



Instituto Politécnico de Beja

Escola Superior de Tecnologia e Gestão
Curso de Tecnologias web e Dispositivos Móveis

Programação de Aplicação ao lado do Cliente

**Trabalho de Grupo
“Projeto”**

**MediMate
Elaborado por:**

Roberto López – 26064
Manuel Machado – 25898
Gonçalo Rosário – 24293

Índice de conteúdos

Introdução.....	4
Trabalho em equipa.....	5
Organização do trabalho.....	6
Tema do Projeto.....	7
Ferramentas Utilizadas.....	8
Etapas da Realização.....	9
Dificuldades na Realização no projeto.....	10
Funcionalidades da Aplicação.....	11
Ecrãs e funcionalidades da aplicação.....	12
Ecrã nº1.....	12
Ecrã nº2.....	13
Ecrã nº3.....	14
Ecrã nº4.....	15
Ecrã nº5.....	16
Ecrã nº6.....	17
Ecrã nº7.....	18
Ecrã nº7.....	19
Ecrã nº7.....	20
Ecrã nº8.....	21
Ecrã nº9.....	22
Desenho da Base de Dados.....	23
Conceção da Base Dados.....	23
Programação da Lógica da Base de Dados.....	24
Conceções do Kotlin.....	25
Ilustração do android studio.....	27
Programação da lógica da aplicação e Base de Dados.....	27
MainActivity.kt.....	27
Repository.kt.....	28
MedicamentoViewmodel.kt.....	29

Outros Aspetos.....	30.
Página web da aplicação.....	32
Página Web.....	32
Conclusão.....	34

Introdução

- Muitas pessoas especialmente idosos, pacientes crônicos e cuidadores, precisam de uma maneira simples e eficiente de gerenciar os seus medicamentos diários, para tal criamos uma aplicação que se aplica a estas necessidades a MediMate.
- Para o seu desenvolvimento foi utilizado a linguagem de programação kotlin e a base de dados sendo utilizada para guardar medicações, horários, etc.
- Ao longo deste relatório serão abordados os principais objetivos do projeto, o processo de desenvolvimento, os desafios enfrentados e as soluções encontradas.

Trabalho em equipa

No nosso trabalho, o trabalho em equipa foi caracterizado por uma abordagem igualitária, onde cada membro realizou as tarefas que lhe foram assignadas:

Manuel Machado

Roberto López

Gonçalo Rosário

Organização do trabalho

Foi criado um repositório no GitHub para o armazenamento e partilha dos documentos produzidos ao longo do projeto nomeadamente o relatório, que tem a seguinte estrutura:

1. Introdução;
2. Tema do projeto;
3. Trabalho em equipa;
4. Dificuldades na realização do projeto;
5. Funcionalidades da aplicação;
6. Desenho da interface da aplicação;
7. Desenho da base de dados;
8. Conceção da base de dados;
9. Programação da lógica da aplicação;
10. Testes com a aplicação;
11. Página web da aplicação e screencast;
12. Conclusão;

Tema do Projeto

A aplicação de gestão de medicamentos (MediMate) é projetada para facilitar a administração de dados dos pacientes de maneira organizada e eficiente. O sistema integrará várias funcionalidades para permitir que os utilizadores, possam acessar, atualizar e gerenciar informações sobre os seus medicamentos

Ferramentas Utilizadas

Para a realização deste projeto foi usado o Android studio tanto para a criação da aplicação como também para a Base de dados, o vs code e o bootstrap para o desenvolvimento do site da aplicação.

Etapas da Realização

Tarefa 1- Criação do site em Vs Code usando, html, css, java Script e json;

Tarefa 2- Criação da aplicação no Android Studio usando a linguagem de programação kotlin;

Tarefa 3 – Criação e ligação da base de dados com a aplicação.

Dificuldades na Realização do Projeto

A maior dificuldade foi a criação e ligação da base de dados com a aplicação usando o android studio.

Funcionalidades da Aplicação

As funcionalidades da aplicação englobam:

- Login com Firebase;
- Adicionar Medicamento;
- Recordatório Diário;
- Alarme;
- Editar Medicamento.

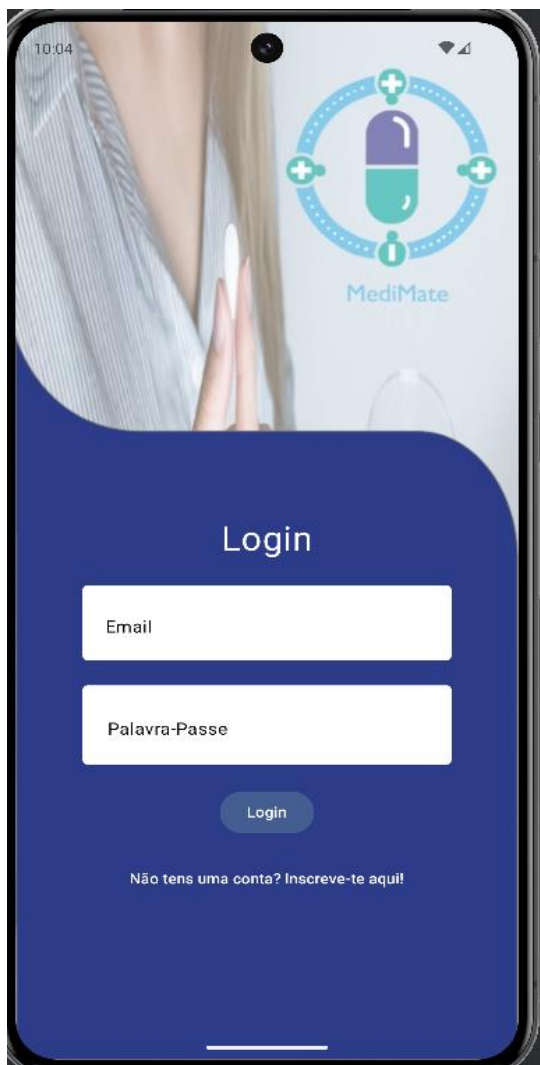
Ecrãs e funcionalidades da aplicação

Com base no código realizado chegámos ao protótipo final com diversos ecrãs diferentes e cada um com a sua funcionalidade.

