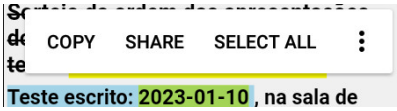
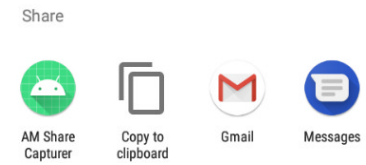
Descrição Geral da App - Só para ajuda e contexto - Não há perguntas nesta página

Nesta prova terá que saber escrever partes de uma app Android de título "AM Share Capturer" que serve para:
- receber partilhas de texto ("shares") de outras apps que existam no mesmo dispositivo;
- guardá-las numa base de dados TSV, na private internal storage;
- e publicar o texto recebido, junto com a data+hora da receção, num serviço Web, em certo URL.

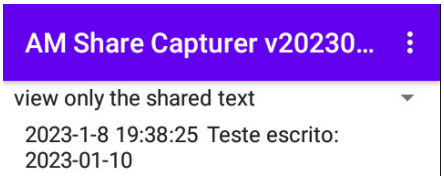
Eis o problema, da perspetiva do utilizador:
- numa qualquer app que suporte partilha de texto, como o Google Chrome, o utilizador seleciona texto e escolhe "share".



- se a sua app tiver registado o intent-filter correto em AndroidManifest.xml (pode assumir que sim), ela vai constar da lista de apps que o sistema apresenta ao utilizador, capazes de receber partilhas de texto.



- se o utilizador escolher a nossa app, o texto escolhido deverá ser recebido pela app.



De uma perspetiva técnica, receber uma partilha de texto faz-se pelo seguinte procedimento:
- obter o Intent que lançou a Activity, com getIntent()
- obter a action do Intent com getAction()
- responder se a action for "ACTION_SEND" e o tipo, obtido com getType(), começar pela frase "text/"
- o texto partilhado está sempre disponível num EXTRA com key "EXTRA_TEXT".

Por exemplo, embora sem cautelas, o seguinte código faz isso numa qualquer Activity:

```
boolean bDevoResponder = getIntent().getAction().equals(Intent.ACTION_SEND);
boolean bFoiTexto = bDevoResponder ? getIntent().getType().startsWith("text/") : false;
String textoPartilhado = bFoiTexto ? getIntent().getStringExtra(Intent.EXTRA_TEXT) : "";
```

Guardar em base de dados TSV, implica saber fazer leituras/escritas de ficheiros TSV na private internal storage.

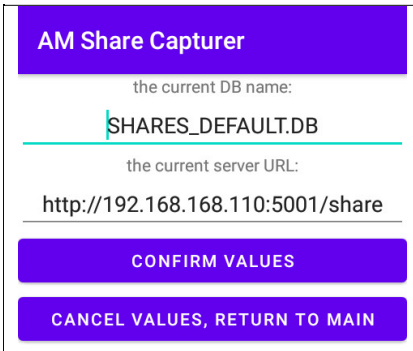
Publicar a data+hora na WWW, implica saber usar o calendário do sistema, para consultar essa informação, e chamar assincronamente um método que NÃO terá que escrever, poderá abstrair, que recebe o URL do serviço, o texto e o momento da partilha.

A app está organizada em duas Activity simples, em relação às quais terá que fazer trabalho, guiado pelas questões seguintes.

As Activity são:
- MainActivity, a Activity de arranque, que terá um menu;
- SettingsActivity, uma Activity secundária, chamável pelo menu da primeira, que serve para alterar dois valores de configuração: o nome do ficheiro da base de dados TSV e o URL do serviço que também recebe a partilha.

Não terá que escrever absolutamente nada do serviço Web. Assuma que está perfeito, online, e utilizável em URL configurável.

➔Q1. Escreva todo o XML necessário para a SettingsActivity, de forma a conseguir um aspeto como o sugerido pela imagem, respeitando as orientações.



