Nome: João Alfredo Teodoro RA: 01191096 Turma: 4 ADS A

URL: https://github.com/GrupoSP4U/SP4U-FRONT-MOBILE

I – Data Class Kotlin

No rojeto foi empregado Data Class Kotlin para criar as classes com os dados que é utilizado durante toda nossa aplicação, utilizando na transferência do mesmo. Para citar um exemplo: Dentro do projeto existe uma pasta"dto" e nela existem duas subpastas "request" e "response", as classes dentro desses arquivos são DTOs, pois nela constam as variáveis imutáveis "val" que posteriormente receberam esse dado do banco, fazendo uma intermediação. Outros arquivos que apresentam essa características são:

- AuthenticateRequestDTO

```
data class AuthenticateRequestDTO(
    var email: String?,
    var password: String?
)
```

- ChangePasswordRequestDTO

- EstabelecimentoDTO

- InteresseD

- UserDTO

II – Chamando outra Activity

Foi necessário colocar isso na nossa aplicação, pois sem essas activities não seria possível fazer os links entre as telas, tornando assim um fluxo de navegação, e nem passar alguns parâmetros para a tela selecionada durante o uso do APP.

```
class AdditionalFiltersActivity : AbstractActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_additional_filters)
        userId = intent.getIntExtra("USER_ID", 0)
        token = intent.getStringExtra("TOKEN")
    }

    fun goToResult(button: View){
        navigate(userId, token, ResultActivity().javaClass)
    }
}
```

Outros arquivos que podem ser encontrados o caso a cima se encontram nas pastas a seguir:

- activity > register

- activity > user

- activity > welcome

III – Dados Offline

Até o momento não foi adicionado dados offline no projeto.

IV - Implementação i18N

Utilizamos a i18N, pois ela serve para realizar a internacionalização da aplicação. Com ela, é possível criar um arquivo “string” com todas os testos do aplicativo, independentemente ser “EditText”, “Toast”, “TextView”, com as traduções. No nosso projeto criamos a tradução para o inglês, outro ponto positivo é a facilidade de manutenção do código. Porque todos os campos que tiverem com esse id-string será atualizado apenas alterando no arquivo “string”, com projetos escaláveis isso resolve um problemão no futuro.

No caso abaixo é o “@string/login_email

```
<string name="titulo_login">login</string>
<string name="login_email">e-mail</string>
<string name="et_login_email">Digite seu e-mail</string>
<string name="login_senha">senha</string>
<string name="et_login_senha">Digite sua senha</string>
<string name="login_cadastrar">cadastre-se</string>
<string name="login_esqueceu_senha">esqueceu senha</string>
<string name="btn_login_entrar">entrarar</string>
```

```
<!-- Start login screen-->
<string name="titulo_login">login</string>
<string name="login_email">e-mail</string>
<string name="et_login_email">Input your e-mail</string>
<string name="login_senha">password</string>
<string name="et_login_senha">Input your password</string>
<string name="login_cadastrar">Sign up</string>
<string name="login_esqueceu_senha">forgot the password</string>
<string name="btn_login_entrar">Sign in</string>
```

```
<TextView
    android:id="@+id/tv_email"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="44dp"
    android:fontFamily="@font/montserrat"
    android:text="@string/login_email"
    android:textColor="@color/White"
    android:textSize="20sp"
    app:layout_constraintTop_toBottomOf="@id/img_inicio"
    tools:layout_editor_absoluteX="66dp" />

<EditText
    android:id="@+id/et_email"
    android:layout_width="280dp"
    android:layout_height="40dp"
    android:layout_marginTop="8sp"
    android:background="@drawable/et_style_blue"
    android:fontFamily="@font/montserrat"
    android:hint="@string/et_login_email"
    android:paddingLeft="10dp"
```

```
    android:inputType="textEmailAddress"
    android:textColor="@color/White"
    android:textColorHint="@color/White"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@id/tv_email" />
```

Esse procesimento se manteve em toda nossa aplicação, portanto todos os arquivos layouts estão com padronização i18N.

V – Scroll

Foi utilizado a “rolagem” de tel, porque por usar devices mobiles, as telas são menores então a rolagem auxilia tanto no fluco do usuário durante seu uso, quanto utilizar menos telas e as “linkagens” necessárias”, podendo não ser muito aceita pelo usuários. Ex: o usuário pode não ter paciência de ficar esperando o retorno do clique ir para outra tela, por isso. A “rolagem” de tela ajuda bastante na experiência do usuário.

Exemplo “ScrollView

```
<ScrollView
    android:id="@+id/ScrollViewTela"
    android:layout_width="match_parent"
    android:layout_height="490dp"
    android:paddingLeft="24dp"
    android:paddingRight="24dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <LinearLayout
        android:id="@+id/linearLayoutTela"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
```

Também encontrado nos arquivos: -

“activity_feels”, “activity_register”,

-“activity_register_establishment_config”,

-“activity_register_establishment_config_description”,

-“activity_register_establishment_music”,

-”activity_result”,

-“activity_user_profile_config”,

-“activity_user_profile_email”,

-“activity_user_profile_password”

Vi – RestAPI

A RestAPI serve para realizar todas as requisições dentro do aplicativo, tais como: “GET”, ‘POST”, “PUT”. Sem elas não é possível gravar, alterar e trazer dados dentro da aplicação.

```
interface AuthenticateService {
    @POST("authenticate")
    fun authenticate(@Body authenticate: AuthenticateRequestDTO):
Call<AuthenticateResponseDTO>
}
```

Também são encontrados nos arquivos abaixo:

- “EstablishmentService”,

- “UserService”

VII- Fragment

Assim como um dos motivos do i18N é deixar o código mais fácil de manutenção o “Fragment” serve para criar partes da aplicação que serão reutilizadas em outras telas, para não precisar codar várias vezes e quando for ser atualizado olhar tela por tela para fazer atualização. Com o “Fragment” só sera necessário mexer nesse arquivos.

Consta no arquivo “fragmente_menu”,

```
class MenuFragment : Fragment() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_menu, container, false)
    }
}
```

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/fr_menu">
    <fragment
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        class="com.example.sp4u_app.fragment.MenuFragment"
        />
</FrameLayout>
```

Outros arquivos que constam a caso a cima são :

-“activity_register_establishment_config”,

-“activity_register_establishment_config_description”,

-“activity_user_profile_config”,

-“activity_user_profile_email”,

-“activity_user_profile_password”