Ecrãs nº1



1ºEcrã da aplicação que apresenta uma imagem de fundo com o logo da aplicação, no meio uma pequena mensagem para atrair o utilizador e finalmente na parte inferior um TextButton cuja função é dirigir ao utilizador para o Ecrã de Login.

Ecrã nº2:

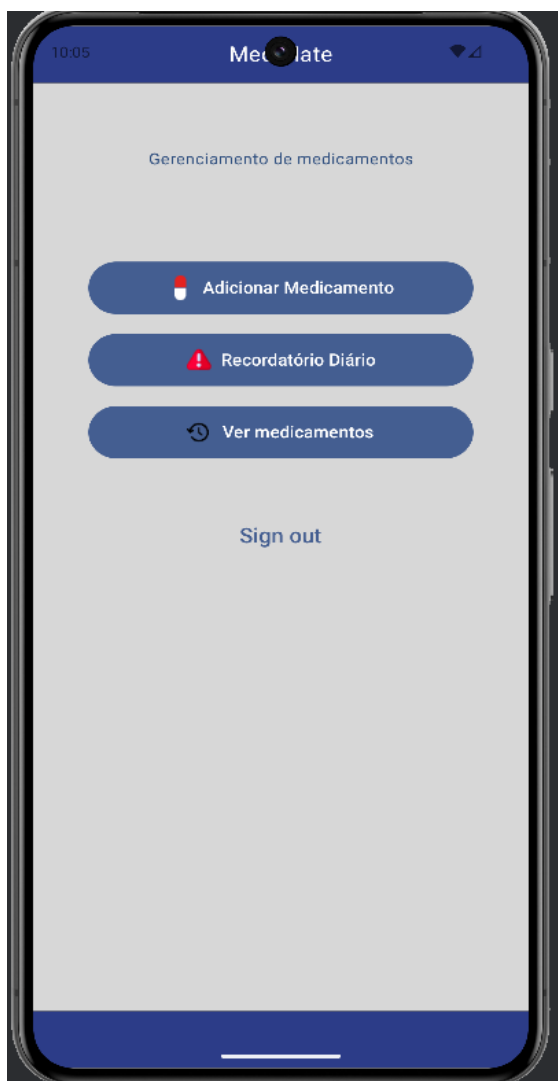
2º Ecrã da aplicação que apresenta características semelhantes ao primeiro ecrã com a adição de caixas de texto onde o utilizador pode fazer login na aplicação, estes dados são guardados no Firebase, se o utilizador não tem conta não consegue fazer login. No caso de não ter conta podemos carregar no texto “Não tens uma conta? Inscreve-te aqui” assim o utilizador é dirigido para o terceiro ecrã.

Ecrã nº3



3º Ecrã da aplicação possui um desenho semelhante ao primeiro ecrã, neste ecrã a diferença do segundo o utilizador pode criar uma conta, estes dados são guardados no Firebase e ao clicar no botão “Criar Conta” o utilizador dirige-se ao menu principal, no caso de se ter enganado e ter navegado para este ecrã pode carregar no TextButton “Já tens uma conta? Login” e volta para o ecrã de Login.

Ecrã nº4



4º Ecrã da aplicação onde o utilizador tem 4 opções:

- Adicionar Medicamento;
- Recordatório Diário;
- Ver medicamentos;
- Sign out.

Ecrã nº 5

10:06 Medicate

Adicionar Medicamento

Nome do Medicamento

ben-nu-ron

Dose

500mg

Frequência

4

Medicamentos

5º Ecrã da aplicação, se o utilizador selecionar no menu principal a opção de Adicionar Medicamento, o utilizador dirige-se para este ecrã onde pode introduzir um novo medicamento, pode adicionar:

- Nome do Medicamento;
- Dose;
- Frequência.

Ao clicar no botão o utilizador dirige-se para o próximo ecrã e esta informação é adicionada na base de dados

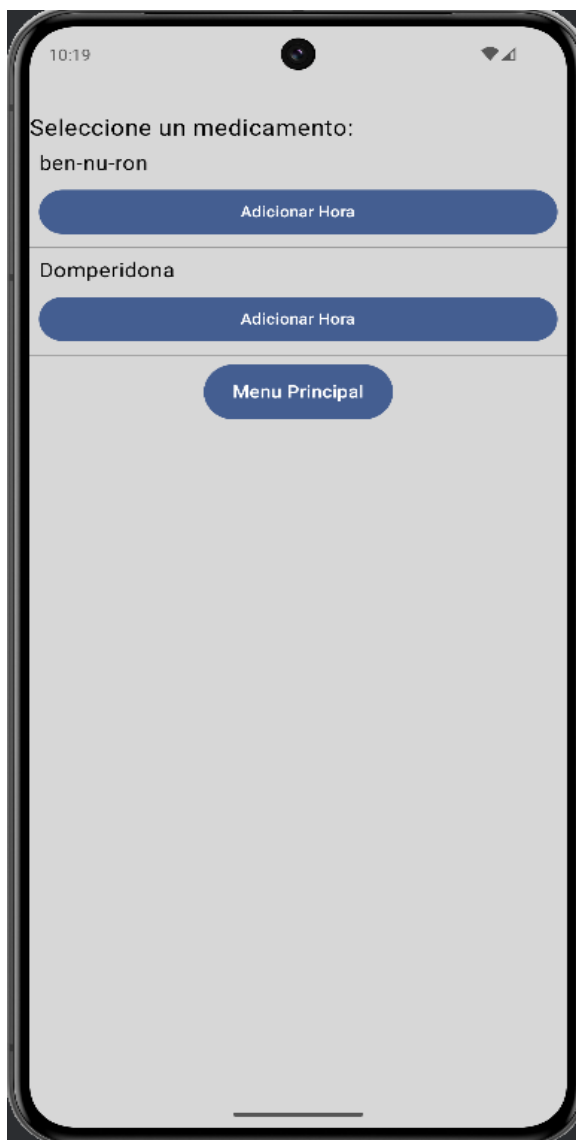
Ecrã nº 6



6º Ecrã da aplicação, neste ecrã o utilizador pode visualizar os medicamentos introduzidos anteriormente, possui um botão que permite excluir o medicamento, ao clicar no botão o utilizador pode excluir completamente o medicamento.

