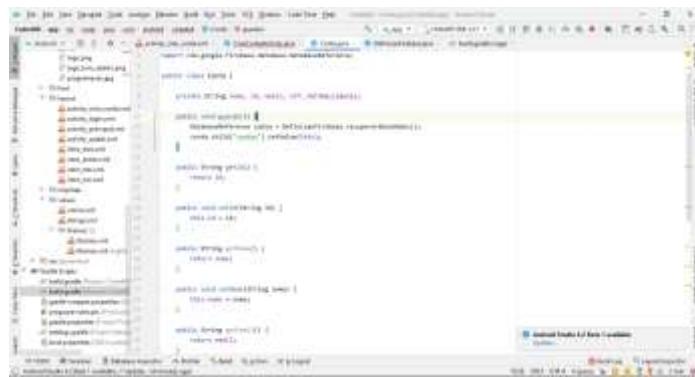


Figura 496: Classe "Conta" após à criação da variável "id" e de um "Getter and Setter" para a mesma

Seguidamente, voltei para o ficheiro “CriarContaActivity”, para fazer a definição da nova variável “id” com o id do utilizador, cheguei a esse ponto com o código “autenticacao.getCurrentUser().getUid()” que me retornaria o “uid” do utilizador, sendo a mesma coisa que “ID”.



Figura 497: Definição do "id" do utilizador

Para dar uso ao campo “id” anteriormente criado, tive então de fazer uma alteração na parte do envio dos dados (na linha 12), e adicionar mais um “child(this.id)”, o código “this.id” retorna-me o valor “atual” da variável “id”, poderia também ter feito doutra maneira e ter colocado “getId()” iria sempre obter o mesmo resultado, mas mais uma vez para variar o tipo de código decidi então fazer desta maneira.



Figura 498: Alteração na linha 12 - child(this.id)

Depois destas alterações no código decidi experimentar novamente a execução da app e a criação de mais um novo utilizador e para isso comecei por novamente, eliminar o utilizador que tinha sido anteriormente criado.



Figura 499: Exclusão do utilizador criado anteriormente em testes

Depois de ter eliminado o utilizador antes criado, voltei a executar mais uma vez a app, para ver se o utilizador já era inserido normalmente.



Figura 500: Criação de mais um utilizador de teste

Mas, na execução da app, após de ter clicado no botão “registar”, obtive este erro na IDE dizendo que passei um argumento “nulo/vazio” para dentro do método “setId”.

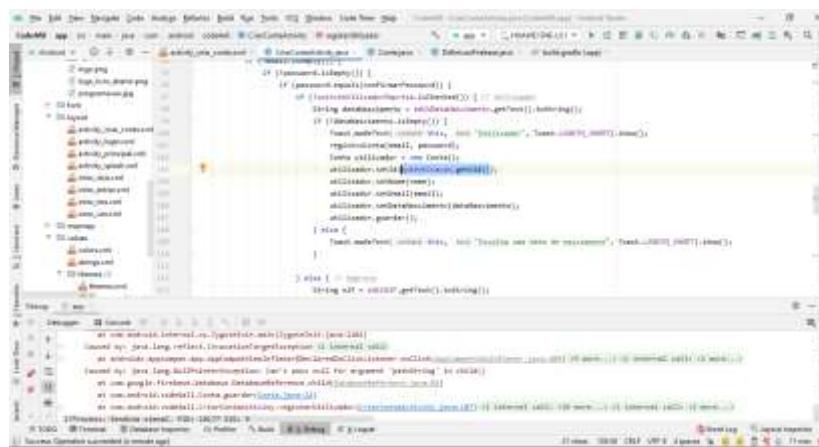


Figura 501: Erro na linha 105 - atribuição de um valor "nulo/vazio"

Coloquei um texto “teste” na definição do método “setId” para ver a reação do código, mas sempre continuava a dar o mesmo erro.

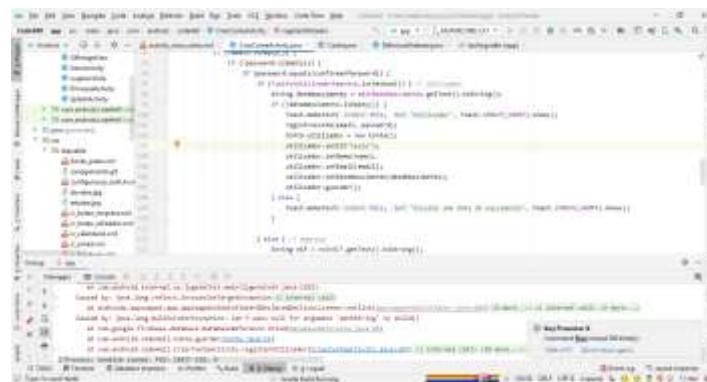


Figura 502: Alteração da String fornecida dentro do método "setId"

Desisti por aqui, eu tinha quase a certeza que o meu código estava bem escrito, tinha apenas uma última opção que era apagar o projeto no Firebase e começar de novo a associação da base de dados, pois uma informação importante que não referi, foi que o banco de dados que selecionei estava em BETA, o que poderia trazer alguns problemas para mim. Fui ao painel da FireBase e apaguei de vez este projeto, como se observa na figura abaixo.

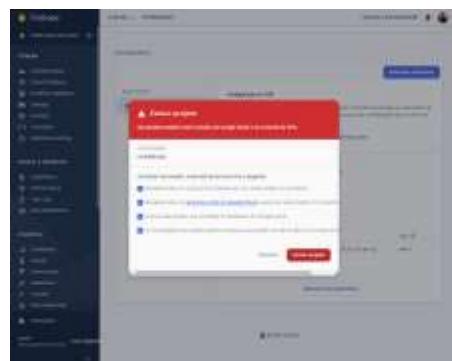


Figura 503: Eliminação do projeto no FireBase

Depois de ter apagado o projeto, decidi criar outro exatamente com o mesmo nome “Code4All” e depois disso cliquei no botão “Continuar” para avançar a configuração deste novo projeto.



Figura 504: Criação de um novo projeto no Firebase

No passo 2, deixei tudo como predefinição e cliquei em “Continuar”.



Figura 505: Ativação do Google Analytics para o projeto "Code4All"

No último passo, configurei a conta do Google Analytics, deixei a conta predefinida pelo o próprio Firebase, visto que é mais rápido e eficaz no processo de criação de um novo projeto.



Figura 506: Último passo para a criação do projeto

Seguidamente, após de ter configurado o projeto, o mesmo foi criado e fui levado para a página principal das definições do projeto, como se vê na imagem abaixo.



Figura 507: Página Inicial do projeto "Code4all"

Cliquei na opção “Authentication” do menu lateral, para ativar o registo/login de utilizadores por email e senha e como se indica na coluna “Status” da imagem, o mesmo foi ativo com sucesso.

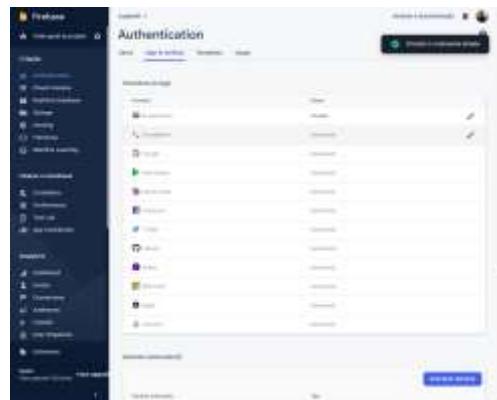


Figura 508: Ativação do Provedor "Email/senha"

Seguidamente, configurei a “Realtime Database” ou “Base de dados em tempo real”, mas desta vez, para tirar as minhas dúvidas coloquei como local do banco de dados os Estados unidos (us-central1), pois este já não está em beta e sim em fase “final”.

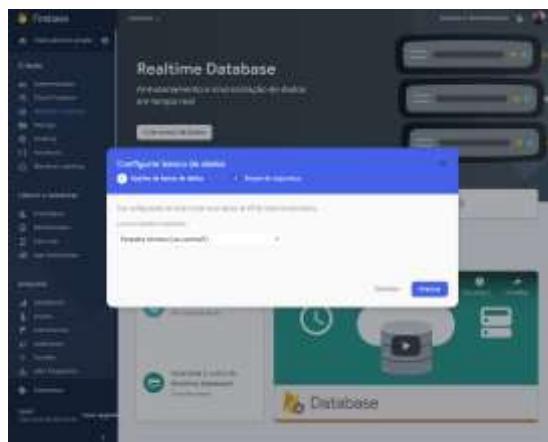


Figura 509: Configuração da localização do banco de dados (FireBase)

Depois veio, a configuração das regras de segurança, como ainda estou em fase de testes para com a app em android, decidi iniciar este banco de dados em modo de teste, como se vê na figura.

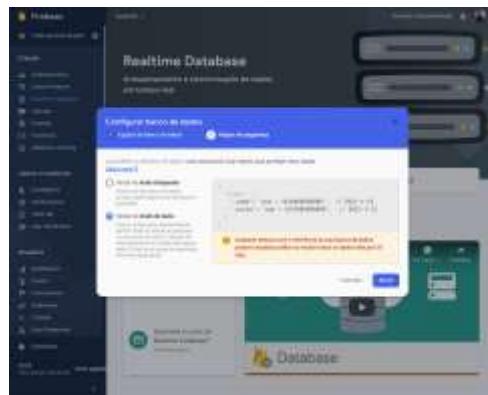


Figura 510: Configuração das regras de segurança relativas ao banco de dados

Depois cliquei em “Ativar”, o que me levou para a página onde ficam os “nós” todos que forem registados na app em Android, como já tinha referido anteriormente. Um aparte que me falta indicar, é que a localização do banco de dados não pode ser alterada uma vez escolhida.

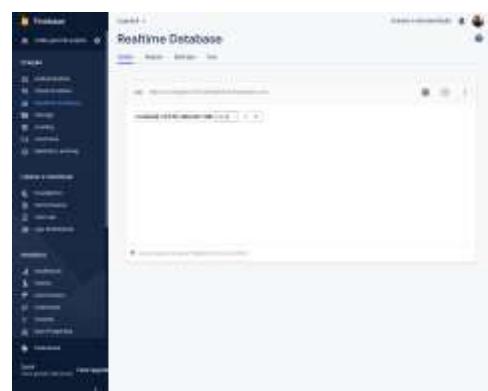


Figura 511: Dados da "Realtime Database"

Depois de ter tudo ativado, à semelhança do projeto Firebase anterior (aquele que foi apagado por mim), tive que adicionar novamente este projeto a uma app de Android (Code4All). Como tinha guardado os dados da mesma, foi só ir buscar os mesmos e fazer um “copy/paste” dos mesmos para os três campos que se encontram na figura.

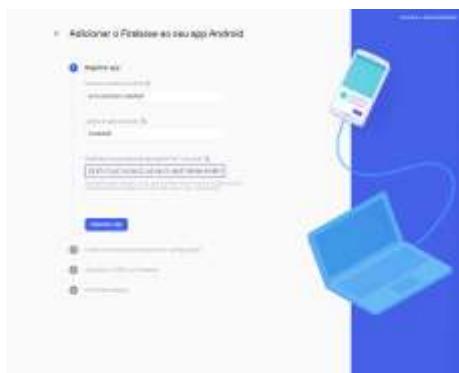


Figura 512: Adesão do projeto Firebase à app Android (Code4All)

No 2º passo, tive que realizar novamente o download do arquivo “google-services.json”, pois como apaguei a outro projeto FireBase, teria que adicionar de novo este ficheiro, pois as configurações deste projeto, não são as mesmas do outro que apaguei anteriormente.

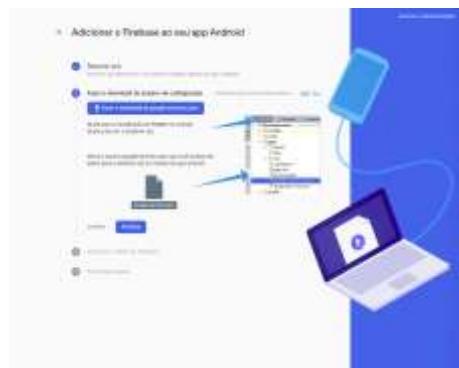


Figura 513: Arquivo de configurações "google-services.json"

Após de ter tido transferido o arquivo anteriormente referido, decidi, desta vez, colocar o mesmo no portfólio deste projeto, dentro da pasta “firebase”, para que se algo acontecer, ter sempre um backup deste ficheiro, pois contém as configurações necessárias para a app comunicar sem problemas com a FireBase e assim sucessivamente.



Figura 514: Colocação do ficheiro "google-services.json" dentro do portfolio

Depois, voltei para a minha IDE (Android Studio) e de imediato mudei o modo de visualização das pastas de “Android” para “Project” e apaguei o ficheiro “antigo” que continha as configurações do projeto Firebase que infelizmente foi apagado.

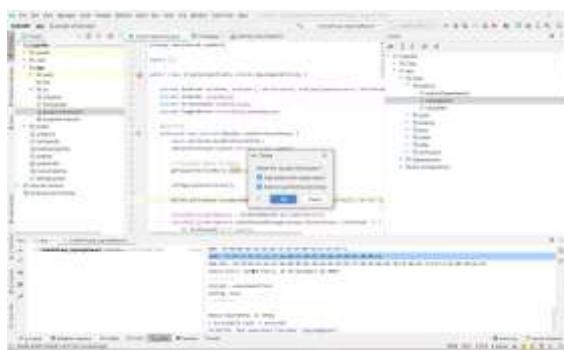


Figura 515: Remoção do antigo ficheiro "google-services.json" do projeto Code4All
(Android Studio)

Com estes passos anteriores realizados, pude dizer que tinha mais uma vez realizado a configuração do Firebase para a app Android com sucesso, saltei o passo três pois já tinha importado o SDK do FireBase no “antigo” projeto e isso sim não muda, os ficheiros são os mesmos que irei utilizar durante o código em “Java”.



Figura 516: Conclusão da adesão do projeto Firebase para a app (Android)

Nesse mesmo momento, voltei para o código desta atividade e adicionei logo uma linha de código (linha 39), para que quando esta atividade fosse iniciada, uns dados de teste fossem inseridos para dentro da Firebase Database. Ficando algo do tipo “teste->Hello, World”.

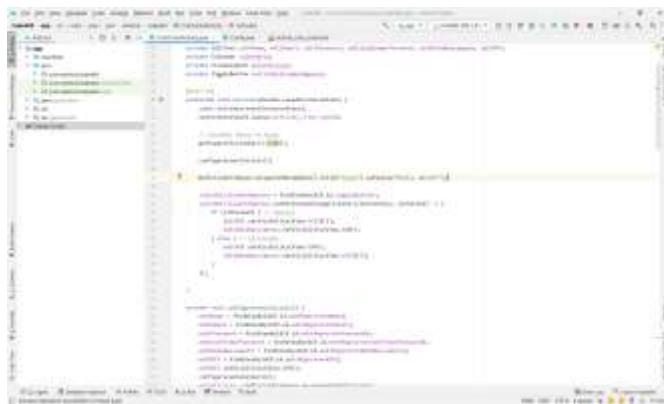


Figura 517: Inserção de uma linha de código para testar a Firebase Database

Com a linha anterior adicionada, podia executar novamente a app para testar a conexão com o novo projeto FireBase e testar ainda se os dados de teste estavam a ser enviados para a Firebase Database.



Figura 518: Criação de um utilizador de teste

Logo após ter efetuado o registo do utilizador, fui ao painel do projeto para ver se o utilizador criado estava já inserido na Firebase Auth e com a imagem abaixo concluo que sim, logo a conexão com o novo projeto da FireBase estava a ser bem-sucedida.

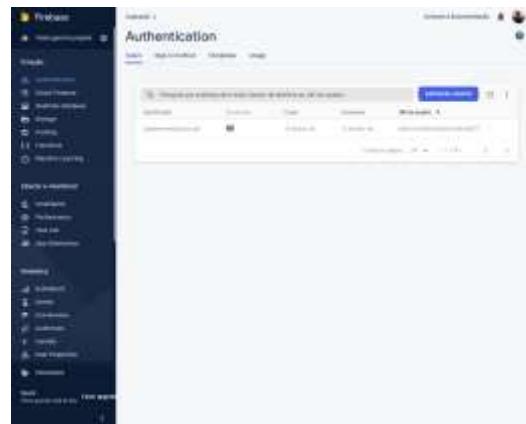


Figura 519: Página "Authentication" com o utilizador de teste já criado

Depois de ter visto que a FireBase Authentication, estava a funcionar às 1000 maravilhas, faltava agora ver se os dados de teste que defini anteriormente foram enviados para a base de dados em tempo real. Na captura de ecrã do meu browser, dá para observar que sim os dados de teste foram enviados, ou seja, posso dizer que o problema não estava no meu código, mas sim na FireBase e no facto do banco de dados que selecionei estar ainda em beta.

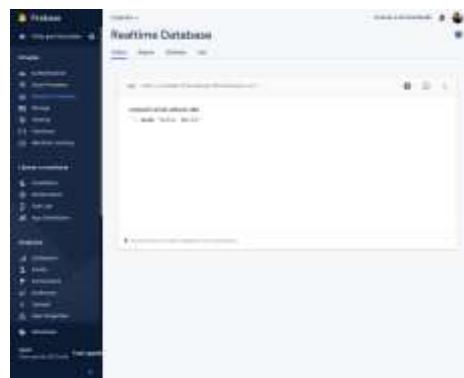


Figura 520: Visualização dos dados de teste na base de dados (FireBase)

Com estes dados de teste enviados com sucesso para a FireBase Database, pude eliminar aquela linha de código que enviava os mesmos.

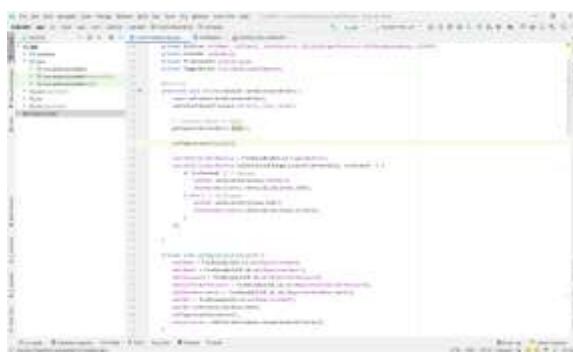


Figura 521: Eliminação da linha de código que enviava dados de teste

Voltando ao método “registroConta”, decidi também alterar a forma de como ia buscar o UID do utilizador (pois antes tinha uma String de “teste” apenas), passando agora a usar o código “task.getResult().getUser().getUid()”, como recomendado pela a documentação da FireBase.



Figura 522: Alteração da Linha 139 - definição certa do uid do utilizador

Seguidamente, voltei para a classe “Conta”, para realizar mais umas modificações. Primeiramente comecei por criar um método “Construtor” vazio, apenas para me orientar.

Depois, fui organizar o código do método “guardar”, definindo a variável “utilizador” como o caminho todo para guardar os dados.

E por fim, usei o código “utilizador.setValue(this)” para guardar todos os dados existentes na classe.

Uma observação é que mudei a forma de como iria buscar o “id” do utilizador e desta vez, fui buscar o id, pelo o método já existente na classe, sendo ele “getId()”.

O código explicado anteriormente, está descrito na imagem que se observa.



Figura 523: Criação de um método "Construtor" e alterações no método "guardar()"

Com a alteração da forma de guardar os dados para a Firebase Database, decidi executar mais uma vez a app, para ver se estava agora tudo a funcionar como suposto.



Figura 524: Criação de um utilizador para testar o envio de dados para a Firebase Database

Novamente realizei o processo de ir ver se os dados foram enviados. Dirigi-me ao painel de controlo do projeto. Com a visualização do print abaixo, posso concluir que agora sim tenho o envio dos dados a funcionar corretamente, as informações que escrevi no código estão a ser passadas para a base de dados.

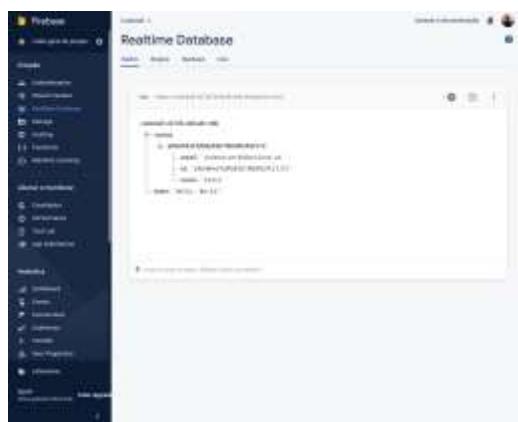


Figura 525: Visualização dos dados

Realizada a parte do envio de dados para a Firebase Database, decidi limpar o banco de dados para proceder aos ajustes no código.

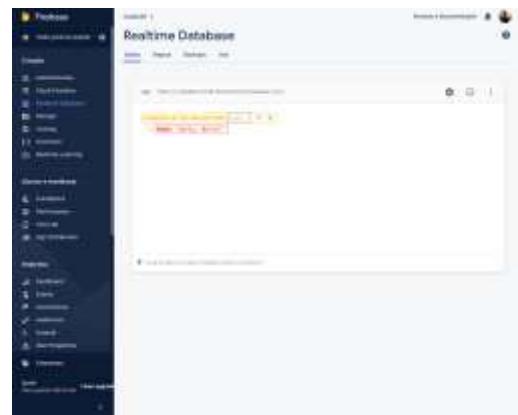


Figura 526: Limpeza da base de dados

Faltava ainda, eliminar o utilizador dentro da Firebase Auth e fui o que realizei, como se observa na figura abaixo.

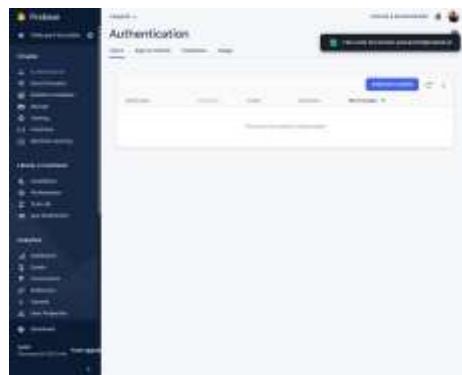


Figura 527: Eliminação da conta de teste anteriormente criada

Com o envio dos dados para o banco de dados tratado com sucesso, fiquei mais descansado e podia agora programar à vontade a classe “Conta” colocando as restantes variáveis que faltavam. Começando pela variável “tipo” que iria conter dois estados “utilizador” ou “empresa”, para de certa forma no futuro quando for feito o login do utilizador/empresa, o mesmo seja direcionado para a atividade correta, com as permissões corretas. Criei ainda um método “Getter and Setter” para a variável “tipo”.

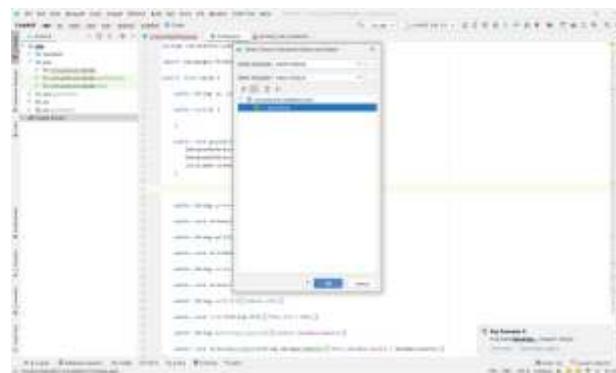


Figura 528: Criação da variável "tipo" e do seu método "Getter and Setter"

No ficheiro da atividade “CriarContaActivity.java”, foi preciso a criação de mais duas variáveis privadas do tipo “String”, sendo elas, “dataNascimento” e “nif”.



Figura 529: Criação de duas variáveis privadas do tipo "String"

Seguidamente fui, ao método “registarUtilizador” e no caso do botão switch estar virado para o utilizador, a variável “dataNascimento” irá receber o texto do campo, se não é mostrado um “Toast” a informar que falta escolher uma data de nascimento. Se o botão switch tiver virado para a empresa, acontecerá a mesma lógica, só que com a variável “nif” e o campo da mesma.



Figura 530: Verificação e recuperação dos valores dos campos de NIF e data de nascimento

Logo depois, fui fazer a definição do método “setTipo()” da variável utilizador da classe “Conta”, inicialmente defini o mesmo como “utilizador” para ver se este era mesmo enviado para a base de dados. Observação: eu sei que realizo muitos testes desnecessários, mas antes prefiro testar 1000 vezes e ter o código a funcionar bem do que não realizar nenhum teste e só testar o código no final.

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Path:** `src/main/java/com/stackroute/EmailVerification.java`
- Code Content:** The code is a Java class named `EmailVerification` with a single method `sendEmailVerification()`. The code uses annotations like `@Service`, `@Autowired`, and `@Transactional`. It interacts with a `MailService` and a `CustomerRepository` to send an email verification link to a user's email address.
- Toolbars:** Standard Java development tools like `Run`, `Build`, `Refactor`, and `Find` are visible at the top.
- Sidebar:** The left sidebar shows the project structure with modules `app` and `db`, and packages `com.stackroute` and `com.stackroute.repository`.
- Bottom:** A navigation bar with tabs like `File`, `Edit`, `View`, `Code`, `Run`, `Build`, `Refactor`, `Find`, `Help`, and `Settings`.

Figura 531: Definição do método "setTipo()" da variável utilizador

Novamente, realizei a execução da app no meu smartphone Android, para testar e confirmar as alterações que foram feitas.

Na captura de ecrã, é mostrado ainda o tão famoso “Toast” para dar algum feedback ao utilizador do que está a ser feito e neste caso indica que o utilizador foi criado com sucesso, ou seja, está tudo a correr conforme o esperado e o código entrou na condição “IF/SE” correta, tal como devia.

A captura de ecrã que estava a ser falada na página anterior, é a que está presente abaixo.



The screenshot shows a registration form titled "Registro". It includes fields for Name (A. José Xavier), Email (xavier46@outlook.pt), Password (two masked input fields), Date of Birth (12/08/03), and Gender (Male). Below the form are two radio buttons: "Empresa" (selected) and "Utilizador", with a note "Utilizar credenciais Google". A large orange "REGISTAR" button is at the bottom.

Na parte da FireBase Authentication, está tudo conforme esperado, o utilizador foi criado e adicionado sem quaisquer problemas.

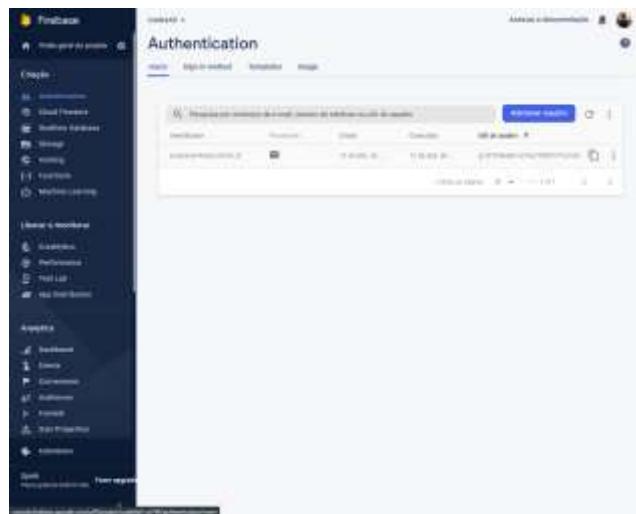


Figura 532: Firebase Authentication - Utilizador criado com sucesso

Na parte da “Realtime Database” todos os dados foram adicionados, inclusive a data de nascimento (graças à criação da variável), o email também está correto, o id também está igual ao que foi apresentado na página “Authetication” (página anterior, onde diz UID do usuário), o nome continua manual (tem de ser alterado consoante o campo nome) e o tipo também está a ir buscar o que introduzi (manualmente).

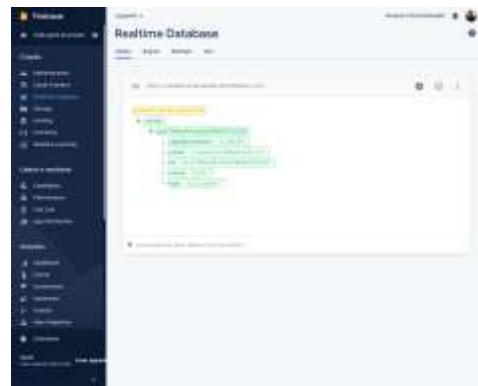


Figura 533: Realtime DataBase - Envio de dados

Chegou a altura de tratar da parte da criação de um utilizador do tipo “empresa”, a alteração que realizei nesta altura, foi só adicionar o método “setTipo()” e configurar o mesmo como “empresa”.



Figura 534: Configuração do método "setTipo()" da classe "Conta" como "empresa"

Voltei a executar a aplicação e criei dois utilizadores, um “utilizador normal” e outro “empresa”.



The screenshot shows the 'Registro' (Registration) page. It has fields for Name (R. Michael Teixeira), Email (michael.teixeira@etpc.pt), Gender (Male), Date of Birth (27/08/88), and a toggle switch between 'Empresa' (Company) and 'Utilizador' (User). The 'Utilizador' option is selected. At the bottom, there's a button labeled 'Utilizador criado com sucesso!' (User created successfully!). Below the button are links for 'Já tem conta?' (Already have an account?) and 'Entrar' (Log in).

Figura 535: Criação de um utilizador Normal



The screenshot shows the 'Registro' (Registration) page. It has fields for Name (Sílvia de Cantanhede), Email (geral@etpc.pt), Gender (Female), Date of Birth (27/08/88), and a phone number (# 55649239232). A toggle switch between 'Empresa' (Company) and 'Utilizador' (User) is shown, with 'Empresa' selected. At the bottom, there's a button labeled 'Utilizador criado com sucesso!' (User created successfully!). Below the button are links for 'Já tem conta?' (Already have an account?) and 'Entrar' (Log in).

Figura 536: Criação de um utilizador do tipo "empresa"

E como se observa na captura de ecrã da base de dados, posso dizer que todos os dados foram introduzidos corretamente, tal como esperado por mim.

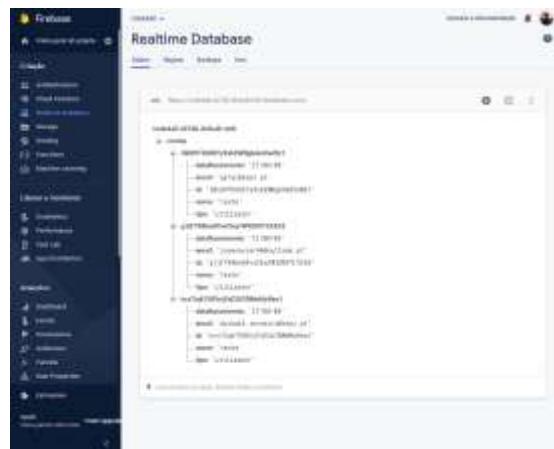


Figura 537: Realtime Database - Dados das contas

Depois, para dar um efeito mais “fixe” ao próprio app, decidi criar um método “limparCampos()” que limpava os campos.

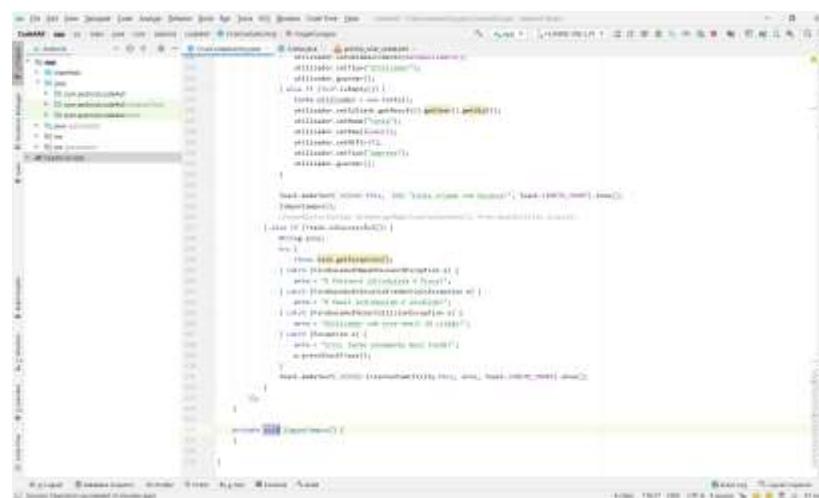


Figura 538: Criação do método "limparCampos()"

Peguei em cada um dos “EditTexts” presentes no layout “activity_criar_conta” e usei o código “getText().clear()” para proceder a limpeza do campo em si, ficando da seguinte forma.

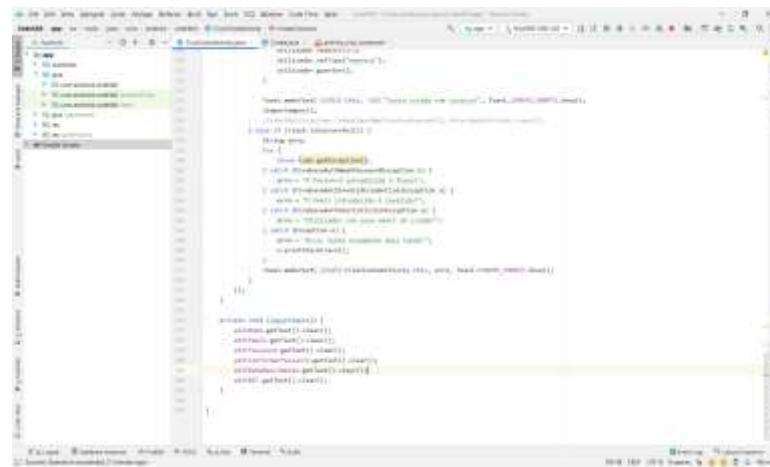


Figura 539: Escrita do código necessário para a limpeza dos campos

Faltava configurar o nome de forma automaticamente, para isso, foi preciso adicionar mais uma variável do tipo “String”, chamada de “nome”.

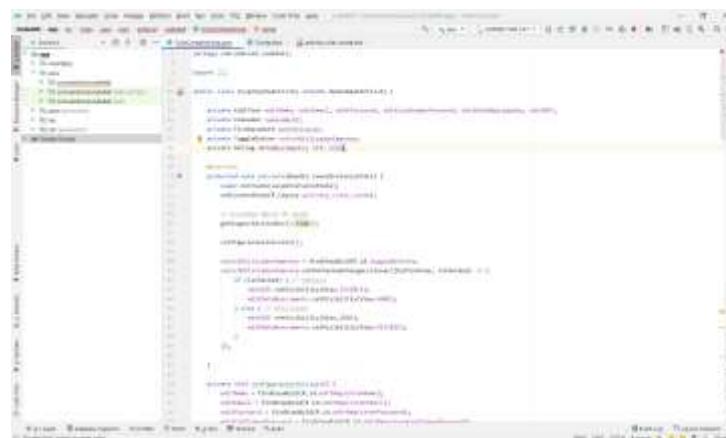


Figura 540: Criação da variável "nome" do tipo "String"

Associei a variável “nome” ao valor introduzido no campo do nome do utilizador e fiz uma alteração na condição da linha 94 e troquei pela a variável “nome”, pois fica mais fácil de entender.

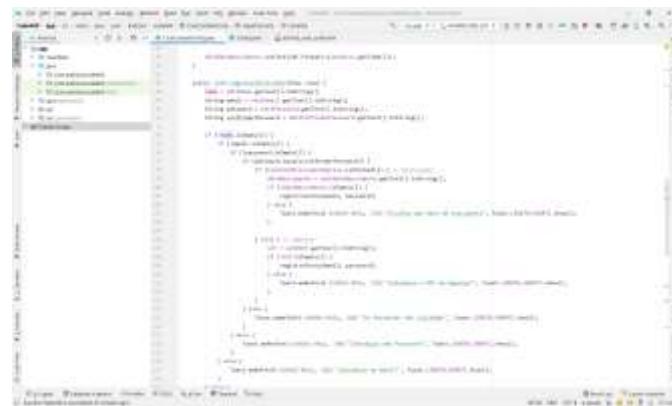


Figura 541: Definição da variável "nome" e alteração da condição "IF/SE" (linha 94)

E dentro do método “registroConta()” pode finalmente alterar a String “teste” para a variável privada “nome” que conteria o nome final do utilizador, logo este já seria enviado em conjunto com os restantes dados do utilizador, tais como, email e tipo.



Figura 542: Alteração do método "setNome()" pela a variável "nome" nos dois tipos de conta

Decidi tentar executar a aplicação mais uma vez, para ver se agora o nome aparecia consoante o que era introduzido no campo de nome, mas fiquei pelo o tentar, pois ao iniciar a atividade a app deu um erro e fechou.

E este erro, deu-se ao facto de não ter inicializado a variável "nif" e "dataNascimento" e para resolver esse problema, bastava declarar as mesmas antes de serem usadas, eu fiz essa declaração no método "registarUtilizador()". Outro motivo para me ter dado erro é devido à linguagem Java não consegue "comparar" referências nulas, tal como diz na consola, na figura abaixo.

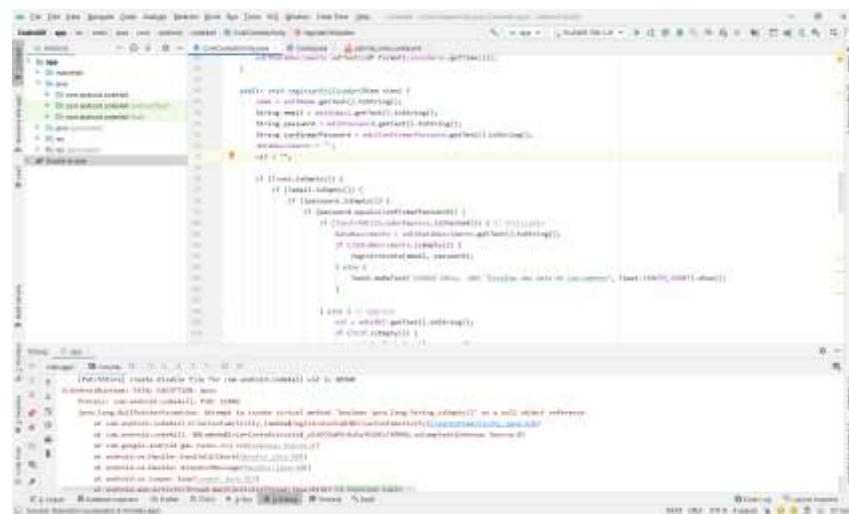


Figura 543: Inicialização das variáveis "nif" e "dataNascimento"

Tentei executar mais uma vez e desta vez a app e atividade foram carregadas sem problemas, passei então à criação de mais um utilizador desta vez do tipo “empresa”.



Figura 544: Criação de uma "empresa" de teste

Fui ver a base de dados e deparei-me que todos os dados introduzidos nos campos desta atividade foram sim enviados com sucesso e todos estão corretamente inseridos. Observação, antes de criar o utilizador, apaguei todos os dados do banco, para que seja mais fácil de perceber o que foi feito.

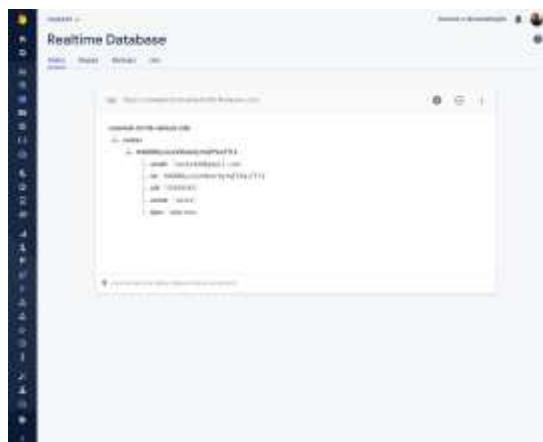


Figura 545: Realtime Database - Dados do tipo de utilizador "empresa"

Depois, passei à criação de um utilizador “normal” para testar se o mesmo bom sucesso de execução do código se aplicava.



The screenshot shows a registration form titled "Registro". It includes fields for a name (Dr. José Xavier), email (xavier46@outlook.pt), gender (male), date of birth (12/06/03), and a toggle switch for "Enviado" (Sent) which is set to "Utilizador" (User). Below the form is an orange "REGISTAR" button. At the bottom, there are links for "Já tenho conta?" (I already have an account?) and "Olhar" (Look).

Figura 546: Criação de um utilizador "normal" de teste

Abaixo, estão apresentados os dados tanto da “empresa” como do utilizador “normal” e como dá para observar todos estão preenchidos corretamente, consoante o que foi inserido na atividade.



Figura 547: Realtime Database - as duas vertentes de contas

Dentro da FireBase Authentication, também está tudo a correr como planeado, os utilizadores são criados e o UID dos mesmos também e este UID também coincide com o que está escrito na base de dados.



Figura 548: Listagem dos dois utilizadores criados anteriormente

Voltei para a atividade, mais concretamente o método “registroConta()”, coloquei um “Toast” para informar ao utilizador que a sua conta havia sido criada com sucesso, adicionei o método “limparCampos()” já anteriormente criado, chamei ainda o método “finish()” para finalizar a atividade e voltar para a “EntrarActivity” e por fim terminei a sessão do utilizador pois quando o registo é realizado o utilizador permanece automaticamente logado, mas como eu não queria esse efeito na minha app, forcei o “signOut” do utilizador.



Figura 549: Complementos ao método “registroConta()”

Para organizar de certa forma o ficheiro da atividade, decidi colocar o código do botão switch (ToggleButton) dentro das configurações, pois tecnicamente é esse o lugar dele.

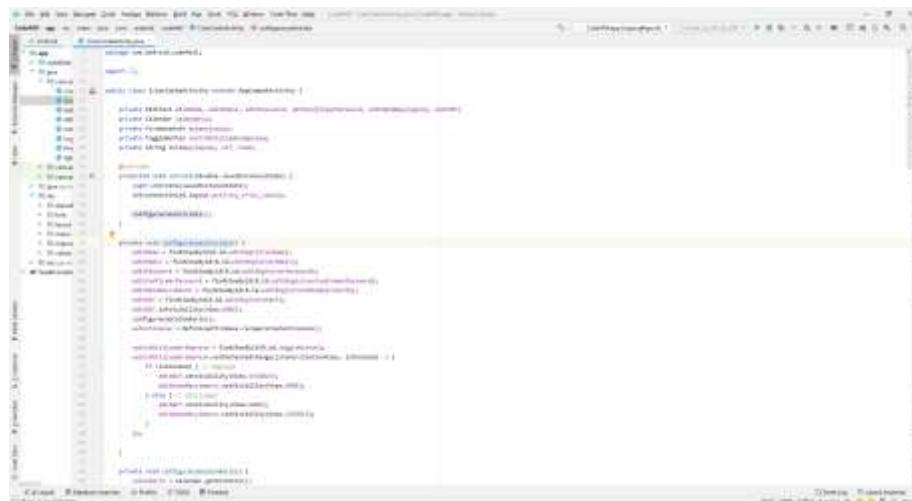


Figura 550: Mudança do código do botão switch do método “onCreate” para o “configuracoesIniciais”

Mudei também o nome do método “registroConta()” para “registroConta()”.

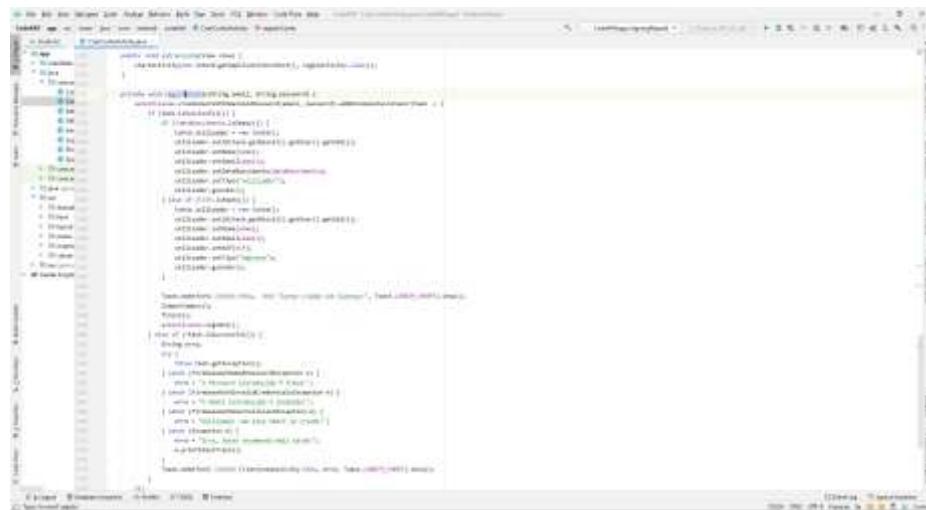


Figura 551: Mudança do nome do método "registroConta()" para "registroConta()"

Mudei também o nome do método “registarUtilizador()” para “verificarDados()”, pois ficava a confundir com o método “registroConta()” e assim já não existe mais a confusão entre estes dois métodos.



Figura 552: Modificação do nome do método "registarUtilizador()" para "verificarDados()"

Visto que as regras da base de dados foram alteradas para “auth!=null”, fui “obrigado” a remover o “logout” do utilizador criado e passar o mesmo para a PrincipalActivity. Tive de realizar este procedimento, pois os dados não estavam a ser enviados devido ao facto do utilizador estar sem a sua sessão iniciada, logo não tem acesso à base de dados.

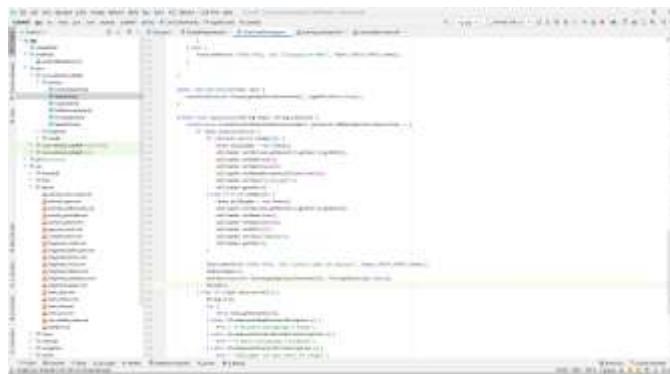


Figura 553: Remoção do “logout” do utilizador e reencaminhamento para a “PrincipalActivity”

Com o método “corAleatoria()” criado, pude passar à atribuição dos métodos “setCorPerfil()” e “setCorFundoPerfil()”.

