MessageService

Modulo: Phone

public class MessageService extends WearableListenerService {  
 Intent messageIntent;  
  
 String attendancePath = "/attendance";  
 String payloadName = "attendance";  
  
 String refreshPath = "/refreshSchedule";

public MessageService() {  
 }  
  
 @Override  
 public void onMessageReceived(MessageEvent messageEvent){  
 if (messageEvent.getPath().equals(attendancePath)){  
  
 final String message = new String(

messageEvent.getData());  
 messageIntent= new Intent();  
 messageIntent.setAction(Intent.*ACTION\_SEND*);  
 messageIntent.putExtra(payloadName, message);  
  
 LocalBroadcastManager.*getInstance*(this)

.sendBroadcast(messageIntent);  
  
  
 } else if (messageEvent.getPath().equals(refreshPath)) {  
 //*TODO : implementare richiesta di aggiornamento* } else{  
 super.onMessageReceived(messageEvent);  
 }  
 }  
}

MessageService è una classe figlia di Wearable- ListenerService.

La sua funzione è gestire la ricezione e il filtraggio di MessageEvents mandati dal modulo Wearable

**attendancePath** e **refreshPath** sono stringhe uniche che identificano i messaggi e permettono di classificarli e richiamare i rispettivi Intent. **payloadName** è il ‘carico’ che viene trasportato dai messaggi di attendance

Alla ricezione del messageEvent si confronta il path contenuto in esso con i path sopra presentati

Se il path ricevuto è **attendancePath ->**Viene chiamata un’ istanza di LocalBroadcast- Manager, che lancia un broadcast all’activity attiva.

Se il path ricevuto è **refreshPath ->**Un intent viene lanciato a Phone\_Activity\_Main con la richiesta di mandare un oggetto Schedule contenente la lista di lezioni odierne

Phone\_Activity\_AddNewSchedule

Modulo: Phone

public class Phone\_Activity\_AddNewSchedule extends  
 AppCompatActivity {  
 private String[] CdL; //TODO ottenerle da internet  
 private String[] years; //TODO ottenerle da CdL  
 final String pathName = getApplicationContext()  
 .getFilesDir() + "next";  
 final InputStream tempResource = getResources()  
 .openRawResource(R.raw.schedule\_stubs);  
  
//TODO implementare priorità di inserimento(cdl-anni)  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(  
 R.layout.phone\_activity\_addnewschedule);

final TextView textField = findViewById(  
 R.id.nameFileField);  
 Button button = findViewById(  
 R.id.createButton);  
 button.setOnClickListener(  
 new View.OnClickListener() {  
  
 @Override  
 public void onClick(View v) {  
 if (textField.getText() != null) {  
 String fileName= textField.getText();  
 createfile(tempResource, fileName);  
 try {  
 tempResource.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 Toast.makeText(  
 getApplicationContext(),

"File created",

Toast.LENGTH\_SHORT).show();

Intent intent = new Intent(getBaseContext(),   
 Phone\_Activity\_Main.class);  
 intent.putExtra("REFRESH", true);  
 startActivity(intent);

} else {

Toast.makeText( getApplicationContext(),  
 "Insert file name",   
 Toast.LENGTH\_SHORT).show();  
 }  
  
 }  
 });  
 }

**Phone\_Activity\_AddNewSchedule** è una Activity.

La sua funzione è rispondere alle interazioni dell’utente sul layout **phone\_activity\_addnew-schedule** per creare nuovi file e salvarli all’interno della memoria dispositivo.

**CdL** e **years** sono entrambe Array di stringhe atte a memorizzare gli input ottenuti dal web rispettivamente: corsi di laura e quanti anni serve conseguire ciascuno di questi.

textField è il campo in cui l’utente immette il nome del file che vuole creare.

button è il pulsante che crea il file e reindirizza alla main activity.

Viene impostato un OnClickListener su button, questo verifica che alla pressione del pulsante ci sia un testo immesso in textField, finalizza quel valore e richiama la funzione **createfile**.

Il Toast rimanda un feedback all’utente, indicando che il file è stato creato o se manca il nome dello stesso. In futuro dovranno essere riempiti anche i campi di corso di laurea e anni.

Viene creato un intent verso la classe **Phone\_Activity\_Main**  di aggiornare il frammento di visualizzazione lista orari e quindi fare un cambio di activity.

private void createfile(InputStream iStream, String fileName){  
 File pathFile = new File(pathName);  
 FileOutputStream oStream;  
 fileName = pathName + File.*separator* + fileName + ".xml";  
   
 if (!pathFile.isDirectory()){  
 pathFile.mkdir();  
 }  
   
 try {  
 File newFile = new File(fileName);  
 if(!newFile.isFile()) {  
 newFile.createNewFile();  
  
 oStream = new FileOutputStream(newFile);  
 IOUtils.*copyStream*(iStream, oStream);  
  
 oStream.flush();  
 oStream.close();  
 } else {  
 Toast.*makeText*(getApplicationContext(), "File   
 already existing, try another name",   
 Toast.*LENGTH\_SHORT*);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}  
  
class DownloadFileFromURL extends AsyncTask<String, String, String> {  
  
 @Override  
 protected String doInBackground(String[] f\_url) {  
 int count;  
 try {  
 URL url = new URL(f\_url[0]);  
 url.openConnection().connect();  
  
 InputStream input = new   
 BufferedInputStream(url.openStream(),  
 8192);  
 //*TODO salvare input* input.close();  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
 return null;  
 }  
  
  
}

**Createfile** verifica che sia presente nel dispositivo la cartella ‘next’ nella giusta directory, se assente la crea. Procede poi a verificare la presenza del file nella suddetta, se presente restituisce tramite Toast il messaggio “file già esistente” suggerendo di cambiare il nome.