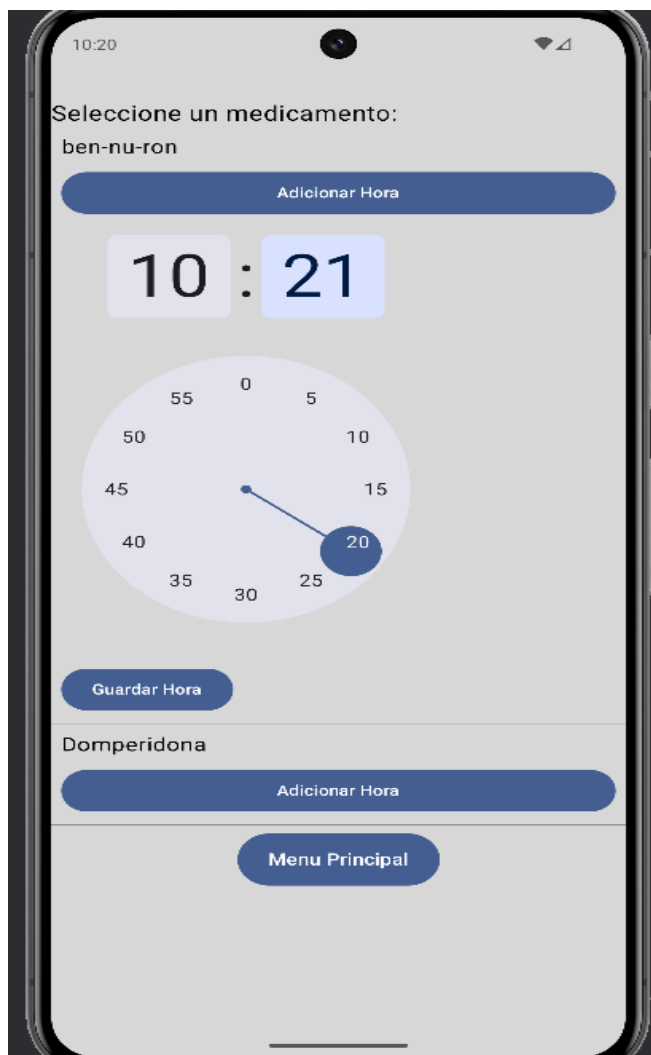
Possui dois botões na parte inferior, o botão de adicionar novo medicamento que permite ao utilizador adicionar um novo medicamento e o botão Menu principal que faz com que o utilizador possa voltar ao menu principal

Ecrã nº7



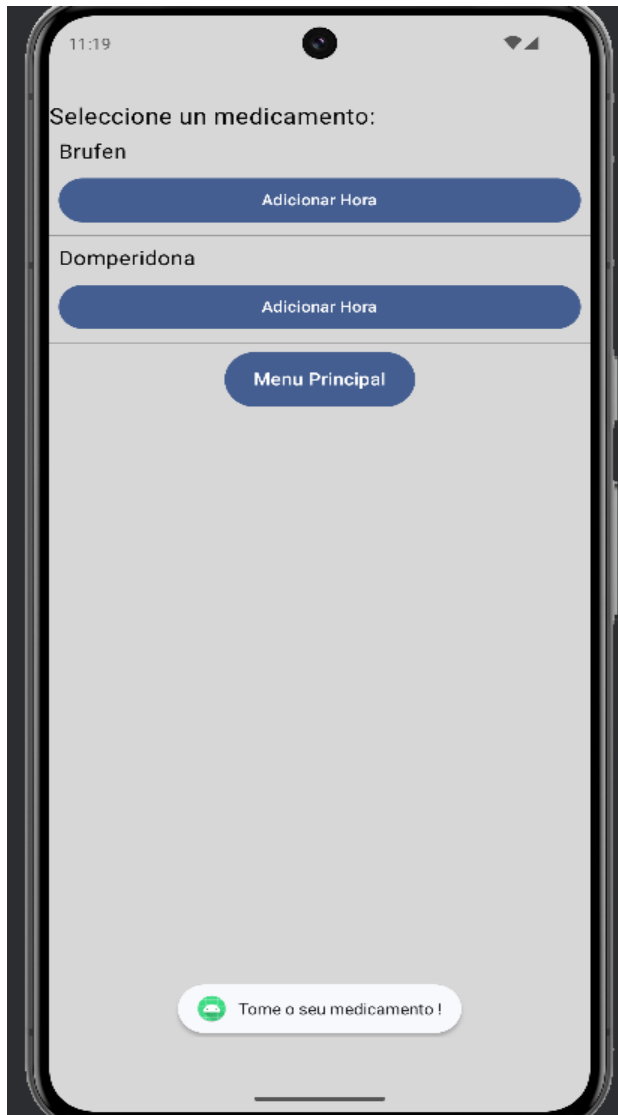
Ecrã onde o utilizador pode introduzir uma hora para cada medicamento e criar um alarme de acordo com a hora introduzida.

Ecrã nº7



Pequena demonstração de como o utilizador pode introduzir a hora para tomar o medicamento.