De cima para baixo, existem elementos com os seguintes ids:
- idTvDBName, um cabeçalho de texto, onde se lê "the current DB name:";
- idEtDBName, um campo de escrita, onde se lê "SHARES_DEFAULT.DB";
- idTvHttpUrl, um cabeçalho de texto, onde se lê "the current server URL:";
- idEtHttpUrl, um campo de escrita, onde se lê "http://192.168.168.110:5001/share";
- idBtnConfirm, um botão, onde se lê "confirm values";
- idBtnCancel, um botão onde se lê "cancel values, return to main".

O elemento raiz do layout deverá ser constraintlayout, ou RelativeLayout.

Deve usar frases de recurso e não frases literais, com nomes idênticos aos dos elementos que as utilizam, mas substituindo "id" por "str".

Deve escrever o strings.xml correspondente.

```
<resources>
<!-- strings.xml -->
    <string name="app_name">AM Share Capturer</string>
    <string name="strTvAbout">The captured data is:</string>
    <string name="strMenuItemLoadImageFromUrl">Load Image From Url</string>
    <string name="strIvCapturedImage">image shared from external app</string>
    <string name="strTitleWithVersion">AM Share Capturer v20230108</string>
    <string name="strMenuShowHttpErrors">show http errors</string>
    <string name="strMenuGoSettings">DB and http settings</string>
    <string name="strMenuClearCurrentDB">clear current DB</string>
    <string name="strEtDBNameHint">DB name</string>
    <string name="strTvDB">the current DB name:</string>
    <string name="strEtServerUrlHint">https://someserver.com/share/</string>
    <string name="strTvHttpUrl">the current server URL:</string>
    <string name="strBtnConfirm">confirm values</string>
    <string name="strBtnCancel">cancel values, return to main</string>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SettingsActivity">

    <TextView
        android:id="@+id/idTvDB"
        android:text="@string/strTvDB"
        android:gravity="center_horizontal"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <EditText
        android:id="@+id/idEtDBName"
        app:layout_constraintTop_toBottomOf="@id/idTvDB"
        android:gravity="center_horizontal"
        android:hint="@string/strEtDBNameHint"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/idTvHttpUrl"
        android:gravity="center_horizontal"
        app:layout_constraintTop_toBottomOf="@id/idEtDBName"
        android:text="@string/strTvHttpUrl"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <EditText
        android:inputType="textNoSuggestions|text"
        android:id="@+id/idHttpUrl"
        android:gravity="center_horizontal"
        app:layout_constraintTop_toBottomOf="@id/idTvHttpUrl"
        android:hint="@string/strEtServerUrlHint"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <Button
        app:layout_constraintTop_toBottomOf="@id/idHttpUrl"
        android:id="@+id/idBtnConfirm"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/strBtnConfirm"/>

    <Button
        app:layout_constraintTop_toBottomOf="@id/idBtnConfirm"
        android:id="@+id/idBtnCancel"
        android:text="@string/strBtnCancel"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

➔ Q2. Escreva todo o Java relevante para operação de um método "init", a chamar em "onCreate" de SettingsActivity, que:

- associe aos elementos XML membros de dados Java com o mesmo nome, mas substituindo "id" por "m"; por exemplo "mTvDBName" para "idTvDBName";

- implemente e atribua o mesmo "handler" de nome "mClickHandler", a ambos os botões. O botão de confirmação deve reagir chamando actionConfirm, que deve abstrair. O botão de cancel deve reagir chamando actionCancel, que também deve abstrair.

- os valores que aparecem nos campos de escrita devem ser consultados de um objeto SharedPreferences, em ficheiro de nome disponível em constante "SETTINGS_DB". O valor do nome da base dados está em KEY_DB_NAME. O valor do URL está em KEY_HTTP_URL. Ambas as keys são mantidas pela MainActivity, mas consultadas aqui, em SettingsActivity. Os valores por defeito são DEFAULT_DB_NAME e DEFAULT_HTTP_URL, respetivamente.