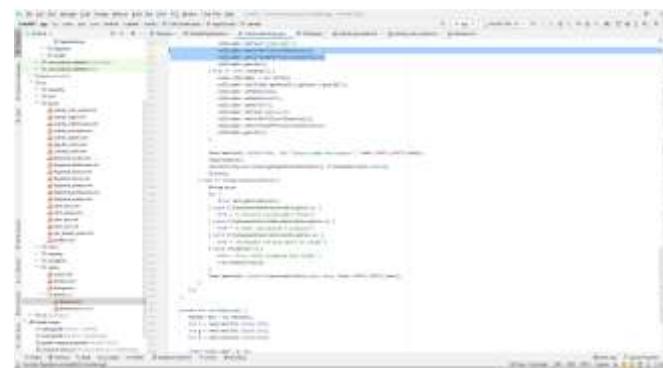


Figura 554: Definição do método "setCorPerfil" e "setCorFundoPerfil"

Seguidamente, passei à implementação do “*Firebase Storage*”, para enviar ficheiros, neste caso as fotos do utilizador.

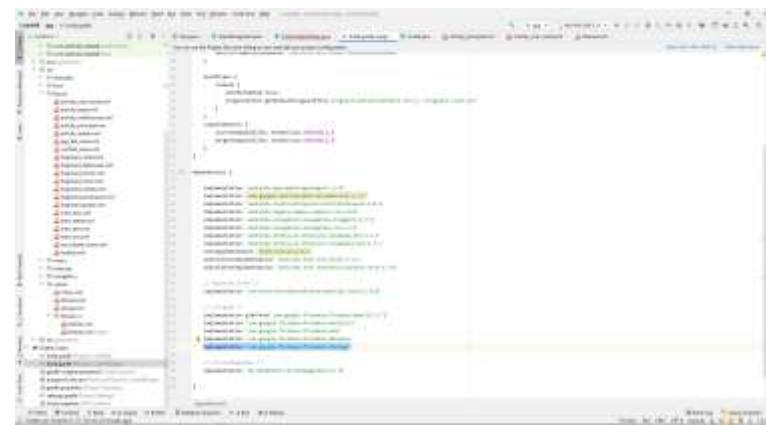


Figura 555: Implementação do Firebase Storage

Seguidamente prossegui, à configuração do Firebase Storage.

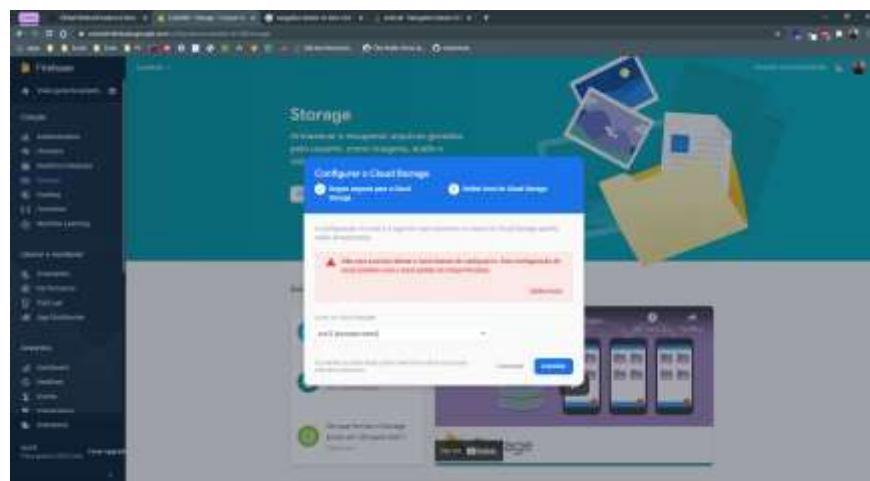


Figura 556: Configuração do Firebase Storage

Seguidamente, decidi procurar como desenhar as iniciais do nome dentro de um bitmap, para depois exportar para a Firebase Storage, como imagem de perfil.

<https://stackoverflow.com/questions/17490975/drawing-transparent-text-on-bitmap-in-android/41972420/>

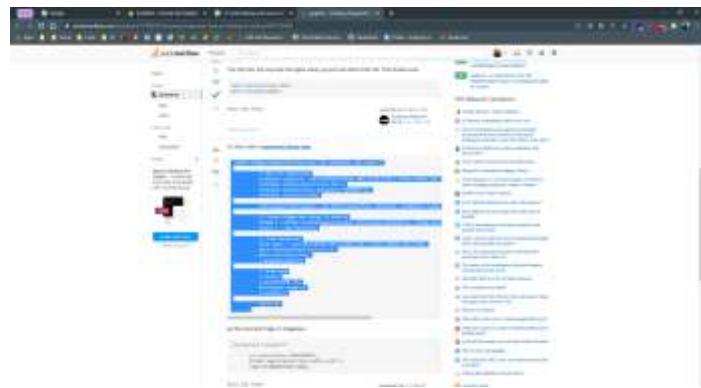


Figura 557: Solução de desenhar as iniciais de um nome como Bitmap

Colocação do método anteriormente visto no *stackoverflow*, para dentro da CriarContaActivity.

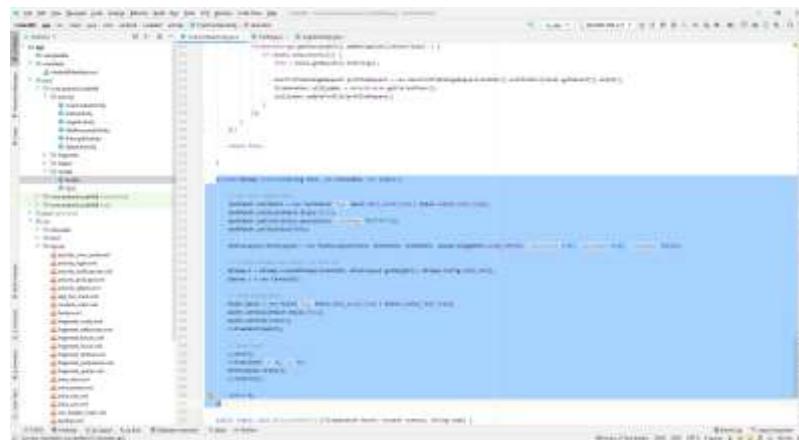


Figura 558: Método “drawText()”

Para me organizar melhor dividi, o código, um para o utilizador e outro para a empresa. Criei também, dois métodos o “guardarNome()” e o “guardarFoto()”.

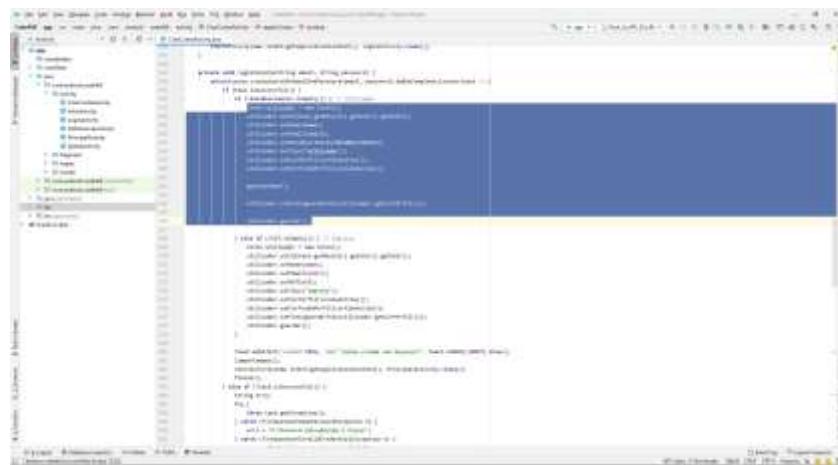


Figura 559: Organização do código e criação de dois novos métodos

O método “guardarNome()” contém o código necessário para guardar o nome do utilizador dentro do próprio utilizador.

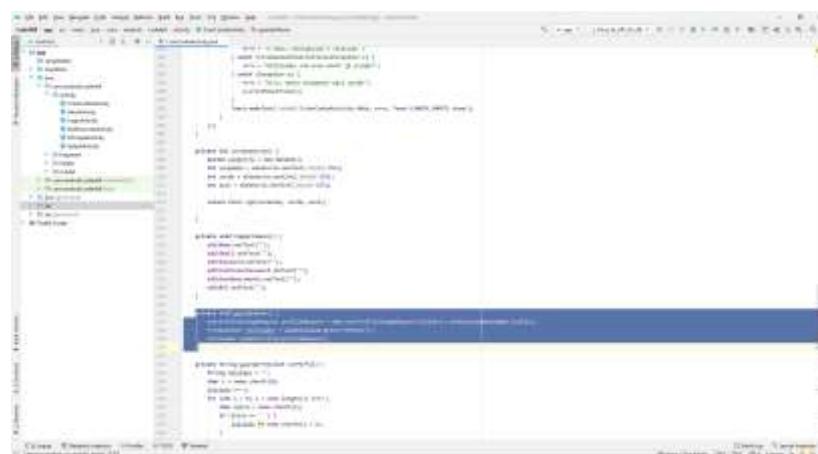


Figura 560: Método "guardarNome()"

O método “guardarFoto()” serve mais para recuperar as iniciais do nome e realizar uma imagem com essas iniciais, enviar para a Firebase Storage e após disso, guardar o URI da imagem dentro do perfil do utilizador.

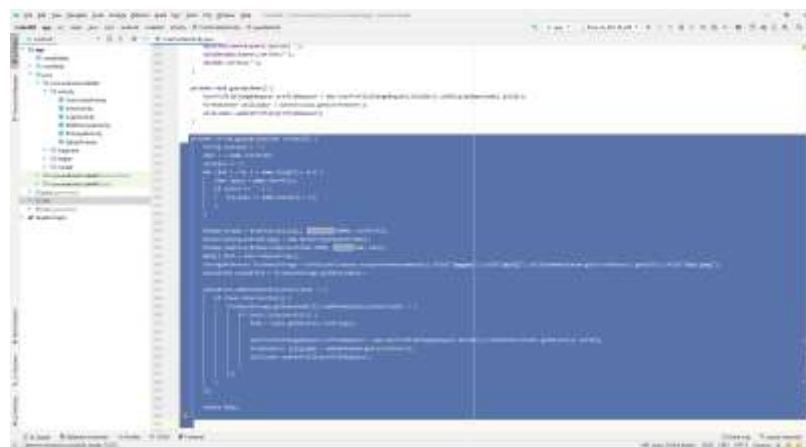


Figura 561: Método guardarFoto()

Adicionei o campo foto dentro do Objeto “Conta” e dos seus devidos “getters and setters”.



Figura 562: Criação de um novo campo String "foto"

Na figura abaixo, podemos ver que a imagem foi enviada com sucesso para a Firebase Storage.

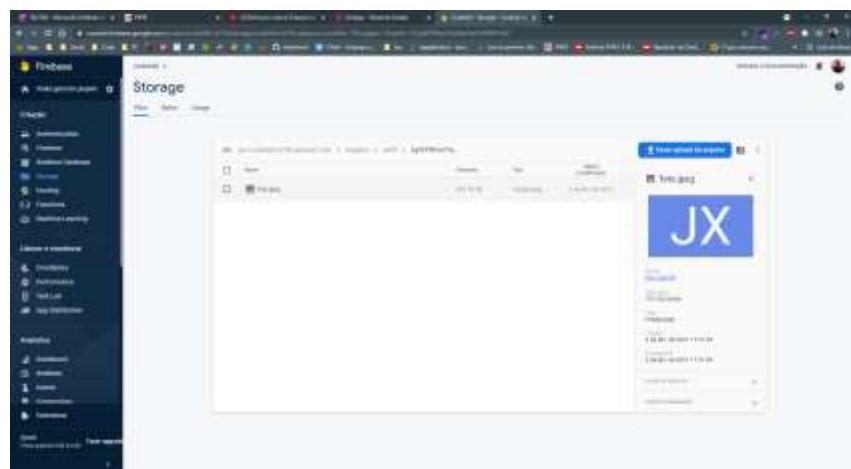


Figura 563: Imagem do Utilizador dentro da Firebase Storage



Figura 564: Criação do Utilizador

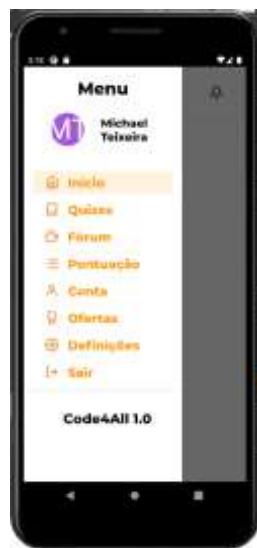


Figura 565: Visualização da foto criada para o Utilizador

Depois de verificar que a imagem era corretamente gerada, tive que atualizar a foto que está gravada diretamente no perfil, portanto, aproveitei o facto de estar a gravar a imagem e coloquei a URI da foto, dentro do próprio utilizador da Firebase, tal como se pode observar na figura abaixo.

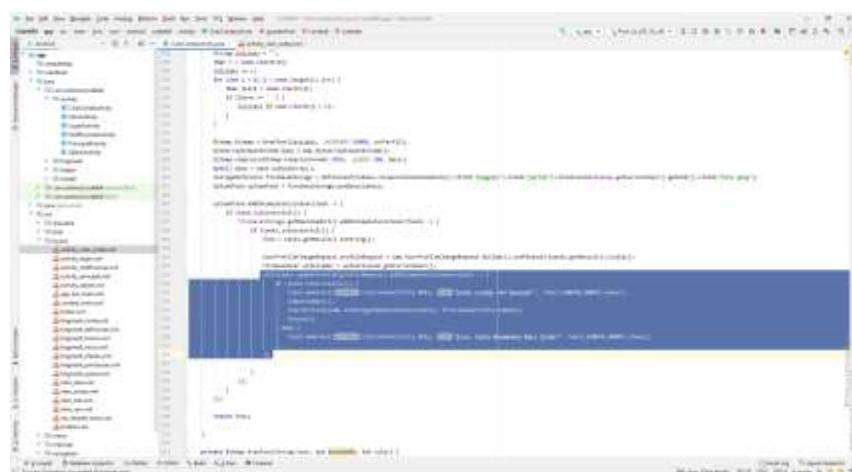


Figura 566: Atualização do perfil, com o URI da foto do utilizador

Coloquei também a variável conta, como private no início do código, para esta ser acessível em todas as zonas desta atividade.

Figura 567: Definição da variável conta como global (privada)

Organizei mais uma vez o código, para o entender melhor, tal como se observa na figura abaixo.

```
1 package org.javatutorial.net;
2
3 import java.util.List;
4 import java.util.ArrayList;
5 import java.util.LinkedList;
6 import java.util.NoSuchElementException;
7
8 public class List<T> implements List<T> {
9
10     private T[] elements;
11     private int size;
12
13     public List() {
14         elements = (T[]) new Object[10];
15         size = 0;
16     }
17
18     public void add(T element) {
19         if (size == elements.length)
20             grow();
21         elements[size] = element;
22         size++;
23     }
24
25     public void remove(int index) {
26         if (index < 0 || index >= size)
27             throw new NoSuchElementException("Index " + index);
28         for (int i = index; i < size - 1; i++)
29             elements[i] = elements[i + 1];
30         size--;
31     }
32
33     public T get(int index) {
34         if (index < 0 || index >= size)
35             throw new NoSuchElementException("Index " + index);
36         return elements[index];
37     }
38
39     public void set(int index, T element) {
40         if (index < 0 || index >= size)
41             throw new NoSuchElementException("Index " + index);
42         elements[index] = element;
43     }
44
45     public boolean contains(T element) {
46         for (int i = 0; i < size; i++)
47             if (elements[i].equals(element))
48                 return true;
49         return false;
50     }
51
52     private void grow() {
53         T[] newElements = (T[]) new Object[elements.length * 2];
54         for (int i = 0; i < size; i++)
55             newElements[i] = elements[i];
56         elements = newElements;
57     }
58 }
```

Figura 568: Organização do código

No método guardarFoto(), depois da imagem ter sido enviada e atualizada no perfil do utilizador, defini a foto, dentro da variável conta do tipo Conta e depois guardei a mesma, como se vê na print.



Figura 569: Definição da foto dentro da variável conta e envio dos dados

Seguidamente, procurei como podia perceber que o “Conta.guardar()”, acabou de executar e encontrei a solução com os dois posts no *StackOverflow*.

<https://stackoverflow.com/questions/50109885/firestore-how-can-read-data-from-outside-void-oncomplete-methods/>

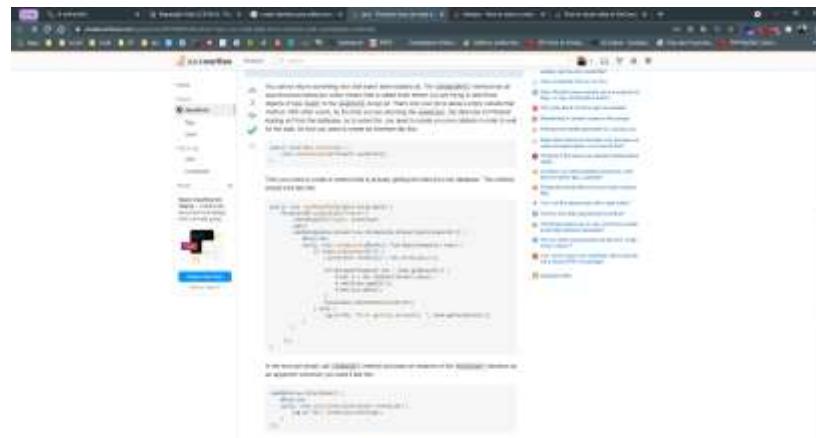


Figura 570: Solução de como perceber se o método “guardar()” foi finalizado

Logo, decidi aplicar a solução que me foi dada, pelo o post visualizável acima e prossegui à criação de uma Interface “Validacao”.

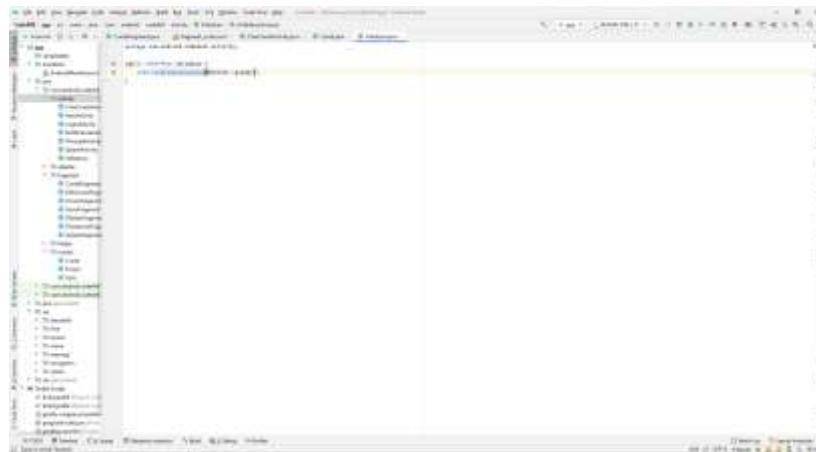


Figura 571: Criação da Interface "Validacao" e do método isValidacaoSucesso

Dentro do método “guardar()”, passei como argumento a tal “Validacao” anteriormente criada e defini a “isValidacaoSucesso” como *true* e *false*, dependendo de como correu a tarefa de guardar os dados do utilizador.

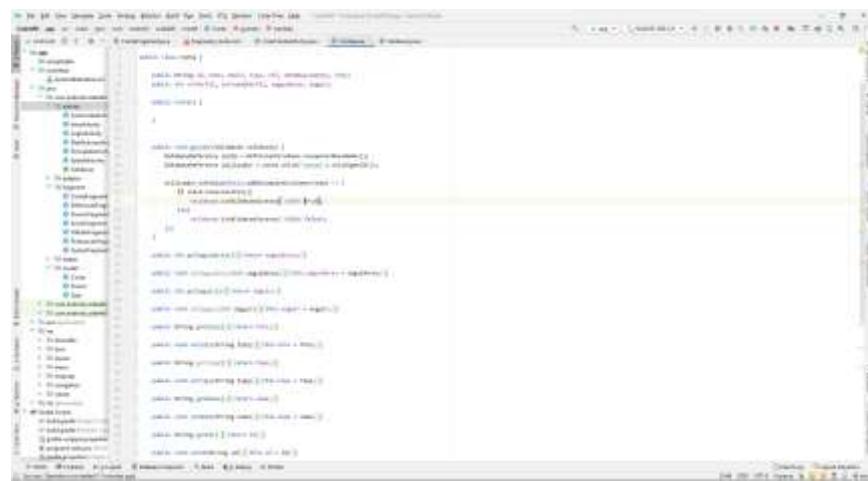


Figura 572: Introdução da Interface "Validacao" dentro do método guardar()

Dentro da CriarContaActivity, foi então necessário ajustar as modificações, a variável sucesso, é a que contém (depois do método ter finalizado), se for true, prossegue o código, se for false, informa o utilizador.

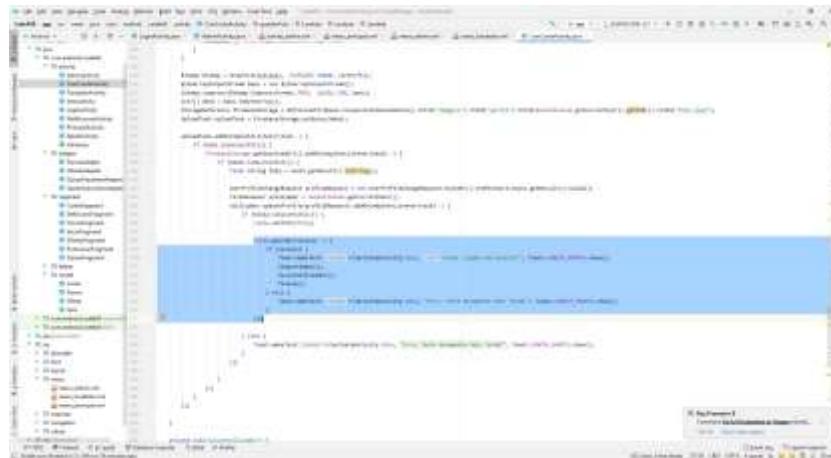


Figura 573: Modificações dentro do método "conta.guardar()"

Com esta verificação bastante importante dada como realizada, só tinha de ir buscar o método já feito “buscarUtilizador()” para me retornar a “Intent” que preciso e iniciar a mesma.



Figura 574: Método "buscarUtilizador()" e início da próxima atividade

Após isso, faltava ainda selecionar o tipo de conta, neste caso só haviam duas, “membro” ou “utilizador”.

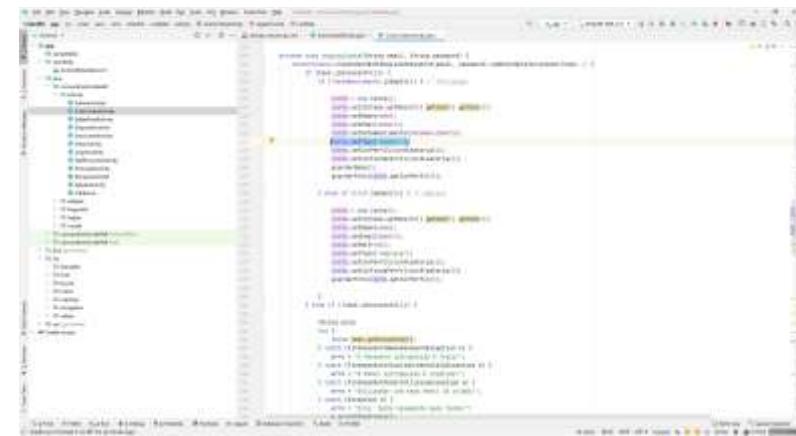


Figura 575: Definição do tipo de conta

Seguidamente, configurei a parte da próxima atividade, no caso de o utilizador criado ser um “membro”, que era as “InscricoesActivity”, onde estariam todos os temas dos “quizes” e o utilizador só teria que escolher aqueles que lhe mais interessam. Na figura abaixo, pode se visualizar o início dessa mesma atividade.

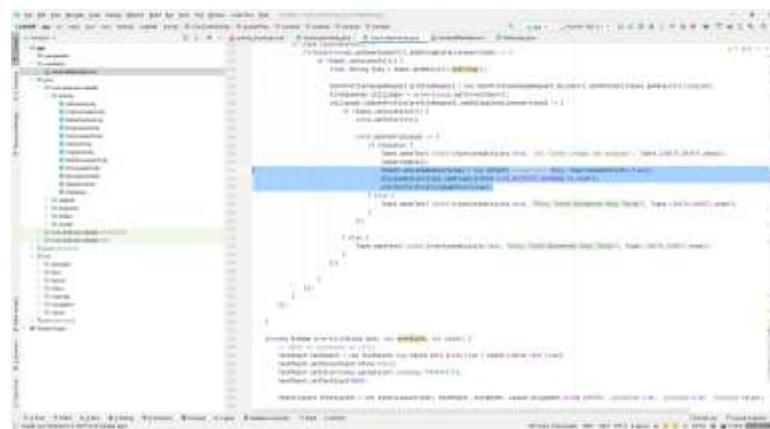


Figura 576: Início da atividade "InscricoesActivity"

De seguida, foi para o objeto “Conta” e removi os seguidores e a Seguir e adicionei o totalXP, que viria a conter o XP total do utilizador em questão.



Figura 577: Objeto "Conta", adição do campo totalXP

Depois, adicionei o parâmetro tipo, dentro do método “guardarFoto()”.

```
private void updatePanel() {
    int max = 100;
    int min = 0;
    int current = 50;
    int step = 1;
    int value = 50;
    int width = 100;
    int height = 100;
    int radius = 50;
    int x = 50;
    int y = 50;
    int x2 = 100;
    int y2 = 100;
    int x3 = 50;
    int y3 = 100;
    int x4 = 0;
    int y4 = 100;
    int x5 = 50;
    int y5 = 150;
    int x6 = 100;
    int y6 = 150;
    int x7 = 50;
    int y7 = 200;
    int x8 = 100;
    int y8 = 200;
    int x9 = 50;
    int y9 = 250;
    int x10 = 100;
    int y10 = 250;
    int x11 = 50;
    int y11 = 300;
    int x12 = 100;
    int y12 = 300;
    int x13 = 50;
    int y13 = 350;
    int x14 = 100;
    int y14 = 350;
    int x15 = 50;
    int y15 = 400;
    int x16 = 100;
    int y16 = 400;
    int x17 = 50;
    int y17 = 450;
    int x18 = 100;
    int y18 = 450;
    int x19 = 50;
    int y19 = 500;
    int x20 = 100;
    int y20 = 500;
    int x21 = 50;
    int y21 = 550;
    int x22 = 100;
    int y22 = 550;
    int x23 = 50;
    int y23 = 600;
    int x24 = 100;
    int y24 = 600;
    int x25 = 50;
    int y25 = 650;
    int x26 = 100;
    int y26 = 650;
    int x27 = 50;
    int y27 = 700;
    int x28 = 100;
    int y28 = 700;
    int x29 = 50;
    int y29 = 750;
    int x30 = 100;
    int y30 = 750;
    int x31 = 50;
    int y31 = 800;
    int x32 = 100;
    int y32 = 800;
    int x33 = 50;
    int y33 = 850;
    int x34 = 100;
    int y34 = 850;
    int x35 = 50;
    int y35 = 900;
    int x36 = 100;
    int y36 = 900;
    int x37 = 50;
    int y37 = 950;
    int x38 = 100;
    int y38 = 950;
    int x39 = 50;
    int y39 = 1000;
    int x40 = 100;
    int y40 = 1000;
    int x41 = 50;
    int y41 = 1050;
    int x42 = 100;
    int y42 = 1050;
    int x43 = 50;
    int y43 = 1100;
    int x44 = 100;
    int y44 = 1100;
    int x45 = 50;
    int y45 = 1150;
    int x46 = 100;
    int y46 = 1150;
    int x47 = 50;
    int y47 = 1200;
    int x48 = 100;
    int y48 = 1200;
    int x49 = 50;
    int y49 = 1250;
    int x50 = 100;
    int y50 = 1250;
    int x51 = 50;
    int y51 = 1300;
    int x52 = 100;
    int y52 = 1300;
    int x53 = 50;
    int y53 = 1350;
    int x54 = 100;
    int y54 = 1350;
    int x55 = 50;
    int y55 = 1400;
    int x56 = 100;
    int y56 = 1400;
    int x57 = 50;
    int y57 = 1450;
    int x58 = 100;
    int y58 = 1450;
    int x59 = 50;
    int y59 = 1500;
    int x60 = 100;
    int y60 = 1500;
    int x61 = 50;
    int y61 = 1550;
    int x62 = 100;
    int y62 = 1550;
    int x63 = 50;
    int y63 = 1600;
    int x64 = 100;
    int y64 = 1600;
    int x65 = 50;
    int y65 = 1650;
    int x66 = 100;
    int y66 = 1650;
    int x67 = 50;
    int y67 = 1700;
    int x68 = 100;
    int y68 = 1700;
    int x69 = 50;
    int y69 = 1750;
    int x70 = 100;
    int y70 = 1750;
    int x71 = 50;
    int y71 = 1800;
    int x72 = 100;
    int y72 = 1800;
    int x73 = 50;
    int y73 = 1850;
    int x74 = 100;
    int y74 = 1850;
    int x75 = 50;
    int y75 = 1900;
    int x76 = 100;
    int y76 = 1900;
    int x77 = 50;
    int y77 = 1950;
    int x78 = 100;
    int y78 = 1950;
    int x79 = 50;
    int y79 = 2000;
    int x80 = 100;
    int y80 = 2000;
    int x81 = 50;
    int y81 = 2050;
    int x82 = 100;
    int y82 = 2050;
    int x83 = 50;
    int y83 = 2100;
    int x84 = 100;
    int y84 = 2100;
    int x85 = 50;
    int y85 = 2150;
    int x86 = 100;
    int y86 = 2150;
    int x87 = 50;
    int y87 = 2200;
    int x88 = 100;
    int y88 = 2200;
    int x89 = 50;
    int y89 = 2250;
    int x90 = 100;
    int y90 = 2250;
    int x91 = 50;
    int y91 = 2300;
    int x92 = 100;
    int y92 = 2300;
    int x93 = 50;
    int y93 = 2350;
    int x94 = 100;
    int y94 = 2350;
    int x95 = 50;
    int y95 = 2400;
    int x96 = 100;
    int y96 = 2400;
    int x97 = 50;
    int y97 = 2450;
    int x98 = 100;
    int y98 = 2450;
    int x99 = 50;
    int y99 = 2500;
    int x100 = 100;
    int y100 = 2500;
    int x101 = 50;
    int y101 = 2550;
    int x102 = 100;
    int y102 = 2550;
    int x103 = 50;
    int y103 = 2600;
    int x104 = 100;
    int y104 = 2600;
    int x105 = 50;
    int y105 = 2650;
    int x106 = 100;
    int y106 = 2650;
    int x107 = 50;
    int y107 = 2700;
    int x108 = 100;
    int y108 = 2700;
    int x109 = 50;
    int y109 = 2750;
    int x110 = 100;
    int y110 = 2750;
    int x111 = 50;
    int y111 = 2800;
    int x112 = 100;
    int y112 = 2800;
    int x113 = 50;
    int y113 = 2850;
    int x114 = 100;
    int y114 = 2850;
    int x115 = 50;
    int y115 = 2900;
    int x116 = 100;
    int y116 = 2900;
    int x117 = 50;
    int y117 = 2950;
    int x118 = 100;
    int y118 = 2950;
    int x119 = 50;
    int y119 = 3000;
    int x120 = 100;
    int y120 = 3000;
    int x121 = 50;
    int y121 = 3050;
    int x122 = 100;
    int y122 = 3050;
    int x123 = 50;
    int y123 = 3100;
    int x124 = 100;
    int y124 = 3100;
    int x125 = 50;
    int y125 = 3150;
    int x126 = 100;
    int y126 = 3150;
    int x127 = 50;
    int y127 = 3200;
    int x128 = 100;
    int y128 = 3200;
    int x129 = 50;
    int y129 = 3250;
    int x130 = 100;
    int y130 = 3250;
    int x131 = 50;
    int y131 = 3300;
    int x132 = 100;
    int y132 = 3300;
    int x133 = 50;
    int y133 = 3350;
    int x134 = 100;
    int y134 = 3350;
    int x135 = 50;
    int y135 = 3400;
    int x136 = 100;
    int y136 = 3400;
    int x137 = 50;
    int y137 = 3450;
    int x138 = 100;
    int y138 = 3450;
    int x139 = 50;
    int y139 = 3500;
    int x140 = 100;
    int y140 = 3500;
    int x141 = 50;
    int y141 = 3550;
    int x142 = 100;
    int y142 = 3550;
    int x143 = 50;
    int y143 = 3600;
    int x144 = 100;
    int y144 = 3600;
    int x145 = 50;
    int y145 = 3650;
    int x146 = 100;
    int y146 = 3650;
    int x147 = 50;
    int y147 = 3700;
    int x148 = 100;
    int y148 = 3700;
    int x149 = 50;
    int y149 = 3750;
    int x150 = 100;
    int y150 = 3750;
    int x151 = 50;
    int y151 = 3800;
    int x152 = 100;
    int y152 = 3800;
    int x153 = 50;
    int y153 = 3850;
    int x154 = 100;
    int y154 = 3850;
    int x155 = 50;
    int y155 = 3900;
    int x156 = 100;
    int y156 = 3900;
    int x157 = 50;
    int y157 = 3950;
    int x158 = 100;
    int y158 = 3950;
    int x159 = 50;
    int y159 = 4000;
    int x160 = 100;
    int y160 = 4000;
    int x161 = 50;
    int y161 = 4050;
    int x162 = 100;
    int y162 = 4050;
    int x163 = 50;
    int y163 = 4100;
    int x164 = 100;
    int y164 = 4100;
    int x165 = 50;
    int y165 = 4150;
    int x166 = 100;
    int y166 = 4150;
    int x167 = 50;
    int y167 = 4200;
    int x168 = 100;
    int y168 = 4200;
    int x169 = 50;
    int y169 = 4250;
    int x170 = 100;
    int y170 = 4250;
    int x171 = 50;
    int y171 = 4300;
    int x172 = 100;
    int y172 = 4300;
    int x173 = 50;
    int y173 = 4350;
    int x174 = 100;
    int y174 = 4350;
    int x175 = 50;
    int y175 = 4400;
    int x176 = 100;
    int y176 = 4400;
    int x177 = 50;
    int y177 = 4450;
    int x178 = 100;
    int y178 = 4450;
    int x179 = 50;
    int y179 = 4500;
    int x180 = 100;
    int y180 = 4500;
    int x181 = 50;
    int y181 = 4550;
    int x182 = 100;
    int y182 = 4550;
    int x183 = 50;
    int y183 = 4600;
    int x184 = 100;
    int y184 = 4600;
    int x185 = 50;
    int y185 = 4650;
    int x186 = 100;
    int y186 = 4650;
    int x187 = 50;
    int y187 = 4700;
    int x188 = 100;
    int y188 = 4700;
    int x189 = 50;
    int y189 = 4750;
    int x190 = 100;
    int y190 = 4750;
    int x191 = 50;
    int y191 = 4800;
    int x192 = 100;
    int y192 = 4800;
    int x193 = 50;
    int y193 = 4850;
    int x194 = 100;
    int y194 = 4850;
    int x195 = 50;
    int y195 = 4900;
    int x196 = 100;
    int y196 = 4900;
    int x197 = 50;
    int y197 = 4950;
    int x198 = 100;
    int y198 = 4950;
    int x199 = 50;
    int y199 = 5000;
    int x200 = 100;
    int y200 = 5000;
    int x201 = 50;
    int y201 = 5050;
    int x202 = 100;
    int y202 = 5050;
    int x203 = 50;
    int y203 = 5100;
    int x204 = 100;
    int y204 = 5100;
    int x205 = 50;
    int y205 = 5150;
    int x206 = 100;
    int y206 = 5150;
    int x207 = 50;
    int y207 = 5200;
    int x208 = 100;
    int y208 = 5200;
    int x209 = 50;
    int y209 = 5250;
    int x210 = 100;
    int y210 = 5250;
    int x211 = 50;
    int y211 = 5300;
    int x212 = 100;
    int y212 = 5300;
    int x213 = 50;
    int y213 = 5350;
    int x214 = 100;
    int y214 = 5350;
    int x215 = 50;
    int y215 = 5400;
    int x216 = 100;
    int y216 = 5400;
    int x217 = 50;
    int y217 = 5450;
    int x218 = 100;
    int y218 = 5450;
    int x219 = 50;
    int y219 = 5500;
    int x220 = 100;
    int y220 = 5500;
    int x221 = 50;
    int y221 = 5550;
    int x222 = 100;
    int y222 = 5550;
    int x223 = 50;
    int y223 = 5600;
    int x224 = 100;
    int y224 = 5600;
    int x225 = 50;
    int y225 = 5650;
    int x226 = 100;
    int y226 = 5650;
    int x227 = 50;
    int y227 = 5700;
    int x228 = 100;
    int y228 = 5700;
    int x229 = 50;
    int y229 = 5750;
    int x230 = 100;
    int y230 = 5750;
    int x231 = 50;
    int y231 = 5800;
    int x232 = 100;
    int y232 = 5800;
    int x233 = 50;
    int y233 = 5850;
    int x234 = 100;
    int y234 = 5850;
    int x235 = 50;
    int y235 = 5900;
    int x236 = 100;
    int y236 = 5900;
    int x237 = 50;
    int y237 = 5950;
    int x238 = 100;
    int y238 = 5950;
    int x239 = 50;
    int y239 = 6000;
    int x240 = 100;
    int y240 = 6000;
    int x241 = 50;
    int y241 = 6050;
    int x242 = 100;
    int y242 = 6050;
    int x243 = 50;
    int y243 = 6100;
    int x244 = 100;
    int y244 = 6100;
    int x245 = 50;
    int y245 = 6150;
    int x246 = 100;
    int y246 = 6150;
    int x247 = 50;
    int y247 = 6200;
    int x248 = 100;
    int y248 = 6200;
    int x249 = 50;
    int y249 = 6250;
    int x250 = 100;
    int y250 = 6250;
    int x251 = 50;
    int y251 = 6300;
    int x252 = 100;
    int y252 = 6300;
    int x253 = 50;
    int y253 = 6350;
    int x254 = 100;
    int y254 = 6350;
    int x255 = 50;
    int y255 = 6400;
    int x256 = 100;
    int y256 = 6400;
    int x257 = 50;
    int y257 = 6450;
    int x258 = 100;
    int y258 = 6450;
    int x259 = 50;
    int y259 = 6500;
    int x260 = 100;
    int y260 = 6500;
    int x261 = 50;
    int y261 = 6550;
    int x262 = 100;
    int y262 = 6550;
    int x263 = 50;
    int y263 = 6600;
    int x264 = 100;
    int y264 = 6600;
    int x265 = 50;
    int y265 = 6650;
    int x266 = 100;
    int y266 = 6650;
    int x267 = 50;
    int y267 = 6700;
    int x268 = 100;
    int y268 = 6700;
    int x269 = 50;
    int y269 = 6750;
    int x270 = 100;
    int y270 = 6750;
    int x271 = 50;
    int y271 = 6800;
    int x272 = 100;
    int y272 = 6800;
    int x273 = 50;
    int y273 = 6850;
    int x274 = 100;
    int y274 = 6850;
    int x275 = 50;
    int y275 = 6900;
    int x276 = 100;
    int y276 = 6900;
    int x277 = 50;
    int y277 = 6950;
    int x278 = 100;
    int y278 = 6950;
    int x279 = 50;
    int y279 = 7000;
    int x280 = 100;
    int y280 = 7000;
    int x281 = 50;
    int y281 = 7050;
    int x282 = 100;
    int y282 = 7050;
    int x283 = 50;
    int y283 = 7100;
    int x284 = 100;
    int y284 = 7100;
    int x285 = 50;
    int y285 = 7150;
    int x286 = 100;
    int y286 = 7150;
    int x287 = 50;
    int y287 = 7200;
    int x288 = 100;
    int y288 = 7200;
    int x289 = 50;
    int y289 = 7250;
    int x290 = 100;
    int y290 = 7250;
    int x291 = 50;
    int y291 = 7300;
    int x292 = 100;
    int y292 = 7300;
    int x293 = 50;
    int y293 = 7350;
    int x294 = 100;
    int y294 = 7350;
    int x295 = 50;
    int y295 = 7400;
    int x296 = 100;
    int y296 = 7400;
    int x297 = 50;
    int y297 = 7450;
    int x298 = 100;
    int y298 = 7450;
    int x299 = 50;
    int y299 = 7500;
    int x300 = 100;
    int y300 = 7500;
    int x301 = 50;
    int y301 = 7550;
    int x302 = 100;
    int y302 = 7550;
    int x303 = 50;
    int y303 = 7600;
    int x304 = 100;
    int y304 = 7600;
    int x305 = 50;
    int y305 = 7650;
    int x306 = 100;
    int y306 = 7650;
    int x307 = 50;
    int y307 = 7700;
    int x308 = 100;
    int y308 = 7700;
    int x309 = 50;
    int y309 = 7750;
    int x310 = 100;
    int y310 = 7750;
    int x311 = 50;
    int y311 = 7800;
    int x312 = 100;
    int y312 = 7800;
    int x313 = 50;
    int y313 = 7850;
    int x314 = 100;
    int y314 = 7850;
    int x315 = 50;
    int y315 = 7900;
    int x316 = 100;
    int y316 = 7900;
    int x317 = 50;
    int y317 = 7950;
    int x318 = 100;
    int y318 = 7950;
    int x319 = 50;
    int y319 = 8000;
    int x320 = 100;
    int y320 = 8000;
    int x321 = 50;
    int y321 = 8050;
    int x322 = 100;
    int y322 = 8050;
    int x323 = 50;
    int y323 = 8100;
    int x324 = 100;
    int y324 = 8100;
    int x325 = 50;
    int y325 = 8150;
    int x326 = 100;
    int y326 = 8150;
    int x327 = 50;
    int y327 = 8200;
    int x328 = 100;
    int y328 = 8200;
    int x329 = 50;
    int y329 = 8250;
    int x330 = 100;
    int y330 = 8250;
    int x331 = 50;
    int y331 = 8300;
    int x332 = 100;
    int y332 = 8300;
    int x333 = 50;
    int y333 = 8350;
    int x334 = 100;
    int y334 = 8350;
    int x335 = 50;
    int y335 = 8400;
    int x336 = 100;
    int y336 = 8400;
    int x337 = 50;
    int y337 = 8450;
    int x338 = 100;
    int y338 = 8450;
    int x339 = 50;
    int y339 = 8500;
    int x340 = 100;
    int y340 = 8500;
    int x341 = 50;
    int y341 = 8550;
    int x342 = 100;
    int y342 = 8550;
    int x343 = 50;
    int y343 = 8600;
    int x344 = 100;
    int y344 = 8600;
    int x345 = 50;
    int y345 = 8650;
    int x346 = 100;
    int y346 = 8650;
    int x347 = 50;
    int y347 = 8700;
    int x348 = 100;
    int y348 = 8700;
    int x349 = 50;
    int y349 = 8750;
    int x350 = 100;
    int y350 = 8750;
    int x351 = 50;
    int y351 = 8800;
    int x352 = 100;
    int y352 = 8800;
    int x353 = 50;
    int y353 = 8850;
    int x354 = 100;
    int y354 = 8850;
    int x355 = 50;
    int y355 = 8900;
    int x356 = 100;
    int y356 = 8900;
    int x357 = 50;
    int y357 = 8950;
    int x358 = 100;
    int y358 = 8950;
    int x359 = 50;
    int y359 = 9000;
    int x360 = 100;
    int y360 = 9000;
    int x361 = 50;
    int y361 = 9050;
    int x362 = 100;
    int y362 = 9050;
    int x363 = 50;
    int y363 = 9100;
    int x364 = 100;
    int y364 = 9100;
    int x365 = 50;
    int y365 = 9150;
    int x366 = 100;
    int y366 = 9150;
    int x367 = 50;
    int y367 = 9200;
    int x368 = 100;
    int y368 = 9200;
    int x369 = 50;
    int y369 = 9250;
    int x370 = 100;
    int y370 = 9250;
    int x371 = 50;
    int y371 = 9300;
    int x372 = 100;
    int y372 = 9300;
    int x373 = 50;
    int y373 = 9350;
    int x374 = 100;
    int y374 = 9350;
    int x375 = 50;
    int y375 = 9400;
    int x376 = 100;
    int y376 = 9400;
    int x377 = 50;
    int y377 = 9450;
    int x378 = 100;
    int y378 = 9450;
    int x379 = 50;
    int y379 = 9500;
    int x380 = 100;
    int y380 = 9500;
    int x381 = 50;
    int y381 = 9550;
    int x382 = 100;
    int y382 = 9550;
    int x383 = 50;
    int y383 = 9600;
    int x384 = 100;
    int y384 = 9600;
    int x385 = 50;
    int y385 = 9650;
    int x386 = 100;
    int y386 = 9650;
    int x387 = 50;
    int y387 = 9700;
    int x388 = 100;
    int y388 = 9700;
    int x389 = 50;
    int y389 = 9750;
    int x390 = 100;
    int y390 = 9750;
    int x391 = 50;
    int y391 = 9800;
    int x392 = 100;
    int y392 = 9800;
    int x393 = 50;
    int y393 = 9850;
    int x394 = 100;
    int y394 = 9850;
    int x395 = 50;
    int y395 = 9900;
    int x396 = 100;
    int y396 = 9900;
    int x397 = 50;
    int y397 = 9950;
    int x398 = 100;
    int y398 = 9950;
    int x399 = 50;
    int y399 = 10000;
    int x400 = 100;
    int y400 = 10000;
    int x401 = 50;
    int y401 = 10050;
    int x402 = 100;
    int y402 = 10050;
    int x403 = 50;
    int y403 = 10100;
    int x404 = 100;
    int y404 = 10100;
    int x405 = 50;
    int y405 = 10150;
    int x406 = 100;
    int y406 = 10150;
    int x407 = 50;
    int y407 = 10200;
    int x408 = 100;
    int y408 = 10200;
    int x409 = 50;
    int y409 = 10250;
    int x410 = 100;
    int y410 = 10250;
    int x411 = 50;
    int y411 = 10300;
    int x412 = 100;
    int y412 = 10300;
    int x413 = 50;
    int y413 = 10350;
    int x414 = 100;
    int y414 = 10350;
    int x415 = 50;
    int y415 = 10400;
    int x416 = 100;
    int y416 = 10400;
    int x417 = 50;
    int y417 = 10450;
    int x418 = 100;
    int y418 = 10450;
    int x419 = 50;
    int y419 = 10500;
    int x420 = 100;
    int y420 = 10500;
    int x421 = 50;
    int y421 = 10550;
    int x422 = 100;
    int y422 = 10550;
    int x423 = 50;
    int y423 = 10600;
    int x424 = 100;
    int y424 = 10600;
    int x425 = 50;
    int y425 = 10650;
    int x426 = 100;
    int y426 = 10650;
    int x427 = 50;
    int y427 = 10700;
    int x428 = 100;
    int y428 = 10700;
    int x429 = 50;
    int y429 = 10750;
    int x430 = 100;
    int y430 = 10750;
    int x431 = 50;
    int y431 = 10800;
    int x432 = 100;
    int y432 = 10800;
    int x433 = 50;
    int y433 = 10850;
    int x434 = 100;
    int y434 = 10850;
    int x435 = 50;
    int y435 = 10900;
    int x436 = 100;
    int y436 = 10900;
    int x437 = 50;
    int y437 = 10950;
    int x438 = 100;
    int y438 = 10950;
    int x439 = 50;
    int y439 = 11000;
    int x440 = 100;
    int y440 = 11000;
    int x441 = 50;
    int y441 = 11050;
    int x442 = 100;
    int y442 = 11050;
    int x443 = 50;
    int y443 = 11100;
    int x444 = 100;
    int y444 = 11100;
    int x445 = 50;
    int y445 = 11150;
    int x446 = 100;
    int y446 = 11150;
    int x447 = 50;
    int y447 = 11200;
    int x448 = 100;
    int y448 = 11200;
    int x449 = 50;
    int y449 = 11250;
    int x450 = 100;
    int y450 = 11250;
    int x451 = 50;
    int y451 = 11300;
    int x452 = 100;
    int y452 = 11300;
    int x453 = 50;
    int y453 = 11350;
    int x454 = 100;
    int y454 = 11350;
    int x455 = 50;
    int y455 = 11400;
    int x456 = 100;
    int y456 = 11400;
    int x457 = 50;
    int y457 = 11450;
    int x458 = 100;
    int y458 = 11450;
    int x459 = 50;
    int y459 = 11500;
    int x460 = 100;
    int y460 = 11500;
    int x461 = 50;
    int y461 = 11550;
    int x462 = 100;
    int y462 = 11550;
    int x463 = 50;
    int y463 = 11600;
    int x464 = 100;
    int y464 = 11600;
    int x465 = 50;
    int y465 = 11650;
    int x466 = 100;
    int y466 = 11650;
    int x467 = 50;
    int y467 = 11700;
    int x468 = 100;
    int y468 = 11700;
    int x469 = 50;
    int y469 = 11750;
    int x470 = 100;
    int y470 = 11750;
    int x471 = 50;
    int y471 = 11800;
    int x472 = 100;
    int y472 = 11800;
    int x473 = 50;
    int y473 = 11850;
    int x474 = 100;
    int y474 = 11850;
    int x475 = 50;
    int y475 = 11900;
    int x476 = 100;
    int y476 = 11900;
    int x477 = 50;
    int y477 = 11950;
    int x478 = 100;
    int y478 = 11950;
    int x479 = 50;
    int y479 = 12000;
    int x480 = 100;
    int y480 = 12000;
    int x481 = 50;
    int y481 = 12050;
    int x482 = 100;
    int y482 = 12050;
    int x483 = 50;
    int y483 = 12100;
    int x484 = 100;
    int y484 = 12100;
    int x485 = 50;
    int y485 = 12150;
    int x486 = 100;
    int y486 = 12150;
    int x487 = 50;
    int y487 = 12200;
    int x488 = 100;
    int y488 = 12200;
    int x489 = 50;
    int y489 = 12250;
    int x490 = 100;
    int y490 = 12250;
    int x491 = 50;
    int y491 = 12300;
    int x492 = 100;
    int y492 = 12300;
    int x493 = 50;
    int y493 = 12350;
    int x494 = 100;
    int y494 = 12350;
    int x495 = 50;
    int y495 = 12400;
    int x496 = 100;
    int y496 = 12400;
    int x497 = 50;
    int y497 = 12450;
    int x498 = 100;
    int y498 = 12450;
    int x499 = 50;
    int y499 = 12500;
    int x500 = 100;
    int y500 = 12500;
    int x501 = 50;
    int y501 = 12550;
    int x502 = 100;
    int y502 = 12550;
    int x503 = 50;
    int y503 = 12600;
    int x504 = 100;
    int y504 = 12600;
    int x505 = 50;
    int y505 = 12650;
    int x506 = 100;
    int y506 = 12650;
    int x507 = 50;
    int y507 = 12700;
    int x508 = 100;

```

Figura 578: Parâmetro "tipo" dentro do método "guardarFoto()"

O que me obrigou depois a alterar o chamamento desse mesmo método, tendo que adicionar o “conta.getTipo()” que me retornaria o tipo da conta.