Ecrã nº7



Pequena demonstração do alarme

Ecrã nº8



Se o utilizador clicar no menu principal no botão de ver medicamentos vai ser dirigido para este menu onde pode ter acesso a todos os medicamentos introduzidos e editar os dados destes , se clicar no botão Editar Medicamentos vai ser dirigido para o seguinte ecrã

Ecrã nº 9



The screenshot shows a mobile application interface for editing medication information. The screen is titled "Ecrã nº 9". It features three input fields: "Nome do Medicamento" (Medication Name) with the value "Brufen", "Dose" (Dose) with the value "600ml", and "Frequência" (Frequency) with the value "4". Below these fields is a blue button labeled "Salvar" (Save). The bottom of the screen shows a standard iOS keyboard with a blue back arrow button.

Ecrã onde o utilizador consegue editar as informações de cada medicamento

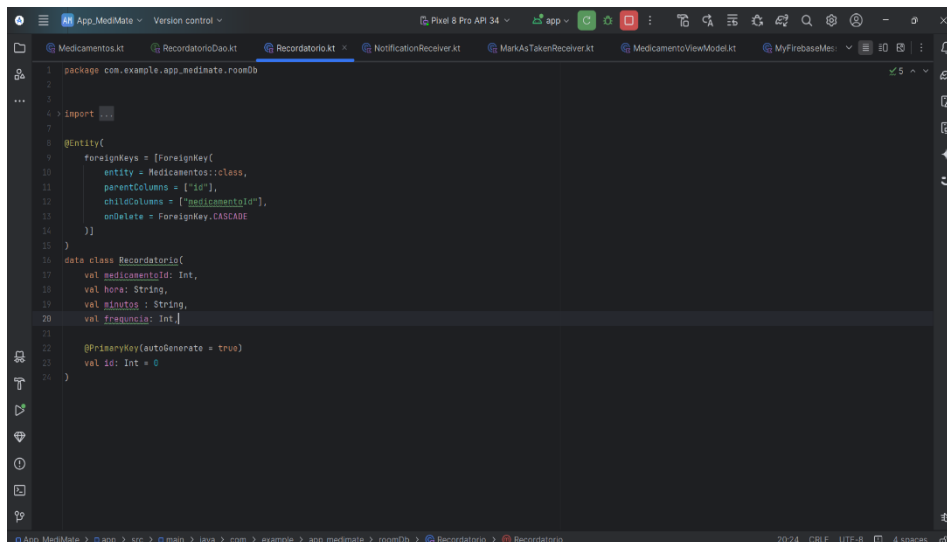
Para a elaboração deste projeto também foi criada uma base de dados pelo próprio android studio usando linguagem de programação.

➤ Ilustração da Base de dados:

➤ **Medicamentos.kt**

```
1 package com.example.app_medimate.roomDb
2
3 > import androidx.room.*
4
5 Explain | Test | Document | Fix | Ask
6 @Entity
7 data class Medicamentos(
8     val nomeMedicamento:String,
9     val dosis: String,
10    val frecuencia: String,
11    @PrimaryKey(autoGenerate = true)
12    val id: Int = 0
13 )
14
```