```
public class SettingsActivity extends AppCompatActivity {

    public final static String DEFAULT_DB_NAME = "SHARES_DEFAULT.DB";
    public final static String DEFAULT_HTTP_URL = "http://192.168.168.110:5001/share";

    Context mContext;

    TextView mTvDBName, mTvHttpUrl;
    EditText mEtDBName, mEtHttpUrl;
    Button mBtnConfirm, mBtnCancel;

    public final static String SETTINGS_DB ="SETTINGS.DB";
    SharedPreferences mSpSettings;

    View.OnClickListener mClickHandler = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            switch(v.getId()){
                case R.id.idBtnConfirm:
                    actionConfirmSettings();
                    break;
                case R.id.idBtnCancel:
                    actionCancelSettings();
                    break;
            }
        }
    };//mClickHandler

   void init(){
        mContext = this;
        mTvDBName = findViewById(R.id.idTvDB);
        mEtDBName = findViewById(R.id.idEtDBName);
        mTvHttpUrl = findViewById(R.id.idTvHttpUrl);
        mEtHttpUrl = findViewById(R.id.idHttpUrl);

        mBtnConfirm = findViewById(R.id.idBtnConfirm);
        mBtnCancel = findViewById(R.id.idBtnCancel);
        mBtnConfirm.setOnClickListener(mClickHandler);
        mBtnCancel.setOnClickListener(mClickHandler);

        mSpSettings = getSharedPreferences(SETTINGS_DB, MODE_PRIVATE);
        if(mSpSettings!=null){
            String strDBName, strHttpUrl;
            strDBName = mSpSettings.getString(MainActivity.KEY_DB_NAME, DEFAULT_DB_NAME);
            strHttpUrl = mSpSettings.getString(MainActivity.KEY_HTTP_URL, DEFAULT_HTTP_URL);
            mEtDBName.setText(strDBName);
            mEtHttpUrl.setText(strHttpUrl);
        }
    }//init
```

➔ Q3. Escreva todo o Java necessário para que a MainActivity possa chamar a SettingsActivity, para obtenção dos valores que forem confirmados nos campos de escrita, utilizando uma versão moderna do padrão SAFR (Start Activity For Result).

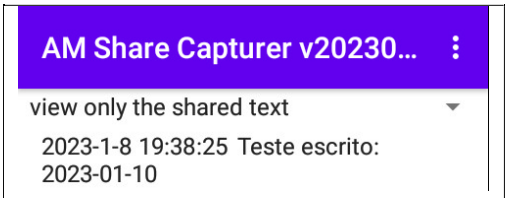
O utilizador vai poder navegar da MainActivity até à SettingsActivity, a partir de uma opção de um menu (que deve abstrair), cujo código completo se segue.

O seu código Java tem que garantir que actionGoSettings vai realmente funcionar.

```
void actionGoSettings(){
    Intent goSettings = new Intent(this, SettingsActivity.class);
    launcher.launch(goSettings); //SAFR
} // actionGoSettings
```

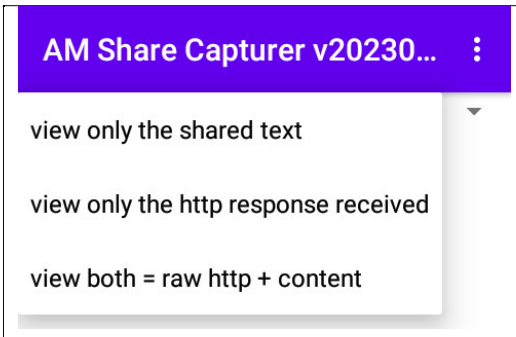
```
//SAFR pattern
public final static String KEY_DB_NAME = "KEY_RESULT_DB_NAME";
public final static String KEY_HTTP_URL = "KEY_RESULT_HTTP_RESULT";
//SAFR caller-side 1
ActivityResultContract<Intent, ActivityResult> safrContract =
        new ActivityResultContracts.StartActivityForResult();

//SAFR caller-side 2
ActivityResultCallback<ActivityResult> safrCallback =
        new ActivityResultCallback<ActivityResult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                boolean bOK = result.getResultCode()==RESULT_OK;
                if(bOK){
                    Intent theResultData = result.getData();
                    if(theResultData!=null){
                        String strDBName = theResultData.getStringExtra(MainActivity.KEY_DB_NAME);
                        String strHttpUrl = theResultData.getStringExtra(MainActivity.KEY_HTTP_URL);
                        // do something with the results
         boolean bDBNamedChanged = MainActivity.this.mCurrentDBName!=strDBName;