Figura 579: Modificação na instância do método "guardarFoto()" com o "conta.getTipo()"

Logo após de ter modificado os parâmetros do método “guardarFoto()”, tive que configurar o funcionamento do mesmo, se a conta criada fosse um “membro” ou “empresa”, se fosse membro abria a “InscricoesActivity”, se fosse uma empresa abria a “EmpresaConfiguracaoActivity”.



Figura 580: Condição se o utilizador for um "membro" ou "empresa"

Voltando ao objeto “Conta”, na configuração do caminho da base de dados, tive de colocar o ID agregado ao utilizador, para de certa forma ficar mais organizado e atingir esse objetivo usando o método “getId()” já obtido dentro da própria classe.



Figura 581: Configuração do caminho da Firebase Database, na criação do utilizador

Antes de usar o método “getId()” é preciso definir o mesmo, com o “setId()” e isso é feito na própria “CriarContaActivity”, depois da criação do utilizador, com o método “task.getResult().getUser().getUid()”.



Figura 582: Definição do ID do utilizador, com o método
"task.getResult().getUser().getUid()"

Tal como referi anteriormente, se a conta fosse um “membro” abria a “InscricoesActivity”, se fosse uma empresa abria a “EmpresaConfiguracaoActivity”, tal como se observa na figura abaixo.



Figura 583: Configuração do utilizador membro e empresa

Seguidamente, achei interessante modificar o drawable que se encontra no EditText, modificando o editTextNome, visto que ele era o mesmo na empresa e no membro, mas para mim era porreiro ele trocar de imagem quando trocasse de utilizador.

<https://stackoverflow.com/questions/4502605/how-to-programmatically-set-drawableleft-on-android-button>

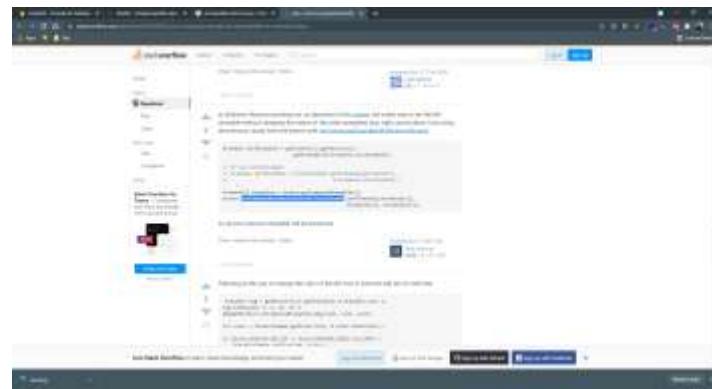


Figura 584: Solução de como definir um DrawableLeft em Java

Depois passei a aplicar o que vi no post do stackoverflow, tal como se observa na figura.



Figura 585: Criação de um Drawable de uma casa e de uma pessoa

Seguidamente, foi atribuir o “drawable” anteriormente criado dentro do “editTextNome”, com o método “setCompoundDrawableWithIntrinsicBounds()”. Se fosse um utilizador, mostrava uma pessoa, se fosse uma empresa mostrava uma casa.



Figura 586: Associação dos “drawables” anteriormente criados

De forma a organizar o código melhor, decidi criar um objeto empresa, deixando a conta para um membro normal apenas e ficou muito mais fácil a interpretação do código.

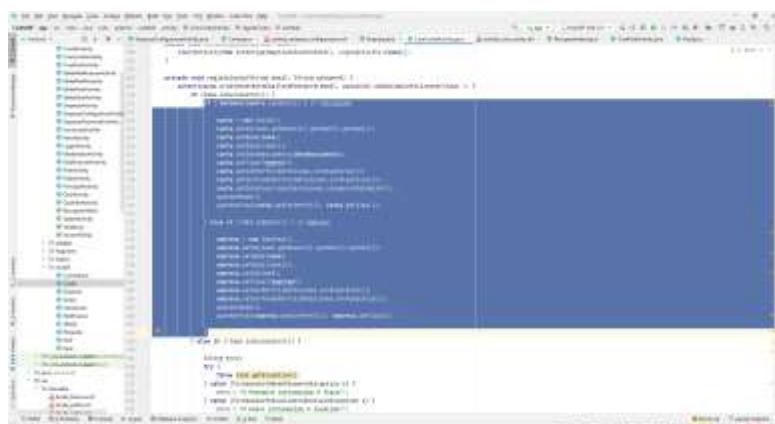


Figura 587: Criação do objeto "Empresa" e associação da empresa

Seguidamente, alterei a parte da empresa, onde a “EmpresaConfiguracaoActivity” era iniciada, passando desta vez, o objeto empresa, no seu início, visto que a atividade EmpresaConfiguracao, era um complemento de configurações desta atividade.



Figura 588: Ajustes no início da “EmpresaConfiguracaoActivity”

Para passar o objeto “Empresa” no ínicio da atividade “EmpresaConfiguracao” teria de colocar o código “implements Serializable”.



Figura 589: Implementação do "Serializable" dentro do Objeto Empresa

Coloquei o código de obter as iniciais do nome, dentro de um método chamado de "obterIniciais()", tal como se pode visualizar na imagem abaixo.

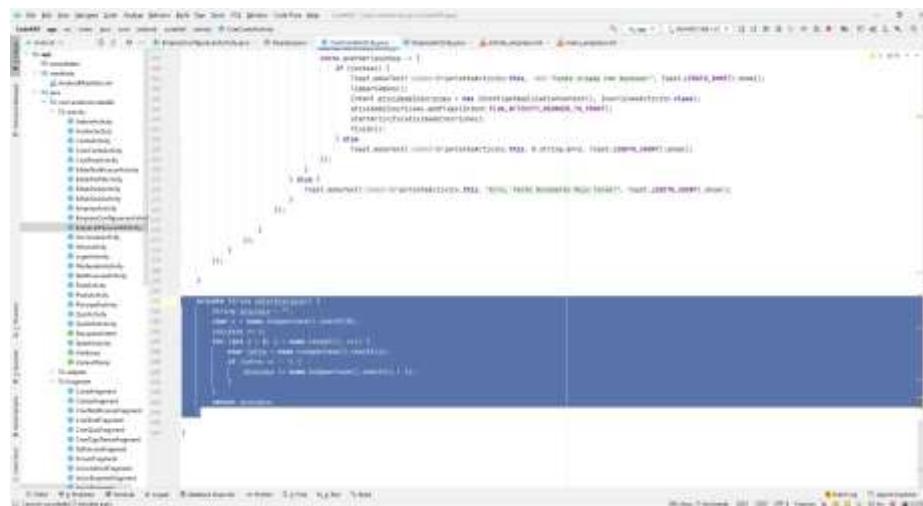


Figura 590: Criação do método "obterIniciais()"

Depois bastou apenas, associar a string iniciais ao retorno desse mesmo método.



Figura 591: String iniciais recebe o retorno do método "obterIniciais()"

Depois na parte de verificação dos campos, coloquei apenas um comprimento de dois, visto que só quero que o utilizador escreva o primeiro e último nome.



Figura 592: Verificação das iniciais do nome puder apenas conter o primeiro e último nome

No método onClick do TextView “Entrar”, é iniciada a atividade LoginActivity e esta (CriarContaActivity) é finalizada.



Figura 593: Configuração do método "onClick" do TextView "Entrar"

Para finalizar, voltei ao objeto “Conta” e adicionei o campo sexo, que poderia a ser útil na construção da biografia.

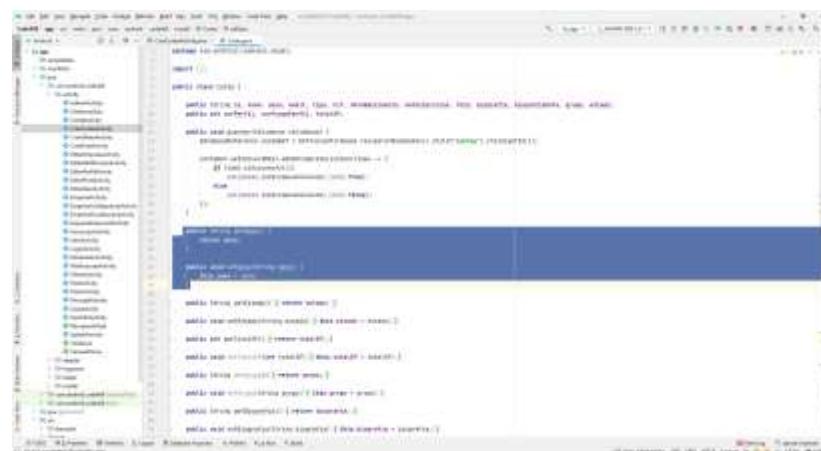


Figura 594: Criação do campo sexo dentro do objeto "Conta"

De seguida, adicionei um novo método “configurarSpinnerSexo()”, que cria uma lista com três valores, como se visualizar na figura.

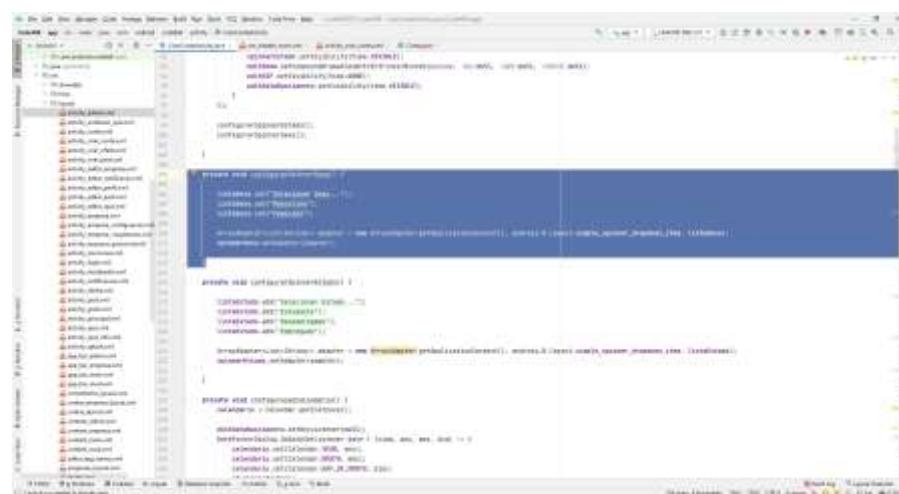


Figura 595: Método "configurarSpinnerSexo()"

Depois, só precisei de definir o sexo, consoante a posição da lista e fazer uma verificação se a posição de seleção era maior que 0, visto que essa era a primeira posição do vetor, que estava preenchida com o item “Selecionar Sexo...”, logo não poderia ser um sexo em si.



Figura 596: Definição do sexo e verificação desse mesmo campo

Tinha também, que esconder esse elemento quando o *switch* estivesse virado para a empresa e mostrar o mesmo, quando o *switch* estivesse virado para o utilizador.

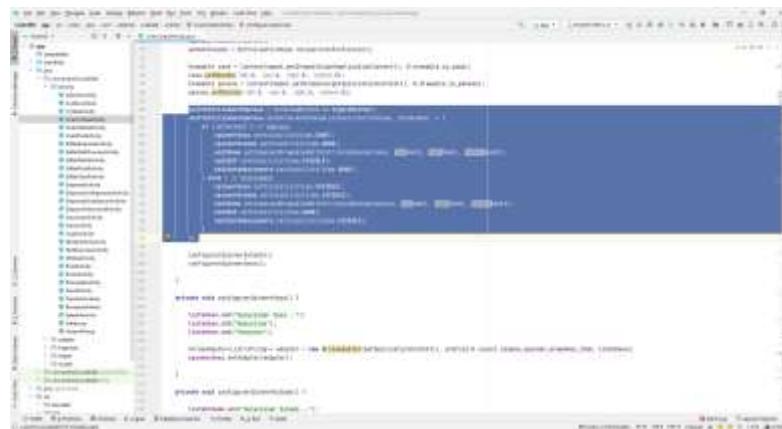


Figura 597: Configuração da visualização do SpinnerSexo

InscricoesActivity

No desenvolvimento desta atividade, comecei de imediato por iniciar o desenvolvimento do layout, como se pode observar na figura abaixo.

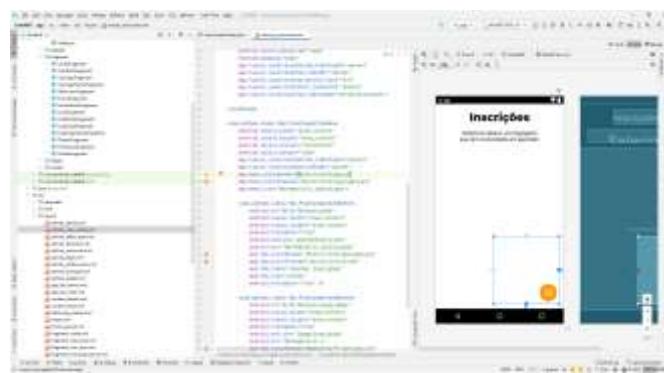


Figura 598: Layout da InscricoesActivity

Seguidamente, fui para o ficheiro Java da atividade, onde comecei por configurar os atributos iniciais.



Figura 599: Configuração dos atributos iniciais

Seguidamente, passei à definição dos atributos iniciais e também do ouvinte “onItemClickListener” da “ListView” com as inscrições do utilizador e se um item dessa lista fosse clicado, tanto o ID da inscrição tanto o seu nome, eram guardados em dois “ArrayList”, para que não se perdesse nenhum dos dados.

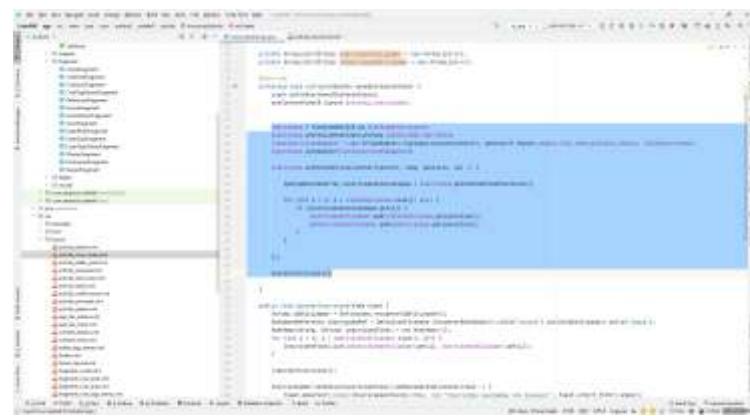


Figura 600: Configurações iniciais e configuração do método “onItemClickListener” do “ListView”

Depois, criei o método “buscarInscricoes()”, que ficaria responsável, pelo o buscar de todas as inscrições disponíveis naquele momento.



Figura 601: Método "buscarInscricoes()"

Logo após, configurei os *FloatingActionButton's*, começando pelo de guardar. Que criava um *HashMap*, para definir as tais inscrições, tal como se pode observar no código que demonstro abaixo.



Figura 602: Método guardarInscricoes()

Seguidamente, veio o método “limparInscricoes()”, utilizado para limpar a seleção do utilizador, limpando todos os vetores utilizados para guardar as inscrições

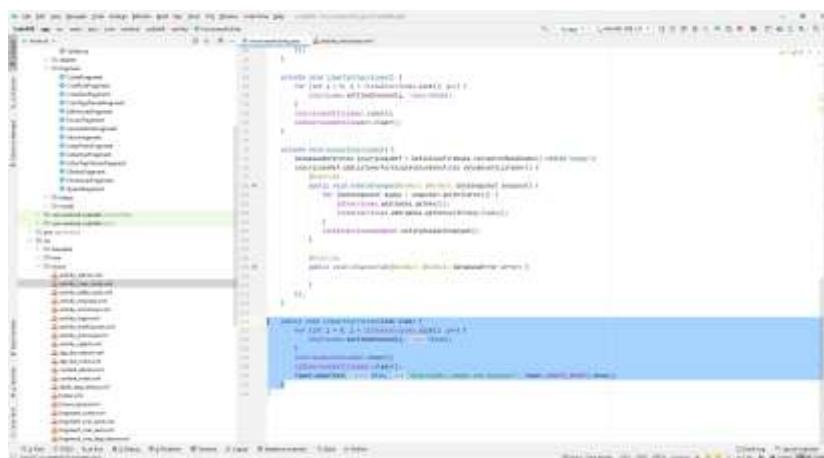


Figura 603: Método limparInscricoes()

Por motivos de otimização do código, decidi apagar o método “libertarInscricoes()”, visto que o “limparInscricoes()” já fazia esse trabalho.

Ficando então um resultado assim do género.



Figura 604: Captura de ecrã

Para finalizar esta atividade, coloquei ainda um Dialog de carregamento, para dar um efeito mais interessante à mesma.



Figura 605: Dialog de Carregamento

PrincipalActivity

Para dar início ao desenvolvimento desta atividade foi preciso criar a mesma. E de imediato coloquei a linha de código necessária para esconder a barra de ações “getSupportActionBar().hide();” dentro do ficheiro “PrincipalActivity.java”.



Figura 606: Linha de código para esconder a barra de ações

Seguidamente, fui para o ficheiro de layout “activity_principal.xml”, onde coloquei um TextView com o nome da atividade, para quando esta fosse executada no meu smartphone, eu soubesse de imediato em qual estaria, pois, a barra de ações está desativada como visto na página anterior.

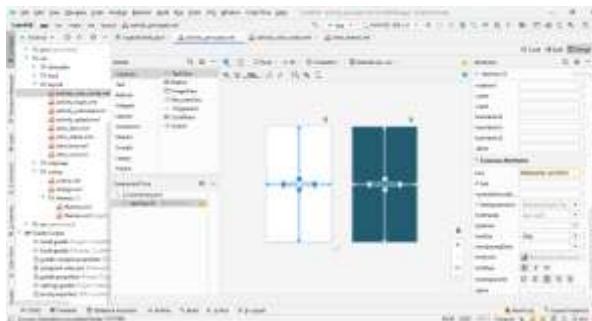


Figura 607: Inserção de um identificador no layout da atividade (TextView)

Depois de ter colocado os dois ícones dentro do layout, alinhei cada um ao topo e lateralmente com “24dp”, para dar um efeito mais interessante no layout em si.

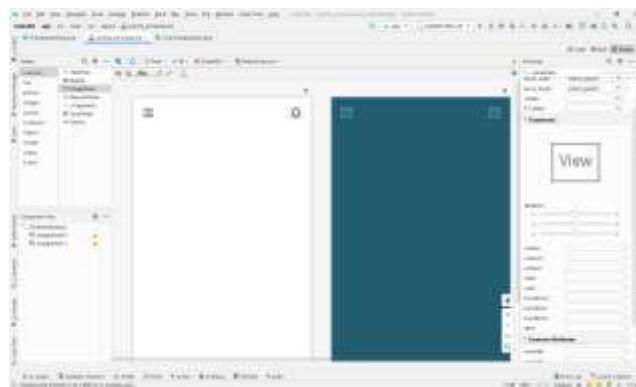


Figura 608: Alinhamento dos ícones de menu e notificações

Decidi também colocar outro ImageView, mas desta vez, com o logo da minha app, mas uma versão horizontal, com o livro aberto, para dar um efeito mais interessante ao próprio layout.

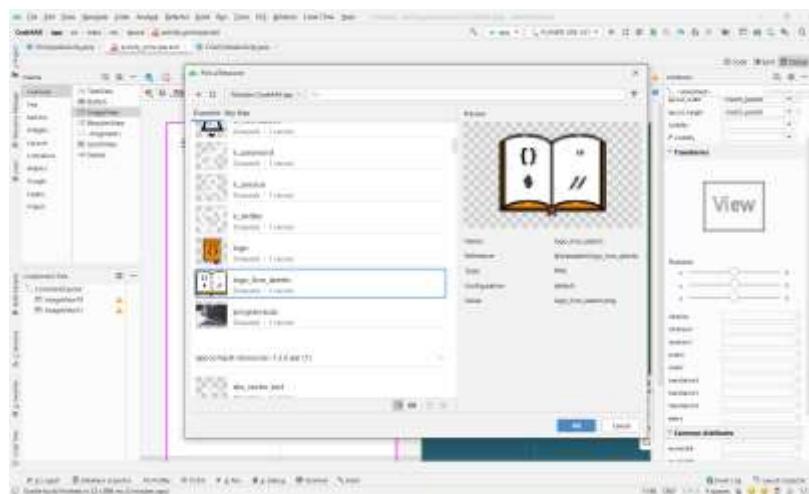


Figura 609: Componente "ImageView" com o logo Code4All (livro aberto)

Depois de ter adicionado o logo com o livro aberto, alinhei o mesmo horizontalmente no meio do layout e verticalmente alinhado no topo do layout com “24dp” de “padding”. Para deixar certo os alinhamentos, alinhei também ambos os ícones (menu e notificações) ao livro para que ficasse tudo uniforme.

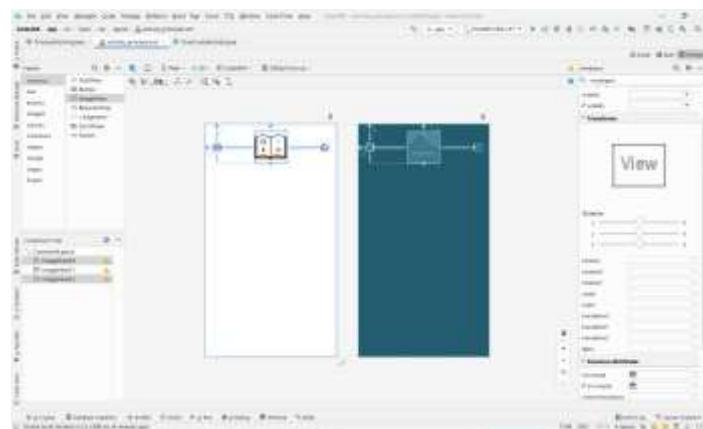


Figura 610: ImageView do logo (livro aberto) e alinhamento dos ícones (menu e notificações)

Depois adicionei dois TextView's, como se pode visualizar na figura apresentada abaixo.

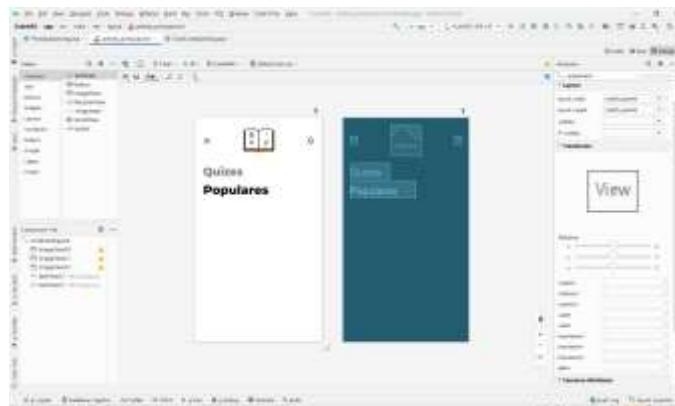


Figura 611: Layout desta atividade após da configuração dos TextView's

Depois destas alterações no layout da atividade, decidi executar a app, para ver se o layout permanecia como na pré-visualização do Android Studio. E como se observa na captura de ecrã retirada do meu dispositivo Android, o layout encontra-se bem alinhado e configurado, tal como se via na IDE.



Figura 612: Visualização do Layout dentro da app em Android

Para orientar de melhor forma o código do layout desta atividade “activity_principal.xml”, decidi então criar, dentro da pasta “res/layout” um ficheiro nomeado de “toolbar”, que iria ter como código XML, o que andei a configurar anteriormente (cabeçalho do layout).

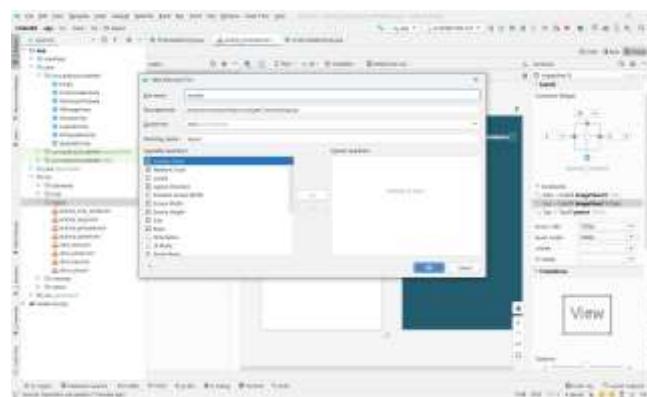


Figura 613: Criação de um ficheiro de layout nomeado de "toolbar"

Passei a abertura desse mesmo ficheiro e transferi o código do “cabeçalho” do layout da atividade para o ficheiro “toolbar.xml”, como se pode ver na figura abaixo

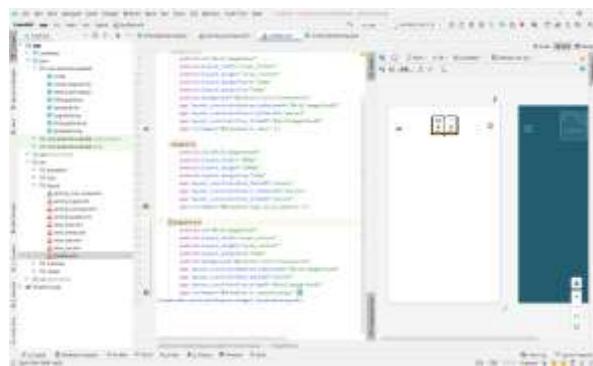


Figura 614: Transferência do código que estava no layout "activity_principal" para o "toolbar" (cabeçalho)

Voltei de novo para o ficheiro “activity_principal.xml” para então incluir o ficheiro “toolbar” no mesmo. Para realizar esse processo foi bastante simples, bastou apenas colocar a tag “include” e configurar o atributo “layout” com o caminho “@layout/toolbar” e colocar o “layout_width” como “match_parent” e o “layout_height” como “wrap_content”, como se visualiza na captura de ecrã do Android Studio.

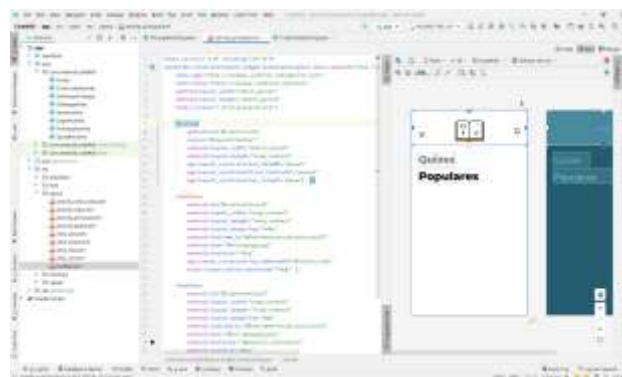


Figura 615: Inclusão do ficheiro "toolbar.xml" dentro do layout "activity_principal.xml"

Após de ter incluído o ficheiro “toolbar.xml” no layout, alinhei verticalmente o TextView “Quizes” ao mesmo, com uma “marginTop” de 24dp, ficando muito mais apelativo graficamente.

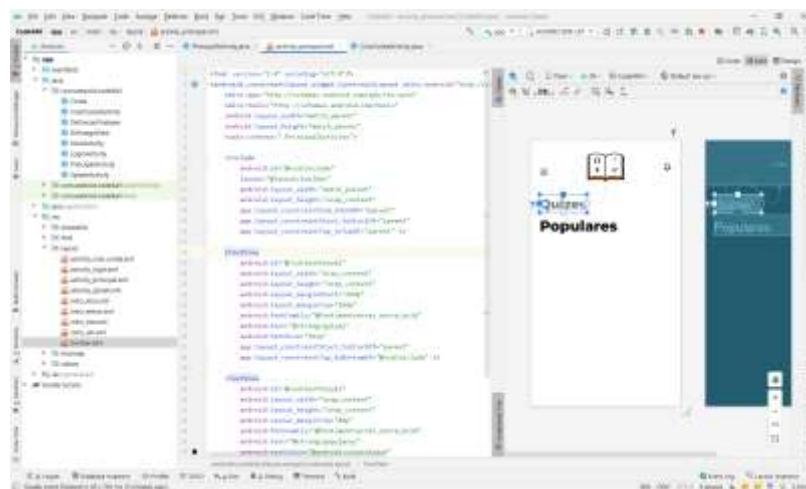


Figura 616: Alinhamentos no TextView "Quizes"

Faltava realizar a configuração do menu lateral e não existe melhor componente que um “Navigation Drawer”, que tratava de todos os meus “requisitos” de uma “cajada” só, poupando recursos e tempo de desenvolvimento.

O próprio Android Studio já trás uma “template” que inclui um “Navigation Drawer”, logo aproveitei essa template e coloquei-a no meu projeto.

Visto que a toolbar estava a dar problemas, decidi esconder a mesma, com o código que tinha colocado inicialmente nesta atividade.

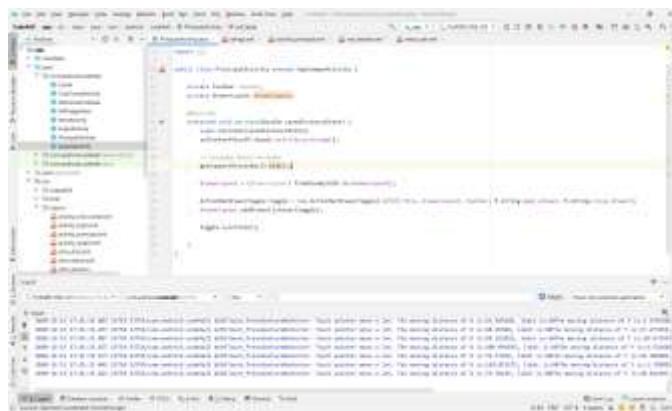


Figura 617: Inserção da linha de código para esconder a barra de ações

E ainda para ter mesmo a certeza que a barra de ações foi mesmo desativada, mudei o tema da aplicação para “Theme.MaterialComponents.DayNight.NoActionBar”. O termo “DayNight” é usado para a aplicação variar entre o tema claro e escuro do smartphone e o termo “NoActionBar” desabilita por completo a barra de ações.

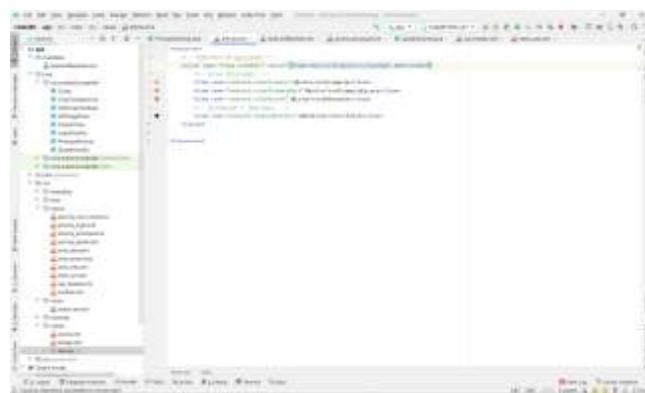


Figura 618: Mudança do tema da aplicação

Visto que desativei por completo a barra de ações, todas as linhas de código que escrevi para esconder a mesma, tiveram de ser apagadas obrigatoriamente para a aplicação conseguir executar, retirei essa linha de código nas seguintes atividades:

- `SplashActivity.java`
- `CriarContaActivity.java`
- `LoginActivity.java`

Já que estava a fazer modificações nos ficheiros todos em geral, decidi abrir o ficheiro “AndroidManifest.xml” e colocar todas as atividades existentes (naquela altura) com o atributo “screenOrientation” como “portrait”, ou seja, todas as atividades iriam executar verticalmente, por outras palavras a aplicação só iria ser executada no modo vertical, sem a possibilidade de rotacionar o ecrã.

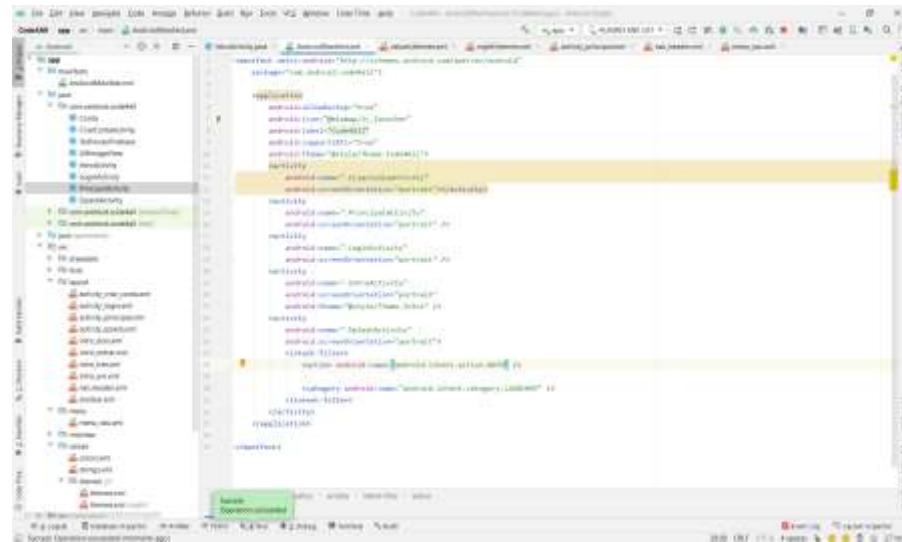


Figura 619: Definição da execução de todas as atividades em modo de "portrait" (vertical)

Seguidamente, passei à criação de uma nova atividade com a template “Navigation Drawer”

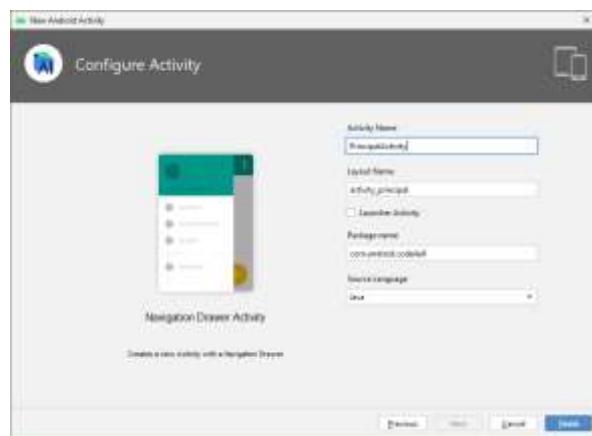


Figura 620: Criação da PrincipalActivity (NavigationDrawer)

Ao executar a app, tive um resultado deste género.



Figura 621: Captura de ecrã (PrincipalActivity)

Este foi o código que o Android Studio gerou automaticamente.

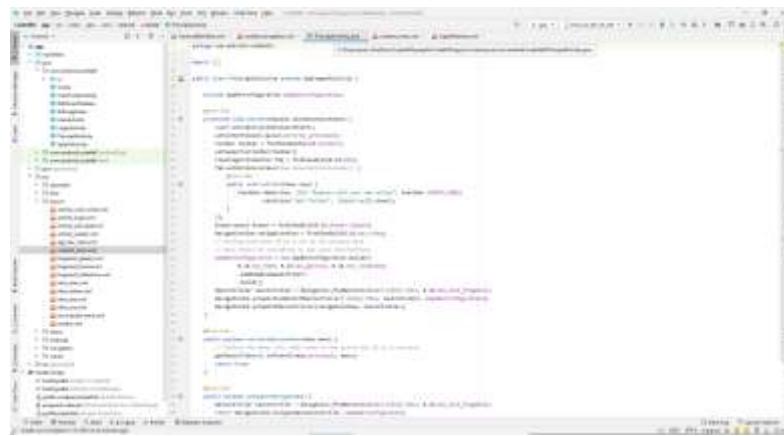


Figura 622: Código automaticamente gerado na PrincipalActivity

Seguidamente, configurei na LoginActivity, se o login fosse bem-sucedido, a PrincipalActivity, iria começar e a LoginActivity fechava, tal como se observa na figura.



Figura 623: Início da atividade (PrincipalActivity) dentro da LoginActivity

Realizei a mesma configuração dentro da `SplashActivity`.

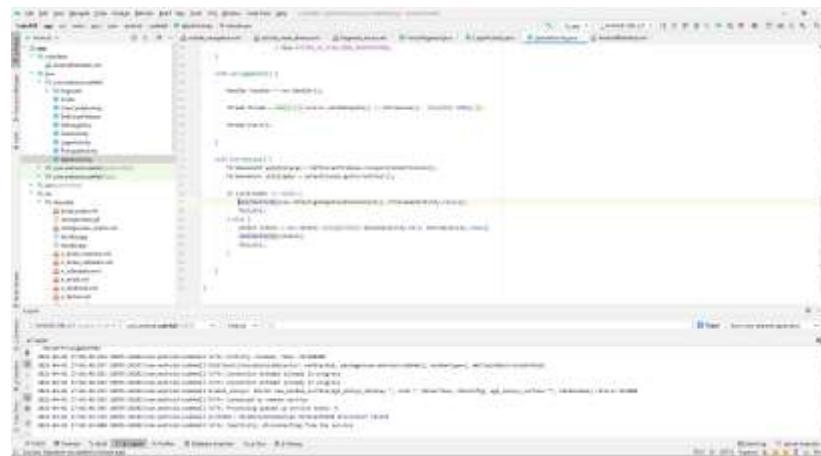


Figura 624: Início da atividade (PrincipalActivity) dentro da SplashActivity

Comecei por editar o ficheiro “`nav_header_main.xml`” responsável pelo o cabeçalho do menu lateral.

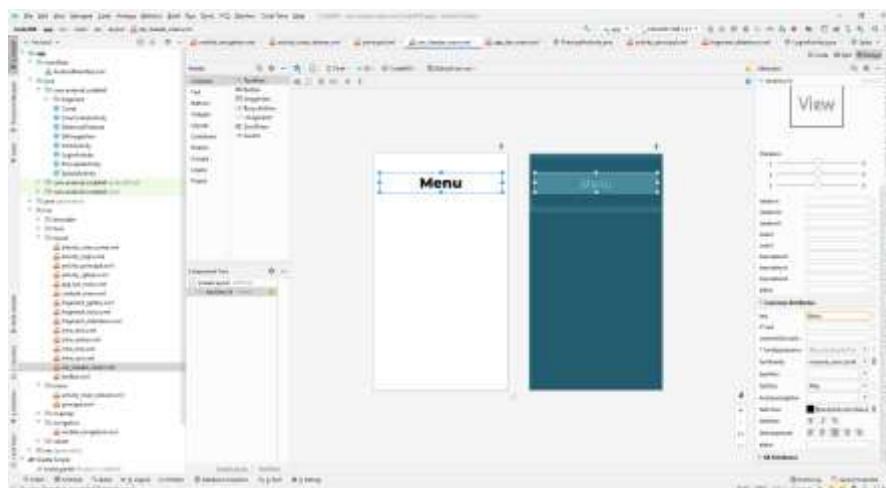


Figura 625: Edição no ficheiro "nav_header_main.xml"

Logo após ter começado a edição do menu, procurei pela solução de como retornar a versão da minha app, para colocar dentro do menu, para lhe dar assim um efeito mais interessante.

<https://stackoverflow.com/questions/4616095/how-can-you-get-the-build-version-number-of-your-android-application>

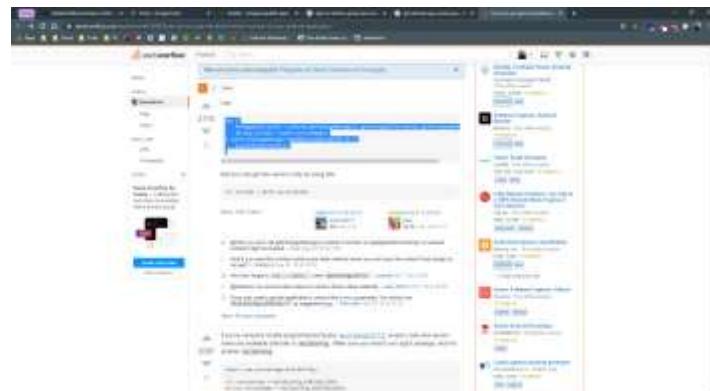


Figura 626: Solução de como obter a versão da app

Seguidamente, fui configurar o ficheiro “mobile_navigation.xml”, um dos responsáveis da troca de *fragments* e layouts do Navigation Drawer.

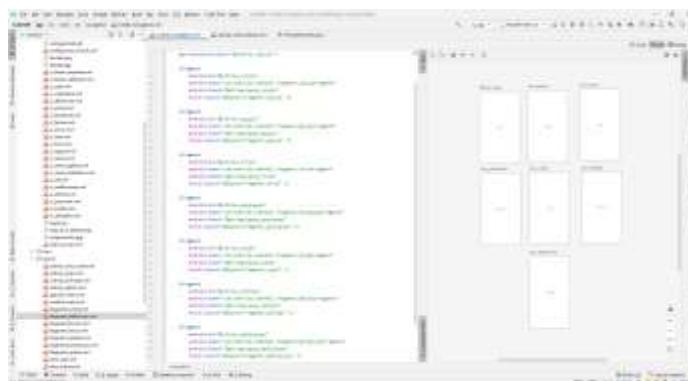


Figura 627: Configuração do ficheiro "mobile_navigation.xml"

Depois, fui configurar o menu que ficou da seguinte forma.