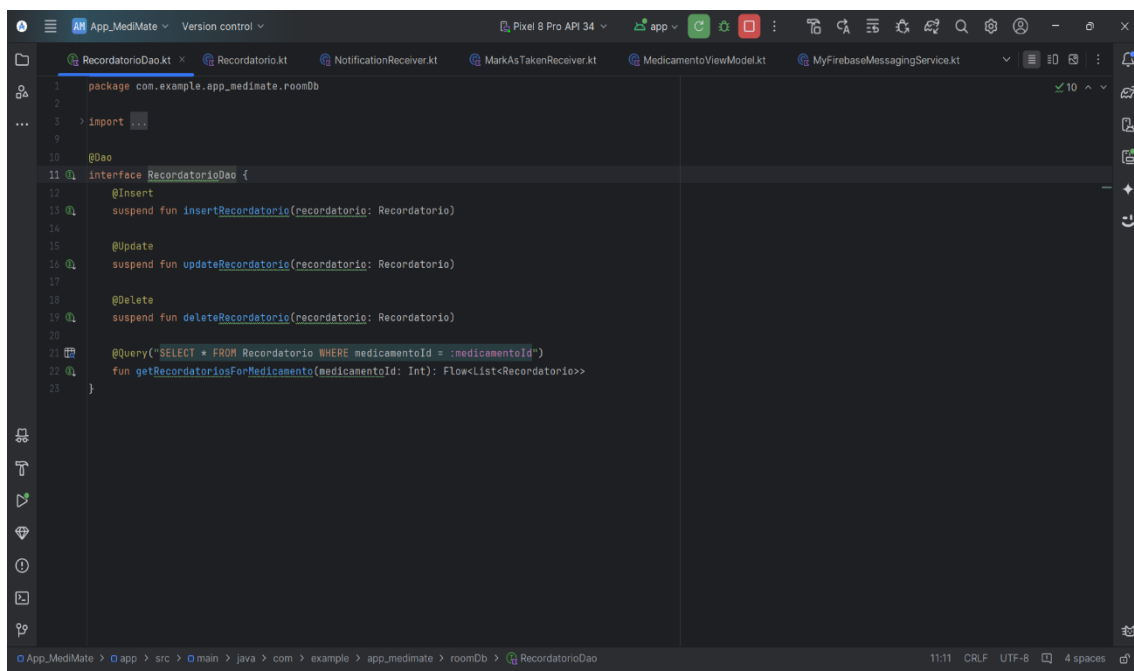
➤ **Recordatorios.kt**



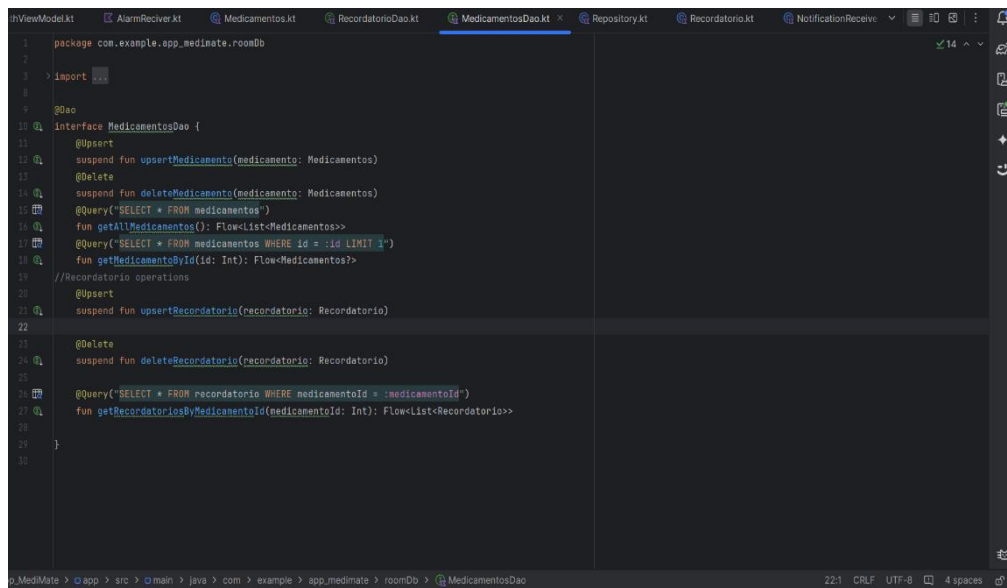
Programação da lógica da Base de dados

Tal como para o desenvolvimento da aplicação, para a criação da base de dados foram criadas também várias classes, sendo elas:

➤ RecordatorioDao.kt



➤ MedicamentoDao.kt

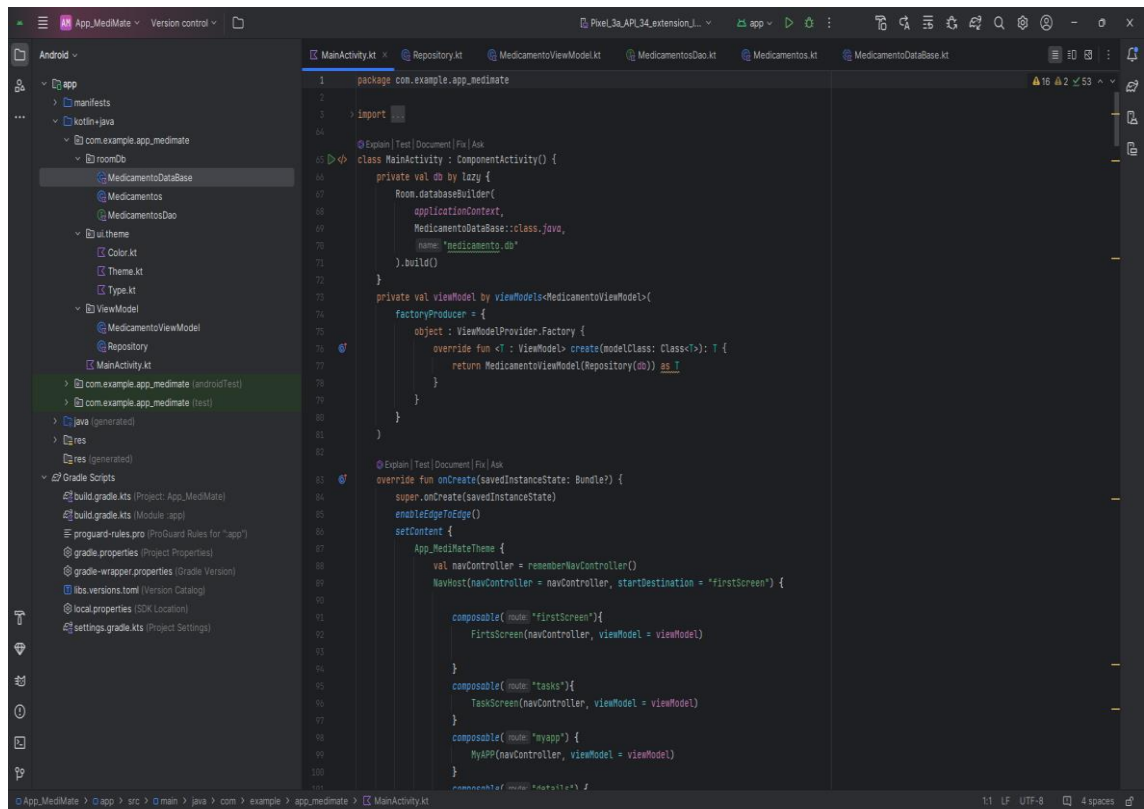


```
1 package com.example.app_medimate.roomdb
2
3 > import androidx.room.*
4
5 @Dao
6 interface MedicamentoDao {
7     @Upsert
8     suspend fun upsertMedicamento(medicamento: Medicamentos)
9
10     @Delete
11     suspend fun deleteMedicamento(medicamento: Medicamentos)
12
13     @Query("SELECT * FROM medicamentos")
14     fun getAllMedicamentos(): Flow<List<Medicamentos>>
15
16     @Query("SELECT * FROM medicamentos WHERE id = :id LIMIT 1")
17     fun getMedicamentoById(id: Int): Flow<Medicamentos?>
18
19     //Recordatorio operations
20     @Upsert
21     suspend fun upsertRecordatorio(recordatorio: Recordatorio)
22
23     @Delete
24     suspend fun deleteRecordatorio(recordatorio: Recordatorio)
25
26     @Query("SELECT * FROM recordatorio WHERE medicamentoId = :medicamentoId")
27     fun getRecordatoriosByMedicamentoId(medicamentoId: Int): Flow<List<Recordatorio>>
28
29 }
30
```

Conceções do Kotlin

Com base nos conhecimentos adquiridos durante as aulas, foi usada a aplicação Android Studio para o desenvolvimento da aplicação usando a linguagem de programação Kotlin.