                        String strDBFeedback = "";
                        if(bDBNamedChanged){
                            MainActivity.this.mCurrentDBName = strDBName;
                            strDBFeedback = "DB Name changed to %s".format(MainActivity.this.mCurrentDBName);
                            refresh();
                        }
                        else{
                            strDBFeedback = "DB Name did NOT change. It remains %s".format(MainActivity.this.mCurrentDBName);
                        }

                        mUtil.fb(strDBFeedback);

                        boolean bHttpUrlChanged = MainActivity.this.mCurrentHttpUrl!=strHttpUrl;
                        if(bHttpUrlChanged){
                            MainActivity.this.mCurrentHttpUrl = strHttpUrl;
                            strDBFeedback = "Http URL changed to %s".format(MainActivity.this.mCurrentHttpUrl);
                        }
                        else{
                            strDBFeedback = "Http URL did NOT change. It remains
%s".format(MainActivity.this.mCurrentHttpUrl);
                        }
                        mUtil.fb(strDBFeedback);
                    }
                }
            }
        };

//SAFR Caller-side 3
ActivityResultLauncher<Intent> launcher = registerForActivityResult(
    safrContract,
    safrCallback
);
```

Q4. A app, conforme já descrito, tem uma Activity de arranque, MainActivity, cujo layout XML pode admitir disponível e perfeito, com o aspeto sugerido pelas imagens.



Tem-se, de cima para baixo:
- um Spinner mSpnViewMode;
- uma ListView mLvTextCaptures.

Neste exemplo (é apenas um exemplo) a ListView mostra uma partilha capturada em "2023-1-8 19:38:25", com o texto "Teste escrito: 2023-01-10".



O Spinner, quando expandido, deve mostrar 3 opções, baseadas em recursos do tipo frase.

Não tem que programar o comportamento do Spinner, mas tem que garantir que um string-array com as opções mostradas é carregado em runtime.

No seu init, a MainActivity, deve determinar se foi chamada para tratar de uma partilha de texto. Se sim, deve fazer chamar um método `void insert(String pDBName, String pTexto)` que recebe texto recebido e o escreve na base de dados pDBName.

➔ Assim, escreva o Java e o XML necessários para:
- garantir que Spinner é devidamente populado em init;
- implementar integralmente o método insert;
- em init, ter o código suficiente para a receção cautelosa do texto partilhado e a sua escrita na base de dados, chamando insert.

Recorde que o nome da base de dados é uma opção mantida em SharedPreferences SETTINGS_DB.

```
void init(Bundle pBundle){
...

    // the Spinner
  mSpnViewMode = findViewById(R.id.idSpnViewMode);
    mAlSpinnerOptions = new ArrayList();
    String[] theOptionsFromResources = getResources().getStringArray(R.array.aShareViewOptions);
    int iHowManyOptions = theOptionsFromResources.length;
    for (String s : theOptionsFromResources){
        mAlSpinnerOptions.add(s);
    }
...
}//init

Boolean insert(String, pDBName, String pInsertWhat){
    Calendar cal = Calendar.getInstance();
    int y = cal.get(Calendar.YEAR);
    int m = cal.get(Calendar.MONTH)+1;
    int d = cal.get(Calendar.DAY_OF_MONTH);
    int hh = cal.get(Calendar.HOUR_OF_DAY);
    int mm = cal.get(Calendar.MINUTE);
    int ss = cal.get(Calendar.SECOND);
    String strTimeStamp = String.format(
        "%d-%d-%d %d:%d:%d", y, m, d, hh, mm, ss);
    pInsertWhat = pInsertWhat.replace("\t", " ");
    pInsertWhat = pInsertWhat.replace("\n", "<br>");
    String strTsv = String.format(
        "%s\t%s\n", //highly relevant for reading,later
   strTimeStamp,
        pInsertWhat
    );
//TODO insert online também
    return writeContentToFileInPIS(pDBName, strTsv, MODE_APPEND);
}//insert
```

```
//chamar isto em init, para a receção do texto
void checkIfCalledByAnotherAppAndReceiveItsSharedData(){
    Intent intentHowWasICalled = getIntent();

    if (intentHowWasICalled!=null){
        String strAction = intentHowWasICalled.getAction();

        boolean bItWasTheUserInterfaceOrAndroidStudio =
            strAction.equals(Intent.ACTION_MAIN);

        boolean bIsItActionSend =
            strAction.equals(Intent.ACTION_SEND); //share!