Figura 628: Configuração do menu da PrincipalActivity

Com o menu devidamente configurado, tive que voltar ao ficheiro da atividade, para proceder à colocação dos ids dos nav's criados no ficheiro "mobile_navigation.xml", tal como se observa na figura abaixo, para que tudo funcione corretamente.

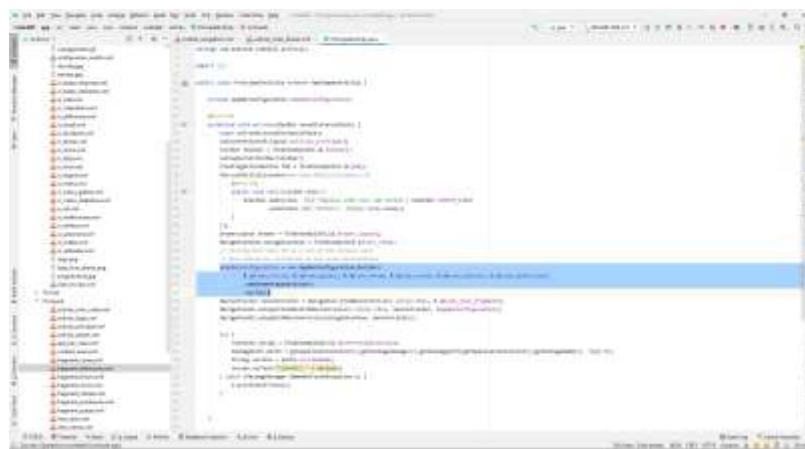


Figura 629: Colocação dos ids dentro da AppBarConfiguration

Como tinha colocado o código da toolbar dentro de um ficheiro à parte (toolbar.xml), agora só tinha que usar a tag “include” e depois a localização do ficheiro e foi o que realizei dentro da “app_bar_main.xml”.

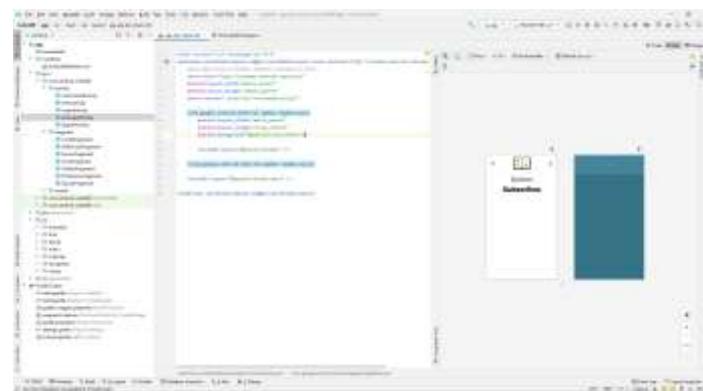


Figura 630: Include do ficheiro "toolbar.xml"

Como estava a configurar o layout da minha PrincipalActivity, aproveitei e fui pesquisar como poderia abrir a Navigation Drawer com a minha toolbar.

<https://stackoverflow.com/questions/22341816/toggle-navigation-drawer-open-on-button-image-click/>

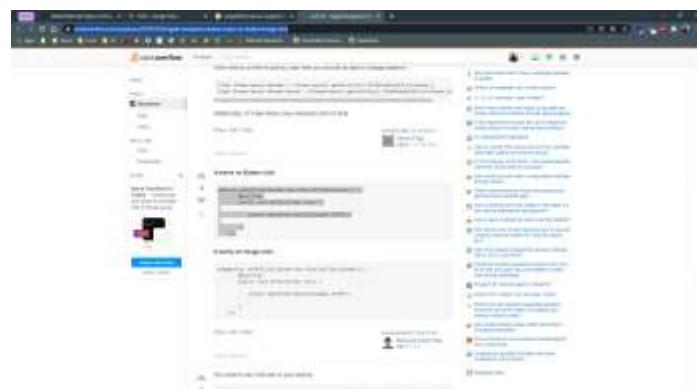


Figura 631: Solução de como abrir a Navigation Drawer no clique de um botão

Até que era bastante simples, só tinha que definir o elemento, neste caso era um “*ImageButton*” (configurado dentro do ficheiro “*toolbar.xml*”), usar o código “*findViewById*” para encontrar esse elemento. Por fim, colocar um ouvinte do clique, e colocar o código “*drawer.openDrawer(GravityCompat.START);*” e estava feito.

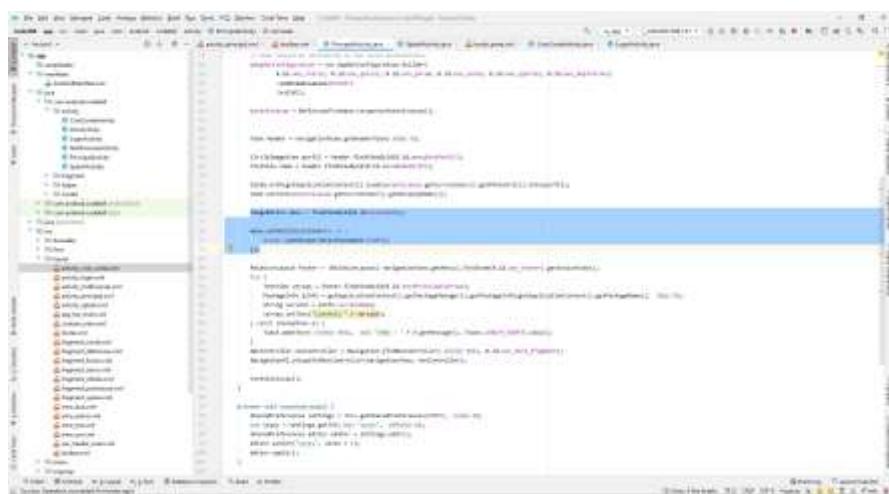


Figura 632: Configuração "onClick" do botão para abrir o Navigation Drawer

Logo depois, surgiu-me a ideia de quando o utilizador pressionar o botão para trás, deveria fechar o Navigation Drawer, se e só se, este estivesse aberto. E mais uma vez encontrei a solução para o meu problema.

<https://stackoverflow.com/questions/26833741/hide-navigation-drawer-when-user-presses-back-button/>

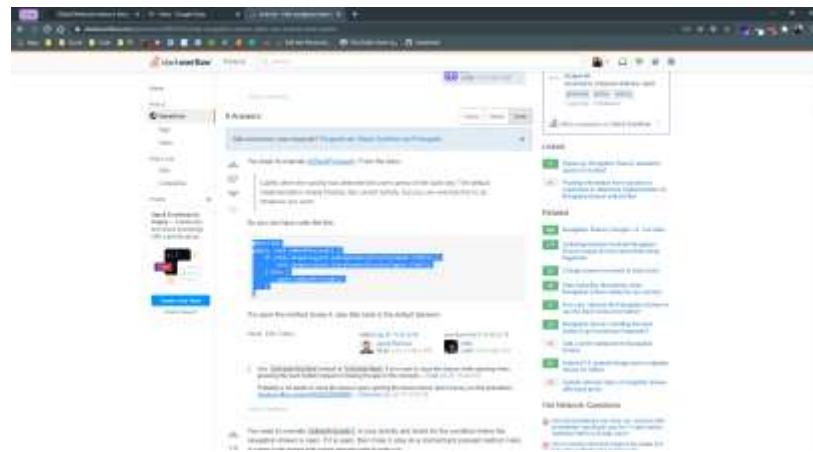


Figura 633: Solução para esconder a Navigation Drawer

Com a solução já dada, só tive de a aplicar dentro do meu projeto, tal como se observa na figura anterior.

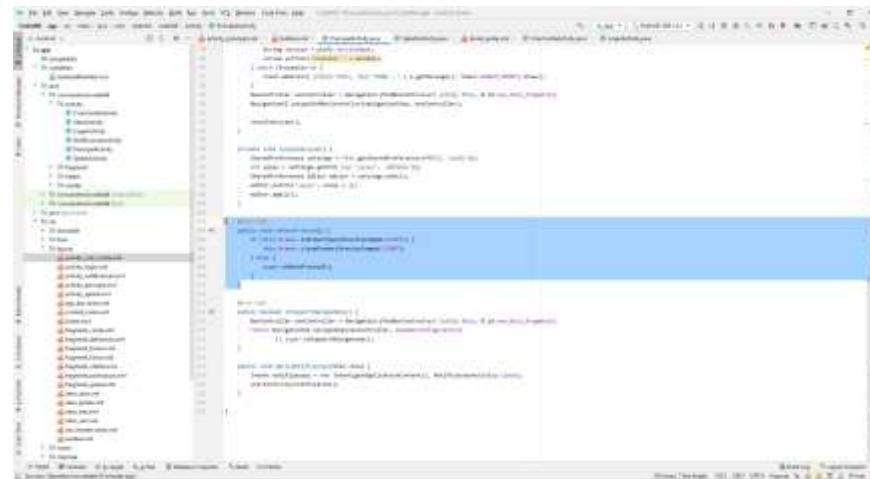


Figura 634: Solução retirada do StackOverflow - Botão para trás fecha o Navigation Drawer

Agora faltava era configurar o botão das notificações, para abrir a atividade “NotificacoesActivity”.

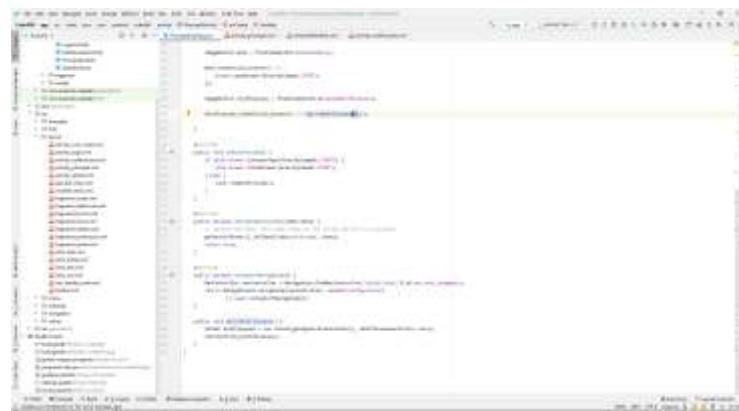


Figura 635: Configuração do botão de notificações para abrir a "NotificacaoActivity"

Depois de configurar os botões da *toolbar*, queria agora mudar a cor do menu, mas como não sabia realizar esse procedimento, fui obrigado a pesquisar novamente.

<https://stackoverflow.com/questions/30886453/change-the-color-of-a-checked-menu-item-in-a-navigation-drawer/>

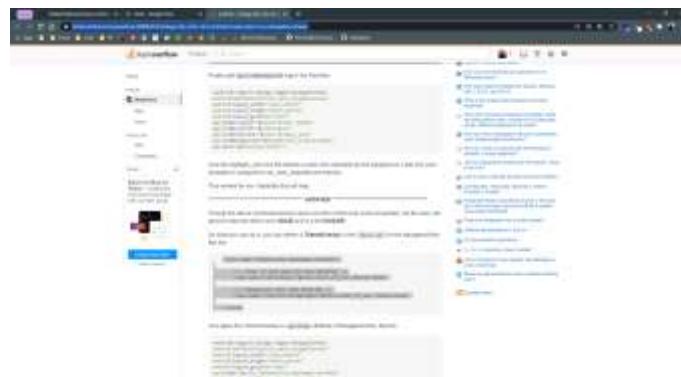


Figura 636: Solução de como mudar a cor das opções do menu (Navigation Drawer)

Com a solução de como mudar as cores das opções, foi só aplicar a mesma dentro do meu projeto, não tem que enganar.

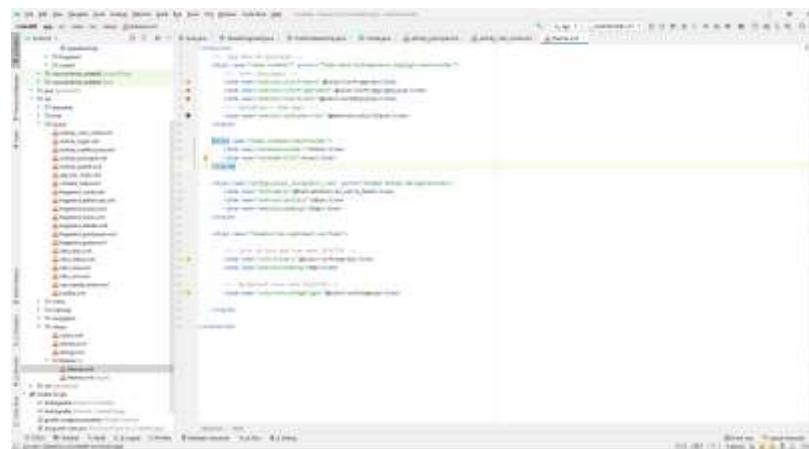


Figura 637: Criação de um novo style "configuracoes_navigation_view"

Depois só tive de aplicar os quatro parâmetros que se podem visualizar na captura de ecrã, para que a cor do menu fosse alterada para a minha cor principal (laranja).

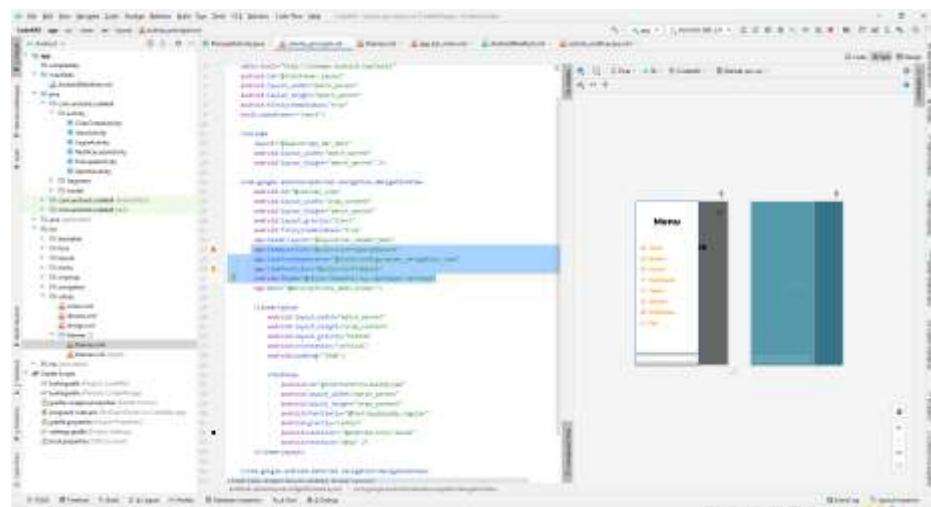


Figura 638: Mudança de cor do menu (Navigation View)

Logo após de ter configurado o Navigation View, decidi codificar a parte do sair, que o que faria era apenas “deslogar” o utilizador e finalizar a atividade em questão.

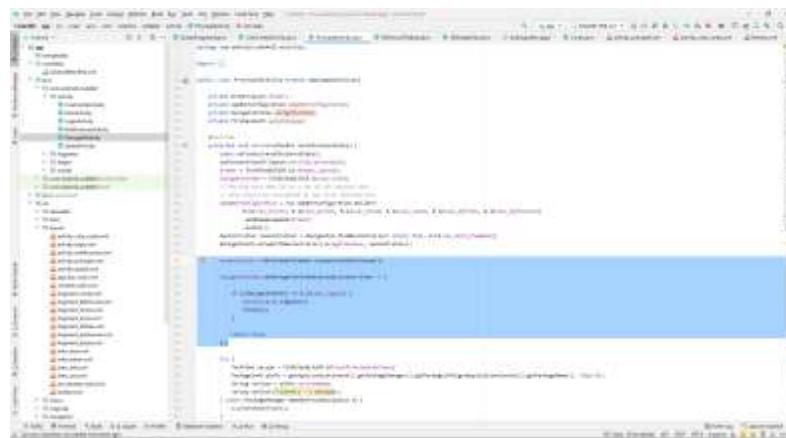


Figura 639: Configuração da opção de sair

Para finalizar a configuração do menu em si, precisava ainda de duas livrarias, uma delas a “*CircleImageView*”, para tornar as minhas imagens redondas, dando-lhes um efeito muito mais bonito e a outra “*Glide*”, para carregar as imagens da Firebase/Internet.

<https://github.com/hdodenhof/CircleImageView/>

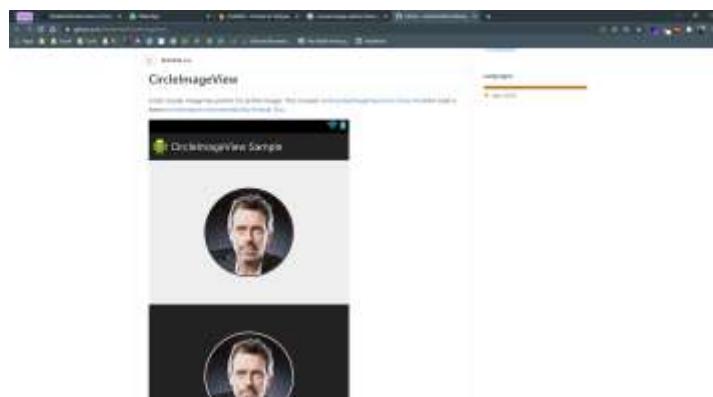


Figura 640: CircleImageView

<https://github.com/bumptech/glide/>

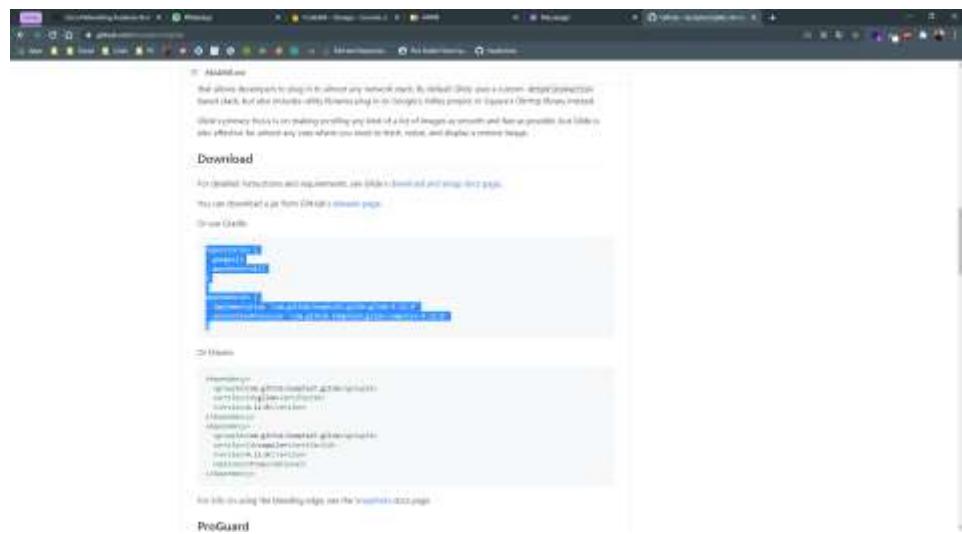


Figura 641: Glide

Com as bibliotecas já encontradas e dentro do meu projeto, precisava de saber como chegar aos elementos do Navigation Header.

<https://stackoverflow.com/questions/43284754/android-how-to-access-the-navigation-drawer-header-items/>

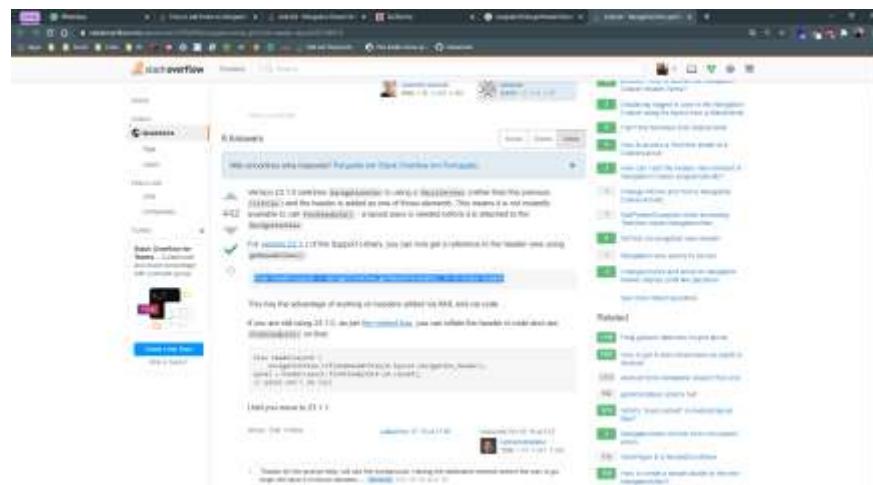


Figura 642: Solução de como recuperar a View do Navigation View Header

Depois bastou aplicar a solução encontrada dentro da atividade, como realizei na captura de ecrã, por motivos de apresentação decidi retirar o *TextView* do email.

```
View header = navigationView.getHeaderView(0);
CircleImageView perfil = header.findViewById(R.id.perfilUserPerfil);
TextView nome = header.findViewById(R.id.nomeUserPerfil);

Glide.with(getApplicationContext()).load(authenticacao.getCurrentUser().getPhotoUrl()).into(perfil);
nome.setText(authenticacao.getCurrentUser().getDisplayName());
```

Figura 643: Configuração do Navigation Header e dos seus elementos

Sem demora, procurei como poderia colocar um rodapé dentro do meu menu, para depois colocar a versão da minha app, tal como havia planeado. A solução que encontrei à primeira vista, foi a seguinte.

<https://stackoverflow.com/questions/30543605/how-to-add-footer-to-navigationview-android-support-design-library>

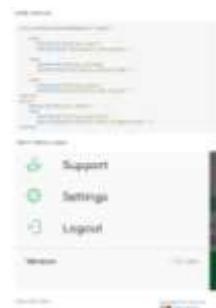


Figura 644: Solução de colocar um rodapé com a versão da app, dentro do Navigation View

Sem demora, passei à criação de uma tag “`<group>`” dentro do ficheiro de menu desta atividade, tal como a solução afirmaria.

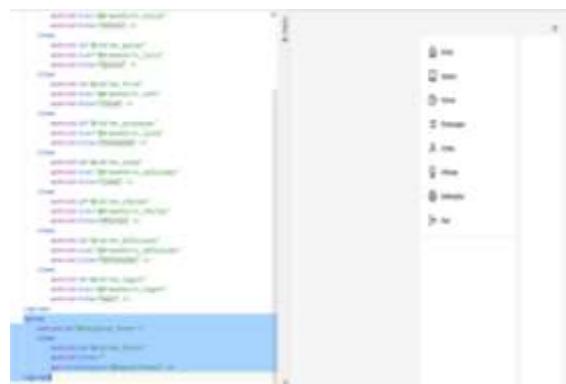


Figura 645: Rodapé do NavigationView

Depois de ter criado a tag “<group>” e associado a mesma ao ficheiro “footer.xml”, tive que o configurar, colocando então um TextView, que seria responsável por apresentar ao utilizador a versão atual da app.



Com o footer criado, fui novamente ao ficheiro da atividade, para que quando a atividade fosse iniciada, o TextView ficasse com o texto “Code4All 1.0”, neste caso é 1.0 visto que é a primeira versão.



Figura 646: Configuração do TextView do rodapé

A seguir, veio-me uma ideia à cabeça, que era o facto de perceber se era possível guardar quantas vezes a app era executada, dentro do próprio dispositivo, para que depois essas mesmas informações fossem mostradas como “easter egg” ao utilizador.

E de facto era possível, como se pode perceber pela a solução que encontrei, situada na próxima página deste relatório espetacular.

<https://stackoverflow.com/questions/10962344/how-to-save-data-in-an-android-app/>



Figura 647: Solução de como guardar dados dentro do dispositivo de execução

Com a solução já devidamente planeada, só a tinha que a executar e colocar dentro da minha atividade, tal como se observa no código abaixo. Apenas procedi à criação de um novo método, que todas as vezes que a PrincipalActivity, fosse executada, a “variável” vezes, seria auto incrementada com mais um.

Aproveitei isto e ainda realizei um código específico dentro da IntroActivity, onde fiz a verificação, se a app já tinha sido executada ou não. Se já tivesse, mostrava apenas o layout de autenticação, se não, mostrava todos.

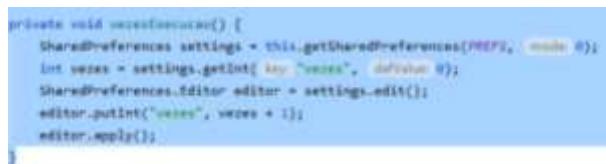


Figura 648: Método "vezesExecucao()"

Seguidamente, decidi realizar um drawable personalizado, para colocar como borda na foto de perfil de todos os utilizadores.

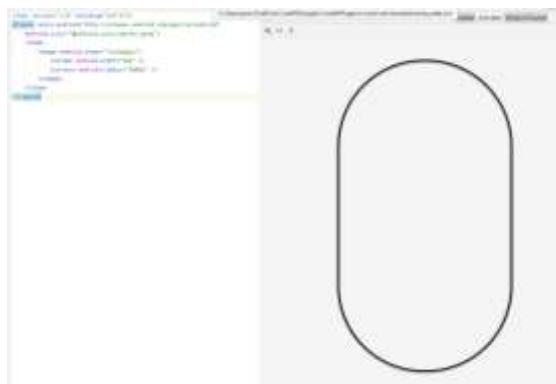


Figura 649: Ficheiro "borda_perfil.xml"

Com o drawable do perfil realizado, bastou apenas definir o atributo “background” do CircleImageView dentro do ficheiro “nav_header_main.xml”, como “borda_perfil”.

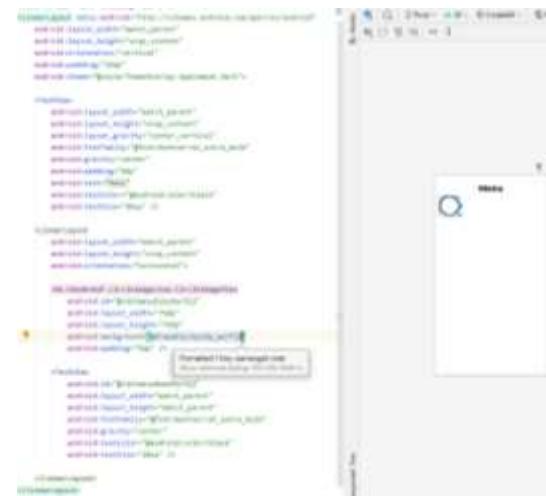


Figura 650: Atribuição do drawable "borda_perfil" como background do CircleImageView (Navigation Header)

De seguida, executei a app, para observar o resultado, da nova borda de perfil e da própria execução da app.

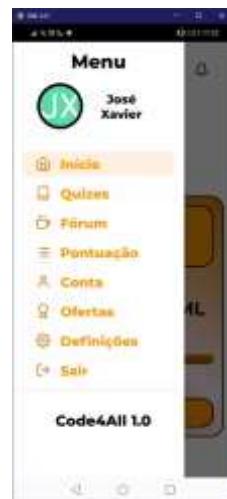


Figura 651: Navigation View da PrincipalActivity

Logo de imediato percebi, que tinha um “bug”, se o utilizador trocasse de foto ou nome, esse mesmo não era atualizado, logo tive que colocar as variáveis de perfil e nome como globais para as alterar em toda a atividade.

Para resolver esse pequeno problema, reenscrevi o método “onResume” que é executado sempre que a atividade, volta à execução.

```
onResume()
protected void onResume() {
    super.onResume();
    Glide.with(getApplicationContext()).load(authenticacao.getCurrentUser().getPhotoUri()).into(perfil);
    nome.setText(authenticacao.getCurrentUser().getDisplayName());
}
```

Figura 652: método "onResume" vai atualiza a foto e nome do utilizador

Com a anterior problema resolvido, decidi reescrever o método “onClick”, para implementar a funcionalidade de o botão quando clicado 5 vezes mostraria um aviso ao utilizador, de quantas vezes a app já foi executada.

Tive esta ideia, devido ao facto de que para ativar o modo de programador, é preciso ir às definições e clicar 5 vezes onde diz “Número de Compilação”.

[versao.onClickListener\(\) -> Definicoes_avonturevezesExecucao\(getApplicationContext\(\)\);](#)

Figura 653: Método "onClick", sobre o TextView da versão da app

O que o método realiza é incrementar a variável “cliques” cada vez que o TextView é clicado, quando chega a 5, informa ao utilizador as vezes de execução da app e por fim definir a variável “cliques” 0, para ficar preparada para o próximo clique.



```
public static int cliques = 0;
public static void quantidadeVezeExecucao(Context context) {
    if (cliques == 5) {
        SharedPreference.setLaranja = context.getSharedPreferences("PREFS", MODE_PRIVATE);
        int vezes = getLaranja.getInt("vezes", 0);
        Toast.makeText(context, "Este app, te foi executado " + vezes + " vezes!", Toast.LENGTH_SHORT).show();
        cliques = 0;
    }
}
```

Figura 654: Método "quantidadeVezeExecucao()

Configurei, também um background para o TextView da versão da app, como um retângulo laranja, tal como se pode observar.

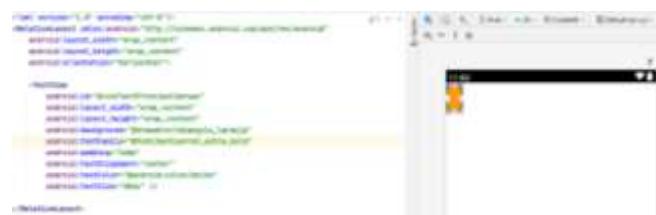


Figura 655: Adição do atributo "background" com o drawable "retangulo_laranja"

Depois de ter colocado, o efeito “retangulo_laranja”, o menu da app, o mesmo ficou com o seguinte efeito.



Figura 656: Execução da PrincipalActivity e visualização do TextView (Versão da App)

Para dar um efeito mais interessante ao menu em geral e para tudo combinar com o laranja, decidi também colocar a borda do perfil, com o fundo laranja.

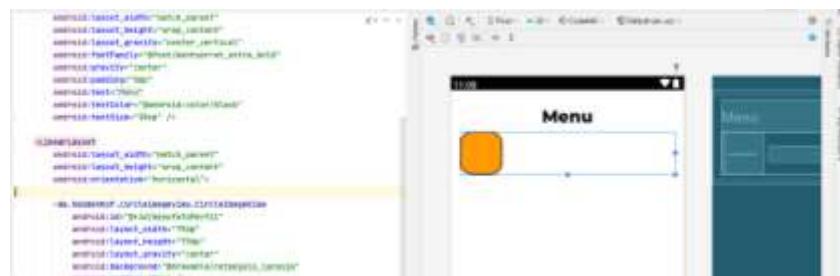


Figura 657: Fundo laranja na imagem do Perfil

NotificacoesActivity

Esta atividade, está destinada exclusivamente apenas de mostrar as notificações de todos os utilizadores, a configuração do seu layout é bastante simples, tal como se pode visualizar.



Figura 658: Layout da NotificacoesActivity

Seguidamente realizei novamente as configurações iniciais, dos elementos básicos e também procedi à regular criação de um método, neste caso, para ir buscar as notificações.



Figura 659: Configurações iniciais e criação do método "buscarNotificacoes()"

Exemplo de uma notificação.



Figura 660: Exemplo de uma notificação

InícioFragment

Este *fragment*, contém os quizzes subscriptos e os quizzes que se encontram finalizados, pelo o utilizador.

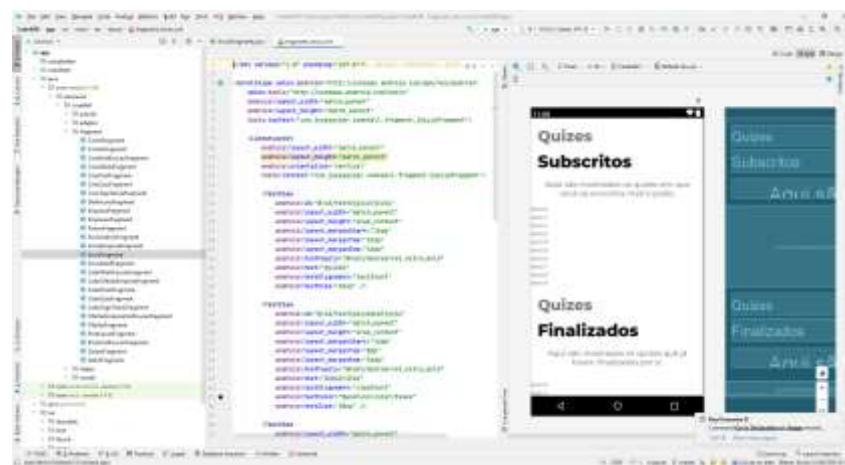


Figura 661: Layout InicioFragment

Depois do layout finalizado, configurei dois recyclerview's um para os quizzes subscriptos e outro para os quizzes finalizados.



Figura 662: Configurações Iniciais

De seguida, configurei o método “buscarQuizes()”, que me vai buscar todos os quizzes e preenchendo as listas do tipo Quiz. Se o progresso for 100, vai para a lista dos quizzes finalizados, se não vai para a lista dos quizzes subscritos, tal como se visualiza na print.

Passei como argumentos os dois recyclerView's, para evitar a criação de duas variáveis globais.



Figura 663: Método "buscarQuizes()"

QuizesFragment

O QuizesFragment, ficou com a função de exibir todos os quizzes, dividindo-os por duas categorias, a dos quizzes populares e a das subscrições, tal como se observa no layout.



Figura 664: Layout do QuizesFragment

Logo após, procedi às configurações básicas deste layout, tal como se observa na figura abaixo.



Figura 665: Configurações Iniciais

É com este método e com o seu argumento “idUtilizador”, este é necessário, visto que a app tem de ir buscar as inscrições que o utilizador colocou na criação da conta

Figura 666: Método "buscarInscricoes()"

Depois de ter a inscrição o método “buscarQuizesSubscricao()” é executado, procurando em todos os quizzes, se existe algum que tenha aquele “tema”.

Figura 667: Método "buscarQuizesSubscricao()"

Com o código da atividade dado como completo, faltava criar o layout do adapter e foi o que eu fiz.

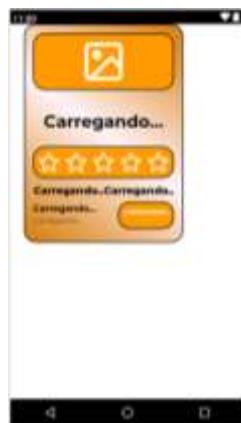


Figura 668: Configuração do layout

Dentro do adapter, configurei todos os campos, normalmente, como sempre fiz. Com a diferença que, o ImageView do Quiz abriria a QuizInfoActivity e também realizei a configuração das estrelas, ou seja, consoante a pontuação as estrelas eram pintadas de amarelo.



Figura 669: Código do Adapter

Seguidamente, veio a configuração do botão de matricular, este excerto de código verifica se o quiz em que estão já se encontra matriculado ou não, se já estiver, coloca o seu texto como “Matriculado” e a sua cor como um laranja escuro. Se não, o texto fica “Matricular” e a sua cor como laranja.

```
String id = definirQuiz.getQuiz().getQuizId();
String nomeQuiz = definirQuiz.getQuiz().getNomeQuiz();
String nomeUtilizador = definirQuiz.getQuiz().getNomeUtilizador();
String nomeQuiz2 = definirQuiz.getQuiz().getNomeQuiz();
String nomeUtilizador2 = definirQuiz.getQuiz().getNomeUtilizador();

if (totalMembros == 0) {
    nomeQuiz = "Matricular";
    nomeUtilizador = "#FF8C00";
} else if (totalMembros > 0) {
    nomeQuiz = "Matriculado";
    nomeUtilizador = "#FF4500";
}
```

Figura 670: Código do Adapter 2

Por fim, mas não menos importante, configurei o que aconteceria se o quiz não tivesse como “Matriculado” e basicamente o que acontece, é que é gerado um nó “inscricoes” seguido do id do quiz em questão e depois são colocados os parâmetros que se podem visualizar no print, o nome do botão também muda para “Matriculado” e a sua cor é alterada para laranja escuro, para finalizar é também incrementado o valor “totalMembros” situado dentro dos quizzes, visto que um utilizador foi matriculado.

A imagem do código, encontra-se na próxima página.

Figura 671: Código do Adapter 3

Continuação do código.

Figura 672: Código do Adapter 4

QuizInfoActivity

Esta atividade, será responsável por mostrar as informações introdutórias aos quizzes. Inicialmente, comecei pela criação de um *token* na *api* do Youtube, visto que a iria utilizar para os vídeos sejam mostrados dentro da minha app e não exteriormente.



Figura 673: Google Cloud Platform - Youtube API

Logo após de ter configurado, a API, tive que fazer download da biblioteca, para usar os métodos disponíveis para controlar que vídeo é carregado e assim sucessivamente.



Figura 674: Download Youtube Android Player API - Download

Depois da API criada foi me fornecida uma chave que é utilizada como credencial para comunicar com o Youtube. Esta chave é fundamental para o bom funcionamento desta atividade.



```
private static final String API_KEY = "AIzaSyCmIyfDQmWvqjyJFscUptOz0U";
```

Figura 675: Definição da API Key dentro de uma variável constante

Logo após de ter configurado a chave da API dentro da atividade, tinha que importar o ficheiro que havia anteriormente descarregado, para dentro do projeto e realizar a implementação do mesmo, dentro do meu “build.gradle”.



```
// YouTubeAPI  
implementation files('libs/YoutubeAndroidPlayerApi.jar')
```

Figura 676: Implementação da Libraria "YoutubeAndroidPlayerApi"

Como já tinha usado esta funcionalidade no curso que realizei, até que foi bastante simples de proceder às suas configurações iniciais. Basicamente o que fiz foi recuperar os elementos do layout e idQuiz, que é passado na abertura desta atividade, como já visto no fragment QuizesFragment.

Após o método de “buscarQuiz()” ser finalizado e a sua execução correr bem, os dados são atribuídos aos devidos elementos e o “youtubePlayerView” é inicializado com a chave da API e com um ouvinte para controlar as ações.



Figura 677: Configurações Iniciais

O método “buscarQuiz()” vai buscar primeiramente, a foto do Quiz e carrega-a para dentro da *ImageView* responsável por tal. Depois disso sim, vai buscar os dados da introdução, tal como se pode observar no código. Por fim, coloca o “isValidacaoSucesso” como *true*, para informar que já acabou e carregar os dados todos.



Figura 678: Método "buscarQuiz()"

Seguidamente, foi também obrigado a realizar um método “obterIDYoutube()”, visto que para carregar um vídeo do Youtube, preciso apenas do id do mesmo, não do link completo, como é passado na criação do quiz.

```
private String extraerYouTube(String linkYouTube) {
    String patron = "(?<=youtu.be/|watch\?v=|videos/|embed/|\\/|\\#\\/(\\?\\?))";
    Pattern patronCompilado = Pattern.compile(patron);
    Matcher matcher = patronCompilado.matcher(linkYouTube);
    if (matcher.find()) {
        return matcher.group();
    } else {
        return "erro";
    }
}
```

Figura 679: Método "obterIDYoutube()"

Visto que esta atividade usa a libraria `YoutubeAndroidPlayerApi`, foi preciso adicionar esta linha de código na declaração da atividade.

```
public class QuizInfoActivity extends YouTubeBaseActivity implements YouTubePlayer.OnInitializedListener
```

Figura 680: Declaração da atividade, com a libreria YoutubeAndroidPlayerApi

Ao ser implementado o método “`YoutubePlayer.OnInitializedListener`”, sou obrigado a instanciar o mesmo dentro da minha atividade.

É neste método que são realizadas todas as configurações relacionadas com o carregamento do vídeo.

O método anteriormente referido “obterIDYoutube()” é aqui utilizado para recuperar esse mesmo ID e é passado como argumento o vídeo definido na criação do quiz. Depois do id do vídeo recuperado, verifiquei se o método executou com sucesso.

Se sim, coloca o vídeo em pendente de carregamento, se não, informa que ocorreu um erro.

```
onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer youtubePlayer, boolean youtubeReady) {
    if (youtubeReady) {
        String videoId = youtubePlayer.getVideoId();
        if (videoId.equals("WzqL_1JmIw")) {
            youtubePlayer.setVolume(0);
        } else {
            Toast.makeText(getApplicationContext(), "Video ID: " + videoId, Toast.LENGTH_SHORT).show();
        }
    }
}

onInitializationFailure(YouTubePlayer.Provider provider, YouTubeInitializationResult youtubeInitializationResult) {
    Toast.makeText(getApplicationContext(), "Error in loading Player", Toast.LENGTH_SHORT).show();
}
```

Figura 681: Método "onInitializationSuccess" do YoutubePlayer

No final, esta atividade ficou da seguinte forma.



Figura 682: Captura de ecrã da QuizInfoActivity

QuizActivity

Esta atividade, é uma das atividades principais deste projeto, visto que é nela que serão feitos e mostrados os quizzes e a sua informação.



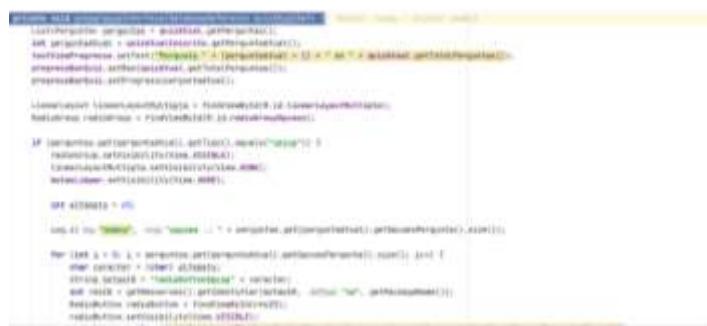
Figura 683: Layout da QuizActivity

Logo depois de ter configurado o layout desta atividade, voltei para o ficheiro java da mesma e realizei as suas configurações iniciais, tal como se pode observar na figura.



Figura 684: Configurações Iniciais

Depois de ter realizado as configurações iniciais e de ir buscar os dados corretos, o método “preparacaoInterface()” é executado e é nele que todos os elementos são carregados, tais como as opções de resposta a pergunta e o progresso atual do utilizador, como se pode observar neste excerto de código do método.



```

        List<Perguntas> perguntas = new ArrayList();
        int perguntaAtual = 0;
        String respostaCorreta = "A";
        String respostaErrada = "B";
        String respostaAlternativa = "C";
        String respostaOutra = "D";
        String respostaUser = null;
        String respostaCorretaUser = null;
        String respostaErradaUser = null;
        String respostaAlternativaUser = null;
        String respostaOutraUser = null;
        String respostaUserFinal = null;
        String respostaCorretaFinal = null;
        String respostaErradaFinal = null;
        String respostaAlternativaFinal = null;
        String respostaOutraFinal = null;

        if (perguntaAtual < 5) {
            perguntas.add(new Pergunta("Qual é o maior continente?", "A. África", "B. Ásia", "C. América do Sul", "D. Europa"));
            perguntas.add(new Pergunta("Qual é o maior oceano?", "A. Oceano Atlântico", "B. Oceano Índico", "C. Oceano Pacífico", "D. Oceano Ártico"));
            perguntas.add(new Pergunta("Qual é o maior país?", "A. Estados Unidos", "B. China", "C. Rússia", "D. Índia"));
            perguntas.add(new Pergunta("Qual é o maior lago?", "A. Lago Niassa", "B. Lago Malawi", "C. Lago Tanganica", "D. Lago Victoria"));
            perguntas.add(new Pergunta("Qual é o maior deserto?", "A. Deserto do Saara", "B. Deserto do Atacama", "C. Deserto do Gobi", "D. Deserto do Kalahari"));
        }
    }
}
    
```

Figura 685: Método "preparacaoInterface()" (excerto de código)

O método “proximaPergunta()”, ficou responsável, pelo o código necessário de trocar de uma pergunta para a próxima. No meio disto tudo, sendo alterados todos os progressos e enviados todos os dados para dentro do nó “inscricoes” dentro do utilizador, visto que é nesse mesmo que todas as informações do utilizador em relação ao quiz são guardadas.



```

        if (perguntaAtual < 5) {
            perguntas.add(new Pergunta("Qual é o maior continente?", "A. África", "B. Ásia", "C. América do Sul", "D. Europa"));
            perguntas.add(new Pergunta("Qual é o maior oceano?", "A. Oceano Atlântico", "B. Oceano Índico", "C. Oceano Pacífico", "D. Oceano Ártico"));
            perguntas.add(new Pergunta("Qual é o maior país?", "A. Estados Unidos", "B. China", "C. Rússia", "D. Índia"));
            perguntas.add(new Pergunta("Qual é o maior lago?", "A. Lago Niassa", "B. Lago Malawi", "C. Lago Tanganica", "D. Lago Victoria"));
            perguntas.add(new Pergunta("Qual é o maior deserto?", "A. Deserto do Saara", "B. Deserto do Atacama", "C. Deserto do Gobi", "D. Deserto do Kalahari"));
        }
    }
}

    
```

Figura 686: Método "proximaPergunta()"

Nas próximas prints, localizam-se os métodos que limpam todas as seleções anteriores feitas pelo próprio utilizador.

```
private void limparTudo() {  
    limparRadioButtons();  
    limparCheckBoxs();  
  
    botaoLimpar.setVisibility(View.GONE);  
    textViewSelucasNuttiga.setVisibility(View.GONE);  
}
```

Figura 687: Método "limparTudo()"

```
private void limparCheckBoxs() {  
    CheckBox cbcheckbox = findViewById(R.id.checkbox1);  
    cbcheckbox.setClickable(false);  
    cbcheckbox.setChecked(false);  
  
    CheckBox cbcheckbox2 = findViewById(R.id.checkbox2);  
    cbcheckbox2.setClickable(false);  
    cbcheckbox2.setChecked(false);  
  
    CheckBox cbcheckbox3 = findViewById(R.id.checkbox3);  
    cbcheckbox3.setClickable(false);  
    cbcheckbox3.setChecked(false);  
  
    CheckBox cbcheckbox4 = findViewById(R.id.checkbox4);  
    cbcheckbox4.setClickable(false);  
    cbcheckbox4.setChecked(false);  
  
    CheckBox cbcheckbox5 = findViewById(R.id.checkbox5);  
    cbcheckbox5.setClickable(false);  
    cbcheckbox5.setChecked(false);  
}
```

Figura 688: Método "limparCheckBoxs()"

```
private void limparRadioButtons() {  
    RadioButton rbcheckbox = findViewById(R.id.radioButton1);  
    rbcheckbox.setClickable(false);  
    rbcheckbox.setChecked(false);  
    rbcheckbox.setText("");  
  
    RadioButton rbcheckbox2 = findViewById(R.id.radioButton2);  
    rbcheckbox2.setClickable(false);  
    rbcheckbox2.setChecked(false);  
    rbcheckbox2.setText("");  
  
    RadioButton rbcheckbox3 = findViewById(R.id.radioButton3);  
    rbcheckbox3.setClickable(false);  
    rbcheckbox3.setChecked(false);  
    rbcheckbox3.setText("");  
  
    RadioButton rbcheckbox4 = findViewById(R.id.radioButton4);  
    rbcheckbox4.setClickable(false);  
    rbcheckbox4.setChecked(false);  
    rbcheckbox4.setText("");  
}
```

Figura 689: Método "limparRadioButtons()"

Depois de ter realizado os métodos para limpar a seleção do utilizador, realizei dois métodos responsáveis, pela a pesquisa de todos os dados necessários para o bom funcionamento desta atividade, neste caso.