La creazione del file consiste nel flusso di dati tra inputStream e outputStream (il file stesso) e la chiusura di questi flussi.

**DownloadFileFromURL** eredita i metodi di una Task asincrona così da non dover interrompere l’interfaccia mentre è in funzione. Serve a scaricare in background in corrispondenza di un URL dei file. Verrà implementata meglio restituendo i file e parametri scaricati dal sito ufficiale edll’orario del PoliBa.

Phone\_Activity\_DailySchedulePager

Modulo: Phone

public class Phone\_Activity\_DailySchedulePager extends FragmentActivity {  
 String dayText;  
 private static final int *NUM\_PAGES* =5;  
 private ViewPager mPager;  
 private PagerAdapter pagerAdapter;  
 private Schedule schedule;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*phone\_activity\_schedulepager*);  
 Bundle scheduleBundle = getIntent().getExtras();

mPager = findViewById(R.id.*pager*);  
 pagerAdapter = new   
 ScreenSlidePagerAdapter(getSupportFragmentManager());  
 mPager.setPageTransformer(true, new   
 ZoomOutPageTransformer());  
 mPager.setAdapter(pagerAdapter);  
   
 TextView dayTextField = findViewById(R.id.*dayText*);  
 dayTextField.setText(dayText);  
 }

@Override  
 public void onBackPressed() {  
 if (mPager.getCurrentItem() == 0) {  
 super.onBackPressed();  
 } else {  
 mPager.setCurrentItem(mPager.getCurrentItem() - 1);  
 }  
 }  
   
 public void setDay(int d){  
 switch (d){  
 case (Calendar.*TUESDAY*):  
 dayText = getString(R.string.*tuesday*);  
 break;  
 case (Calendar.*WEDNESDAY*):  
 dayText = getString(R.string.*wednesday*);  
 break;  
 case (Calendar.*THURSDAY*):  
 dayText = getString(R.string.*thursday*);  
 break;  
 case (Calendar.*FRIDAY*):  
 dayText = getString(R.string.*friday*);  
 break;  
 default:  
 dayText = getString(R.string.*monday*);  
 break;  
 }  
 }

**Phone\_Activity\_DailySchedulePager** è un’Acti-vity che eredita I metodi di FragmentActivity. La sua funzione è di gestire e rispondere alle intera-zioni utente all’interfaccia ***phone\_activity\_sche-dulepager.*** Gestisce i passaggi e le animazioni tra i Fragment **Phone\_Fragment\_DailySchedule** tramite ViewPager.

***NUM\_PAGES*** indica il numero di frammenti tra cui poter scorrere (5 poiché il calendario settimanale va da lunedì a venerdì).

Quando viene creata l’Activity chiede i dati con cui è stata chiamata e li salva nel Bundle **scheduleBundle**, questo permette di aggiornare i valori interni (per essere implementato la classe Stub deve ereditare i metodi di Parceable).

**mPager** viene istanziato e impostato come FragmentManager, viene impostata anche l’animazione ZoomOutTransformer poi definita.

**dayTextField** è l’indicatore della giornata corrispondente al Fragment, per lo stato attuale di implementazione non viene aggiornato

onBackPressed viene eseguita quando l’utente utilizza il pulsante di default per tornare indietro, e fa scorrere indietro nella settimana. In futuro verrà eliminato.

setDay imposta la variabile **dayText** in base al valore int d in input.

private class ScreenSlidePagerAdapter extends   
 FragmentStatePagerAdapter {  
 public ScreenSlidePagerAdapter(FragmentManager fm) {  
 super(fm);  
 }  
  
 @NonNull  
 @Override  
 public Fragment getItem(int position) {  
 return new Phone\_Fragment\_DailySchedule();  
 }  
  
 @Override  
 public int getCount() {  
 return *NUM\_PAGES*;  
 }  
 }

private class ZoomOutPageTransformer implements   
 ViewPager.PageTransformer {  
 private static final float *MIN\_SCALE* = 0.85f;  
 private static final float *MIN\_ALPHA* = 0.5f;  
d  
 public void transformPage(View view, float position) {  
 int pageWidth = view.getWidth();  
 int pageHeight = view.getHeight();  
  
 if (position < -1) { // [-Infinity,-1)  
 // This page is way off-screen to the left.  
 view.setAlpha(0f);  
  
 } else if (position <= 1) { // [-1,1]  
 float scaleFactor =   
 Math.*max*(*MIN\_SCALE*, 1 - Math.*abs*(position));  
 float vertMargin =   
 pageHeight \* (1 - scaleFactor) / 2;  
 float horzMargin =   
 pageWidth \* (1 - scaleFactor) / 2;  
 if (position < 0) {  
 view  
 .setTranslationX( horzMargin - vertMargin /2);  
 } else {  
 view  
 .setTranslationX(-horzMargin + vertMargin /2);  
 }  
  
 // Scale the page down (between MIN\_SCALE and 1)  
 view.setScaleX(scaleFactor);  
 view.setScaleY(scaleFactor);  
  
 // Fade the page relative to its size.  
 view.setAlpha(*MIN\_ALPHA* +  
 (scaleFactor - *MIN\_SCALE*) /  
 (1 - *MIN\_SCALE*) \* (1 - *MIN\_ALPHA*));  
  
 } else { // (1,+Infinity]   
 view.setAlpha(0f);  
 }  
 }  
 }  
}

**ScreenSlidePagerAdapter** eredita i metodi di FragmentStatePagerAdapter per creare un pager efficace.

Il costruttore rimanda al costruttore della classe super.

getItem restituisce l’oggetto che viene mostrato tramite la sua posizione.

ZoomOutPageTransformer è un’animazione di cui non abbiamo credito:   
<https://gist.github.com/xalexchen/9423660>.