➤ Ilustração do Android Studio:



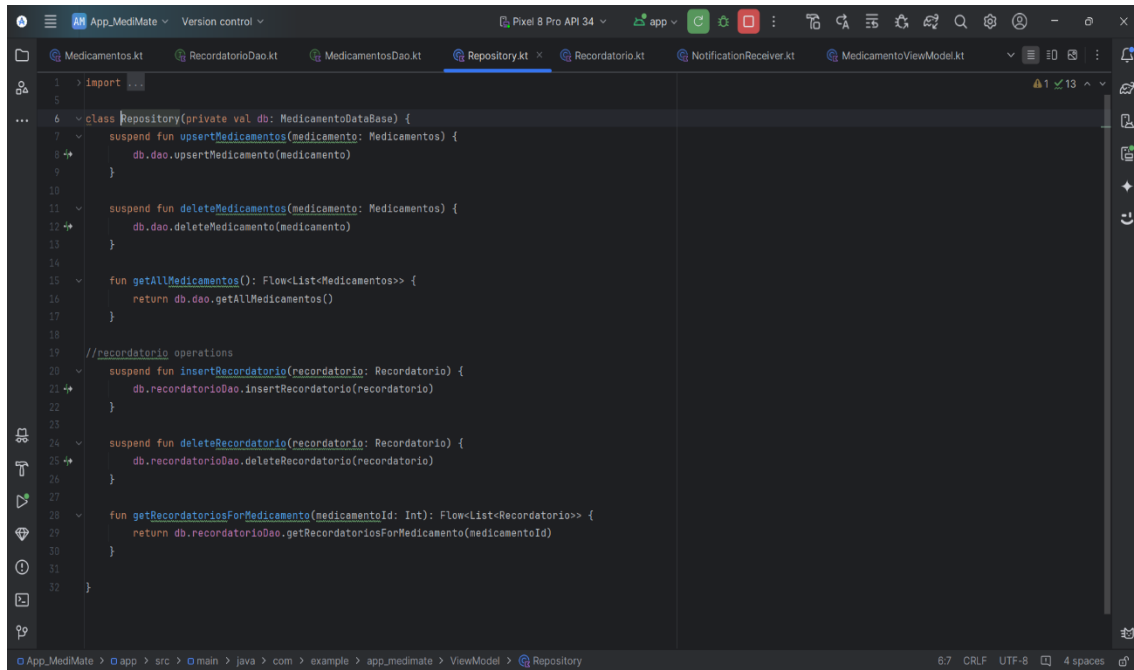
Programação da lógica da aplicação e Base de Dados

Para o desenvolvimento do nosso projeto foi necessário a criação de várias classes, entre elas estão:

➤ MainActivity.kt:

```
183 fun TaskScreen(navController: NavController, modifier: Modifier = Modifier, viewModel: MedicamentoViewModel) {  
184     ) {  
211         Box(  
212             modifier = Modifier  
213                 .fillMaxWidth()  
214                 .padding(top = 120.dp)  
215         ) {  
216             Column(  
217                 modifier = Modifier  
218                     .fillMaxWidth()  
219                     .wrapContentSize(Alignment.Center)  
220                     .offset(x = 0.dp, y = -60.dp)  
221             ) {  
222                 Text(  
223                     text = "Gerenciamento de medicamentos",  
224                     fontSize = 15.sp,  
225                     color = Color(color = 0xFF2F4F7F),  
226                     textAlign = TextAlign.Center  
227                 )  
228             }  
229             Column(  
230                 modifier = Modifier  
231                     .fillMaxWidth()  
232                     .padding(50.dp),  
233                 horizontalAlignment = Alignment.CenterHorizontally,  
234                 verticalArrangement = Arrangement.Center  
235             ) {  
236                 Button(  
237                     onClick = {  
238                         navController.navigate(route = "myapp") },  
239                     modifier = Modifier  
240                         .fillMaxWidth()  
241                         .height(50.dp)  
242                 ) {  
243                     Row(  
244                         verticalAlignment = Alignment.CenterVertically,  
245                         horizontalArrangement = Arrangement.SpaceBetween  
246                     ) {  
247                         Image(  
248                             painter = painterResource(id = R.drawable.md),  
249                             contentDescription = null  
250                         )  
251                     }  
252                 }  
253             }  
254         }  
255     }  
256 }
```