O método “buscarInfoQuizAtual()”, é responsável por ir buscar as informações do utilizador em relação ao quiz escolhido, para perceber se este já foi iniciado ou não ou em que posição estava quando foi a última vez.



Figura 690: Método "buscarInfoQuizAtual()"

Por outro lado, o método “buscarQuiz()” é responsável por ir buscar todas as informações do quiz em si, perguntas, o XP das mesmas e o mais importante, a solução correta de cada uma das perguntas.

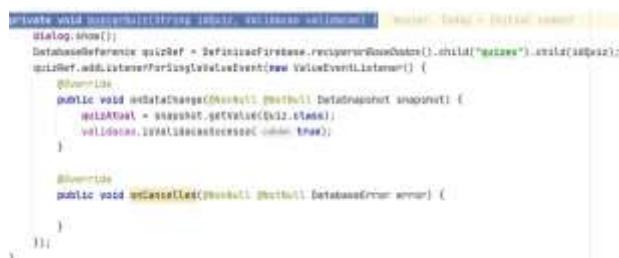


Figura 691: Método "buscarQuiz()"

Para finalizar esta atividade, criei o método “adicionarSolucao()” que é executado quando uma opção múltipla é pressionada, visto que foi a forma mais fácil que eu encontrei para realizar todo este processo de verificação, para o caso da solução ser múltipla.

O método “limparSolucaoMultipla()” foi mais para ser utilizado no caso do utilizador se enganar na seleção da sua solução final, pois ele limpa todas as variáveis e as checkboxes selecionadas, para que o utilizador faça a sua seleção novamente.

```
public void adicionaSolucoes(View view) {
    int soma = Integer.parseInt(editTextSoma.getText().toString());
    int resultado = soma + Integer.parseInt(editTextSolucoesMultiple.getText());
    textViewSolucoesMultiple.setText("Solução Atual: " + resultado);
}

public void limparSolucoesMultiple(View view) {
    editTextSolucoesMultiple.setText("");
    textViewSolucoesMultiple.setText("");
    Toast.makeText(this, "Solução Limpa com Sucesso!", Toast.LENGTH_SHORT).show();
}
```

Figura 692: Método "adicionarSolucao()" e "limparSolucaoMultipla()"

Visto que esta atividade chegou ao seu fim, queria ainda informar que existem várias condições que proíbem o utilizador de escolher todas as opções possíveis, no caso de a solução ser múltipla. E é executada uma fórmula específica para dividir os pontos consoante as certas e as erradas.

```

    for (int j = 0; j < sentence.length(); j++) {
        if (sentence.charAt(j) == sentence.charAt(j+1)) {
            count++;
        }
    }
}

int sahlepanta = sentence.substring(0, sentence.indexOf(" ")).length();

if (count > 44 && count < sentence.length() / 2) {
    double desconto = (double) count / sentence.length();
    sahlepanta = (int) Math.round(sahlepanta * desconto);
} else if (count == 80) {
    sahlepanta = 40;
}

```

Figura 693: Uma das condições do caso de a solução ser múltipla

AvaliacaoQuiz

Esta atividade, ficou destinada para ser exclusivamente executada no final de todos os quizzes, para que o utilizador desse a sua avaliação, alterando assim a pontuação geral do quiz.

Quanto maior a pontuação do quiz melhor, visto que será logo mostrado ao utilizador, quando este entrar na app.

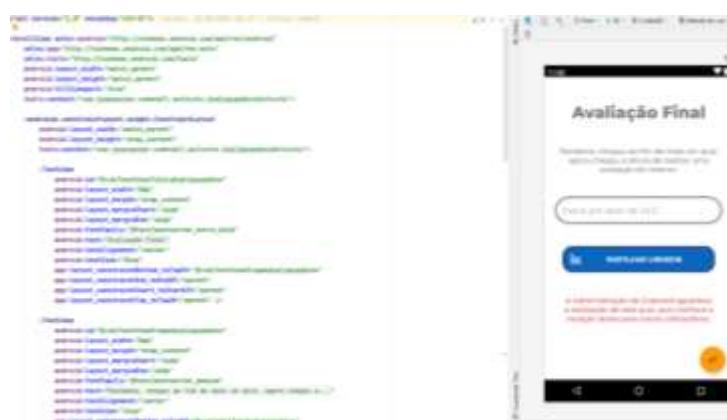


Figura 694: Layout da AvaliacaoQuizActivity

Seguidamente, configurei a partilha no *Linkedin*, com o link da minha *Landing Page*, tal como se pode observar na figura abaixo.



Figura 695: Método "partilharLinkedIn()"

Por fim, veio a configuração do *FloatingActionButton*, responsável por pegar na classificação dada pelo utilizador e realizar uma média com a classificação atual do quiz, depois dessa média estar feita envia e atualiza a média do quiz.

```

onOptionsItemSelected(MenuItem item) {
    DatabaseReference mediaRef = DatabaseReference.getInstance().getReference().child("users").child(userID);
    String avaliação = editTextAvaliação.getText().toString();
    if (avaliação.isEmpty()) {
        if (double.parseDouble(avaliação) < 0) {
            mediaRef.addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    double média = dataSnapshot.getDouble("media");
                    double novaMédia = (máxima + média) / 2;
                    double classificacao =	Double.parseDouble(editTextClassificação.getText());
                    double classificaçãoFinal = (classificacao + Double.parseDouble(avaliação)) / 2;
                    TaskManager tarefa = new TaskManager();
                    tarefa.setTask("classificação", classificaçãoFinal);
                    mediaRef.updateChildren(tarefa).addOnCompleteListener(task -> {
                        if (task.isSuccessful()) {
                            Toast.makeText(getApplicationContext(), "Sua avaliação foi salvo!", Toast.LENGTH_SHORT).show();
                            if (isUpdateClicked)
                                startMainActivity(getApplicationContext(), PrincipalActivity.class);
                        } else
                            Toast.makeText(getApplicationContext(), "Erro: Tente novamente mais tarde!", Toast.LENGTH_SHORT).show();
                    });
                }

                @Override
                public void onCancelled(DatabaseError error) {
                }
            });
        } else
            toast.showError("Insira uma avaliação maior que 0!");
    } else
        toast.showError("Insira uma avaliação válida!");
}
    
```

Figura 696: Método "onClick" do FAB

ForumFragment

Neste fragment, são apresentadas todas as publicações aceites e aprovadas pelo os administradores, ou criadas pela a administração.

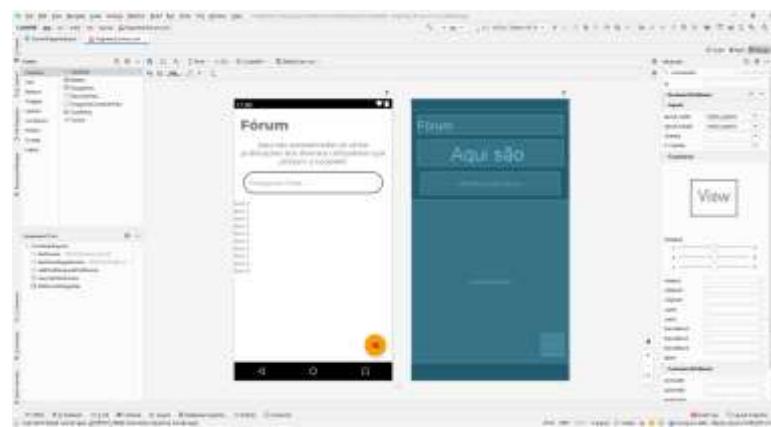


Figura 697: Layout do ForumFragment

De seguida, realizei as configurações iniciais dos elementos básicos de operação com o layout.



Figura 698: Configurações iniciais

Defini também o evento de clique dentro do *RecyclerView*, visto que quando um *post* fosse clicado, era preciso ser aberta a atividade “*PostActivity*”, que seria responsável por mostrar todos os dados desse mesmo *post*.



Figura 699: Configuração do clique no RecyclerView

Depois de ter configurado o evento “*onClick*”, tive de configurar o que o método “*cliqueCurto()*” no *RecyclerView* fazia e neste caso abriria a “*PostActivity*” como já mencionado anteriormente, passando o id do post selecionado. O método “*filtrar()*”, é responsável por filtrar os dados, como já dito neste relatório e por fim o método “*buscarForum()*” que vai buscar todos os posts.



Figura 700: Métodos “*cliqueCurto()*”, “*filtrar()*” e “*buscarForum()*”

Com a atividade devidamente configurada, passei à configuração do *adapter*, começando então pelo o layout do mesmo.

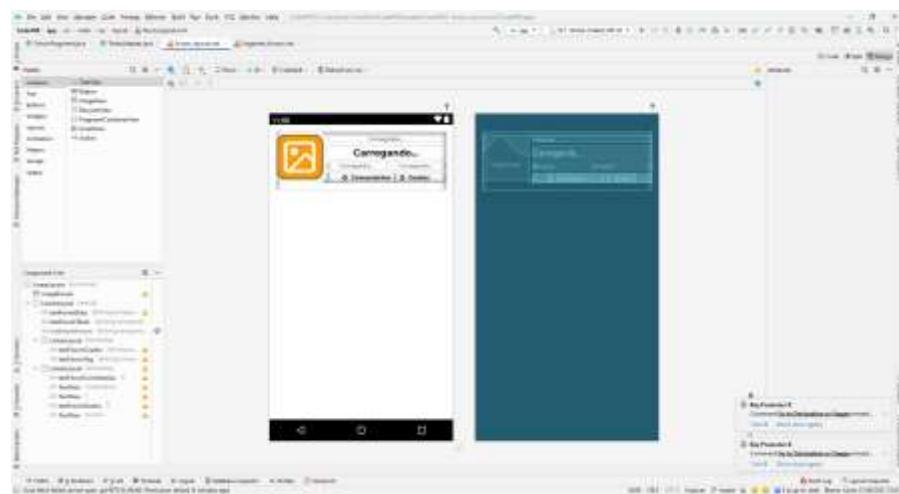


Figura 701: Layout do PostsAdapter

Seguidamente, no método “*onBindViewHolder*”, apenas precisei de associar os dados aos seus corretos elementos e adicionei uma verificação para que no *PostsActivity*, o utilizador perceba se o seu post já foi aceite ou não.

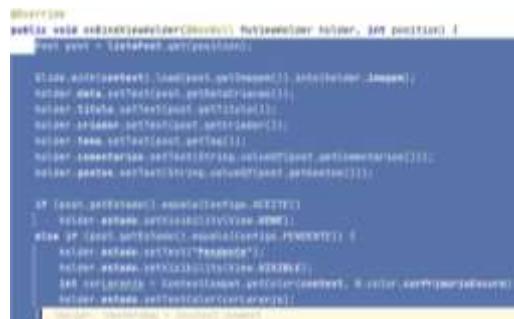


Figura 702: Método "onBindViewHolder"

Para finalizar esta atividade, coloco uma captura de ecrã da mesma



Figura 703: Execução do ForumFragment

PostActivity

Esta atividade, fica destinada à visualização do post que foi clicado dentro do *ForumFragment*, comecei então por estruturar o layout da seguinte forma.



Figura 704: Layout da PostActivity

Depois de ter finalizado a construção do layout, comecei por desenvolver as configurações básicas para esta atividade arrancar sem problemas, tal como se pode observar na figura.



Figura 705: Configurações Iniciais

Depois de ter realizado as configurações iniciais, decidi configurar a funcionalidade do botão de gosto. Sendo o seu texto igual a “Gostei”, significa que já existe um gosto por parte do utilizador no post, logo, se ele clicar novamente o gosto irá ser removido e o texto do botão passará a ser “Gosto”, tal como se vê na figura.



Figura 706: Método "onClick" do botão gosto

Se não, um novo objeto do tipo “Gosto”, será criado e enviado para a base de dados e o texto do botão ser atualizado para “Gostei”. Quero ainda ressaltar que sendo adicionado ou removido um gosto, o nó de “gostos” do post também é atualizado consoante a operação realizada pelo o utilizador.



Figura 707: Método "onClick" do botão gosto 2

Com o botão de gosto totalmente configurado, passei à configuração do botão de comentário, usando um AlertDialog para me facilitar o processo.

```

    else (String button button = new String ("button"); button.setPriority (0));
        button.setCaption ("button"); button.setMnemonic (0));
        button.setAccelerator (null);
        button.addActionListener (new ActionListener () {
            public void actionPerformed (ActionEvent e) {
                System.out.println ("button clicked!");
            }
        });
        contentPanel.add (button, button.getLayout ().getVerticalTextPosition ());
        button.repaint ();
    }
}

```

Figura 708: Método "onClick" do botão comentar

Se a opção “Confirmar” fosse clicada, era criado um objeto do tipo “Comentario” e guardados todos os dados necessários. De seguida, após do comentário ser enviado, a contagem dos comentários era também incrementada.

Se a opção clicada fosse “Cancelar”, nada era realizado, a não ser o fecho do próprio *Dialog*.



Figura 709: Opção positiva do AlertDialog anteriormente criado

Para finalizar a criação do *AlertDialog*, é obrigatório, colocar o código “builder.show()”, se este não for colocado, o mesmo não é aberto. Depois, passei à pesquisa do post com base do seu id.



Figura 710: Finalização da configuração do *AlertDialog*

Logo após, criei dois métodos responsáveis pela a pesquisa dos dados, tanto do *Post* em si, como os comentários do mesmo, como demonstro na figura.



Figura 711: Método "buscarComentarios()" e "buscarPost()"

Com a configuração da atividade mais ou menos orientada, passei à configuração do *adapter* dos comentários, o seu layout está presente na figura abaixo. Um layout bastante básico, por si mesmo.

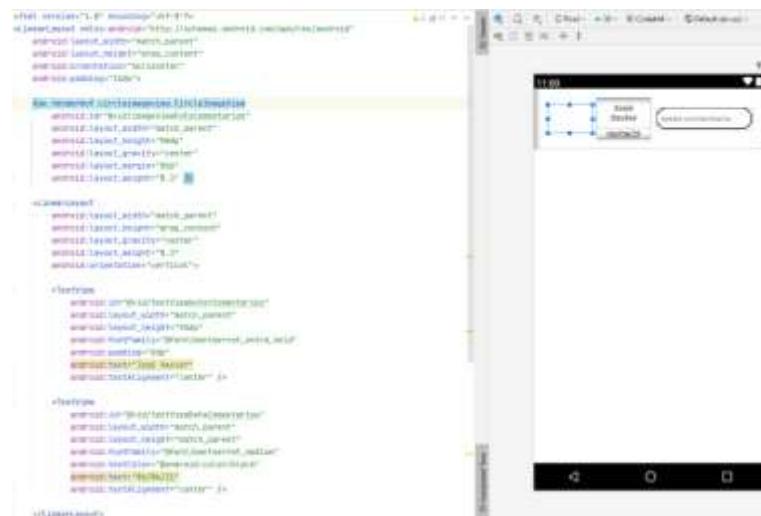


Figura 712: Configuração do layout do ComentariosAdapter

Com o layout finalizado, só tive de colocar os dados a corresponder aos elementos corretos.

```
    @Override
    public void onBindViewHolder(@NonNull ViewHolder viewHolder, int position) {
        Comentario comentario = listaComentarios.get(position);
        if (comentario != null) {
            viewHolder.textViewNome.setText(comentario.getNome());
            viewHolder.textViewData.setText(comentario.getData());
            viewHolder.textViewMensagem.setText(comentario.getMensagem());
        }
    }
```

Figura 713: Método "onBindViewHolder()"

Depois, tive de voltar para o ficheiro da atividade, para realizar as últimas configurações no mesmo. Começando com a possibilidade dos administradores e moderadores de eliminar qualquer comentário em qualquer post, mas como é óbvio um utilizador normal (membro), apenas pode eliminar o seu.



Figura 714: Condição que possibilita a edição da administração de todos os comentários

Configurei também o clique curto no recyclerView, configurando-o para a edição do comentário clicado, tal como se pode observar na figura abaixo.



Figura 715: Criação do AlertDialog para a edição do comentário

Por fim, faltava ainda configurar a remoção do comentário e deixei essa função, quando um clique longo fosse executado dentro, tal como se visualiza código abaixo. Quero ainda realçar que se um comentário é eliminado a contagem dos comentários também é removido um valor.



Figura 716: Configuração da remoção do comentário

Para finalizar, encontra-se uma captura de ecrã da atividade.



Figura 717: Captura de ecrã da PostActivity

PostsActivity (Posts do Utilizador)

Esta atividade, ficou destinada exclusivamente para os membros saberem os posts que já realizaram, o controlo dos mesmos e se esses foram já aceites ou não.

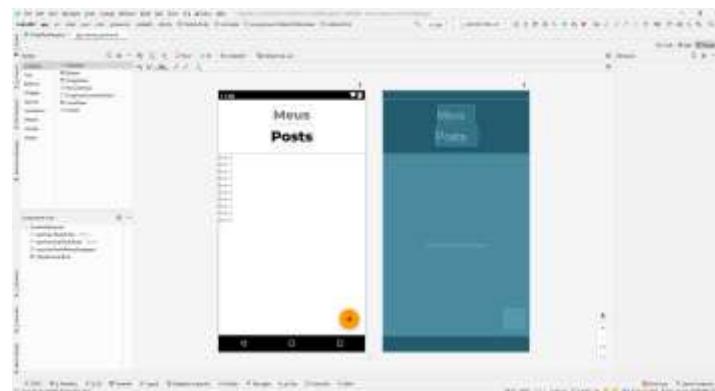


Figura 718: Layout da PostsActivity

Depois de ter configurado o layout, mais uma vez procedi à realização das configurações básicas de todos os elementos.

Quero ainda informar, que esta atividade, reutiliza o *PostsAdapter*, reutilizando assim o *adapter* criado dentro do *ForumFragment*.



Figura 719: Configurações Básicas

Como já efetuado nas outras atividades/*fragments*, deixei configurado o clique curto, como a edição do post, sendo começada a *EditarPostActivity*. No clique longo, defini a remoção do post.

```

recyclerPosts.addOnItemTouchListener(new RecyclerItemClickListener(this, recyclerPosts, new RecyclerItemClickListener.OnItemTouchListener() {
    @Override
    public void onItemTClick(View view, int position) { // Editar Post
        if (listPosts.get(position).getEstado().equals(PostActivity.PENDENTE)) {
            AlertDialog.Builder builder = new AlertDialog.Builder(PostActivity.this);
            builder.setTitle("Editar Post");
            builder.setNegativeButton("Cancelar", dialog);
            builder.setPositiveButton("Confirmar", dialog, which) -> {
                Intent editarPost = new Intent(getApplicationContext(), EditarPostActivity.class);
                editarPost.putExtra("idPost", listPosts.get(position).getId());
                startActivity(editarPost);
            });
            builder.setNegativeButton("Remover", dialog, which) -> dialog.dismiss();
            builder.show();
        }
    }

    @Override
    public void onItemLClick(View view, int position) { // Remover Post
        AlertDialog.Builder dialog = new AlertDialog.Builder(PostActivity.this);
        dialog.setTitle("Remover Post");
        dialog.setMessage("Tem a certeza que pretende eliminar o Post " + listPosts.get(position).getTitulo() + "?");
        dialog.setPositiveButton("Sim", (dialog, which) -> {
            DialogRemovePosts(dialog);
            DatabaseReference ref = FirebaseDatabase.getInstance().getReference("posts").child(listPosts.get(position).getId()).removeValue(error, ref) -> {
                if (error == null) {
                    dialog.dismiss();
                    Toast.makeText(getApplicationContext(), "Post removido com sucesso!", Toast.LENGTH_SHORT).show();
                } else {
                    dialog.dismiss();
                    Toast.makeText(getApplicationContext(), "Ops, Verifique sua conexão.", Toast.LENGTH_SHORT).show();
                }
            });
            dialog.setOnDismissListener();
        });
        dialog.setNegativeButton("Não", (dialog, which) -> dialog.dismiss());
        dialog.show();
    }
})

```

Figura 720: Clique curto e clique longo do RecyclerView

Seguidamente, decidi configurar o *FloatingActionButton*, como aquela que iria criar os posts dos membros, abrindo então a *CriarPostActivity*

```

FloatingActionButton fabCriarPost;
fabCriarPost = findViewById(R.id.fabAdicionarPost);
fabCriarPost.setOnClickListener() -> {
    Intent criarPostActivity = new Intent(getApplicationContext(), CriarPostActivity.class);
    criarPostActivity.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
    startActivity(criarPostActivity);
};

fabCriarPost.setOnClickListener();
view.setOnClickListener(v -> fabCriarPost.performClick());

```

Figura 721: Configuração do clique do FloatingActionButton

CriarPostActivity

Esta atividade fica disponível apenas para uso dos membros, para eles criarem os posts, mas como é óbvio se o post é criado por esta via, é óbvio que na finalização da sua criação, o mesmo não irá logo ser aceite, este irá precisar de uma confirmação adicionar da administração da app.



Figura 722: Layout do CriarPostActivity

Depois de ter configurado o layout, realizei, à semelhança de todos os outros ficheiros, as configurações iniciais.



Figura 723: Configurações Iniciais

Depois de ter realizado as configurações básicas da atividade, coloquei este método “verificarCampos()”, dentro do clique do botão, passando como argumentos todos os dados que precisava para submeter um post e fazendo a verificação de um a um, tal como se observa na figura abaixo.

```
private void verificarCampos(String titulo, String descricao, int tag) {
    if (!titulo.isEmpty()) {
        if (!descricao.isEmpty()) {
            if (tag > 0) {
                if (image != null) {
                    dialog.show();
                    criarPost(titulo, descricao, tag);
                } else {
                    Toast.makeText(getApplicationContext(), "Escolha uma Imagem para o Post!", Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(getApplicationContext(), "Selecione uma Tag!", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(getApplicationContext(), "Introduza uma Descrição para o Post!", Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(getApplicationContext(), "Introduza um Título para o Post!", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 724: Método "verificarCampos()"

Posteriormente, realizei o método para ir buscar todas as *tags* disponíveis dentro da *Firebase Database* e colocar as mesmas dentro do *spinner*, para que o utilizador escolha a que *Tag* o seu *Post* irá ficar agregado.

```
private void buscarTags(Spinner spinner) {
    dialogCarregamento.show();
    DatabaseReference referenciaTags = DefinicoesFirebase.recuperarBaseDados().child("tags");
    referenciaTags.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            tags.add("Selecionar Tag...");
            for (DataSnapshot nodes : snapshot.getChildren()) {
                tags.add(nodes.getValue(String.class));
            }
            dialogCarregamento.dismiss();
            ArrayAdapter<String> adapter = new ArrayAdapter(getApplicationContext(), android.R.layout.simple_spinner_dropdown_item, tags);
            spinner.setAdapter(adapter);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
        }
    });
}
```

Figura 725: Método "buscarTags()"

Para finalizar esta atividade, criei o método “criarPost()”, responsável pela a criação do Post em si. Como se pode ver na captura de ecrã, o estado do post, é colocado como “pendente”, logo não é aceite de imediato, uma medida bastante importante, impedindo vários possíveis problemas.

```
private void criarPost(String titulo, String descricao, int posicao) {
    DatabaseReference postRef = DefinicoesFirebase.recuperarBaseDados().child("posts");

    Post post = new Post();

    post.setId(postRef.push().getKey());
    post.setCriador(Configs.recuperarIdUtilizador());
    post.setDescricao(Configs.recuperarDescricao());
    post.setTitulo(titulo);
    post.setEstado(Configs.PENDENTE); // alterar
    post.setDescricao(descricao);
    post.setTag(tags.get(posicao));
    post.setUtilizador(Configs.recuperarIdUtilizador());
    post.guardarImagen(dadosImagen, validar -> {
        if (validar) {
            post.guardar(validar2 -> {
                if (validar2) {
                    dialog.dismiss();
                    Toast.makeText(getApplicationContext(), "Post Criado com Sucesso!", Toast.LENGTH_SHORT).show();
                    finish();
                } else {
                    dialog.dismiss();
                    Toast.makeText(getApplicationContext(), "Erro, Tente Novamente Mais Tarde!", Toast.LENGTH_SHORT).show();
                }
            });
        } else {
            Toast.makeText(getApplicationContext(), "Erro, Tente Novamente Mais Tarde!", Toast.LENGTH_SHORT).show();
        }
    });
}
```

Figura 726: Método "criarPost()"

PontuacaoFragment

A *PontuacaoFragment*, ficaria destinada a todos os membros e apenas seriam mostradas as primeiras 10 posições, funcionando como uma “*LeaderBoard*” para haver mais despike entre os utilizadores.



Figura 727: Layout da PontuacaoFragment

Seguidamente, veio a configuração do RecyclerView, responsável por mostrar os utilizadores, configurei também, o clique curto, como aquele que abria a ContaActivity, para mostrar assim os dados da conta em específico.



Figura 728: Configuração do RecyclerView

Seguidamente, configurei o adapter e instanciei o método “buscarUtilizadores()”:

```
    // Configurando Adapter  
    PuntuacionAdapter * new PuntuacionAdapter(ListaContato, getContext());  
    recyclerView.setAdapter(puntuacionAdapter);  
  
    buscarUtilizadores();
```

Figura 729: Configuração do adapter

Seguidamente, codifiquei o método “buscarUtilizadores()”, responsável por ir buscar todos os utilizadores do tipo “membro”, depois dos ir buscar todos, é realizada uma organização tendo em conta o xp total de todos.

Figura 730: Método "buscarUtilizadores()"

Seguidamente, precisei de proceder à criação de um adapter para fazer a listagem dos utilizadores que viriam a aparecer dentro da PontuacaoFragment, para isso criei o PontuacaoAdapter. O layout do mesmo, encontra-se na página seguinte.



Figura 731: Layout do PontuacaoAdapter

Com o layout realizado, codifiquei o método “onBindViewHolder()”, para que os dados fossem corretamente visualizados. Coloquei ainda uma condição, que se o utilizador autenticado no momento, fosse apresentado, teria uma cor diferente dos outros para que fosse distingível.

Como queria mostrar apenas o top 10 dos utilizadores, limitei a visualização do adaptador apenas para 10, com uma variável global (privada).

```

private void onBindViewHolder(@NonNull ViewHolder viewHolder, int position) {
    User listItemUser = listItemUsers.get(position);
    viewHolder.rank.setText(listItemUser.getRank());
    viewHolder.name.setText(listItemUser.getName());
    viewHolder.ac.setScore("Top " + position + "º");
    viewHolder.points.setText("Pontos: " + listItemUser.getPoints());
    viewHolder.rank.setTextColor(position == 0 ? Color.BLUE : Color.GRAY);
}

@Override
public int getItemCount() {
    // Limita o número final em 10000, caso seja exibir mais de 10000 não aparece mais o resultado
    if (listItemUsers.size() > 10000) {
        return 10000;
    } else {
        return listItemUsers.size();
    }
}

```

Figura 732: Método "onBindViewHolder()" e "getItemCount()"

Para organizar melhor os privilégios da app, decidi colocar dentro do ficheiro de configurações, todos os grupos e estados, tal como se observa na figura.

```
public class Definições {  
    private static final String grupo = "memoria", "red", "azul", "verde";  
    private static final String PERMIS = "insert_update";  
    public static int idgroup = 0;  
    public static String PERSISTENTE = "persistente";  
    public static String REVISAO = "revisao";  
    public static String ACESSE = "acessa";  
    public static String CONCLUIR = "concluiu";  
    public static final int DELEZAR_CAMARA = 100;  
    public static final int DELEZAR_MATERIA = 200;
```

Figura 733: Previlépios da app

Para finalizar esta atividade, coloco uma captura de ecrã do meu smartphone perante a sua execução.



Figura 734: Captura de ecrã da PontuacaoFragment

ContaFragment

A ContaFragment, ficaria destinada apenas para mostrar ao utilizador como está o seu perfil na vista dos outros e para o editar.

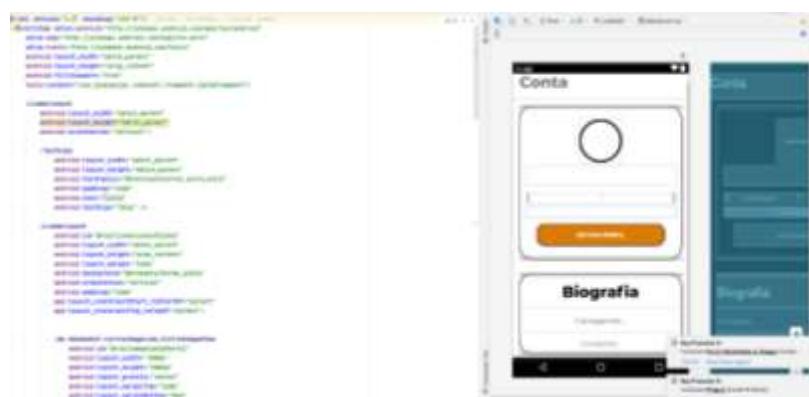


Figura 735: Layout ContaFragment

Depois de ter configurado o layout, procedi às configurações iniciais, que se baseavam na recuperação dos elementos do layout e dos dados do utilizador e também da definição do método “onClick” do botão de editar perfil, como a abertura do EditarPerfilActivity.

```
Configs.recuperarIDUtilizador(idUtilizador --> recuperarDadosUtilizador(idUtilizador))

fundoPerfil = root.findViewById(R.id.linearLayoutConta);
fotoPerfil = root.findViewById(R.id.imageContaPerfil);
nomeConta = root.findViewById(R.id.textContaNome);
biografia = root.findViewById(R.id.textContaBiografia);
biografiasData = root.findViewById(R.id.textContaBiografiaData);
contaxP = root.findViewById(R.id.textContaXP);
estado = root.findViewById(R.id.textContaEstado);
nascimento = root.findViewById(R.id.textContaNascimento);

Button buttonEditarPerfil;
buttonEditarPerfil = root.findViewById(R.id.buttonEditarPerfil);
buttonEditarPerfil.setOnClickListener(v --> abrirEditarPerfil());
```

Figura 736: Configurações Iniciais

Em seguida, realizei o método “recuperarDadosUtilizador()”, que era responsável por ir buscar os dados do utilizador e configura-los da forma correta colocando cada informação no seu devido lugar.

Para finalizar, ainda coloquei a funcionalidade de perceber se o dia atual, era o dia de aniversário do utilizador, sendo esse o caso, é mostrada uma mensagem de parabéns.

São os pequenos detalhes que fazem a diferença.

Figura 737: Método "recuperarDadosUtilizador()"

EditarPerfilActivity

Depois de ter configurado o fragment de visualização da conta, decidi criar esta atividade, para o utilizador conseguir editar alguns aspectos do seu perfil.

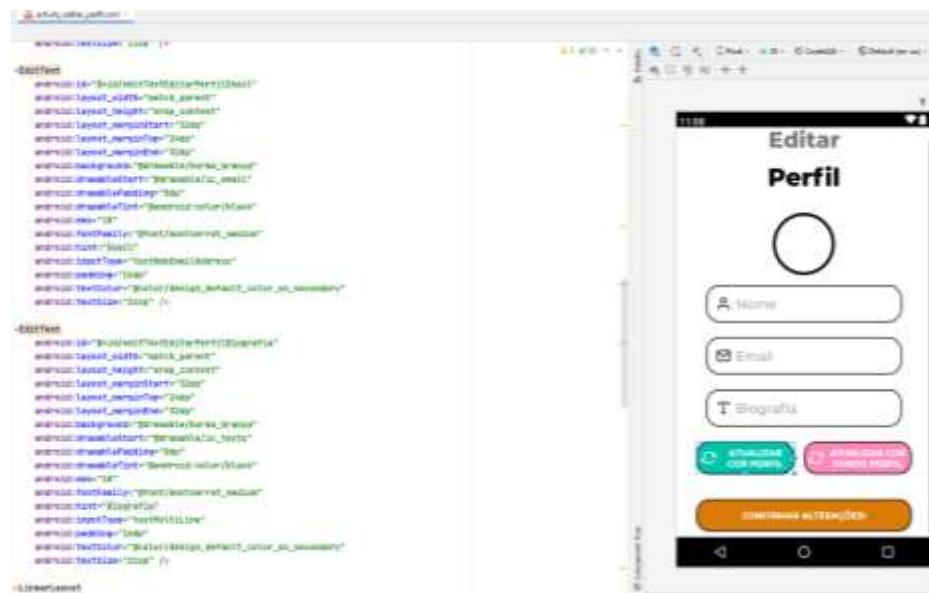


Figura 738: Layout do *EditarPerfilActivity*

Após de ter finalizado o layout, realizei as configurações desta atividade.

```

dialog = new SpotsDialog.Builder().setContext(this).setMessage("Guardando Alterações...").setTheme(R.style.dialog_correngimento).setCancelable(false).build();
dialogCorrengimento = new SpotsDialog.Builder().setContext(this).setMessage("Carregando Dados...").setTheme(R.style.dialog_correngimento).setCancelable(true).build();

linmarLayoutCorrendoPerfil = findViewById(R.id.linmarLayoutCorrendoPerfil);
imagePerfil = findViewById(R.id.imageCorrendoPerfil);
editTextNome = findViewById(R.id.editTextEditorPerfilNome);
editTextBio = findViewById(R.id.editTextEditorPerfilBio);
editTextBio.setOnClickListener(this);

recuperardados();
    
```

Figura 739: Configurações Iniciais

Como já feito anteriormente, este método, é responsável por ir buscar todos os dados do utilizador e colocar os mesmos no seu devido lugar.

```

private void recuperarDados() {
    DatabaseReference perfilRef = DefinicoesFirebase.recuperarDados().child("contas").child(idUtilizador);
    perfilRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            DialogCarregamento.show();
            Conta conta = dataSnapshot.getValue(Conta.class);

            Drawable drawable = ContextCompat.getDrawable(getApplicationContext(), R.drawable.borda_preta);
            drawble.setDither(true);
            drawble.setCornerRadius(10);
            drawble.setRadius(10);

            LinearLayoutCorFundo.setBackground(drawable);
            Glide.with(getApplicationContext()).load(conta.getFoto()).into(imagemPerfil);
            editNome.setText(conta.getNomeUtilizador());
            editEmailLogradouro.setText(conta.getEmail());
            editTextEmail.setText(conta.getEmail());
            editTextEmail.setSingleLine(false);
            DialogCarregamento.dismiss();
        }

        @Override
        public void onCancelled(DatabaseError error) {
        }
    });
}
    
```

Figura 740: Método "recuperarDados()"

O método “atualizarCorPerfil()”, fica responsável por gerar uma nova cor aleatória e trocar a cor de fundo do perfil, enviando essa cor para a Firebase Database e depois a foto seria atualizada consoante a cor que lá permanecesse, com o método “guardarFoto()” já anteriormente desenvolvido.

```

public void atualizarCorPerfil(View view) {
    dialog.show();
    DatabaseReference utilizadorRef = DefinicoesFirebase.recuperarDados().child("contas").child(idUtilizador);
    Random random = new Random();
    int corAleatoria = Config.corAleatoria();
    corPerfil.putInt("corPerfil", corAleatoria);
    utilizadorRef.updateChildren(corPerfil).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            guardarFoto(corAleatoria, validar -> {
                if (validar) {
                    dialog.dismiss();
                    Toast.makeText(EditorPerfilActivity.this, "Cor Atualizada com sucesso!", Toast.LENGTH_SHORT).show();
                } else {
                    dialog.dismiss();
                    Toast.makeText(EditorPerfilActivity.this, "Erro, Teste Novamente Mais Tarde!!", Toast.LENGTH_SHORT).show();
                }
            });
        } else {
            dialog.dismiss();
            Toast.makeText(getApplicationContext(), "Erro, Teste Novamente Mais Tarde!!", Toast.LENGTH_SHORT).show();
        }
    });
}
    
```

Figura 741: Método "atualizarCorPerfil()"

Depois, decidi codificar o botão de atualizar a cor fundo de perfil, que era bastante semelhante à anterior. Criei um HashMap e enviei o mesmo para dentro do nó do utilizador, como se verifica no excerto de código.

```
public void atualizarCorFundoPerfil(View view) { // mudar a cor do botão para a cor gerada
    dialog.show();
    DatabaseReference utilizadorRef = DefinicaoFirebase.recuperarBaseDados().child("contas").child(idUtilizador);
    HashMap<String, Object> corPerfil = new HashMap<>();
    corPerfil.put("corFundoPerfil", Configs.corAleatoria());
    utilizadorRef.updateChildren(corPerfil).addOnCompleteListener(task ->
    {
        if (task.isSuccessful())
            Toast.makeText(getApplicationContext(), "Cor do Fundo de Perfil atualizada com sucesso!", Toast.LENGTH_SHORT).show();
        else
            Toast.makeText(getApplicationContext(), "Erro, tenta novamente mais tarde!", Toast.LENGTH_SHORT).show();
        dialog.dismiss();
    });
}
```

Figura 742: Método "atualizarCorFundoPerfil()"

No botão de confirmar as alterações, realizei as verificações dos campos e prossegui à atualização dos dados com o método “atualizarDadosPerfil()”..

```
public void confirmarAlteracoesPerfil(View view) {
    String nome = editTextNome.getText().toString();
    String biografia = editTextBiografia.getText().toString();

    if (!nome.isEmpty()) {
        if (!biografia.isEmpty()) {
            dialog.show();
            atualizarDadosPerfil();
        } else {
            dialog.dismiss();
            Toast.makeText(context: this, text: "Introduza algo na Biografia!", Toast.LENGTH_SHORT).show();
        }
    } else {
        dialog.dismiss();
        Toast.makeText(context: this, text: "Introduza um Nome!", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 743: Método "confirmarAlteracoesPerfil()"

Por fim, o método “atualizarDadosPerfil()” é responsável por atualizar tudo em geral, tal como se pode observar na figura abaixo.

```

programa main {
    databaseReference currentUser = inicializarResumo.usuarioResumo();
    currentUser.child("dadosPerfil").setValue(null);
}

String nome, biografia;
nome = editTextNome.getText().toString();
biografia = editTextBiografia.getText().toString();

if (nome.isEmpty() || nome.length() < 3) {
    if (biografia.isEmpty()) {
        toast("Insira pelo menos 3 caracteres no nome!");
    } else {
        toast("Insira pelo menos 3 caracteres na biografia!");
    }
} else {
    if (biografia.length() < 3) {
        toast("Insira pelo menos 3 caracteres na biografia!");
    } else {
        nome = nome.replace(" ", "%20");
        dadosPerfil.child("name", nome);
        dadosPerfil.child("biografia", biografia);
        dadosPerfil.child("data", System.currentTimeMillis());
        currentUser.updateChildren(dadosPerfil).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                guardarNome();
                validar2 = false;
                if (validar2) {
                    editarNome();
                    Toast.makeText(getApplicationContext(), "Dados salvos com sucesso!", Toast.LENGTH_SHORT).show();
                    finish();
                } else {
                    Toast.makeText(getApplicationContext(), "Erro, não pode haver espaços na biografia!", Toast.LENGTH_SHORT).show();
                    finish();
                }
            } else {
                Toast.makeText(getApplicationContext(), "Erro, Tente novamente mais tarde!", Toast.LENGTH_SHORT).show();
            }
        });
    }
}

```

Figura 744: Método "atualizarDadosPerfil()"

ContaActivity

Esta atividade, foi planeada para servir apenas de exibição de dados de um utilizador em específico, visto que esta pode ser começada em vários locais, necessitando apenas do id do utilizador que é para visualizar.

4



Figura 745: Layout da ContaActivity

Seguidamente, procedi às configurações básicas ao nível do layout e realizei o método “buscarConta()” responsável por buscar todos os dados da conta a que esta atividade foi “mandada” executar.



Figura 746: Método "buscarConta()"

Para finalizar esta atividade, coloco um print da mesma em execução.



Figura 747: Execução da ContaActivity (exemplo)

OfertasFragment

O OfertasFragment destinei-o, para ser uma listagem de todas as ofertas que o utilizador tem no momento.



Figura 748: Layout (OfertasFragment)

Seguidamente, realizei as configurações necessárias para com o RecyclerView

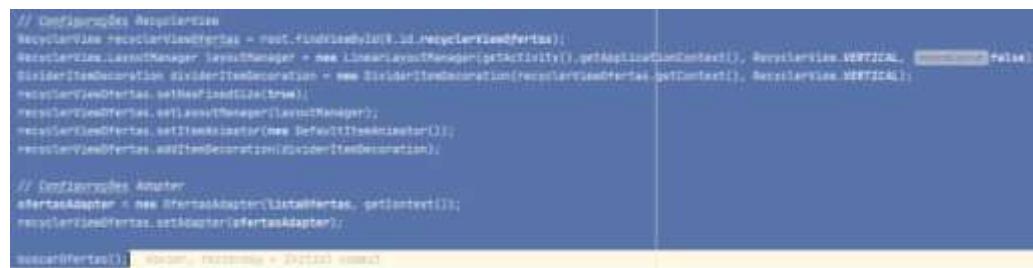


Figura 749: Configurações Básicas do RecyclerView

Criei também o método “`buscarOfertas()`”, para que sejam pesquisadas as ofertas, com a condição de pertencerem ao utilizador autenticado no momento e nas quais o seu estado seja “pendente”.



Figura 750: Método "buscarOfertas()"

Com a criação deste método, a codificação da atividade estava terminada, prossegui para a criação do *adapter*.

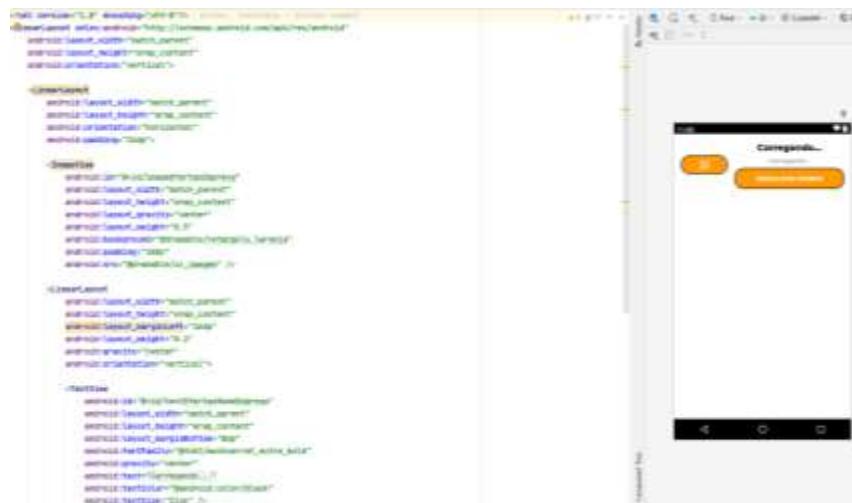


Figura 751: Layout (OfertasAdapter)

Com o layout do OfertasAdapter devidamente configurado, apenas tive de codificar o método “*onBindViewHolder()*”, para que este direcionasse os dados para cada elemento do layout.

Coloquei dois ouvintes “*onClick*”, um no *ImageView* da empresa, para que o utilizador aceda ao perfil da empresa, com este código a EmpresaVisualizacaoActivity é aberta, sendo passada o id da empresa e o outro coloquei no botão de visualizar oferta, que irá iniciar a OfertaActivity responsável também pela visualização da oferta e se a mesma será aceite ou recusada, consoante a escolha do utilizador.



Figura 752: Método "onBindViewHolder()"

DefinicoesFragment

Este *fragment*, tem como função de alterar as definições mais privadas de cada utilizador, um à parte é que apenas é visualizável o botão “Modificar Inscrições” no caso do utilizador ser “membro”.



Figura 753: Layout (DefinicoesFragment)

Seguidamente, realizei as configurações iniciais deste *fragment*, começando pelos elementos básicos e a recuperação do grupo, para que, tal como dito anteriormente, o botão “Modificar Inscrições” seja visível ou não.



Figura 754: Configurações Iniciais

No método “*onClick*” do botão de alterar password, usei um *AlertDialog*, para facilitar o meu trabalho e coloquei um *EditText*, para o utilizador escrever a sua password, depois realizei a verificação de perceber se o campo havia sido preenchido e usei o método “*updatePassword()*” da *Firebase*, para atualizar a password, tendo em conta que também tratei dos erros, como se pode ver no código.

```
AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
builder.setTitle("Alterar Password");
builder.setMessage("Para alterar a sua password, digite uma nova no campo abaixo!");
builder.setView(editText);
builder.setPositiveButton(R.string.confirmar, (dialog, which) -> {
    String password = editText.getText().toString();
    if (password.isEmpty() || password.equals("")) {
        dialog.dismiss();
        return;
    }
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    user.updatePassword(password).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            dialog.dismiss();
            Toast.makeText(getContext(), R.string.password_update_sucedida, Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(getContext(), MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(intent);
        } else {
            String erro;
            try {
                throw task.getException();
            } catch (FirebaseAuthUserNotFoundException e) {
                erro = "A Password introduzida é fraca!";
            } catch (FirebaseAuthRecentLoginRequiredException e) {
                erro = "Credenciais Inválidas!";
            } catch (Exception e) {
                erro = "Sorry, Tente Novamente Mais tarde!";
                e.printStackTrace();
            }
            dialog.dismiss();
            Toast.makeText(getContext(), erro, Toast.LENGTH_SHORT).show();
        }
    });
});
else {
    Toast.makeText(getContext(), R.string.introduza_a_password_desejada_anter_de_confirmar, Toast.LENGTH_SHORT).show();
}
```

Figura 755: Método “*onClick*” do botão de atualizar password

Para finalizar este *fragment*, configurei também o método “*onClick*” do botão de remover conta. Mais uma vez, utilizei o *AlertDialog*, porque poupa muito o código e é muito mais rápido de realizar verificações.