   //is this a share situation?
   if (bIsItActionSend){
            String strType =
                intentHowWasICalled.getType(); //text/html text/plain

            boolean bIsItSharedText =
                strType.startsWith("text/");

            if (bIsItSharedText){
                //receive the shared text
      String strSharedText =
                    intentHowWasICalled.getStringExtra(
                        Intent.EXTRA_TEXT
       );

                //display the received text in mTvCapturedText
      mTvCapturedText.setText(strSharedText);

                insert(strSharedText);

                refresh(); // to sync the list view with the database
     }//if
        }//if
  }//if
}//checkIfCalledByAnotherAppAndReceiveItsSharedData
```

➔ Q5. Explique, escrevendo os troços de código que considere fundamentais, como é que a MainActivity poderia fazer manutenção de estado da sua ListView, que deve constantemente "espelhar" o conteúdo da base de dados TSV.

Uma vez que a ListView deve espelhar os dados no ficheiro de texto, basta escrever para o ficheiro sempre que uma share é recebida e notificar o Adapter da ListView quando isso acontece. No código da questão anterior, isso é feito pelo método "refresh()", então abstraído.

```
void refresh(){
    //mAlTextCaptures = new ArrayList<>();

 ArrayList<String> temp = readAllPrevSharedTexts(this.mCurrentDBName);
   if (temp!=null && temp.size()>=0){
       mAlTextCaptures.clear();
  int iHowManyRecords = temp.size();
       for (int idx=0; idx<iHowManyRecords; idx++){
           String record = temp.get(idx);
           mAlTextCaptures.add(0, record);
       }//for
  mAd.notifyDataSetChanged();
   }
}//refresh
```

EM ALTERNATIVA:

Em geral, todos os dados "especiais" que se queiram considerar na manutenção de estado, podem ser mantidos por put/get feitos em onSavesInstanceState/onRestoreInstanceState. Exemplifica-se isso para um inteiro "especial", de seguida.

```
//save stuff here!
@Override
protected void onSaveInstanceState(@NonNull Bundle outState) {
    if (outState!=null){
        outState.putInt("KEY_SPECIAL", especial);
    }
    super.onSaveInstanceState(outState);
}

//recover stuff here
@Override
protected void onRestoreInstanceState(@NonNull Bundle savedInstanceState) {
    if (savedInstanceState!=null){
        boolean bHasKey =
            savedInstanceState.containsKey("KEY_SPECIAL");

        if (bHasKey){
            especial = savedInstanceState.getInt("KEY_SPECIAL");
        }
    }
    super.onRestoreInstanceState(savedInstanceState);
}
```

➔ Q6. Admita disponível um método postShare, com a assinatura seguinte, capaz de fazer http-post de dados para um URL e retornar a resposta do serviço ao pedido, na forma de uma String.

```
public static String postShare(
  String pUrl, //URL para o qual fazer POST
  String pWhen, //frase com data+hora da text share
  String pWhat //texto partilhado
)
```

- Utilizando alguma técnica Android assíncrona, utilize postShare, e escreva o Java que permita que, na sequência dos eventos que conduzem a um insert, se faça também a publicação no http url que estiver configurado em SETTINGS_DB.

- O que aconteceria se se utilizasse postShare diretamente, sem preocupações de assincronia?

Se se tentasse invocar postShare, diretamente na user thread, haveria uma exceção de runtime, porque essa operação sobre um recurso externo não é permitida, por poder causar situações ANR (App Not Responding).

```
class MyAsyncTaskThatPostsToAWebServiceTheWhenAndTheWhatThatWasShared
extends AsyncTask <String /*inputs*/, Void /*progress*/, String /*return of doInBackground*/> {

    @Override
    protected String doInBackground(String... strings) {
        String strWhen = strings[0];
        String strWhat = strings[1];
        String strServerResponse =
            postShare(
                MainActivity.this.mCurrentHttpUrl,
                strWhen,
                strWhat
            );
        return strServerResponse;
    }//doInBackground
}//MyAsyncTask
```