➤ Repository.kt



➤ MedicamentoViewModel.kt

```

1
2
3 > import androidx.lifecycle.ViewModel
4
5
6
7
8
9
10
11
12
13
14 class MedicamentoViewModel(private val repository: Repository) : ViewModel() {
15     val medicamentosFlow: StateFlow<List<Medicamentos>> = repository.getAllMedicamentos()
16     .stateIn(viewModelScope, SharingStarted.Eagerly, emptyList())
17
18
19
20
21     fun upsertMedicamento(medicamentos: Medicamentos) {
22         viewModelScope.launch { this: CoroutineScope
23             repository.upsertMedicamentos(medicamentos)
24         }
25     }
26
27     fun deleteMedicamento(medicamentos: Medicamentos) {
28         viewModelScope.launch { this: CoroutineScope
29             repository.deleteMedicamentos(medicamentos)
30         }
31     }
32
33     fun getMedicamentoById(id: Int): Flow<Medicamentos?> {
34         return medicamentosFlow.map { medicamentos -> medicamentos.find { it.id == id } }
35     }
36
37     // Recordatorio operations
38     fun insertRecordatorio(recordatorio: Recordatorio) {
39         viewModelScope.launch { this: CoroutineScope
40             repository.insertRecordatorio(recordatorio)
41         }
42     }
43
44
45
46
47
48
49
50
51

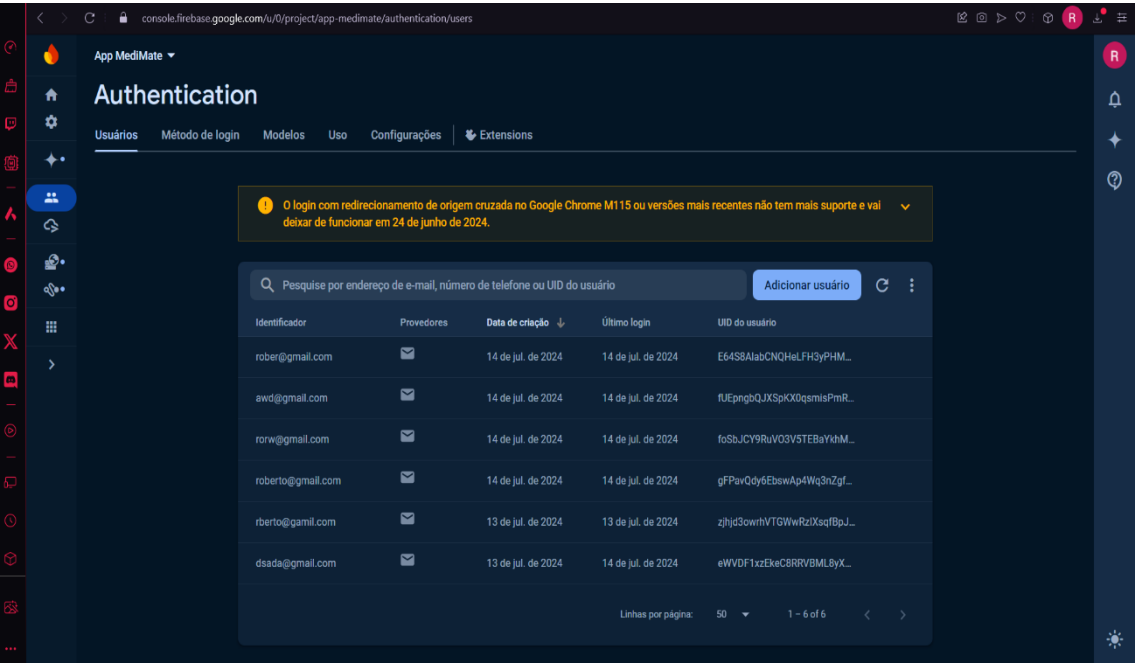
```

App_MediMate | Version control | Pixel 8 Pro API 34 | app | Medicamentos.kt | RecordatorioDao.kt | MedicamentosDao.kt | Repository.kt | Recordatorio.kt | NotificationReceiver.kt | MedicamentoViewModel.kt

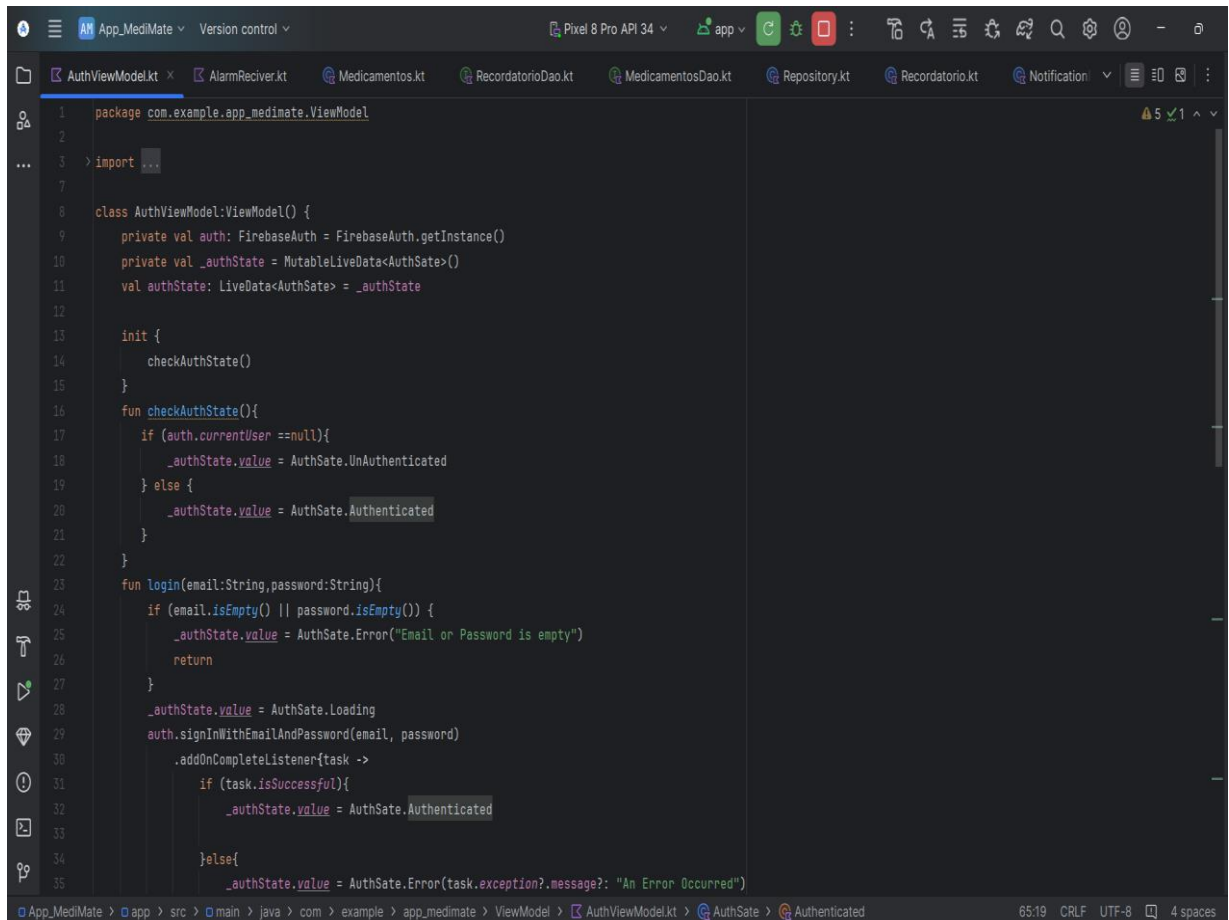
17:1 CRLF UTF-8 4 spaces

Outros Aspetos

Para realizar um Login e Criar Conta na aplicação foi criado um projeto no Firebase que permite guardar os dados do email e palavra-passe dos utilizadores