Para remover a conta o procedimento é um pouco diferente do alterar password, visto que irá eliminar todos os registos da conta dentro da base de dados (*Firebase*).

No *AlertDialog*, coloquei um *EditText*, para confirmar a password, como é óbvio. Depois, realizei a verificação desse mesmo campo e se estiver preenchido continua com a execução do código.

Depois vai autenticar com a password fornecida, se for a correta, começa por eliminar os dados da *Firebase Database*, de seguida elimina a foto dentro da *Firebase Storage* e por fim sim elimina o utilizador da *Firebase Authentication* e quando isso tudo estiver feito e bem executado, o utilizador é “deslogado” e informado da situação, é levado de volta a *IntroActivity*.

```

        builder.setTitle("Remover conta");
        builder.setMessage("Tem 100% de certeza que pretende remover a sua conta da ContaUT?");
        builder.setNegativeButton("Não", dialog, null) -> {
            String password = editText.getText().toString();
            if (password.isEmpty()) {
                dialog.cancel();
            }
            FirebaseAuth auth = FirebaseAuth.getInstance();
            FirebaseUser user = auth.getCurrentUser();
            DatabaseReference ref = FirebaseDatabase.getInstance().getReference("users");
            ref.child(user.getUid()).removeValue();
            if (user.isEmailVerified()) {
                auth.signOut();
                Intent intent = new Intent(getApplicationContext(), IntroActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            } else {
                dialog.dismiss();
                String error;
                try {
                    throw new FirebaseAuthException();
                } catch (FirebaseAuthException e) {
                    error = "Credenciais Inválidas!";
                } catch (Exception e) {
                    error = "Oops, Tente Novamente mais Tarde!";
                }
            }
        };
        builder.setPositiveButton("Sim", dialog, (dialog1, which) -> {
            String password = editText.getText().toString();
            if (password.isEmpty()) {
                dialog.cancel();
            }
            FirebaseAuth auth = FirebaseAuth.getInstance();
            FirebaseUser user = auth.getCurrentUser();
            DatabaseReference ref = FirebaseDatabase.getInstance().getReference("users");
            ref.child(user.getUid()).removeValue();
            if (user.isEmailVerified()) {
                auth.signOut();
                Intent intent = new Intent(getApplicationContext(), IntroActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            } else {
                dialog.dismiss();
                String error;
                try {
                    throw new FirebaseAuthException();
                } catch (FirebaseAuthException e) {
                    error = "Credenciais Inválidas!";
                } catch (Exception e) {
                    error = "Oops, Tente Novamente mais Tarde!";
                }
            }
        });
    }
}

```

Figura 756: Método “onClick” do botão de remover conta

SobreFragment

Este *SobreFragment*, foi pensado mesmo na última fase, para de certa forma, fazer uma “publicidade” para mim, indicando o desenvolvedor desta app espetacular.



Figura 757: Layout (SobreFragment)

Seguidamente, passei à configuração das ações dos botões colocando o link de cada rede social minha no botão indicado, como se pode visualizar. Por fim, ainda, configurei o botão de enviar email, para que o utilizador se pretender envie diretamente um email para mim.

```

ImageButton facebook, instagram, github, linkedin;
facebook = root.findViewById(R.id.buttonFacebook);
instagram = root.findViewById(R.id.buttonInstagram);
github = root.findViewById(R.id.buttonGitHub);
linkedin = root.findViewById(R.id.buttonLinkedIn);

Button enviarEmail;
enviarEmail = root.findViewById(R.id.buttonEnviarEmail);

facebook.setOnClickListener(v -> startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.facebook.com/saonaviera54"))));
instagram.setOnClickListener(v -> startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.instagram.com/saonaviera54"))));
github.setOnClickListener(v -> startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("https://github.com/saonaviera54"))));
linkedin.setOnClickListener(v -> startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.linkedin.com/in/saonaviera54"))));
enviarEmail.setOnClickListener(v -> Config_recuperarUtilizador.nomeUtilizador -> {
    Intent intent = new Intent(Intent.ACTION_SENDTO);
    intent.setType("message/rfc822");
    intent.putExtra("to", "saonaviera54");
    intent.putExtra(Intent.EXTRA_EMAIL, new String[]{Config_recuperarUtilizador.emailGmail});
    intent.putExtra(Intent.EXTRA_SUBJECT, "Mensagem - Code4All");
    intent.putExtra(Intent.EXTRA_TEXT, "Olá,\n\nAgradeço a sua ajuda e parceria.\n\nAtenciosamente,\n" + nomeUtilizador);
    startActivity(intent);
});
    
```

Figura 758: Configurações Iniciais e Definição das ações dos botões

AdminActivity

A AdminActivity é basicamente uma cópia da PrincipalActivity, só que serve apenas para os utilizadores em que o grupo é “admin”, tendo mais privilégios que as restantes pessoas, nas próximas prints estão as configurações que tive de realizar.



Figura 759: Configuração da AppBarConfiguration, com os ids dos fragments utilizados nesta atividade



Figura 760: Configuração do ficheiro "mobile_navigation_admin.xml"

Figura 761: Configuração do ficheiro "menu_admin.xml"

InicioAdminFragment

Este *fragment*, é aquele que é apresentado sempre que um utilizador do tipo “admin” é logado.



Figura 762: Layout InicioAdminFragment

Após ter configurado o layout, realizei as configurações básicas para depois manipular os TextView's com os dados corretos adquiridos da base de dados.

```
textPostsFazer = root.findViewById(R.id.textCountAdminPostsFazer);

textMembros = root.findViewById(R.id.textCountAdminMembros);
textEmpresas = root.findViewById(R.id.textCountAdminEmpresas);
textModos = root.findViewById(R.id.textCountAdminModos);
textAdmins = root.findViewById(R.id.textCountAdminAdmins);
textQuizes = root.findViewById(R.id.textCountAdminQuizes);
textPosts = root.findViewById(R.id.textCountAdminPosts);
textOfertas = root.findViewById(R.id.textCountAdminOfertas);

buscarInfo();
```

Figura 763: Configurações Iniciais

Estes dados são recebidos através do método “*buscarInfo()*”, e depois foi só contabilizar os dados pretendidos com as devidas condições, tal como se observa nas próximas figuras.

Figura 764: Método "buscarInfo()"

```
for (DataSnapshot todos : snapshot.getChildren()) {
    if (todos.getValue(Post.class).getEstadoId().equals(Configs.ABRETO)) {
        abertos++;
    } else if (todos.getValue(Post.class).getEstadoId().equals(Configs.PENDENTE)) {
        pendentes++;
    }
}

textPosts.setText(String.valueOf(abertos));
textPostsPrazo.setText(String.valueOf(pendentes));

referenciaReferal.child("abertos").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull @NotNull DatabaseReference snapshot) {

        int abertos = (int) snapshot.getChildrenCount();
        textoAbertos.setText(String.valueOf(abertos));
    }

    @Override
    public void onCancelled(@NonNull @NotNull DatabaseError error) {
    }
});
```

Figura 765: Método "buscarInfo" 2

ContasFragment

Seguidamente, criei um fragment que iria listar todas as contas que têm como tipo “membro” e “mod”, para alterar o seu privilégio, visto que o utilizador é admin, ele tem a permissão de o realizar.



Figura 766: Layout (ContasFragment)

Sem demora, configurei o RecyclerView responsável por mostrar as contas dos utilizadores, visto que essa era a função principal.



Figura 767: Configurações iniciais e do RecyclerView

Sem delonga, codifiquei o método “cliqueCurto()” que edita os privilégios do utilizador, se o utilizador (admin) alterasse alguma permissão, a mesma era de imediato gravada dentro da *Firebase Database*, como se pode observar no excerto de código.

```

private void cliqueCurto(ListView listaCategorias) {
    // Clique sobre
    // Fazer scroll com o tipo de permissões, permitindo a edição
    // de Permissões tipo diretamente no utilizador, sem a sua necessidade
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    LayoutInflater inflater = getLayoutInflater();
    View dialogView = LayoutInflater.inflate(R.layout.preencher_tipos_layout, null);
    builder.setView(dialogView);

    spinnerTiposCategorias = dialogView.findViewById(R.id.spinnerTiposCategorias);
    preencherTipos(listaCategorias.get(position)); // Preencher tipos disponíveis à lista de utilizador ligado (usuário)

    builder.setTitle("editar Privilégios do Utilizador");
    builder.setNegativeButton("esta atualmente a editar o utilizador " + listaCategorias.get(position).getNome() + " " + listaCategorias.get(position).getSobrenome(), null);
    builder.setPositiveButton("Confirmar", (dialog, which) -> {
        String[] tipos = spinnerTiposCategorias.getSelectedItem().toString().split(",");
        HashMap<String, Object> listaCategorias = new HashMap<>();
        int posicao = spinnerTiposCategorias.getSelectedItemPosition();

        listaCategorias.setEditable(true);
        listaCategorias.setEditable(true, posicao);

        DatabaseReference ref = FirebaseDatabase.getInstance().getReference().child("users").child("name").child(listaCategorias.get(position).getNome());
        ref.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    Toast.makeText(getContext(), "Tipo de Conta atualizado com sucesso!", Toast.LENGTH_SHORT).show();
                    dialog.dismiss();
                } else {
                    Toast.makeText(getContext(), "Erro! Temos de encontrar esta conta!", Toast.LENGTH_SHORT).show();
                    dialog.dismiss();
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                dialog.dismiss();
            }
        });
        builder.setNegativeButton("Cancelar", (dialog, which) -> dialog.dismiss());
        spinnerTiposCategorias.setSelection(0);
        builder.show();
    });
}

```

Figura 768: Método "cliqueCurto()

Depois precisei do método “preencherTipos()”, visto que era ele que me iria dar a lista dos tipos do utilizador, para controlar os privilégios decidi realizar um “switch case”.

Primeiramente, comecei por recuperar o grupo do utilizador autenticado. Se fosse “*admin*”, conseguiria alterar os privilégios até ao nível, se fosse “*mod*” a mesma coisa aconteceria, tal como se observa no código da próxima página.

```

private void premiosTipos(String tipoUtilizadorSeleccionado) {
    tipos.clear();
    Configs.recuperarGrupos(grupo -> {
        switch (grupo) {
            case "miembros":
                tipos.add(tipoUtilizadorSeleccionado);
                if (!tipoUtilizadorSeleccionado.equals("miembro"))
                    tipos.add("miembro");
                if (!tipoUtilizadorSeleccionado.equals("med"))
                    tipos.add("med");
                break;
            case "admin":
                tipos.add(tipoUtilizadorSeleccionado);
                if (!tipoUtilizadorSeleccionado.equals("miembro"))
                    tipos.add("miembro");
                if (!tipoUtilizadorSeleccionado.equals("med"))
                    tipos.add("med");
                if (!tipoUtilizadorSeleccionado.equals("admin"))
                    tipos.add("admin");
                break;
        }
        spinnerAdapter = new ArrayAdapter(getContext(), android.R.layout.simple_spinner_item, tipos);
        spinnerTiposContato.setAdapter(spinnerAdapter);
    });
}

```

Figura 769: Método "preencherTipos()"

O último método, mas não o menos importante, é responsável por ir buscar todas as contas organizadas pelo o seu tipo. Se o grupo do utilizador autenticado fosse “mod” este só teria acesso, aos utilizadores “membro” e “mod”, se ele fosse “admin”, teria acesso aos mesmos utilizadores, incluído o próprio “admin”.

Figura 770: Método "buscarContas()"

Com a atividade devidamente orientada, passei à configuração do *adapter*, começando então pelo o seu layout.

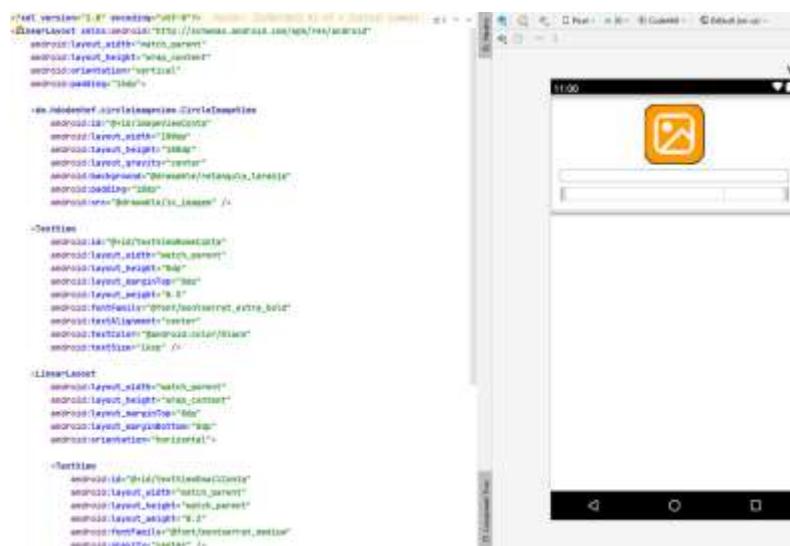


Figura 771: Layout (ContasAdminAdapter)

Para finalizar este *fragment*, configurei todos os elementos anteriormente colocados no layout, para que estes recebem corretamente os dados.

```

@Override
public void onBindViewHolder(@NonNull @NotNull MyViewHolder holder, int position) {
    holder.setIsRecyclable(false);
    Conta conta = listaContas.get(holder.getAdapterPosition());

    Glide.with(context).load(conta.getFoto()).into(holder.imagem);
    holder.textNome.setText(listaContas.get(position).getNome());
    holder.textTipo.setText(listaContas.get(position).getTipo());
    holder.textEmail.setText(listaContas.get(position).getEmail());
}

```

Figura 772: Método "onBindViewHolder()"

PostsVerificacaoFragment

Neste fragment, aparecem todos os posts que necessitam de ser verificados, ou seja, todos os posts realizados por membros.



Figura 773: Layout (PostsVerificacaoFragment)

Depois de ter configurado o layout, passei à configuração do *RecyclerView* que iria conter todos os *posts* em que o estado fosse igual a “pendente”, como já disse anteriormente.



Figura 774: Configurações Iniciais e o método "buscarPosts()"

Seguidamente, após de ter realizado as alterações que necessitava dentro da atividade, passei à criação de mais um *adapter*, depois de o criar desenvolvi o layout, tal como se apresenta na figura.



Figura 775: Layout (PostsVerificacaoAdapter)

Para finalizar este fragment, só tive de associar os dados recuperados aos elementos certos, como já feito várias vezes.

```
@Override  
public void onBindViewHolder(@NotNull @NotNull MyViewHolder holder, int position) {  
  
    Post post = listaPosts.get(position);  
  
    Glide.with(context).load(post.getImagen()).into(holder.imagem);  
    holder.criador.setText(post.getCriador());  
    holder.titulo.setText(post.getTitulo());  
    holder.descricao.setText(post.getDescricao());  
    holder.tag.setText(post.getTag());  
    holder.botaoAceitar.setOnClickListener(v -> atualizarEstado(listaPosts.get(position).getId(), Configs.ACEITE));  
    holder.botaoRecusar.setOnClickListener(v -> atualizarEstado(listaPosts.get(position).getId(), Configs.RECUSADO));  
}
```

Figura 776: Método "onBindViewHolder()"

CriarQuizFragment

Este CriarQuizFragment, é um dos mais importantes, visto que é nele que serão criados todos os quizzes da Code4All. Posso ainda adiantar, que foi este o fragment que me deu mais trabalho a desenvolver, visto que só ele tem ~600 linhas de código, que serão aqui explicadas muito mais resumidamente, como é óbvio.

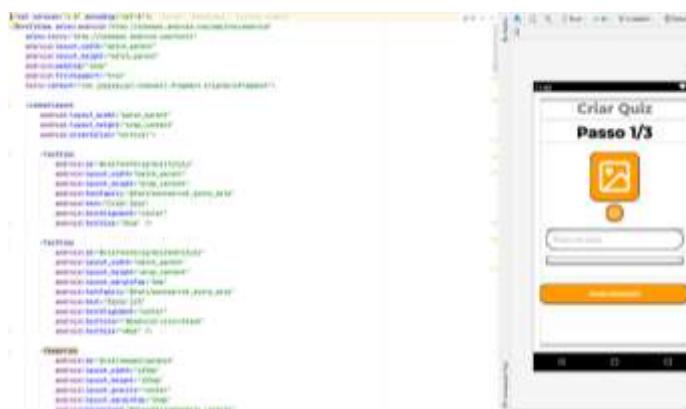


Figura 777: Layout (CriarQuizFragment)

Seguidamente, configurei todos os parâmetros que viria a necessitar durante o desenvolvimento desta parte da app.

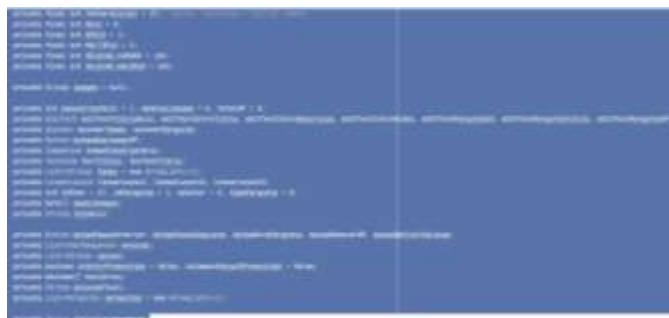


Figura 778: Parâmetros Iniciais

Com os parâmetros todos mais ou menos orientados, fui configurar os elementos todos, um a um, para que tivesse o total controlo sobre o layout apresentado ao utilizador.

```

        id="etpcLogo" x="160" y="100" width="100" height="100" style="background-color: #0070C0; color: white; text-align: center; padding: 10px;">
            ETPC
        
```

```

        id="titleText" x="300" y="150" width="300" height="50" style="font-size: 24px; font-weight: bold; border: 1px solid black; padding: 5px; margin-bottom: 10px;">
            Quiz
        
```

```

        id="subTitleText" x="300" y="180" width="300" height="50" style="font-size: 16px; font-weight: bold; border: 1px solid black; padding: 5px; margin-bottom: 10px;">
            Desenvolvido por
        
```

```

        id="text1" x="300" y="210" width="300" height="50" style="font-size: 14px; border: 1px solid black; padding: 5px; margin-bottom: 10px;">
            ETPC
        
```

```

        id="text2" x="300" y="240" width="300" height="50" style="font-size: 14px; border: 1px solid black; padding: 5px; margin-bottom: 10px;">
            Fundação para a Ciência e a Tecnologia
        
```

```

        id="text3" x="300" y="270" width="300" height="50" style="font-size: 14px; border: 1px solid black; padding: 5px; margin-bottom: 10px;">
            Programa POC do FSE
        
```

```

        id="text4" x="300" y="300" width="300" height="50" style="font-size: 14px; border: 1px solid black; padding: 5px; margin-bottom: 10px;">
            Projeto Portugal 2020
        
```

```

        id="text5" x="300" y="330" width="300" height="50" style="font-size: 14px; border: 1px solid black; padding: 5px; margin-bottom: 10px;">
            União Europeia
        
```

Figura 779: Configurações Iniciais dos elementos do Layout

Seguidamente, chamei o método “buscarTemas()”, responsável por ir buscar todos os temas em que um quiz se pode identificar.

Também configurei um ouvinte “onClick”, no botão de avançar, para que o layout se adaptasse consoante o passo em que o utilizador está, dividi este por 3 passos, como se pode observar.

No 1º Passo, apenas é pedido o título do quiz e o tema do mesmo. No 2º passo é realizada a introdução do quiz, os dados que estarão apresentáveis dentro da QuizInfoActivity. No 3º Passo e último, é onde são gravadas todas as perguntas adicionadas pelo o utilizador e o quiz é enviado para a Firebase Database, claro que por este meio todo, todos os campos são verificados para que não haja erros de “NullPointerException”, suficientes para causar um Crash completo na app.

Nas capturas de ecrã abaixo, encontra-se o código realizado.

Figura 780: Método "onClick" do botão passo seguinte

Figura 781: Método "onClick" do botão passo seguinte 2

Seguidamente, configurei tanto o método “guardarFoto()”, responsável por guardar a foto do quiz e o método “guardarPergunta()” é executado cada vez que o utilizador clica no botão “Nova Pergunta”

Figura 782: Método "guardarFoto()" e "guardarPergunta"

Mais uma vez, utilizei o método “onActivityResult()” para que o utilizador escolha uma imagem e essa seja atribuída para uma variável posteriormente enviada com o método “guardarFoto()”, anteriormente visto.

Figura 783: Método "onActivityResult()"

Continuando, criei também o método “limparCampos()”, que limparia todos os campos usados na criação de uma pergunta, tal como se observa na figura abaixo.

O método “novaPergunta()”, executado como clique do botão nova pergunta, executa primeiramente uma verificação de todos os campos e se esses estão preenchidos, se estiverem, a pergunta é guardada, os campos são limpos, o id da pergunta é incrementado e por fim, o EditText que indica o número da pergunta é atualizado.

```
private void limparCampos(View root) {
    EditText primeiraOpcao = root.findViewById(idCharInicial);
    editTextPerguntaTitulo.setText("");
    editTextPerguntaXP.setText("");
    spinnerPergunta.setSelection(0);
    opSelecionada = 0;
    for (int i = idCharInicial; i <= idChar; i++) {
        removerOpcao(root);
    }
    primeiraOpcao.setText("");
    solucoes.clear();
    isVetorPreenchido = false;
}

private void novaPergunta(View root) {
    verificarCamposPasso3(root);
    if (isCamposPasso3Preenchidos) {
        guardarPergunta();
        limparCampos(root);
        idPergunta++;
        editTextPerguntaID.setText("ID da Pergunta : " + idPergunta);
    }
}
```

Figura 784: Método "limparCampos()" e "novaPergunta()"

É no método “verificarCamposPasso3()” que todos os campos da pergunta são verificados e se estão preenchidos ou não, tal como se observa na figura da próxima página.

Figura 785: Método "verificarCamposPasso3()"

Logo de seguida, codifiquei o método “abrirDialog()”, responsável pelo clique no botão “Definir Solução”, é nele que a solução da pergunta a vir a ser criada é definida, sendo ela de escolha única ou múltipla, como se observa nas próximas figuras.

Figura 786: Método "abrirDialog()"

```

        dialog.dismiss();
    });
    builder.setPositiveButton(null, "Confirmar Seleção", (dialog, which) -> {
        definirSolucaoPergunta();
        dialog.dismiss();
    });
}
if (tipoPergunta != NULO) {
    AlertDialog dialog = builder.create();
    dialog.show();
} else {
    Toast.makeText(getApplicationContext(), "É preciso definir o tipo de seleção!", Toast.LENGTH_SHORT).show();
}

```

Figura 787: Método "abrirDialog()" 2

De seguida, tratei do método “definirSolucaoPergunta()”, que como o próprio nome diz, iria definir a solução da pergunta e guardar numa variável global. O método “passoAnterior()”, serve mais para se o utilizador (admin) se enganar em alguma configuração do quiz, consiga voltar para trás sem problemas, efetuando as devidas configurações e o preparando o layout para tal mudança.

```

private void definirRelacoesPergunta() {
    if (selecao.size() == 2) {
        seletacoFinal = selecao.get(0).toString();
    } else if (selecao.size() == 1) {
        for (int i = 0; i < selecao.size(); i++) {
            seletacoFinal += selecao.get(i).toString();
        }
    } else {
        Toast.makeText(getApplicationContext(), "É preciso definir a seleção da Pergunta!", Toast.LENGTH_SHORT).show();
        return;
    }
    Toast.makeText(getApplicationContext(), "Relação Atual: " + seletacoFinal, Toast.LENGTH_SHORT).show();
}

private void possoAnterior() {
    if (passoCriarQuiz == 2) {
        textoTítulo.setText("Criar Quiz");
        textoSubTítulo.setText("Passo 2/2");
        passoCriarQuiz = 1;
        linearLayout1.setVisibility(View.GONE);
        linearLayout.setVisibility(View.VISIBLE);
        botaoPossoAnterior.setVisibility(View.GONE);
        linearLayout2.setVisibility(View.GONE);
        botaoDefinirSeloco.setVisibility(View.GONE);
    } else if (passoCriarQuiz == 1) {
        textoTítulo.setText("Entregue!");
        textoSubTítulo.setText("Passo 3/2");
        passoCriarQuiz = 2;
        linearLayout1.setVisibility(View.GONE);
        linearLayout2.setVisibility(View.VISIBLE);
        linearLayout.setVisibility(View.GONE);
        botaoPossoAnterior.setVisibility(View.GONE);
        botaoDefinirSeloco.setVisibility(View.GONE);
        botaoAdicionarOP.setVisibility(View.GONE);
        botaoNovoPergunta.setVisibility(View.GONE);
        botaoNovoSeguinte.setVisibility(View.GONE);
        botaoPossoSeguinte.setText("Passo Seguinte");
    }
}

```

Figura 788: Método "definirSolucaoPergunta()" e "passoAnterior()"

No método “configuracaoSpinnerPergunta()” realizei a configuração da “*combo box*” e dos seus elementos para que o utilizador escolhesse que tipo de solução iria aplicar dentro da pergunta.

```

private void configuraSpinnerPergunta() {
    List<String> opcionesPergunta = new ArrayList<>();
    opcionesPergunta.add("Introducir respuesta");
    opcionesPergunta.add("Borrar");
    opcionesPergunta.add("Nuevo");
    opcionesPergunta.add("Modificar");

    ArrayAdapter<String> adapter = new ArrayAdapter(getContext(), android.R.layout.simple_spinner_dropdown_item, opcionesPergunta);
    spinnerPergunta.setAdapter(adapter);
    if (tipoPregunta != null) {
        spinnerPergunta.setSelection(tipoPregunta);
    }

    spinnerPergunta.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            tipoPregunta = position;
        }
    });

    @Override
    public void onNothingSelected(AdapterView<View> parent) {
    }
}

```

Figura 789: Método "configuracaoSpinnerPergunta()"

No botão de “Adicionar Pergunta”; cliquei como execução do seu clique, o método “adicionarEditText()”, para que quando pressionado, seja adicionado um novo EditText para que o utilizador consiga escrever uma nova opção na sua pergunta.

Quero ainda referir que só é possível realizar até 6 opções por pergunta.

Figura 790: Método "adicionarEditText()"

Como tenho um método “adicionarEditText()” também precisa de um “removerOpcao()”, responsável por remover as opções que o utilizador pretende-se começando da última até chegar na primeira, impedindo o mesmo de a apagar.

```
private void removerOpcao(View root) {
    if (idChar == 66) { // PRIMEIRA OP // BUG
        Toast.makeText(getApplicationContext(), text: "Operação Inválida!", Toast.LENGTH_SHORT).show();
    } else if // SE NÃO PODE APAGAR :(
    {
        isVetorPreenchido = false;
        if (!solucaoFinal.isEmpty() || !solucaoFinal.equals("")) {
            solucao.clear();
            solucaoFinal = "";
        }
        idChar--;
        ViewGroup layout = (ViewGroup) root.findViewById(idChar).getParent();
        if (layout != null) {
            layout.removeView(root.findViewById(idChar));
        }
    }
}
```

Figura 791: Método "removerOpcao()"

Por fim e finalmente, tratei do método “buscarTemas()”, responsável por ir buscar os temas presentes no início da configuração do quiz

```
private void buscarTemas() {
    DatabaseReference referenciaTemas = DefinicoesFirebase.recuperarBaseDados().child("temas");
    referenciaTemas.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            temas.add("Selecionar Tema");
            for (DataSnapshot dados : snapshot.getChildren()) {
                temas.add(dados.getValue(String.class));
            }
            ArrayAdapter<String> adapter = new ArrayAdapter(getContext(), android.R.layout.simple_spinner_dropdown_item, temas);
            spinnerTemas.setAdapter(adapter);
        }

        @Override
        public void onCancelled(@NotNull DatabaseError error) {
        }
    });
}
```

Figura 792: Método "buscarTemas()"

ListarQuizFragment

Na *ListaQuizFragment*, comecei por planejar o layout e desenvolver o mesmo ficando da seguinte forma.



Figura 793: Layout (*ListarQuizFragment*)

Seguidamente, fui à procura de uma librarria que me oferecesse um ouvinte de cliques para o RecyclerView, visto que ele não tem uma por definição, de imediato encontrei uma no GitHub e importei para dentro do projeto.

<https://github.com/jamiltondamasceno/RecyclerItemClickListener/>

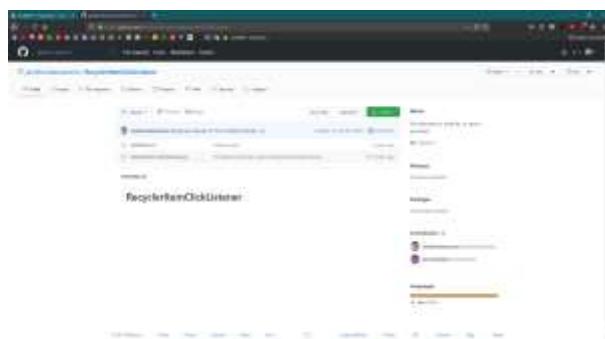


Figura 794: RecyclerItemClickListener - GitHub

Depois, configurei todos os elementos do layout e também o *RecyclerView*, reescrevendo os métodos de clique curto e longo, informo ainda que este *fragment* tem uma filtragem de itens que já foi explicada anteriormente neste relatório.



Figura 795: Configurações iniciais e do RecyclerView

Dentro do método “cliqueLongo()”, coloquei o código necessário para ser executada a remoção de um quiz, eliminando toda a sua existência, tanto foto como os dados do mesmo.



Figura 796: Método "cliqueLongo()"

Seguidamente, configurei o método “cliqueCurto()”, que abre a atividade EditarQuizActivity, responsável pela edição do quiz em si, passando também o id do quiz clicado.

```
private void cliqueCurto(int position, List<Quiz> listaQuizes) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
    builder.setTitle("Editar informações do Quiz");
    builder.setMessage("Está atualmente a editar o Quiz '" + listaQuizes.get(position).getTitulo() + "'");
    builder.setPositiveButton("Confirmar", (dialog, which) -> {
        Intent editarPost = new Intent(getApplicationContext(), EditarQuizActivity.class);
        editarPost.putExtra("idQuiz", listaQuizes.get(position).getId());
        startActivity(editarPost);
        dialog.dismiss();
    });
    builder.setNegativeButton("Cancelar", (dialog, which) -> dialog.dismiss());
    builder.show();
}
```

Figura 797: Método "cliqueCurto()"

Logo após de ter criado e configurado os eventos de clique do *RecyclerView*, tive de realizar mais um *adapter* e um layout, ficando ele como se demonstra na figura.



Figura 798: Layout (QuizesEmpresaAdapter)

Por fim, configurei os dados para que esses fossem devidamente apresentados ao utilizador.

```
@Override  
public void onBindViewHolder(@NotNull @NotNull MyViewHolder holder, int position) {  
  
    Quiz quiz = listaQuizes.get(position);  
  
    Glide.with(c).load(quiz.getImagen()).into(holder.imagem);  
    holder.titulo.setText(quiz.getTitulo());  
    holder.criador.setText(quiz.getCriador());  
    holder.classificacao.setText(String.format("%.2f", quiz.getClassificacao()));  
    holder.membros.setText("Membros : " + quiz.getTotalMembros());  
    holder.perguntas.setText("Perguntas : " + quiz.getTotalPerguntas());  
}
```

Figura 799: Método "onBindViewHolder()"

EditarQuizActivity

É nesta atividade, que serão realizadas as edições ao nível do quiz, comecei então a mesma, por planear e realizar o seu layout.

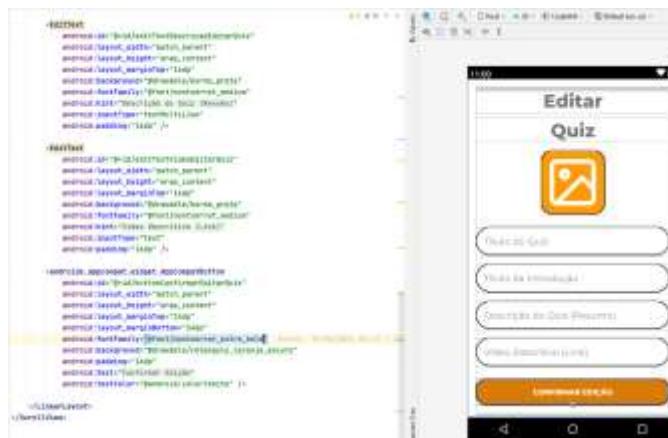


Figura 800: Layout (EditarQuizActivity)

Com o layout configurado passei à configuração de todos os elementos, tal como se observa na figura, informo também que no início desta atividade é recuperado o id do quiz que foi previamente selecionado.

```

Dialog dialog = new DialogBuilder().setContext(this).setMessage("São dados dados...").setTheme(R.style.dialog_carregamento).setCancelable(false).build();
dialogCarregamento = new ProgressDialog.Builder().setContext(this).setMessage("Carregando Alterações...").setTheme(R.style.dialog_carregamento).setCancelable(false).build();

String idquiz = getIntent().getStringExtra("idquiz");
imageViewEditorQuiz = findViewById(R.id.imageViewEditorQuiz);
editTextTitleQuiz = findViewById(R.id.editTextTitleEditQuiz);
editTextTitleExtra = findViewById(R.id.editTextTitleExtraEditQuiz);
editTextDescriçao = findViewById(R.id.editTextDescriçaoEditQuiz);
editTextExtra = findViewById(R.id.editTextExtraEditQuiz);
    
```

Figura 801: Configurações Iniciais

Continuando, defini o método “onClick” do botão confirmar alterações, começando pela verificação dos campos e se esses se encontram preenchidos.

Seguidamente, tive a necessidade de criar um *HashMap* para colocar os dados que vão ser atualizados e depois foi só proceder à atualização desses valores.

```

        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String titleQuiz = editTextTitleQuiz.getText().toString();
                String titleExtra = editTextTitleExtra.getText().toString();
                String description = editTextDescription.getText().toString();
                String view = editTextView.getText().toString();

                if (!titleQuiz.isEmpty()) {
                    if (!titleExtra.isEmpty()) {
                        if (!description.isEmpty()) {
                            if (!view.isEmpty()) {
                                dialogCarregamento.dismiss();
                                DatabaseReference quizRef = FirebaseDatabase.getInstance().getReference("quizes").child(titleQuiz);
                                DatabaseReference quizRefQuiz = quizRef.child("quiz");
                                DatabaseReference quizRefExtra = quizRef.child("extra");
                                DatabaseReference quizRefView = quizRef.child("view");

                                quizRef.updateChildren(quizRefQuiz.addValueEventListener(task -> {
                                    if (task.isSuccessful()) {
                                        Quiz quiz = task.getValue(Quiz.class);
                                        quizRefQuiz.put("title", titleQuiz);
                                        quizRefExtra.put("title", titleExtra);
                                        quizRefView.put("view", view);

                                        quizRefQuiz.updateChildren(quizRefQuiz.addValueEventListener(task2 -> {
                                            if (task2.isSuccessful()) {
                                                dialogCarregamento.dismiss();
                                                Toast.makeText(getApplicationContext(), "Altereções guardadas com sucesso!", Toast.LENGTH_SHORT).show();
                                            } else {
                                                dialogCarregamento.dismiss();
                                                Toast.makeText(getApplicationContext(), "Erro. Tente novamente mais tarde!", Toast.LENGTH_SHORT).show();
                                            }
                                        });
                                    }
                                }));
                            } else {
                                dialogCarregamento.dismiss();
                                Toast.makeText(getApplicationContext(), "Erro. Tente novamente mais tarde!", Toast.LENGTH_SHORT).show();
                            }
                        } else {
                            dialogCarregamento.dismiss();
                            Toast.makeText(getApplicationContext(), "Erro. Tente novamente mais tarde!", Toast.LENGTH_SHORT).show();
                        }
                    } else {
                        dialogCarregamento.dismiss();
                        Toast.makeText(getApplicationContext(), "Erro. Tente novamente mais tarde!", Toast.LENGTH_SHORT).show();
                    }
                } else {
                    dialogCarregamento.dismiss();
                    Toast.makeText(getApplicationContext(), "Erro. Tente novamente mais tarde!", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}

```

Figura 802: Método "onClick" do botão confirmar alterações

```
        Toast.makeText(getApplicationContext(), "Introduza um video (Link ou YouTube) para o Streamer!", Toast.LENGTH_SHORT).show();
    }
} else {
    Toast.makeText(getApplicationContext(), "Introduza um Descrição para o Streamer!", Toast.LENGTH_SHORT).show();
}
} else {
    Toast.makeText(getApplicationContext(), "Introduza o Título para o Streamer!", Toast.LENGTH_SHORT).show();
}
} else {
    Toast.makeText(getApplicationContext(), "Introduza um Titulo!", Toast.LENGTH_SHORT).show();
}
}

private void verificaOrientacao() {
}
```

Figura 803: Método "onClick" do botão confirmar alterações 2

CriarPostFragment

Neste *fragment*, são criados os *posts*, pelo o lado da administração, o layout deste, pode ser visualizado.



Figura 804: Layout (*CriarPostFragment*)

Com o layout feito, passei a realizar as configurações iniciais tal como de costume, aproveitei e configurei o botão de criar o post, com o método “*verificarCampos()*” e recuperando os valores de todos os campos.

```
criarPost = findViewById(R.id.criarPost);
etitle = new EditText(getApplicationContext());
etdescription = new EditText(getApplicationContext());
etimage = new ImageView(getApplicationContext());
btncriar = new Button(getApplicationContext());
ettitle.setHint("Título");
etdescription.setHint("Descrição");
etimage.setImageResource(R.drawable.ic_image);
btncriar.setText("Criar");

etitle.setOnFocusChangeListener(new View.OnFocusChangeListener() {
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
        if (!hasFocus) {
            etitle.clearFocus();
            etitle.setHint("Título");
        }
    }
});

etdescription.setOnFocusChangeListener(new View.OnFocusChangeListener() {
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
        if (!hasFocus) {
            etdescription.clearFocus();
            etdescription.setHint("Descrição");
        }
    }
});

btncriar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String title = etitle.getText().toString();
        String description = etdescription.getText().toString();
        String image = etimage.getDrawable().getHref();
        String url = "http://10.0.2.2:8080/api/post";
        JSONObject json = new JSONObject();
        try {
            json.put("title", title);
            json.put("description", description);
            json.put("image", image);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        String jsonStr = json.toString();
        RequestBody body = RequestBody.create(jsonStr, MediaType.parse("application/json; charset=utf-8"));
        Call<String> call = cliente.createPost(url, body);
        call.enqueue(new Callback<String>() {
            @Override
            public void onResponse(Call<String> call, Response<String> response) {
                if (response.isSuccessful()) {
                    Log.d("Response", "Success: " + response.body());
                } else {
                    Log.d("Response", "Error: " + response.errorBody().string());
                }
            }

            @Override
            public void onFailure(Call<String> call, Throwable t) {
                Log.d("Failure", "Error: " + t.getMessage());
            }
        });
    }
});
```

Figura 805: Configurações Iniciais e configuração do método “*onClick*” do botão criar post

Como já visto neste relatório, tive a necessidade de incluir este método para recuperar a imagem introduzida pelo o utilizador.

```

@Override
public void onActivityResult(int requestCode, Intent resultCode, IntentData data, String retorno) {
    super.onActivityResult(requestCode, resultCode, data);
    image = null;
    try {
        if (requestCode == SELECAO_CAMERA) {
            Uri uri = Uri.parse(data.getStringExtra("data"));
            image = uri;
        } else if (requestCode == SELECAO_GALERIA) {
            Uri uri = Uri.parse(data.getStringExtra("data"));
            image = uri;
        }
    } catch (Exception e) {
        e.printStackTrace();
        Toast.makeText(getApplicationContext(), "Erro ao Recuperar Imagem", Toast.LENGTH_SHORT).show();
    }
}

private void abreGaleria() {
    Intent i = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    if (i.resolveActivity(getApplicationContext()) != null) {
        startActivityForResult(i, SELECAO_GALERIA);
    }
}

```

Figura 806: Método "onActivityResult"

E também configurei o método “abrirGaleria()”, para que este abrisse a galeria de fotos do utilizador.

```

private void abrirGaleria() {
    Intent i = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    if (i.resolveActivity(getApplicationContext()) != null) {
        startActivityForResult(i, SELECAO_GALERIA);
    }
}

```

Figura 807: Método "abrirGaleria()"

Logo após, codifiquei o método “verificarCampos()”, responsável pela a verificação de todos os campos, tal como o seu próprio nome indica.

```
private void verificarCampos(String titulo, String descricao, int tag) {
    if (titulo.isEmpty()) {
        if (descricao.isEmpty()) {
            if (tag > 0) {
                if (image != null) {
                    dialog.show();
                    criarPost(titulo, descricao, tag);
                } else {
                    Toast.makeText(getApplicationContext(), "Introduza uma Imagem para o Post!", Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(getApplicationContext(), "Selecione uma Tag!", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(getApplicationContext(), "Introduza uma Descrição para o Post!", Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(getApplicationContext(), "Introduza um Título para o Post!", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 808: Método "verificarCampos()"

Se a verificação de campos for bem sucedida, o método “criarPost()” é executado, o objeto do tipo *Post* também é inserido para a Firebase Database, quando todos os métodos executam com sucesso.

```
private void criarPost(String titulo, String descricao, int posicao) {
    DatabaseReference postRef = FirebaseDatabase.getInstance().getReference().child("posts");
    Post post = new Post();

    post.setId(postRef.push().getKey());
    post.setReader(Configs.recuperaNomeUtilizador());
    post.setWriter(Configs.recuperaSetorUser());
    post.setTitle(titulo);
    post.setStatus(Configs.PENDENTE); // alterar
    post.setDescription(descricao);
    post.setTag(Configs.getTags());
    post.setReader(Configs.recuperaUser());
    post.guardarImagen(dadasImagen, validar -> {
        if (validar) {
            post.guardarValidar2 = true;
            if (validar2) {
                dialog.dismiss();
                Toast.makeText(getApplicationContext(), "Post Criado com Sucesso!", Toast.LENGTH_SHORT).show();
                finish();
            } else {
                dialog.dismiss();
                Toast.makeText(getApplicationContext(), "Erro, Tente Novamente Mais Tarde!", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(getApplicationContext(), "Erro, Tente Novamente Mais Tarde!", Toast.LENGTH_SHORT).show();
        }
    });
}
```

Figura 809: Método "criarPost()"

ListarPostsFragment

Este *fragment*, é praticamente igual ao *ListarQuizFragment*, mudando apenas a visualização dos itens do *RecyclerView*.



Figura 810: Layout (*ListarPostsFragment*)

Configurei o *RecyclerView*, para esta atividade e realizei todas as configurações básicas para colocar a atividade a iniciar sem problemas, controlando todos os elementos da mesma. Passei logo então, para a criação do *adapter* e por consequência o seu layout.

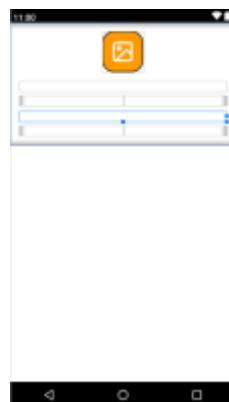


Figura 811: Layout (*PostsAdminAdapter*)

Para finalizar, método “onBindViewHolder()” recuperei todos os dados e encaminhei os mesmos para os elementos corretos.

```
@Override
public void onBindViewHolder(@NotNull @NotNull MyViewHolder holder, int position) {

    Post post = posts.get(position);

    Glide.with(c).load(post.getImagen()).into(holder.imagem);
    holder.criador.setText(post.getCriador());
    holder.titulo.setText(post.getTitulo());
    holder.tag.setText(post.getTag());
    holder.descripcion.setText(post.getDescripcion());
    holder.comentarios.setText("Comentários :: " + post.getComentarios());
    holder.gostos.setText("Gostos :: " + post.getGostos());
}
```

Figura 812: Método "onBindViewHolder()"

EditarPostActivity

É nesta atividade, que os *posts* podem ser editados, apenas a administração e os utilizadores de cada post (quando o seu estado é pendente), tem acesso a esta atividade.



Figura 813: Layout (EditarPostActivity)

Sem demora, realizei logo as configurações iniciais, como se verifica na figura.

Figura 814: Configurações iniciais

Depois de ter realizado as configurações iniciais, foi necessário recuperar todos os dados, primeiramente fui buscar as informações “post” e depois também fui buscar as “tags”.

```

private void buscarTags(String tag) {
    DatabaseReference reference = FirebaseDatabase.getInstance().getReference("tags");
    reference.orderByValue().equalTo(tag).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for (DataSnapshot data : dataSnapshot.getChildren()) {
                String nomePost = data.getKey();
                String nomeTag = data.getValue().toString();
                Log.d("TAG", nomePost + " - " + nomeTag);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            Log.d("TAG", "Error: " + error.getMessage());
        }
    });
}

private void buscarPost() {
    DatabaseReference reference = FirebaseDatabase.getInstance().getReference("posts");
    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for (DataSnapshot data : dataSnapshot.getChildren()) {
                String nomePost = data.getKey();
                String nomeUser = data.getValue().toString();
                Log.d("POST", nomePost + " - " + nomeUser);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            Log.d("POST", "Error: " + error.getMessage());
        }
    });
}

```

Figura 815: Métodos "buscarTags()" e "buscarPost()

Para finalizar a atividade, codifiquei o método “onClick()” do botão confirmar, que trata das verificações dos campos e também executa a função “atualizarPost()”, função esta que envia os dados para a *Firebase Database*.

```

private void confirmarEdicaoPost() {
    EditText title = findViewById(R.id.title);
    EditText description = findViewById(R.id.description);
    EditText author = findViewById(R.id.author);
    EditText tags = findViewById(R.id.tags);

    if (title.getText().toString().isEmpty() || description.getText().toString().isEmpty() || author.getText().toString().isEmpty() || tags.getText().toString().isEmpty()) {
        Toast.makeText(getApplicationContext(), "Todos os campos são obrigatórios!", Toast.LENGTH_SHORT).show();
    } else {
        String titleText = title.getText().toString();
        String descriptionText = description.getText().toString();
        String authorText = author.getText().toString();
        String tagsText = tags.getText().toString();

        DatabaseReference reference = FirebaseDatabase.getInstance().getReference("posts").child("post").child("postID");
        reference.updateChildren(new DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError databaseError, DatabaseReference databaseReference) {
                if (databaseError != null) {
                    Log.e("TAG", "Error: " + databaseError.getMessage());
                } else {
                    Toast.makeText(getApplicationContext(), "Post alterado com sucesso!", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}

```

Figura 816: Métodos "confirmarEdicaoPost()" e "atualizarPost()

CriarTagsTemasFragment

Este *fragment*, ficou destinado para a criação de novos *temas* e *tags*. Os temas estão relacionados com os *quizes* e as *tags* estão relacionadas com os *posts*.

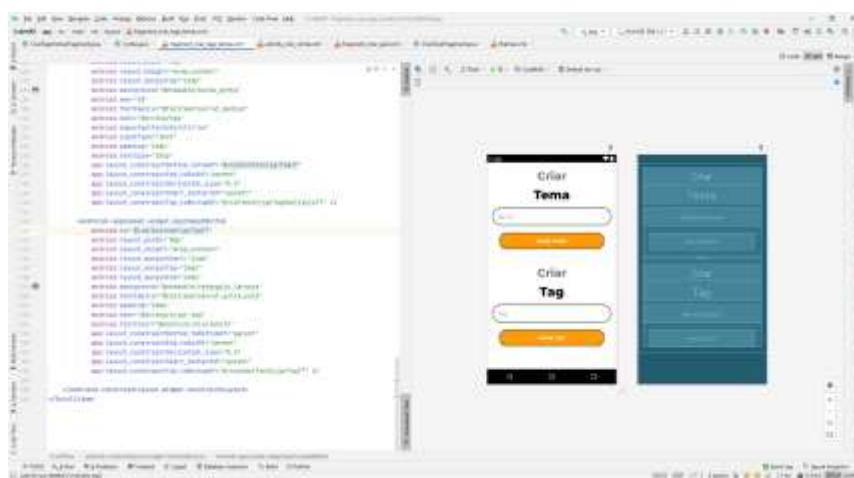


Figura 817: Layout (CriarTagsTemasFragment)

Mais uma vez, realizei as configurações iniciais obrigatórias, para obter os dados passados no layout. Aproveitei e defini os métodos “*onClick*” de ambos os botões com o método “*criarTT()*”.

```
Dialog = new AlertDialog.Builder().setContext(getApplicationContext()).setMessage("Criar TT...").setTheme(R.style.dialog_criartemastag).setCancelable(false).build();  
  
Tema = root.findViewById(R.id.edtTextCriarTema);  
tag = root.findViewById(R.id.edtTextCriarTag);  
  
Button criarTema, criarTag;  
criarTema = root.findViewById(R.id.buttonCriarTema);  
criarTag = root.findViewById(R.id.buttonCriarTag);  
  
criarTema.setOnClickListener(v -> criarTT("tema", tema.getText().toString()));  
criarTag.setOnClickListener(v -> criarTT("tag", tag.getText().toString()));
```

Figura 818: Configurações Iniciais

No método “criarTT()”, são testados os valores passados nos campos e é verificado se ele foi preenchido ou não. Se foi preenchido, envia para a base de dados.

Este método trata tanto a criação de uma *tag* como a criação de um *tema*, como se pode ver nas próximas prints.

```

private void onDatabaseError(String message) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    } else {
        AlertDialog alertDialog = new AlertDialog.Builder(this).create();
        alertDialog.setTitle("Error");
        alertDialog.setMessage(message);
        alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
                (dialog, which) -> dialog.dismiss());
        alertDialog.show();
    }
}

private void onSuccess(String message) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    } else {
        AlertDialog alertDialog = new AlertDialog.Builder(this).create();
        alertDialog.setTitle("Success");
        alertDialog.setMessage(message);
        alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
                (dialog, which) -> dialog.dismiss());
        alertDialog.show();
    }
}

private void onSuccessWithImage(String message, Uri uri) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    } else {
        AlertDialog alertDialog = new AlertDialog.Builder(this).create();
        alertDialog.setTitle("Success");
        alertDialog.setMessage(message);
        alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
                (dialog, which) -> dialog.dismiss());
        alertDialog.show();
    }
}
}

```

Figura 819: Método "criarTT()"

```

        tagsRef.child(String.valueOf(valorAtivo)).setvalue(texto).addCompleteListener(task -> {
            if (task.isSuccessful()) {
                dialog.dismiss();
                linearProgress();
                Toast.makeText(getApplicationContext(), "Tag adicionada com sucesso!", Toast.LENGTH_SHORT).show();
            } else {
                dialog.dismiss();
                Toast.makeText(getApplicationContext(), "Error, Teste Novamente Mais Tardar!", Toast.LENGTH_SHORT).show();
            }
        });
    }

    @Override
    public void onCancelled(@NonNull @NotNull DatabaseError error) {
    }
}

} else { // andr. Informa o utilizador que preencher
    Toast.makeText(getApplicationContext(), "Introduza algo no campo inserido!", Toast.LENGTH_SHORT).show();
}

```

Figura 820: Método "criarTT()" 2

Para finalizar este *fragment*, criei ainda o método “limparCampos()”.

```
private void limparCampos() {  
    tag.setText("");  
    tema.setText("");  
}
```

Figura 821: Método "limparCampos()"

ListarTagsTemasFragment

Neste listar, decidi colocar tanto o listar das *tags* e o listar dos temas, mas desta vez, tive de realizar de uma forma totalmente diferente da qual estava habituado.



Figura 822: Layout (ListarTagsTemasFragment)

Depois de criar o layout, procurei como realizar a configuração de um ListView, pois já não me recordava muito bem de como o fazia.

<https://www.tutorialspoint.com/how-to-make-a-listview-in-android/>



Figura 823: Solução de como criar um ListView

Depois de ter visto a anterior solução, só tive de aplicar para dentro do meu *fragment* e fui isso que realizei, com a ajuda de um *AlertDialog* como é certo. Começando com o clique curto em ambas as listagens que tem a função de editar.

```

    listview = view.findViewById(R.id.list_view);
    listAdapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1, list_items);
    listview.setAdapter(listAdapter);
    listview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView parent, View view, int position, long id) {
            Intent intent = new Intent(getApplicationContext(), AddEditActivity.class);
            Bundle bundle = new Bundle();
            bundle.putString("id", list_items.get(position).get("id"));
            bundle.putString("name", list_items.get(position).get("name"));
            bundle.putString("email", list_items.get(position).get("email"));
            intent.putExtras(bundle);
            startActivity(intent);
        }
    });
}

```

Figura 824: Clique curto ListViewTemas

Figura 825: Clique curto ListViewTags

Logo após ter configurado o clique curto, passei à codificação, desta vez, do clique longo, que neste caso tem a função de eliminar.

```

temas.setOnItemLongClickListener((parent, view, position, id) -> {
    AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
    builder.setTitle("Remover Tema");
    builder.setCancelable(false);
    builder.setMessage("Tem a certeza, que pretende eliminar o Tema \"" + listaTemas.get(position) + "\" ?");
    builder.setPositiveButton("Sim", (dialog, which) -> {
        DatabaseReference tagsRef = DefinicoesFirebase.recuperarBaseDados().child("temas");
        tagsRef.child(idTemas.get(position)).removeValue().addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                Toast.makeText(getContext(), "Tema Eliminado com Sucesso!", Toast.LENGTH_SHORT).show();
                listaTemasAdapter.notifyDataSetChanged();
                dialog.dismiss();
            } else {
                Toast.makeText(getContext(), "Erro, Tente Novamente Mais Tarde!", Toast.LENGTH_SHORT).show();
                dialog.dismiss();
            }
        });
    });
    builder.setNegativeButton("Cancelar", (dialog, which) -> dialog.dismiss());
    builder.show();
    return true;
});

```

Figura 826: Clique Longo ListViewTemas

```

tags.setOnItemLongClickListener((parent, view, position, id) -> {
    AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
    builder.setTitle("Remover Tag");
    builder.setCancelable(false);
    builder.setMessage("Tem a certeza, que pretende eliminar a Tag \"" + listaTags.get(position) + "\" ?");
    builder.setPositiveButton("Sim", (dialog, which) -> {
        DatabaseReference tagsRef = DefinicoesFirebase.recuperarBaseDados().child("tags");
        tagsRef.child(idTags.get(position)).removeValue().addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                Toast.makeText(getContext(), "Tag Eliminada com Sucesso!", Toast.LENGTH_SHORT).show();
                listaTagsAdapter.notifyDataSetChanged();
                dialog.dismiss();
            } else {
                Toast.makeText(getContext(), "Erro, Tente Novamente Mais Tarde!", Toast.LENGTH_SHORT).show();
                dialog.dismiss();
            }
        });
    });
    builder.setNegativeButton("Cancelar", (dialog, which) -> dialog.dismiss());
    builder.show();
    return true;
});

```

Figura 827: Clique Longo ListViewTags

Seguidamente, configurei os métodos de ir buscar os dados, tanto *tags* como temas e neste caso o código mudou um pouco visto que teria de carregar *dois ArrayList's*, um com os ids e outro com os nomes, tal como se pode visualizar nas próximas figuras.

```
private void buscarTemas() {
    dialogCarregamento.show();
    DatabaseReference temasRef = DefinicaoFirebase.recuperarBaseDados().child("temas");

    temasRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull @NotNull DataSnapshot snapshot) {
            listaTemas.clear();
            idTemas.clear();
            for (DataSnapshot dados : snapshot.getChildren()) {
                idTemas.add(dados.getKey());
                listaTemas.add(dados.getValue(String.class));
            }
            dialogCarregamento.dismiss();
            listaTemasAdapter.notifyDataSetChanged();
        }

        @Override
        public void onCancelled(@NonNull @NotNull DatabaseError error) {
        }
    });
}
```

Figura 828: Método "buscarTemas()"

```
private void buscarTags() {
    dialogCarregamento.show();
    DatabaseReference temasRef = DefinicaoFirebase.recuperarBaseDados().child("tags");

    temasRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull @NotNull DataSnapshot snapshot) {
            listaTags.clear();
            idTags.clear();
            for (DataSnapshot dados : snapshot.getChildren()) {
                idTags.add(dados.getKey());
                listaTags.add(dados.getValue(String.class));
            }
            dialogCarregamento.dismiss();
            listaTagsAdapter.notifyDataSetChanged();
        }

        @Override
        public void onCancelled(@NonNull @NotNull DatabaseError error) {
        }
    });
}
```

Figura 829: Método "buscarTags()"

Para finalizar, criei o método “onClick” do FloatingActionButton, que troca a visualização de ambas as listagens com a variável *boolean* de controlo “isTemaVisivel”, trocando a visibilidade dos *LinearLayout’s*.

```
private void trocarTT(View root) {
    FloatingActionButton fab;
    fab = root.findViewById(R.id.fabTrocarTT);

    LinearLayout linearLayoutTag, linearLayoutTema;
    linearLayoutTag = root.findViewById(R.id.linearLayoutListarTag);
    linearLayoutTema = root.findViewById(R.id.linearLayoutListarTema);

    fab.setOnClickListener(v -> {

        if (isTemaVisivel) {
            linearLayoutTag.setVisibility(View.GONE);
            linearLayoutTema.setVisibility(View.VISIBLE);
            isTemaVisivel = false;
        } else {
            linearLayoutTag.setVisibility(View.VISIBLE);
            linearLayoutTema.setVisibility(View.GONE);
            isTemaVisivel = true;
        }
    });
}
```

Figura 830: Método "trocarTT()"

CriarNotificacaoFragment

Este fragment, foi destinado para a criação de notificações, visíveis para todos os utilizadores.



Figura 831: Layout (CriarNotificacaoFragment)

Após ter terminado o layout, procedi à realização das configurações iniciais de mais um fragment.

```

dialogCorrespondents = new AlertDialog.Builder(getActivity()).setContext(getContext()).setMessage("Criadas Notificações...").setItems(R.style.dialog_correspondents).setCancelable(false).show();

ImageViewCriarNota = root.findViewById(R.id.imageViewCriarNotificacao);

editTextTitulo = root.findViewById(R.id.editTextTituloCriarNotificacao);
editTextDescrição = root.findViewById(R.id.editTextDescriçãoCriarNotificacao);

Imagebutton_imagemGaleria;
Imagebutton_imagemGaleria = root.findViewById(R.id.imagebutton_imagemGaleria);
imagebutton_imagemGaleria.setOnClickListener(view -> abrirGaleria());

Button_buttonCriarNotificacao;
buttonCriarNotificacao = root.findViewById(R.id.buttonCriarNotificacao);

```

Figura 832: Configurações Iniciais

Seguidamente, configurei o método “onClick” do botão para criar a notificação, mais uma vez são verificados todos os campos e se estão todos preenchidos.

Se isso realmente acontecer, um objeto do tipo “Notificacao” é criado e enviado para a *Firebase Database* e a sua imagem, é enviada para a *Firebase Storage*, tal como se pode verificar no código.

```

buttonCreateNotification.setOnClickListener(v -> {
    String titulo = editTextTitulo.getText().toString();
    String descricao = editTextDescricao.getText().toString();
    String nomeImage = editTextNomeImage.getText().toString();

    if (titulo.isEmpty() || nomeImage.isEmpty() || descricao.isEmpty()) {
        Toast.makeText(getApplicationContext(), "Todos os campos devem ser preenchidos!", Toast.LENGTH_SHORT).show();
    } else {
        if (image != null) {
            dialogCarregamento.show();
            notificacao.notificacao = new Notificacao();
            notificacao.notificacao.titulo = titulo;
            notificacao.notificacao.descricao = descricao;
            notificacao.notificacao.image = nomeImage;
            notificacao.notificacao.ref = FirebaseDatabase.getInstance().getReference("notificacao");
            notificacao.notificacao.urlImage = storageRef.child("notificacao").getDownloadUrl().toString();

            notificacaoRef.child("notificacao").setValue(notificacao).addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    notificacao.guardarImagenes(mandoImage, editor -> {
                        if (editor != null) {
                            editor.commit();
                            dialogCarregamento.dismiss();
                            Toast.makeText(getApplicationContext(), "Notificação criada com sucesso!", Toast.LENGTH_SHORT).show();
                            getActionBar().setHomeButtonEnabled(true);
                        } else {
                            dialogCarregamento.dismiss();
                            Toast.makeText(getApplicationContext(), "Erro: Teste Novamente Mais Tardar!", Toast.LENGTH_SHORT).show();
                        }
                    });
                } else {
                    dialogCarregamento.dismiss();
                    Toast.makeText(getApplicationContext(), "Erro: Teste Novamente Mais Tardar!", Toast.LENGTH_SHORT).show();
                }
            });
        } else {
            dialogCarregamento.dismiss();
            Toast.makeText(getApplicationContext(), "Escolha uma imagem!", Toast.LENGTH_SHORT).show();
        }
    }
});
    
```

Figura 833: Método "onClick" botão de criar notificação

```

        } else {
            Toast.makeText(getApplicationContext(), "Escolha uma imagem!", Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(getApplicationContext(), "Introduza uma Descrição!", Toast.LENGTH_SHORT).show();
    }
} else {
    Toast.makeText(getApplicationContext(), "Introduza um Titulo!", Toast.LENGTH_SHORT).show();
}
});
```

Figura 834: Método "onClick" botão de criar notificação 2

Mais uma vez, foi preciso incluir o método “onActivityResult” para recuperar a imagem escolhida pelo o utilizador.

```

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    image = null;
    try {
        switch (requestCode) {
            case Confirma_SELECAO_GALERIA:
                image = (Bitmap) data.getExtras().get("data");
                break;
            case Confirma_SELECAO_BAIXAR:
                Uri localImagemBaixada = data.getData();
                if (Build.VERSION.SDK_INT >= 24) {
                    // Usar método para obter URI da imagem
                    ImageDecoder.Source source = ImageDecoder.createSource(getActivity().getContentResolver(), localImagemBaixada);
                    image = ImageDecoder.decodeImage(source);
                } else {
                    // Usar versão mais antiga do código
                    image = MediaStore.Images.Media.getBitmap(getActivity().getContentResolver(), localImagemBaixada);
                }
                break;
        }
    } catch (Exception e) {
        e.printStackTrace();
        Toast.makeText(getApplicationContext(), "Erro ao Recuperar Imagem!" + e.getMessage(), Toast.LENGTH_SHORT).show();
    }
}

// CONVERTE IMAGEM PARA
if (image != null) {
    ByteArrayOutputStream basicArrayOutputStream = new ByteArrayOutputStream();
    image.compress(Bitmap.CompressFormat.PNG, 100, basicArrayOutputStream);
    byte[] imagens = basicArrayOutputStream.toByteArray();
}
}

```

Figura 835: Método "onActivityResult()"

Para finalizar a criação da notificação precisei do seguinte método, para abrir a galeria de fotos do utilizador.

```

private void abrirGaleria() {
    Intent i = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    if (i.resolveActivity(getActivity().getPackageManager()) != null) {
        startActivityForResult(i, SELECAO_GALERIA);
    }
}

```

Figura 836: Método "abrirGaleria()"

ListarNotificacoesFragment

Nesta parte da app, são listadas todas as notificações, o layout que coloquei foi o seguinte.



Figura 837: Layout (ListarNotificacoes)

Depois de finalizar o layout, procedi às configurações iniciais do ficheiro principal deste fragment. Comecei por recuperar os elementos básicos e depois, atribui os métodos “cliqueCurto()” e “cliqueLongo()” no RecyclerView.

O código está presente na próxima página.

Figura 838: Configurações Iniciais

Depois de ter realizado as configurações iniciais, aproveitei e realizei logo o método responsável por buscar todas as notificações.

```
private void buscarNotificacoes() {
    dialogCarregamento.show();
    DatabaseReference notificacoesRef = DefinicaoFirebase.recuperarBaseDados().child("notificacoes");

    notificacoesRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull @NotNull DataSnapshot snapshot) {
            listaNotificacoes.clear();

            for (DataSnapshot dados : snapshot.getChildren()) {
                listaNotificacoes.add(dados.getValue(Notificacao.class));
            }

            dialogCarregamento.dismiss();
            notificacoesAdapter.notifyDataSetChanged();
        }

        @Override
        public void onCancelled(@NonNull @NotNull DatabaseError error) {
        }
    });
}
```

Figura 839: Método "buscarNotificacoes()"

Configurei o clique curto, como a edição da notificação, como se pode observar na figura.

```
private void cliqueCurto(int position, List<Notificacao> listaNotificacoes) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
    builder.setTitle("Editar Notificação");
    builder.setMessage("Está atualmente a editar o Notificação " + listaNotificacoes.get(position).getTitulo() + "!");
    builder.setPositiveButton(text: "Confirmar", (dialog, which) -> {
        Intent editarPost = new Intent(getContext(), EditarNotificacaoActivity.class);
        editarPost.putExtra(name: "idNotificacao", listaNotificacoes.get(position).getId());
        startActivity(editarPost);
    });
    builder.setNegativeButton(text: "Cancelar", (dialog, which) -> dialog.dismiss());
    builder.show();
}
```

Figura 840: Método "cliqueCurto()"

Configurei também o clique longo, mas este remove a notificação em que for executado. O código do mesmo encontra-se na próxima página.

```
private void cliqueLongo(int position, List<Notificacao> listaNotificacoes) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
    builder.setTitle("Remover um Notificação");
    builder.setMessage("Tem a certeza de querer eliminar a Notificação " + listaNotificacoes.get(position).getTitulo() + "?");
    builder.setNegativeButton(text: "Não", (dialog, which) -> {
        dialog.dismiss();
    });
    builder.setPositiveButton(text: "Sim", (dialog, which) -> {
        StorageReference fotoRef = FirebaseStorage.getInstance().getReference("images").child(listaNotificacoes.get(position).getFoto() + ".jpg");
        fotoRef.getDownloadUrl().addOnSuccessListener(task -> {
            if (task.isSuccessful()) {
                FirebaseFirestore database = FirebaseFirestore.getInstance();
                database.collection("notificacoes").whereEqualTo("foto", task.getResult().toString()).get(position).getRef().removeValue(error, ref) -> {
                    if (error == null) {
                        dialog.dismiss();
                        Toast.makeText(getContext(), "Notificação removida com sucesso!", Toast.LENGTH_SHORT).show();
                    } else {
                        dialog.dismiss();
                        Toast.makeText(getContext(), "Erro: Tente novamente mais tarde!", Toast.LENGTH_SHORT).show();
                    }
                };
            }
        });
        database.collection("notificacoes").document(position).delete();
        dialog.dismiss();
        Toast.makeText(getContext(), "Nota: Removendo nota!", Toast.LENGTH_SHORT).show();
    });
    builder.setNegativeButton(text: "Cancelar", (dialog, which) -> dialog.dismiss());
    builder.show();
}
```

Figura 841: Método "cliqueLongo"

Não foi preciso configurar um *adapter* novo, pois este utiliza o mesmo do que a *NotificacoesActivity*.

EditarNotificacaoActivity

Esta atividade, foi planeada para, como o seu nome indica, editar exclusivamente notificações, o seu layout está presente na imagem abaixo.



Figura 842: Layout (EditarNotificacaoActivity)

Sem demora, realizei as configurações iniciais desta atividade, começando por recuperar os elementos básicos de operação que se encontram no layout anteriormente mostrado.

Configurei também o método “onClick” para verificar os campos e se estão preenchidos e de seguida atualizar os dados, no caso de algum ser modificado. O código destas configurações encontra-se na próxima página.

Figura 843: Configurações Iniciais

Para finalizar esta atividade, configurei ainda o método mais importante, que é o “`buscarNotificacao()`”, que recebe o id da notificação que previamente foi escolhida antes de a execução desta atividade.

```
private void buscarNotificacao(String idNotificacao) {
    DatabaseReference notificacaoRef = FirebaseDatabase.getInstance().getReference("notificacoes").child(idNotificacao);

    notificacaoRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            dialogCarregamento.show();
            Notificacao notificacao = snapshot.getValue(Notificacao.class);

            Glide.with(getApplicationContext()).load(notificacao.getImagem()).into(imageViewEditorNotificacao);
            editTextTitulo.setText(notificacao.getTitulo());
            editTextDescrição.setText(notificacao.getDescrição());
            dialogCarregamento.dismiss();
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
        }
    });
}
```

Figura 844: Método "buscarNotificacao()"

ModeradorActivity

Esta atividade é basicamente uma cópia da *AdminActivity*, porque as configurações são bastante semelhantes, o que muda é apenas as opções visualizáveis pelo utilizador, visto que esta será apenas para o grupo “mod”.



Figura 845: Definição dos ids dentro da AppBarConfiguration

Seguidamente, configurei o ficheiro “mobile_navigation_mod.xml” colocando apenas as opções que convinham e as quais o “mod” tem acesso

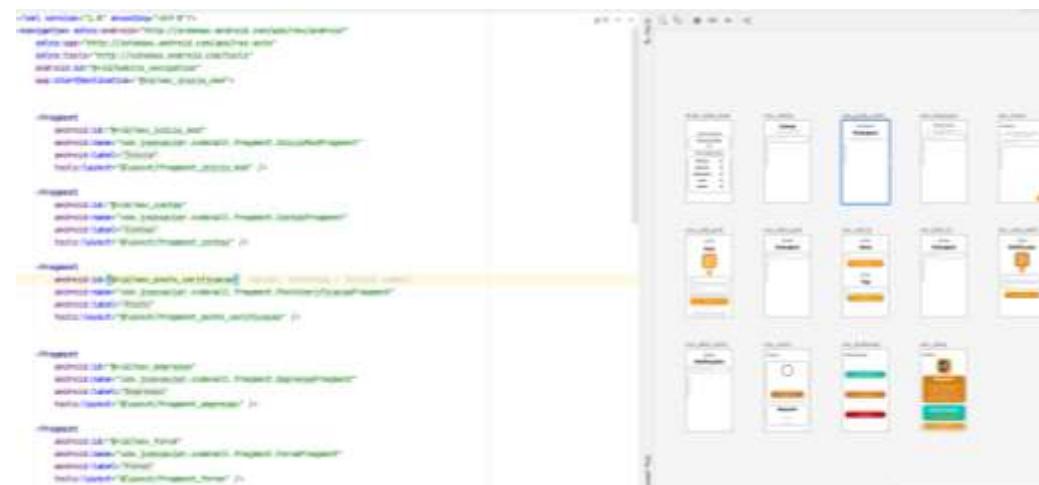


Figura 846: Configuração do ficheiro "mobile_navigation_mod.xml"

E para finalizar a configuração desta atividade, criei mais um ficheiro “menu_mod.xml” que contém todas as opções a qual o “mod” tem acesso.

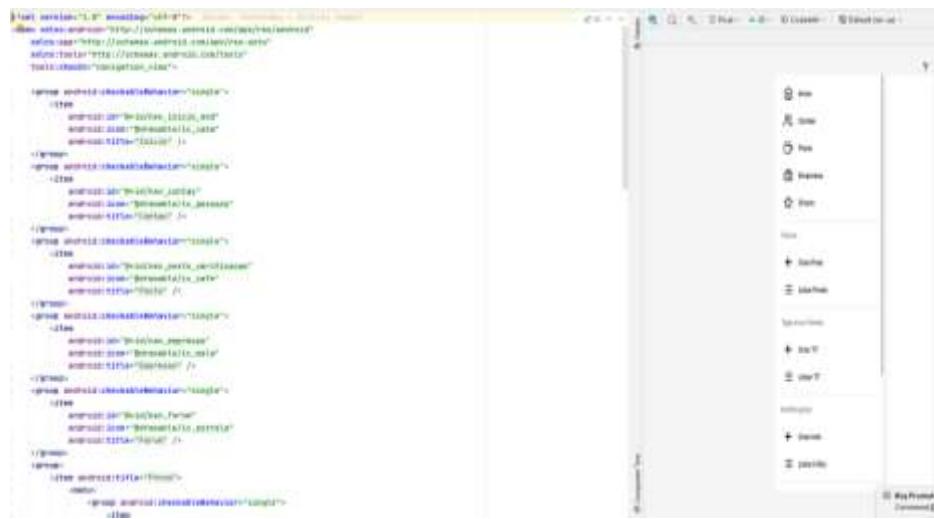


Figura 847: Ficheiro "menu_mod.xml"

InícioModFragment

Este fragment, é aquele que é apresentado sempre que um utilizador do tipo “admin” é logado.



Figura 848: Layout (InicioModFragment)

Depois do layout estar pronto, mais uma vez, realizei as configurações iniciais do mesmo.

```
textPostsFazer = root.findViewById(R.id.textCountModPostsFazer);
textMembros = root.findViewById(R.id.textCountModMembros);
textEmpresas = root.findViewById(R.id.textCountModEmpresas);
textModeradores = root.findViewById(R.id.textCountModModeradores);
textPosts = root.findViewById(R.id.textCountModPosts);  
  
buscarInfo();  
  
return root;
```

Figura 849: Configurações Iniciais

Logo depois, codifiquei o método “buscarInfo()”, responsável por ir buscar todas as informações presentes no layout anteriormente criado.



Figura 850: Método "buscarInfo()"

Depois, executei a app, para ver se os dados eram realmente bem encaminhados e atualizados.



Figura 851: Captura de ecrã

Para finalizar e impedir o fragment de “crashar”, tive que realizar uma verificação para que só sejam vistos os dados em que o tipo da conta, seja diferente de vazio, tal como se observa na figura.