Foi também criado um viewModel onde se programou a logica dos menus de Login e Criar Conta

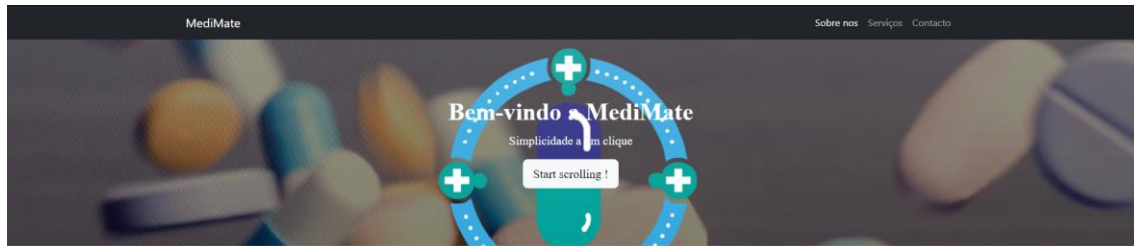


```
1 package com.example.app_medimate.ViewModel
2
3 > import ...
4
5
6
7
8 class AuthViewModel:ViewModel() {
9     private val auth: FirebaseAuth = FirebaseAuth.getInstance()
10     private val _authState = MutableLiveData<AuthState>()
11     val authState: LiveData<AuthState> = _authState
12
13     init {
14         checkAuthState()
15     }
16     fun checkAuthState(){
17         if (auth.currentUser == null){
18             _authState.value = AuthState.UnAuthenticated
19         } else {
20             _authState.value = AuthState.Authenticated
21         }
22     }
23     fun login(email:String,password:String){
24         if (email.isEmpty() || password.isEmpty()) {
25             _authState.value = AuthState.Error("Email or Password is empty")
26             return
27         }
28         _authState.value = AuthState.Loading
29         auth.signInWithEmailAndPassword(email, password)
30             .addOnCompleteListener{task ->
31             if (task.isSuccessful){
32                 _authState.value = AuthState.Authenticated
33             }else{
34                 _authState.value = AuthState.Error(task.exception?.message?: "An Error Occurred")
35             }
36         }
37     }
38 }
```

Página web da aplicação

Para o Desenvolvido da página web da aplicação precisamos de usar o vs code e o Bootstrap para a sua realização.

Página web:

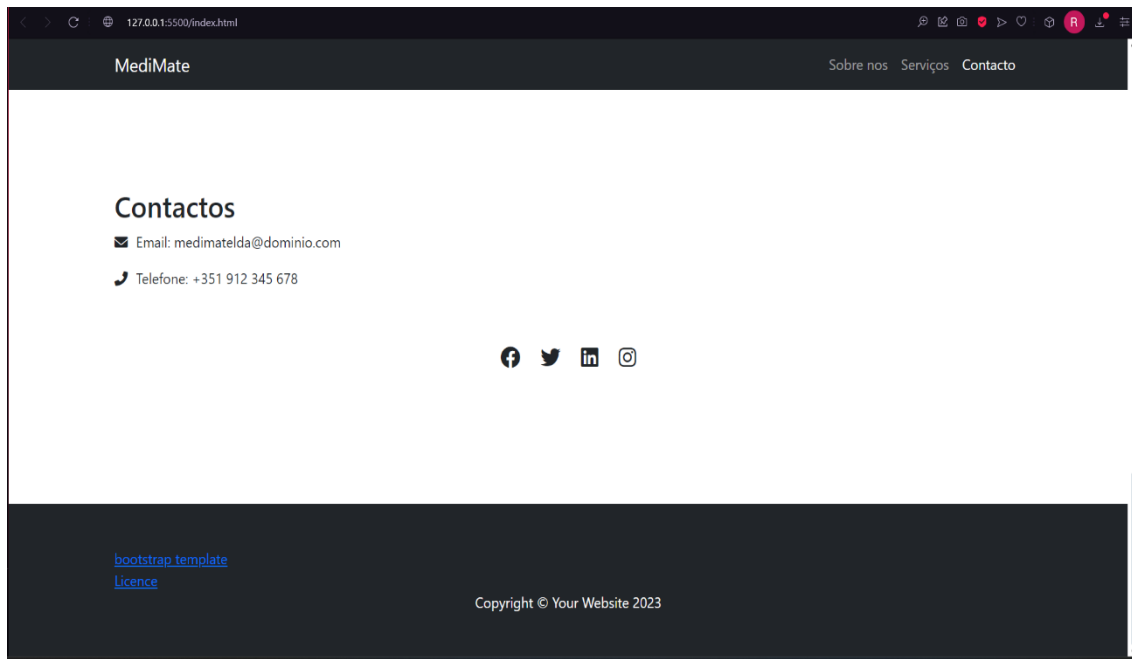


Sobre MediMate

A aplicação MediMate é uma ferramenta intuitiva e fácil de usar, projetada para ajudar os utilizadores a gerenciar seus medicamentos diários. Com ela, você pode registrar seus medicamentos, configurar lembretes diários, marcar como tomados e visualizar um histórico simples.

- Fácil de utilizar devido a um layout simples;
- Com o download desta aplicação não se precisa precupar a medicação;
- Experimente gratuitamente sem nenhum custo;
- Com MediMate pode garantir erros mínimos.





Conclusão

Com a realização dos trabalhos em sala de aula e a ajuda dos professores foi possível a realização deste projeto assim ficamos a entender os mil e um projetos que são possíveis realizar com a programação e com os diferentes tipos de linguagem de programação existentes.