```

public static void main(String[] args) {
    try {
        // ... (rest of the try block)
    } catch (Exception e) {
        if (classe != null) {
            if (!classe.equals("vazio")) {
                System.out.println("classe != vazio");
            }
        }
    }
}

```

Figura 852: Condição para impedir o aparecimento de erros

EmpresaConfiguracaoActivity

Mais uma vez, comecei pela realização do layout desta atividade, que ficou da seguinte forma.

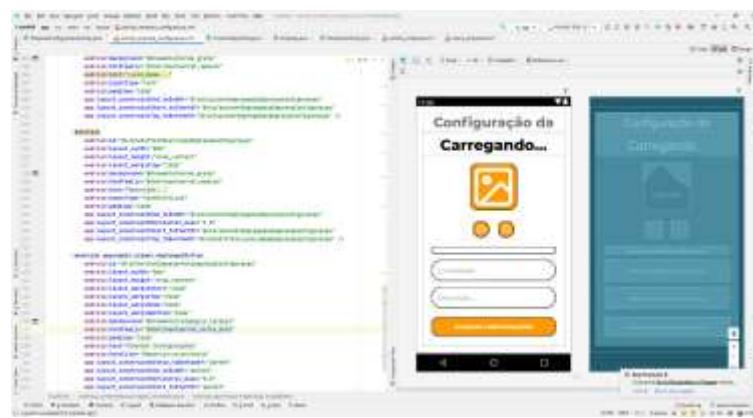


Figura 853: Layout da EmpresaConfiguracaoActivity

Depois vieram as configurações iniciais da atividade, como se pode visualizar.

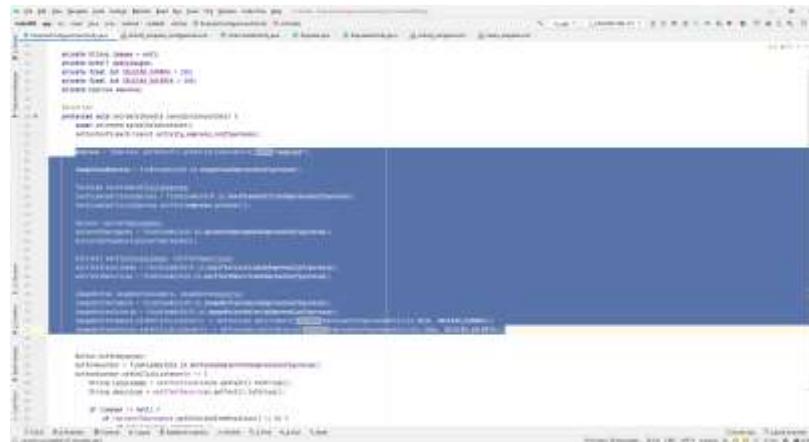


Figura 854: Configurações iniciais

No início desta atividade, foi recuperada a empresa, passada da atividade anterior (CriarContaActivity). Seguidamente, fiz a configuração do método “onCLick” do botão de guardar, fazendo as verificações normais e o envio de dados para a firebase database e o envio da imagem para a firebase storage.

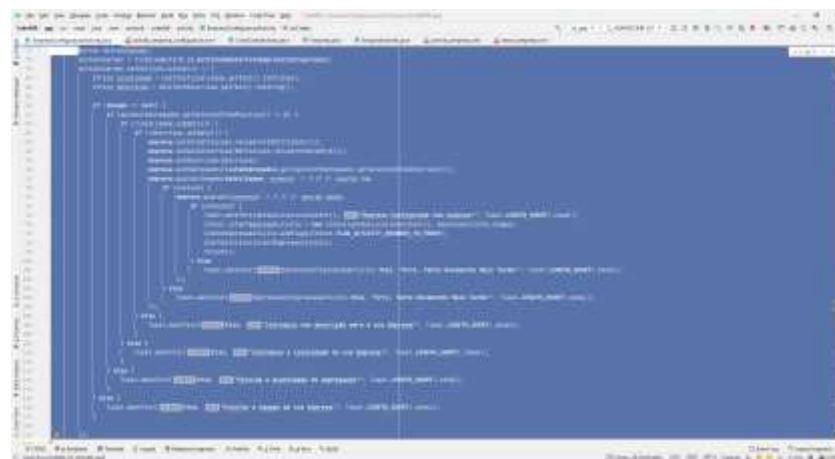


Figura 855: Método "onClick" do botão de guardar

Seguidamente, subscrevi o método “onActivityResult”, com o código necessário para recuperar a imagem escolhida, sendo ela da câmara ou da própria galeria do utilizador.

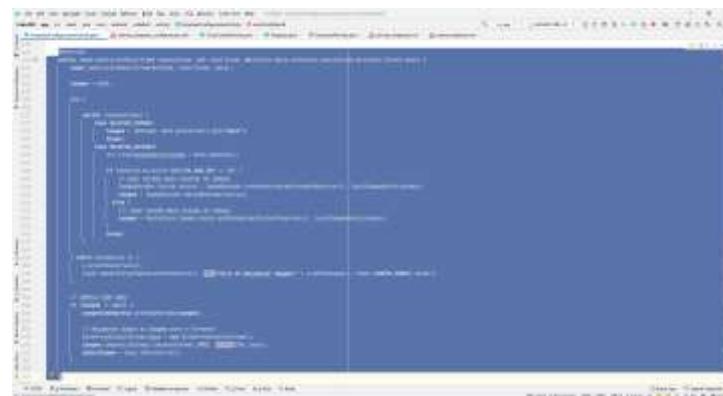


Figura 856: Método "onActivityResult"

Antes de finalizar esta atividade, faltava ainda um método “buscarEmpregados()”, que iria buscar uns valores já definidos e criados manualmente por mim na base de dados.

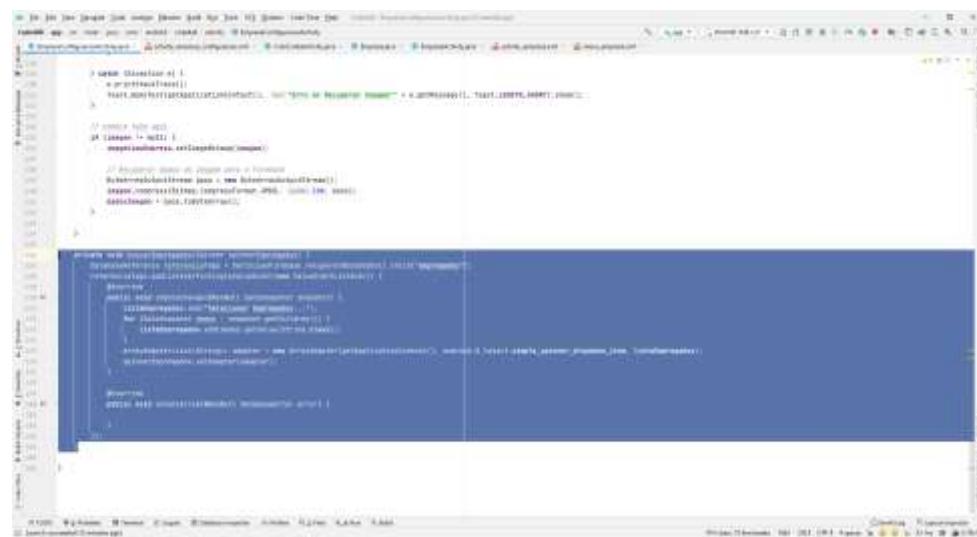


Figura 857: Método “buscarEmpregados()”

EmpresaActivity

Comecei por copiar o código da PrincipalActivity e modificar o mesmo, mas desta vez para a EmpresaActivity, tal como se observa na figura abaixo.



Figura 858: Configuração dos ids da AppBarConfiguration

Seguidamente, passei à configuração do ficheiro “mobile_navigation_empresa.xml”, responsável pelos os itens da empresa

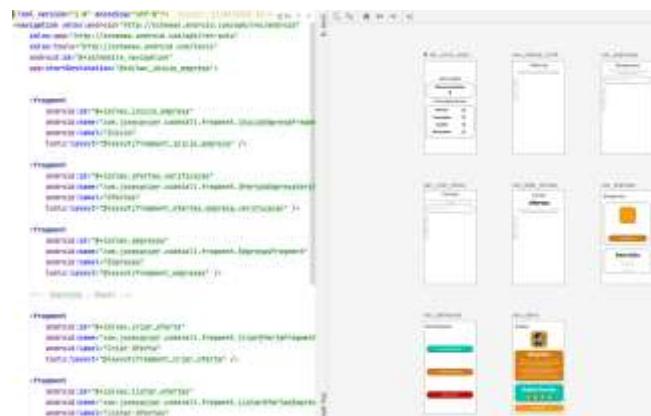


Figura 859: Configuração do ficheiro "mobile_navigation_empresa.xml"

Para finalizar a configuração desta atividade, configurei o menu que seria apresentado, ficando da seguinte forma.

Figura 860: Ficheiro "menu_empresa.xml"

InicioEmpresaFragment

Neste fragment, comecei por tratar do layout, visto que é a parte mais fácil.

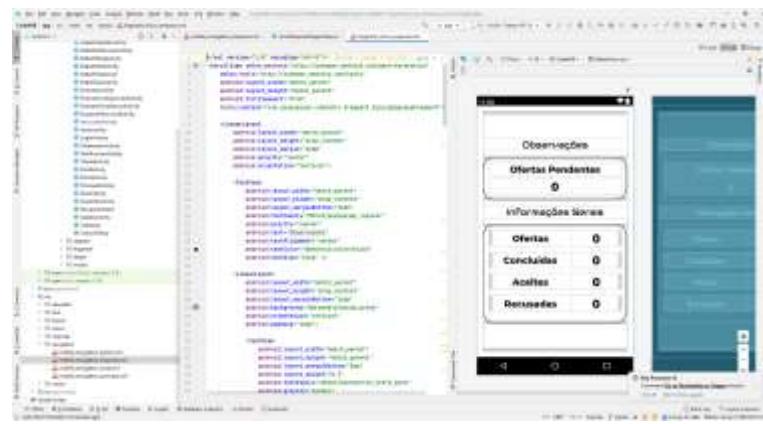


Figura 861: Layout InicioEmpresaFragment

De seguida, passei às configurações iniciais do mesmo.



Figura 862: Configurações Iniciais

Depois, criei o método “buscarInfo()” que é responsável por buscar todas as informações relacionadas com a empresa em si, neste caso das ofertas, tal como visto no layout.

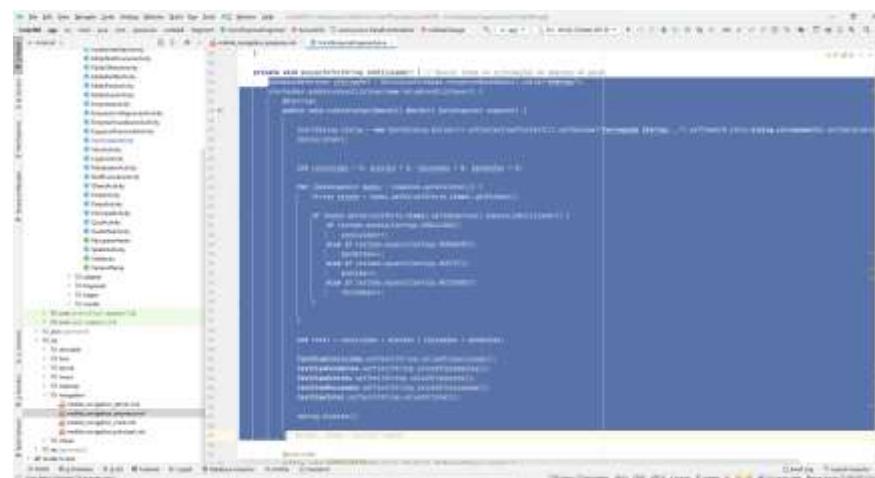


Figura 863: Método "buscarInfo()"

OfertasEmpresaVerificacaoFragment

Na print apresentada abaixo, consegue-se visualizar o layout final deste fragment.



Figura 864: OfertasEmpresaFragment Layout

Após ter finalizado o desenvolvimento do layout, passei às configurações iniciais, neste caso, com a configuração do RecyclerView e do seu Adapter.



Figura 865: Configuração do RecyclerView e do seu Adapter

De seguida, passei à criação do método “buscarOfertas()”, que é responsável por ir buscar todas as ofertas que tenham como estado “aceite”, querendo dizer que o utilizador a aceitou e está à espera agora de um email, por parte da empresa.

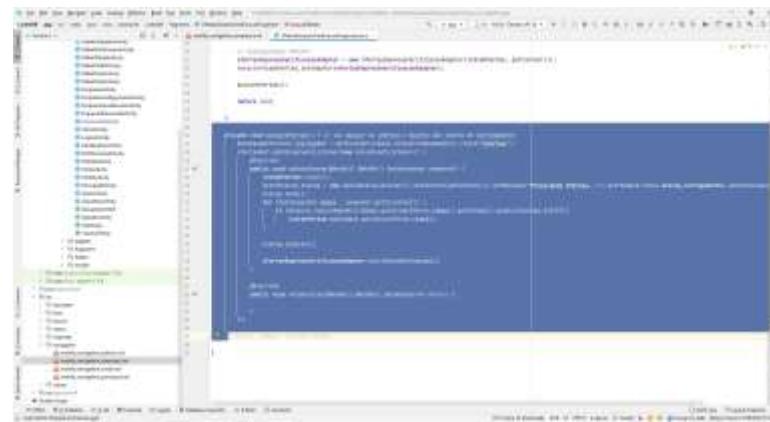


Figura 866: Método "buscarOfertas()"

Depois, passei à criação do adapter deste fragment, começando mais uma vez pelo o layout, uma das minhas partes favoritas.



Figura 867: Layout do OfertasEmpresaVerificacaoAdapter

No método “onBindViewHolder”, são recuperados todos os valores e carregados para dentro do layout anteriormente criado, tal como se observa abaixo.

Deve-se ainda destacar, que a imagem de perfil, tem um evento de clique, que leva para a ContaActivity e passa o id da conta que foi pressionada, para mostrar as informações desse utilizador. Ainda existe outro evento “onClick” do botão que serve para enviar um email ao utilizador.



Figura 868: Método "onBindViewHolder()"

EmpresasFragment

No EmpresasFragment, como realizei nos fragments anteriores, comecei por mais uma vez, configurar o seu layout.

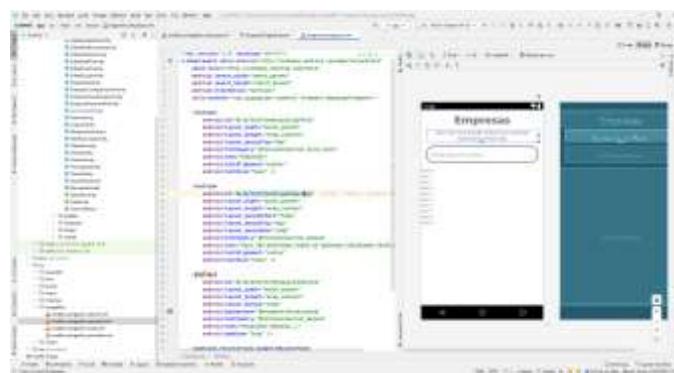


Figura 869: Configuração do Layout EmpresasFragment

Este fragment, à semelhança do anterior também leva um RecyclerView e por si mesmo um Adapter. Mas neste caso, também leva um EditText, que permite pesquisar os elementos do RecyclerView, facilitando assim a sua visualização.

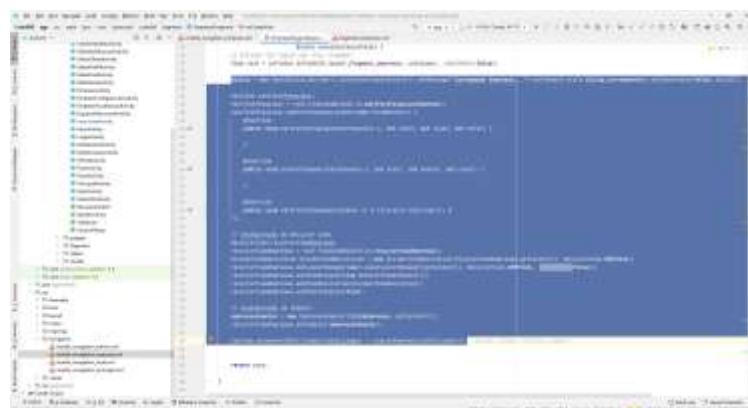


Figura 870: Configurações Iniciais

No EditText, anteriormente referido, reescrevi o método “addTextChangeListener()”, para que quando fosse escrito algo no mesmo, ele executasse de imediato o método “filtrar()”, que filtraria todos os elementos do RecyclerView.

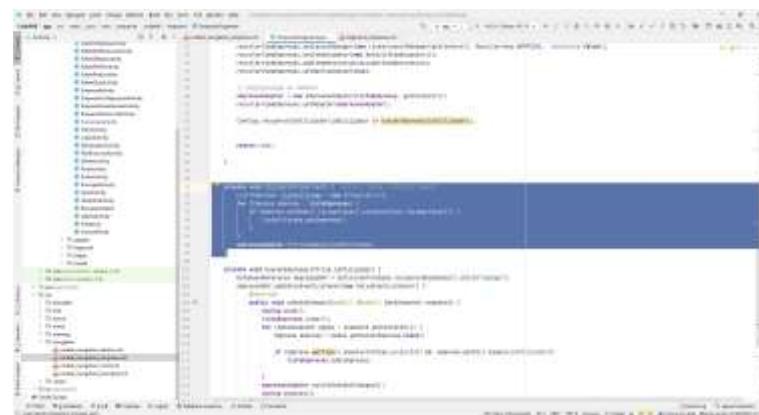


Figura 871: Método "filtrar()"

Seguidamente, escrevi o método “buscarEmpresa()” que recebe como parâmetro o idUtilizador, que é o ID da empresa que se encontra autenticada, evitando que a mesma apareça na listagem do RecyclerView.

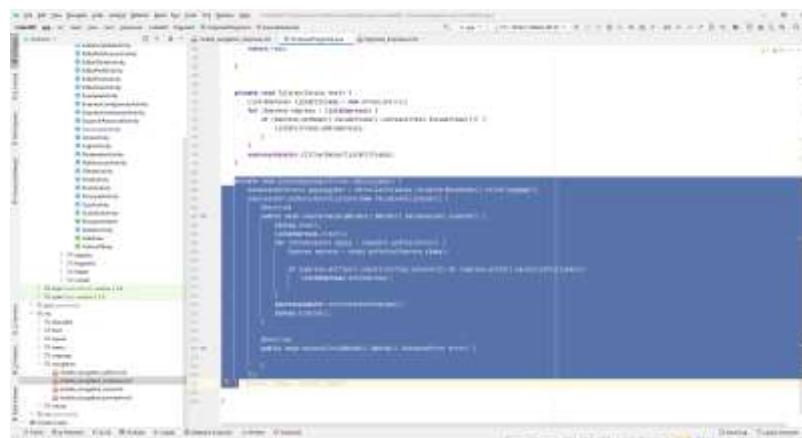


Figura 872: Método "buscarEmpresas()"

Logo depois, construí o layout do “EmpresasAdapter”, que é o adapter deste fragment.

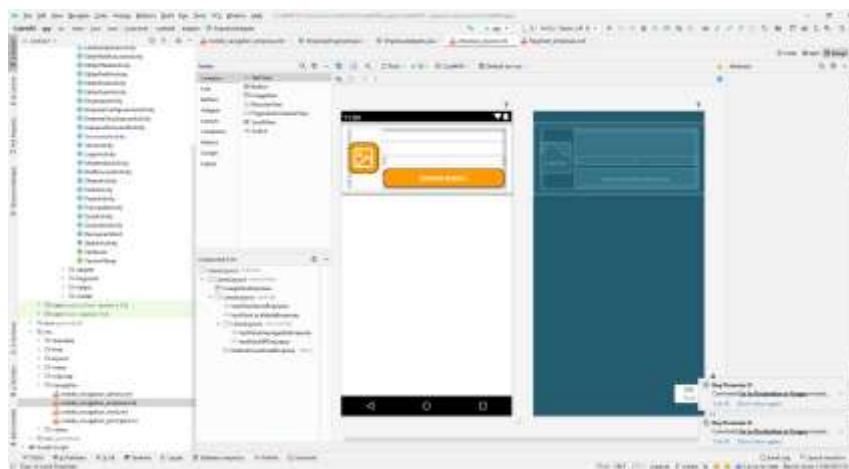


Figura 873: Layout do EmpresasAdapter

No método “onBindViewHolder()”, fiz as associações normais para com o layout e novamente configurei um botão para enviar um email.



Figura 874: Configuração do método "onBindViewHolder"

CriarOfertaFragment

Neste *fragment*, inicialmente, são mostrados todos os utilizadores, ordenados da maior pontuação para aqueles que têm menos, o layout deste *fragment* é o presente na figura.



Figura 875: Layout do CriarOfertaFragment

À semelhança do *EmpresasFragment*, nas configurações iniciais configurei o *RecyclerView*, o seu *adapter* e um *EditText*, relacionado com a filtragem de utilizadores.

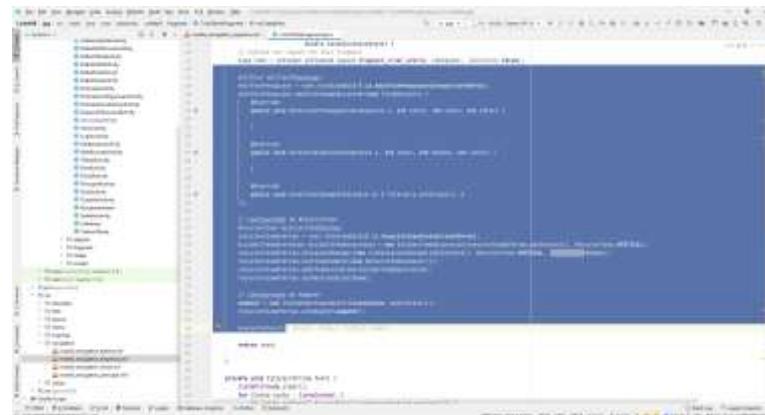


Figura 876: Configurações Iniciais

Copiei o método já realizado no fragment anteriormente referido e adaptei a este e escrevi um método “buscarContas()”, que iria buscar todas as contas nas quais o seu tipo fosse “membro”. Depois, organizei pela a pontuação em decrescente.

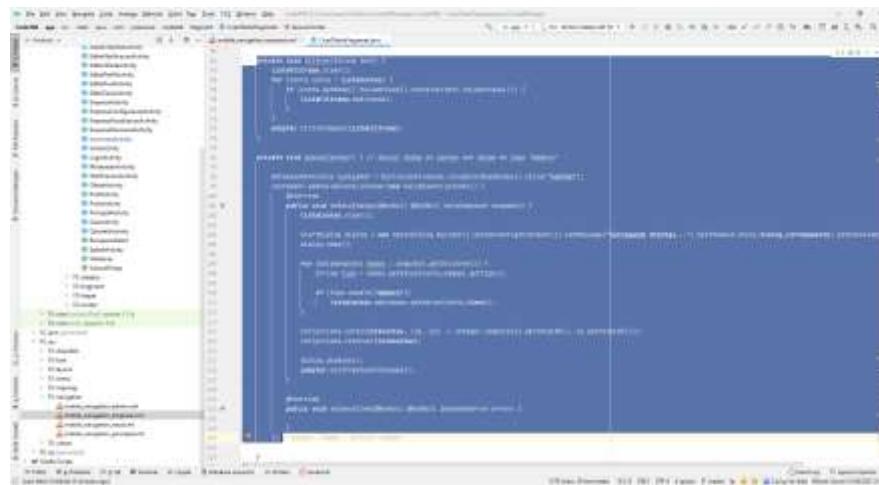


Figura 877: Método "filtrar()" e "buscarContas()"

Seguidamente, criei o layout do ContasEmpresaAdapter, ficando assim.

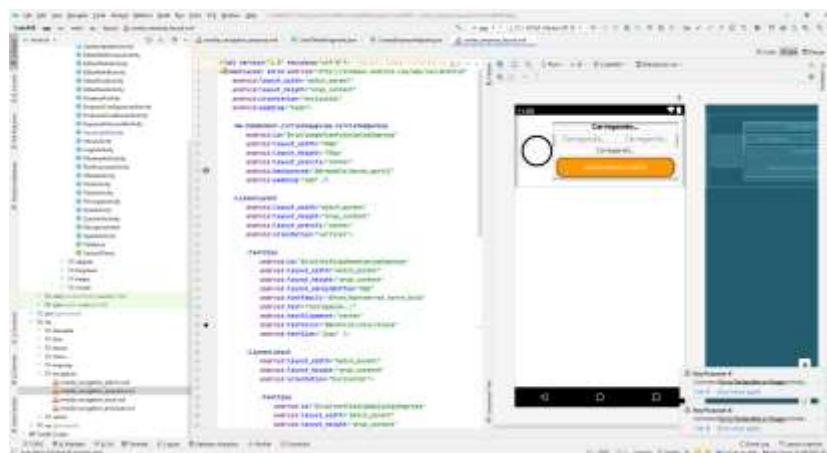


Figura 878: Layout do ContasEmpresaAdapter

Os aspetos mais importantes deste *adapter*, são o facto de ele calcular a idade do seu utilizador automaticamente, tendo em conta a data de nascimento e a data atual, tal como se pode observar na figura. A imagem do utilizador, tem também, à semelhança dos outros adapters, o método “onClick” que abre o perfil do mesmo. E por fim, tem um botão de selecionar que chama, quando clicado, a CriarOfertaActivity.

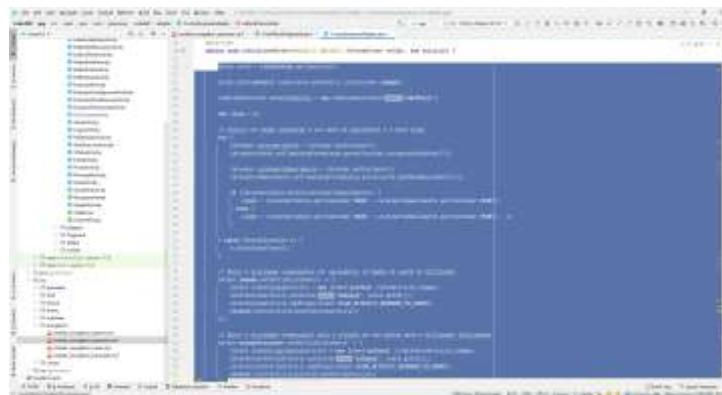


Figura 879: Método "onBindViewHolder"

Quero ressaltar que todos os adapter que contêm uma filtragem de dados, tal como este, precisam obrigatoriamente deste método.

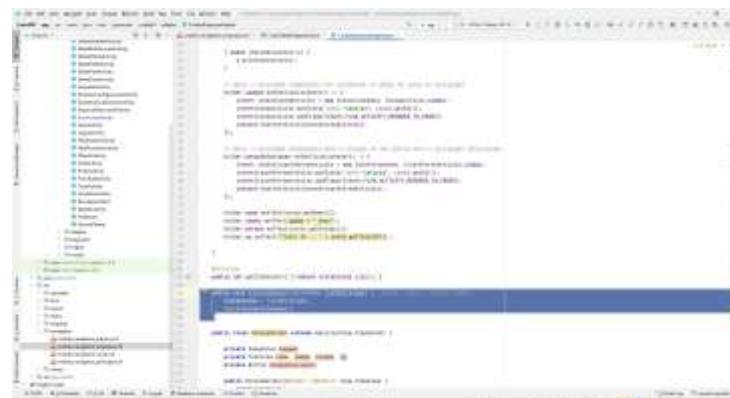


Figura 880: Método "filtrarDados()"

CriarOfertaActivity

Esta atividade, foi criada como complemento à CriarOfertaFragment, visto que só é possível entrar nesta atividade, se previamente um utilizador tenha sido selecionado.



Figura 881: Layout da CriarOfertaActivity

Seguidamente, passei às configurações iniciais desta atividade, tal como se observa na figura abaixo.



Figura 882: Configurações Iniciais

Seguidamente, configurei o botão de criar oferta, onde são verificados todos os campos nomeadamente se estes estão preenchidos, depois recuperei o caminho da base de dados e criei um objeto do tipo “Oferta” atribuindo-lhe todos os campos necessários, o ID desta oferta é gerado automaticamente pela Firebase.



Figura 883: Método "onClick" do botão de criar oferta

Decidi também implementar nesta atividade, um dialog de carregamento para lhe dar um efeito mais interessante e impedir que o utilizador tenha acesso à interface antes desta estar carregada.

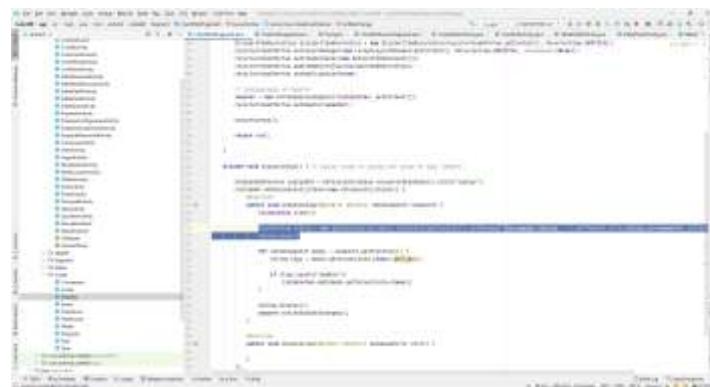


Figura 884: Dialog de Carregamento

ListarOfertasEmpresaFragment

Neste fragment, estão todas as ofertas que a empresa autenticada, já realizou.



Figura 885: Layout ListarOfertasEmpresaFragment

Seguidamente, configurei mais uma vez, os elementos básicos do layout e criei o método “buscarOfertas()” que vai buscar apenas as ofertas da empresa.

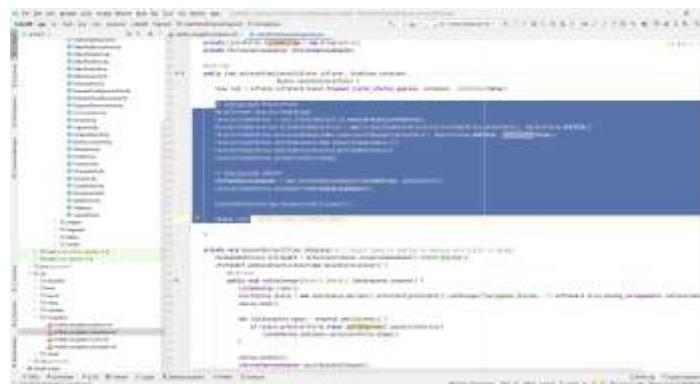


Figura 886: Configurações Iniciais

Seguidamente, configurei e codifiquei o layout do OfertasEmpresaAdapter, que ficou da seguinte forma.



Figura 887: Layout do OfertasEmpresaAdapter

No método “onBindViewHolder”, realizei a configuração normal do layout, colocando os dados certos em cada elemento, mais uma vez adicionei a ação de quando a imagem de perfil é clicada, uma atividade com os dados do mesmo é aberta. E neste caso, visto que as ofertas têm vários estados, decidi colocar uma cor para cada um deles. Se for aceite fica a verde, se for pendente fica a amarelo, se for recusado fica a vermelho e se for concluído, aparece a laranja, tal como se pode observar na figura.



Figura 888: Método "onBindViewHolder()"

No layout anteriormente mostrado, existiam ainda dois botões incorporados dentro de um *LinearLayout*. No restante código deste método, decidi incluir a função de editar e remover a oferta, fazendo referência aos tais dois botões que se apresentavam como “gone” dentro do *LinearLayout*. Isto apenas acontece, quando e só quando, a oferta fosse “pendente”.

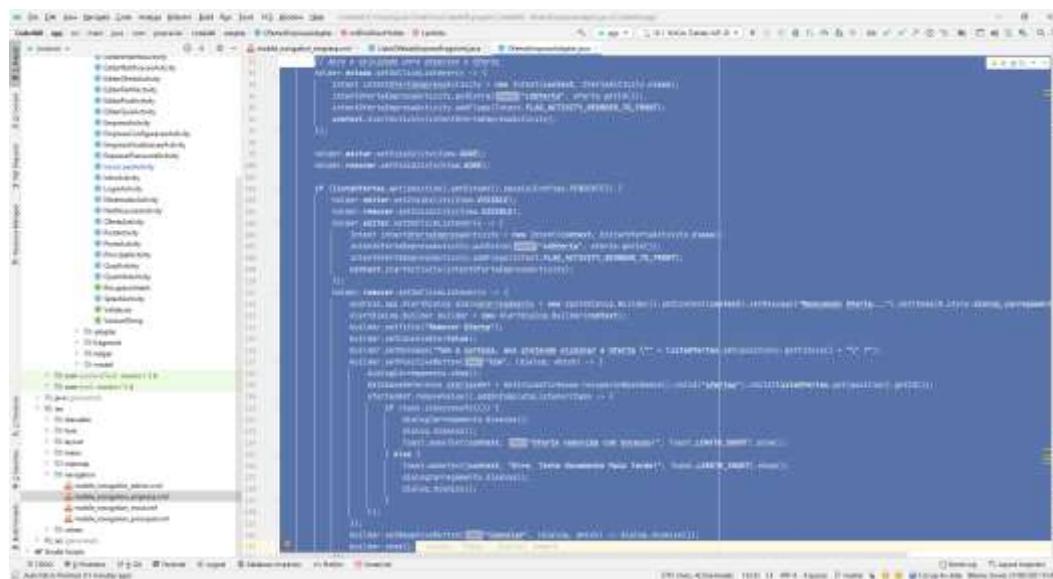


Figura 889: Função de remover/editar a oferta

OfertaActivity

Com a listagem das ofertas da empresa faltaria uma atividade para mostrar os dados dessa atividade e foi o que eu realizei.

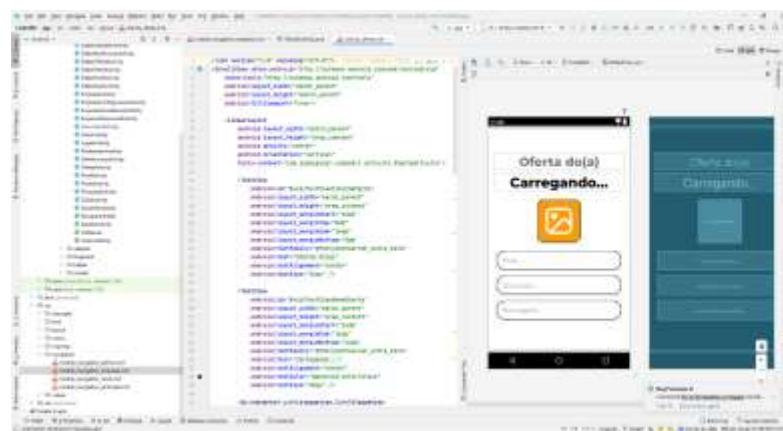


Figura 890: Layout da OfertaActivity

Depois de ter configurado o layout corretamente, passei à programação da atividade em si, que é bastante simples, mas uma vez tinha que armazenar o id da oferta, passado no início desta atividade e fazer uma pesquisa pelo os dados e mostrar os mesmos dentro dos elementos corretos.

Para dar um efeito mais interessante a esta atividade, decidi também colocar um *dialog* de carregamento, para que o utilizador tivesse apenas acesso a todos os dados quando estes fossem totalmente carregados.

A captura de ecrã encontra-se na próxima página.

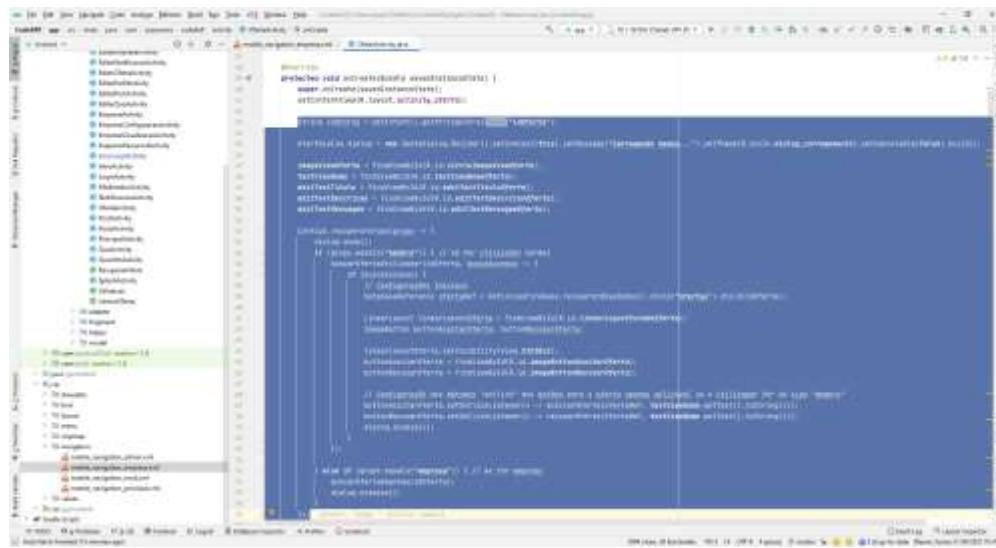


Figura 891: Configurações iniciais da atividade

EditarOfertaActivity

Depois de ter codificado a atividade de visualização de uma oferta, faltava agora configurar a atividade de edição da mesma, comecei pelo desenvolvimento do layout desta atividade



Figura 892: Layout da *EditarOfertaActivity*

Com o layout devidamente configurado e ajustado, passei à realização da configuração dos elementos básicos de operação deste layout.

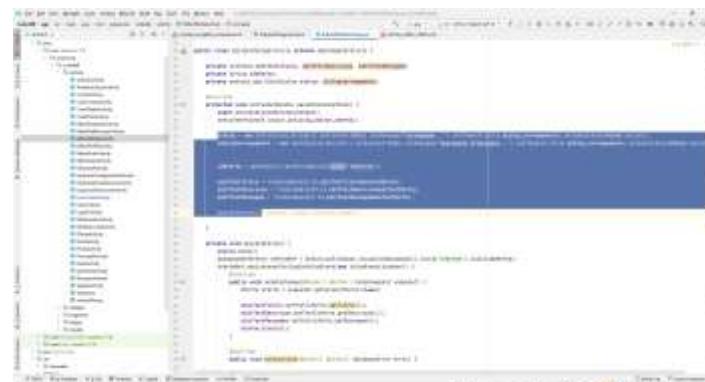


Figura 893: Configurações Iniciais

Por fim, fiz a configuração do botão de confirmar alteração, mais concretamente do método “onClick” do mesmo. Fazendo como das outras vezes, primeiro a verificação dos campos e depois disso sim, coloca os dados num HashMap e envia para a Firebase.

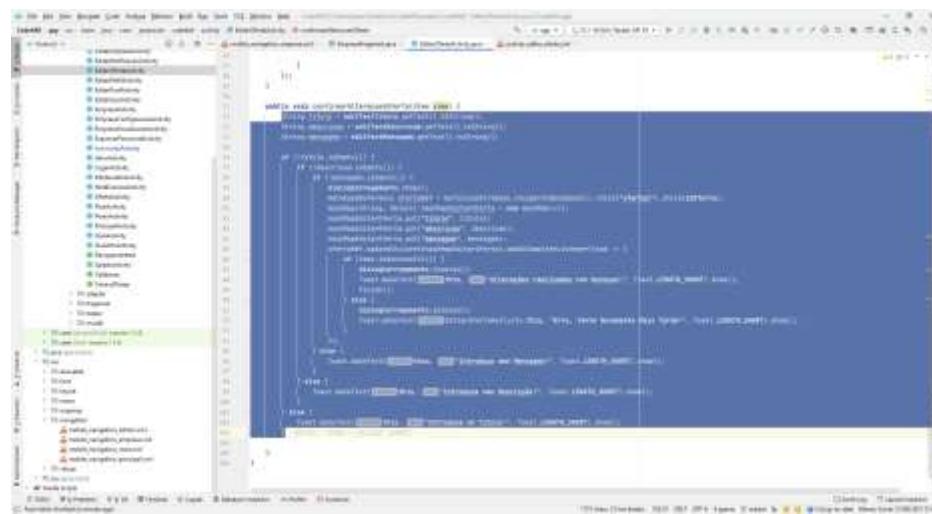


Figura 894: Configuração do botão de confirmar alterações

EmpresaVisualizacaoActivity

Criei esta atividade, para mostrar todas as empresas (individualmente), o layout desta atividade encontra-se na seguinte figura:

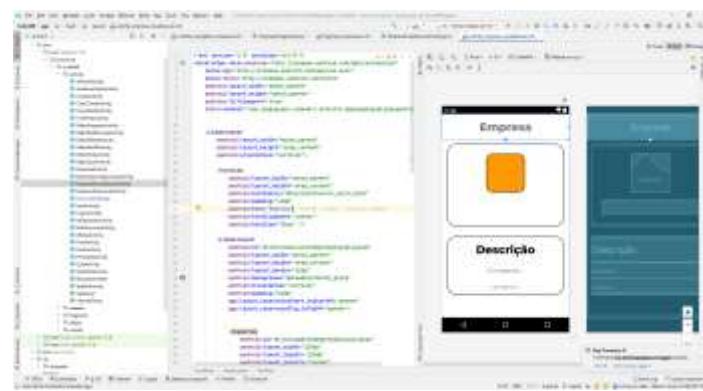


Figura 895: Layout da EmpresaVisualizacaoActivity

Seguidamente, realizei as configurações iniciais da atividade, começando pelo elemento mais importante, que é o id passado no início da mesma, pois é baseado nesse id que vão ser buscados os dados da empresa a ser visualizada.

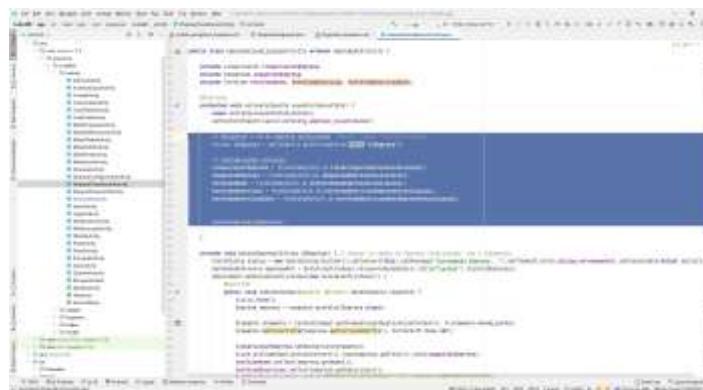


Figura 896: Configurações iniciais e o método "buscarEmpresa()"

Publicação da App (Google Play Store)

Devido ao facto de ter colocado (inicialmente), o nome do pacote como “com.android.code4all”, isso gerou-me problemas ao enviar a app para a play store, visto que o nome de pacote assim, não é autorizado.



Figura 897: Tentativa falhada de envio da app

Logo fui obrigado a renomear o pacote para “com.josexavier.code4all”, como se observa na figura.



Figura 898: Modificação do nome do pacote

Visto que mudei o nome de pacote tive de refazer as configurações da firebase.



Figura 899: Reconfiguração da Firebase

Depois de realizar a reconfiguração da FireBase voltei à configuração da app, para enviar para a Google Play Store

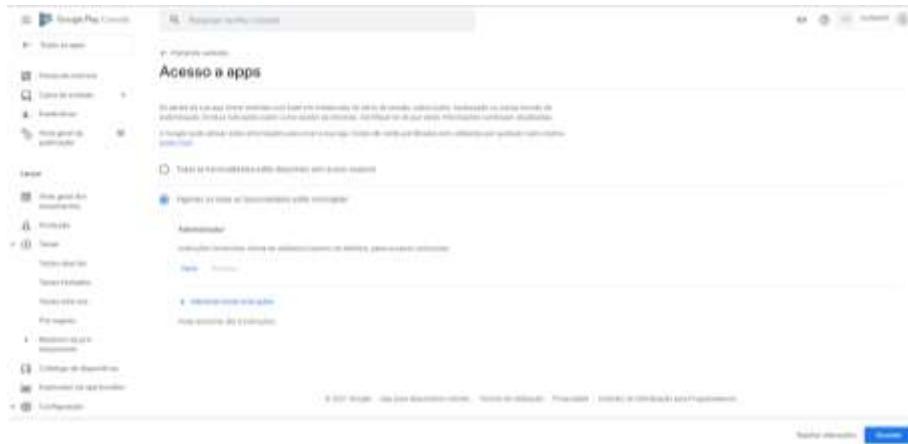


Figura 900: Configuração do Acesso a apps

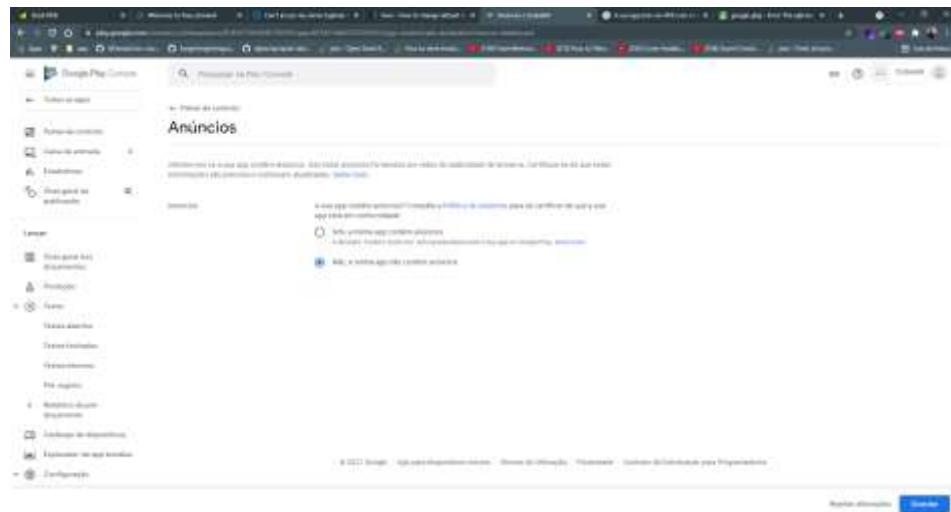


Figura 901: Configuração dos Anúncios

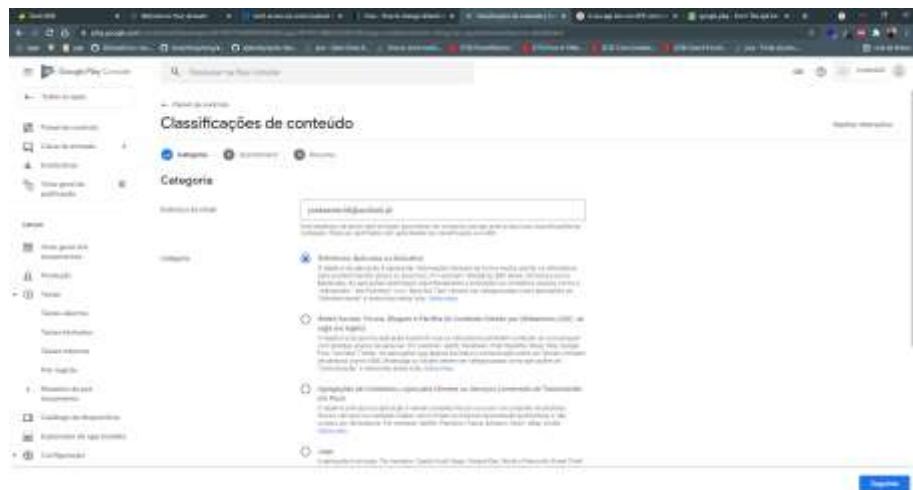


Figura 902: Classificações de Conteúdo - Categoria



Figura 903: Classificações de Conteúdo – Questionário



Figura 904: Classificações de Conteúdo – Resumo



Figura 905: Público-alvo e conteúdo - Faixa etária de segmentação



Figura 906: Público-alvo e conteúdo – Presença na Google Play Store



Figura 907: Público-alvo e conteúdo - Resumo



Figura 908: Apps de notícias



Figura 909: Definições da Loja

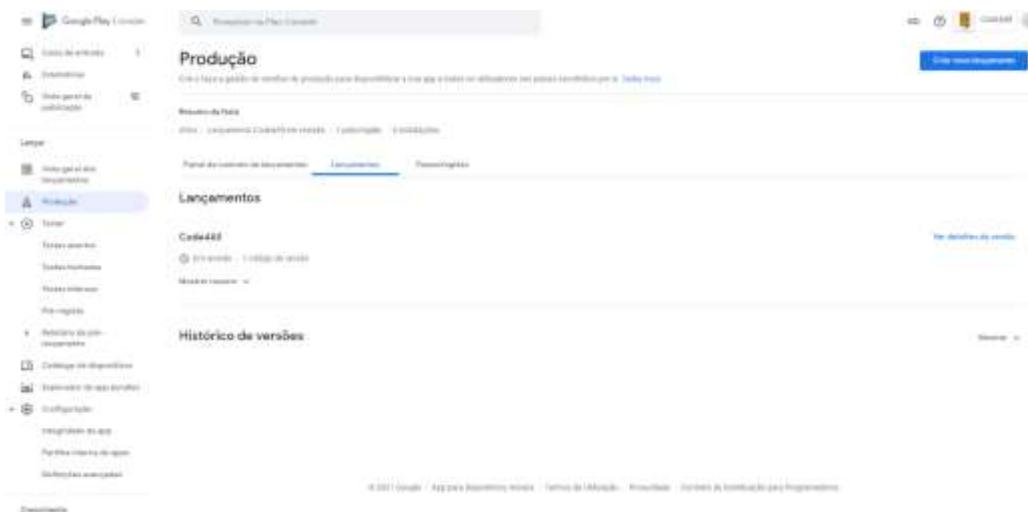


Figura 910: Lançamento da App - Em revisão

Criação do repositório Code4All (GitHub)

Como prometido, quando cheguei à versão 1.0, submeti a mesma para o GitHub, pois este projeto é totalmente *open-source*.



Figura 911: Code4All - Repositório GitHub

<https://github.com/rednexx46/Code4All>

Conclusão

O meu projeto consistiu em desenvolver uma aplicação para Android, para conseguir concluir este projeto foi preciso ter uma grande capacidade de autonomia e persistência, aspectos esses que se foram desenvolvendo à medida que este projeto era concebido.

A realização deste projeto fez-me solidificar uma série de conhecimentos, foi uma experiência diferente de todo o desenvolvimento do curso.

Pela primeira vez tive a experiência de elaborar um projeto de raiz com o objetivo de o tornar possível de realizar fazendo uma pequena amostra disso mesmo, preparando tudo para a realização.

Foi, sem dúvida, uma boa experiência ter elaborado um projeto de tão grande importância para mim, pois permitiu-me melhorar o meu conhecimento sobre como começar um projeto do zero e os seus passos do início até ao fim.

Com este projeto aprendi várias lógicas de programação, que antes da sua realização, não sabia, ou seja, solidifiquei ainda mais a minha base de conhecimentos de programação, um dos meus objetivos propostos no início deste trabalho.

Confesso, que no início não estava à espera que esta prova tomasse proporções tão grandes, mas ainda bem que assim foi, aprendi bastante e espero levar este projeto para a frente, uma vez que realmente tem importância para todas as pessoas que pretendem começar no mundo da informática (programação), visto que isto é o futuro.

Para finalizar, graças ao programa Cloc aka “Counts Lines of Code” realizado pelo o “AlDanial”, pude contabilizar quantas linhas de código realizei.

```
C:\Users\josesv\OneDrive\Ambiente de Trabalho\Cloc c:\users\josesv\desktop\códigos\projeto\Code4All\app\src\main\java\com\josesv\code4all  
85 text files.  
85 unique files.  
8 files ignored.  
github.com/AlDanial/cloc v 1.80 T=1.07 s (89.6 files/s, 11299.3 lines/s)  
Language           Files      blank     comment      code  
Java              89       2479      135      9432  
SUM:              89       2479      135      9432  
  
C:\Users\josesv\OneDrive\Ambiente de Trabalho\Cloc c:\users\josesv\desktop\códigos\projeto\Code4All\app\src\main\resources\lipmd  
85 text files.  
85 unique files.  
3 files ignored.  
github.com/AlDanial/cloc v 1.80 T=0.53 s (161.8 files/s, 14729.3 lines/s)  
Language           Files      blank     comment      code  
XML               85       039       0       6985  
SUM:              85       039       0       6985  
  
C:\Users\josesv\OneDrive\Ambiente de Trabalho
```

Figura 912: Linhas de código escritas (JAVA + XML)

É com o maior orgulho possível que fecho este projeto, que me deu muitas dores de cabeça e noites sem dormir, mas foi com ele que mais aprendi.

Bibliografia

<https://developer.android.com/>
<https://firebase.google.com/docs/>
<https://developer.android.com/studio/releases/platform-tools/>
<https://developer.android.com/studio/command-line/adb/>
<https://romannurik.github.io/AndroidAssetStudio/>
<https://fonts.google.com/>
<https://developer.mozilla.org/pt-BR/>
<https://developer.android.com/guide/components/activities/activity-lifecycle/>
<https://github.com/heinrichreimer/material-intro/>
<https://pt.wikipedia.org>
<https://vanya.jp.net/vtree/>
<https://github.com/Genymobile/scrcpy>
<https://github.com/dybarsky/spots-dialog>
<https://github.com/AIDanial/cloc>
<https://play.google.com/console/u/0/developers>

Anexos

**FICHA 1 - PAP
ANTE-PROJETO**

Curso: Técnico de Gestão e Programação de Sistemas Informáticos		
Triénio de Formação: 2018/2021	Ano letivo: 2019/2020	Data: 21/09/20
Aluno: José Xavier Nabão	N.º 16	Turma TGPSI

Tema proposto
O tema que proponho para a minha Prova de Aptidão Profissional é uma aplicação e decidi optar pelo nome "Code4All".

Fundamentação da escolha
Escolhi este projeto pois, recentemente, instalei uma aplicação no meu dispositivo Android nomeada de "Mimo", que consiste em "quizes" de várias linguagens de programação, mas infelizmente esta App tem "in-app purchases", ou seja, para vermos todos os conteúdos que a aplicação nos disponibiliza temos de os comprar.
Com isto, queria fazer uma aplicação semelhante, mas com a diferença de ser totalmente grátis, e assim fazer com que todas as pessoas tenham acesso à mesma e que aprendam a programar com apenas alguns cliques no seu dispositivo Android.
E de certa forma, a função principal da PAP é reunir os conhecimentos obtidos durante os 3 anos de curso, e com isso, gostaria de implementar nesta aplicação em Android, os conhecimentos, os truques e dicas, que fui adquirindo/aprendendo, com a realização deste curso e não só, mas também nos cursos exteriores que realizei, como por exemplo na plataforma Udemy.
Falando da Udemy, estou atualmente a realizar um curso de desenvolvimento de aplicações para Android, para aperfeiçoar os meus conhecimentos sobre a programação para dispositivos móveis e, como é óbvio, paguei pelo curso para o ter e, quando o completar, irei receber um certificado de conclusão.
Com a realização deste projeto, gostaria de aprofundar a área do desenvolvimento para dispositivos móveis, pois ao fazê-lo irei adquirir e consolidar bastantes conhecimentos nesta mesma área, e com isso evoluir neste ramo da programação.
As disciplinas que pretendo aprofundar no desenvolvimento deste projeto são de certa forma, a disciplina de PSI (Programação e Sistemas de Informação) como é óbvio para o desenvolvimento da minha aplicação, a disciplina de Inglês para melhorar de certa forma a minha fala e escrita.

Figura 913: Ficha 1 - Parte 1



FICHA 1 - PAP
ANTE-PROJETO

desta língua, já que me vai ser necessária nas mais variadas situações, nomeadamente vou precisar dela para quando estiver a traduzir a minha aplicação, ou simplesmente, quando estiver a colocar os "mini-cursos" na aplicação e, por último, a disciplina de RC (Redes de Comunicação), visto que vou fazer também uns "mini-cursos" sobre grande parte da matéria que demos em RC, como por exemplo HTML, CSS, entre outras...

Por fim, uma das outras razões para escolher este projeto, é eu ter o objetivo pessoal de fazer algo que de certa forma influencie a comunidade, neste caso gostaria que as pessoas conseguissem, através de uma aplicação, aprender como programar de uma forma intuitiva, pois a programação em si, não é como o que muitos dizem um "bicho de 7 cabeças", e deixar com que as pessoas tenham essa ideia sobre a programação e realmente vejam, o quanto divertido é programar!

Descrição sumária

Decidi o nome "Code4All" como o nome do projeto devido à junção de "Code" que significa programação em Inglês, com "4All" que significa para todos. No final, a tradução quer dizer "Programação para Todos". Após alguns BrainStorms que realizei, cheguei a este nome que para mim faz totalmente sentido, neste projeto e não só, mas acho que é um nome bastante moderno. O modo de funcionamento da aplicação que pretendo obter será o seguinte: primeiro o utilizador terá de se registrar, depois realizar o login, e aí terá acesso logo, aos "mini-cursos"; claro que primeiro tem de começar pelos cursos mais básicos e à medida que os completa pode avançar para cursos mais difíceis. A aplicação incluirá, ainda, um grau de dificuldade e, assim, antes do utilizador iniciar o curso terá uma noção da dificuldade do mesmo, para fazer a sua análise e escolhas antes de avançar.

Eu próprio farei "minicursos" e desafios de HTML, CSS, etc e, deste modo, o utilizador poderá clicar em algum dos disponíveis, e começar a aprender a programar de uma forma intuitiva, à medida que vai concluindo o curso e os seus desafios, vai ganhando experiência e moedas, que depois poderão ser usadas para comprar outros "minicursos" e/ou desafios interessantes. Com esta experiência também vai existir um sistema de níveis e ranking entre utilizadores que utilizam a aplicação. Desta forma, irá gerar um interesse e um certo despike entre utilizadores, que levará a que todos tenham interesse em aprender os vários "minicursos" que irei disponibilizar na aplicação.

Para completar este sistema de ranking, gostava de colocar uma "leaderboard", para que todos os utilizadores vissem o seu ranking na aplicação.

Cofinanciado por:
   

201

Figura 914: Ficha 1 - Parte 2



FICHA 1 - PAP
ANTE-PROJETO

As tecnologias que pretendo utilizar são: O Android Studio para programar a aplicação em si, e como é óbvio vou programar a aplicação 100% em Java. Vou também precisar da preciosa ajuda do Firebase da Google, que me vai permitir fazer o código da aplicação de uma forma muito mais rápida e eficaz, o registo, o login, o nível de experiência, as moedas que cada utilizador tem, os cursos que cada utilizador já tem feito e os seus progressos, vão ser todos tratados pelo Firebase, com ajuda da sua documentação e da minha codificação, claro.

No entanto, para diferenciar a aplicação da de inspiração "Mimo", quero fazer a minha aplicação totalmente gráts e que, no final de cada desafio, esta emita um certificado de realização daquele "minicurso" para o utilizador que acabou de o realizar o mesmo e, quem sabe, também gostaria de implementar cursos que a comunidade poderá sugerir e fazer. Primeiro teriam de ser aprovados por mim, como é óbvio, mas depois seriam disponibilizados aos utilizadores, mas como cursos da comunidade, não cursos realizados por mim.

Para completar tudo, gostava de colocar também um fórum de perguntas e respostas, caso os utilizadores encontrem alguma dúvida na explicação/questão colocada na aplicação.

Nisto tudo, ao fazer esta aplicação estou a fazer dois trabalhos ao mesmo tempo, a melhorar a minha experiência com o desenvolvimento de aplicações em dispositivos móveis, e não só, mas também, a melhorar os meus conhecimentos, porque ao fazer os "minicursos" estarei a rever a matéria que dei durante estes 3 anos de curso.

Com isto quero fazer com que todas as pessoas saibam o mínimo de programação e que em qualquer situação, no seu quotidiano, consigam resolver problemas simples de programação.

O aluno: José Xavier Nabão

O Diretor de Curso:

Cofinanciado por:



33

Figura 915: Ficha 1 - Parte 3



Figura 916: Certificado de conclusão - Curso Android

Glossário

- IDE: Do inglês Integrated Development Environment ou Ambiente de Desenvolvimento Integrado, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.
- SDK: Kit de desenvolvimento de software, também conhecido como Software development kit, SDK ou "devkit", é tipicamente um conjunto de ferramentas de desenvolvimento de software que permite a criação de aplicativos para um certo pacote de software, framework, plataforma de hardware, sistema de computador, console de videogame, sistema operacional, ou plataforma de desenvolvimento similar. Para criar aplicativos, deve-se utilizar um kit de desenvolvimento de software específico, sendo geralmente acompanhadas de um Ambiente de desenvolvimento integrado.
- Android: É um sistema operacional baseado no núcleo Linux, desenvolvido por um consorcio de desenvolvedores conhecido como Open Handset Alliance, sendo o principal colaborador o Google.
- GPU: GPU (Graphics Processing Unit, ou Unidade de Processamento Gráfico), conhecido também como VPU ou unidade de processamento visual, é um tipo de microprocessador especializado em processar gráficos em computadores pessoais, estações de trabalho ou consolas de jogos.
- Commit: No contexto de ciência da computação e gerenciamento de dados, commit refere-se à ideia de fazer permanentes um conjunto de mudanças experimentais. Uma utilização popular está no fim de uma transação. Um commit é o ato de enviar, ou seja, fazer um upload dos códigos para um banco de dados.

- XML: XML, do inglês eXtensible Markup Language, é uma linguagem de marcação recomendada pela W3C para a criação de documentos com dados organizados hierarquicamente, tais como textos, banco de dados ou desenhos vetoriais. A linguagem XML é classificada como extensível porque permite definir os elementos de marcação.
- GIF: Graphics Interchange Format é um formato de imagem de bitmap que foi desenvolvido por uma equipe do provedor de serviços on-line CompuServe, liderado pelo cientista de computação americano Steve Wilhite em 15 de junho de 1987. Desde então, tem vindo a ser amplamente utilizado na Internet devido ao seu amplo suporte e portabilidade entre muitas aplicações e sistemas operacionais.
- UI: A interface do utilizador, no campo de desenho industrial da interação homem-máquina, é o espaço onde a interação entre humanos e máquinas ocorre.
- SP: (Scale-independent Pixels) é considerada como o tamanho da fonte que está a ser utilizado. É recomendado que essa unidade seja usada para especificar o tamanho de uma fonte, para que esta seja automaticamente ajustada conforme as preferências da tela do usuário.
- DP: (Density-independent Pixels) Esta unidade é relativa à resolução do ecrã. Por exemplo se a resolução da tela é de 160 dpi, significa que um dp representa 1 pixel em um total de 160.

- API: Interface de Programação de Aplicações (português), cuja sigla API provém do Inglês Application Programming Interface, é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.
- DPI: Pontos por polegada (ppp; em inglês dots per inch, dpi) é uma medida de densidade relacionada à composição de imagens, que expressa o número de pontos individuais que existem em uma polegada linear na superfície onde a imagem é apresentada. Também é comum encontrar referências a essa densidade pelo termo "resolução de imagem" ou simplesmente "resolução". De maneira geral, quanto maior o número de pontos por polegada, mais detalhada e bem definida é a imagem.
- Commit: Refere-se à ideia de fazer permanentes um conjunto de mudanças experimentais. Uma utilização popular está no fim de uma transação. Um commit é o ato de enviar, ou seja, fazer um upload dos códigos para um banco de dados.
- SHA-1: É uma função de dispersão criptográfica (ou função hash criptográfica) projetada pela Agência de Segurança Nacional dos Estados Unidos e é um Padrão Federal de Processamento de Informação dos Estados Unidos publicado pelo Instituto Nacional de Padrões e Tecnologia.

- **SVG:** É a abreviatura de Scalable Vector Graphics que pode ser traduzido do inglês como gráficos vetoriais escalonáveis. Trata-se de uma linguagem XML para descrever de forma vetorial desenhos e gráficos bidimensionais, quer de forma estática, quer dinâmica ou animada. Uma das principais características dos gráficos vetoriais, é que não perdem qualidade ao serem ampliados. A grande diferença entre o SVG e outros formatos vetoriais, é o fato de ser um formato aberto, não sendo propriedade de nenhuma empresa. Foi criado pela World Wide Web Consortium, responsável pela definição de outros padrões, como o HTML e o XHTML.
- **PNG:** (Portable Network Graphics) é um formato de dados utilizado para imagens, que surgiu em 1996 como substituto para o formato GIF, devido ao facto de este último incluir algoritmos patenteados. Esse formato livre é recomendado pela W3C, suporta canal alfa, tem uma maior gama de profundidade de cores, alta compressão (regulável), além de outras características.
- **AKA:** A expressão aka é considerada uma sigla da língua inglesa. A estrutura é um vocábulo que se originou das letras ou sílabas iniciais de outra palavra. Assim, a expressão aka é, na verdade, a abreviação das palavras da expressão also known as, cuja tradução para o português seria “também conhecido como” ou “vulgo”.