



UNIVERSITÀ DI PISA



InformaticaUmanistica

Professioni nascoste:

**La tecnologia e l'innovazione a
sostegno della società**



Laboratorio di Cultura Digitale

Relazione di

Margherita Mori

Gli strumenti tecnologici di ultima generazione, semplici e diretti, sia nello sviluppo che nell'uso, possono giocare un ruolo di supporto nell'interazione quotidiana della vita di una persona? Nel mio percorso mi sono trovata a dare delle risposte positive e così ho sviluppato il mio interesse ed entusiasmo in questo campo. Nella relazione espongo l'utilizzo di questi strumenti in settori di utilità sociale, quali la didattica e il supporto psicologico, affrontando una sfida verso la possibilità di una professione¹ che possa coniugare il mio coinvolgimento negli ambiti psicosociali con le competenze acquisite nel corso di laurea d'Informatica Umanistica.

¹ F. M. Zanzotto 2013, nel seminario <http://www.labcd.unipi.it/seminari/f-m-zanzotto-artisti-per-videogioco/>

Sommario

1.	Introduzione	3
2.	Supporto didattico per l'apprendimento di forme geometriche e numeri	5
2.1	Scratch	6
2.2	Makey Makey	6
3.	Kotlin	7
3.1	Perché Kotlin	7
3.2	App in Kotlin	8
3.2.1	Creazione del layout da codice	8
3.2.2	Salvataggio nella cartella <i>values</i>	9
3.2.3	Controllo dei campi	10
3.2.4	Launch screen	11
3.2.5	WebView	12
3.2.6	DatePicker	13
3.2.7	Dialog	15
4.	Conclusioni	18
4.1	Ricerca attiva del tirocinio	18
4.2	Soft skills	18
4.3	L'innovazione in un prototipo	19
4.4	La tecnologia al servizio delle comunità	19
5	Sitografia	20

1. Introduzione

L'occasione di poter seguire dal vivo e partecipare ai seminari del corso di Cultura Digitale, oltre alla possibilità di accedere al vasto archivio degli anni passati mi ha permesso di allargare l'orizzonte dell'uso delle tecnologie a 360° gradi e ha rappresentato quindi per me un arricchimento. Il mio interesse si è focalizzato sull'uso della tecnologia come supporto sociale.

Già nel 2016 il corso Tecnologie assistive per la didattica mi ha stimolato profondamente dandomi l'occasione di sviluppare un progetto piccolo, ma per me significativo, in cui le competenze teoriche che via via acquisivo potevano trasformarsi in strumenti utili per un settore svantaggiato come ad esempio i bambini non vedenti.

Nel 2017/2018 ho frequentato un corso IFTS nell'ambito dello sviluppo e progettazione di applicazioni informatiche, al termine del quale ho svolto un tirocinio presso una startup dell'Incubatore del Polo scientifico e tecnologico di Navacchio. Mi sono quindi avvicinata, con curiosità mista ad apprensione, a questa realtà, piccola e dinamica. Ho assistito e partecipato fin dall'inizio alla creazione di un'applicazione mobile, in particolare un'applicazione mobile per il sociale richiesta da un'associazione che si occupa della tragedia dell'amnesia dissociativa. Nel caso specifico si tratta del disturbo dissociativo per cui un genitore, abituato alla routine di ogni giorno, può dimenticarsi di accompagnare il figlio all'asilo lasciandolo in auto purtroppo con esito mortale².

Queste due esperienze, apparentemente lontane, mi hanno permesso di interessarmi sempre più al mondo della tecnologia per il sociale oltre a sviluppare nuove competenze informatiche, come lo sviluppo di un'applicazione Android. Infatti l'applicazione di cui sopra è stata richiesta in linguaggio nativo ed è stato scelto Kotlin; nel paragrafo 3.1 verranno spiegate le motivazioni di questa scelta.

In questa relazione presento lo sviluppo di queste due esperienze: nel primo caso l'esperienza, pur limitata ad un prototipo creato con mezzi "handmade", ha avuto più una valenza di stimolo motivazionale. Nel secondo caso mi concentrerò sul linguaggio di programmazione Kotlin e sul suo utilizzo.

² http://www.ansa.it/sito/notizie/cronaca/2017/06/07/bimba-morta-in-auto-un-dramma-che-si-ripete_07f3cdd9-5f8e-40e4-9396-d46ba2be3993.html

Ciò che lega questi due momenti rappresenta per me un percorso di apprendimento che estendendo i miei interessi e motivazioni spero mi porti ad un possibile sbocco professionale. Come ha detto il professor F.M. Zanzotto³ nel suo incoraggiante seminario⁴ sulla ricerca delle professioni nascoste l'essenziale è "trovare degli strumenti facili da utilizzare che ci consentano di fare delle cose che prima erano impensabile".

³ <http://art.uniroma2.it/zanzotto/>

⁴ <http://www.labcd.unipi.it/seminari/f-m-zanzotto-artisti-per-videogioco/>

2. Supporto didattico per l'apprendimento di forme geometriche e numeri

“Le Tecnologie Assistive (TA) sono diventate parte integrante della vita di ogni persona che vive una condizione di disabilità. Le TA sono la chiave di volta di un processo educativo, moderno ed efficace, che si rivolge agli studenti disabili. Esse possono supportare e aiutare la piena partecipazione di questi studenti al processo di apprendimento, sia permettendo loro di superare il danno o la menomazione, sia superando le barriere create dalle tradizionali metodologie educative. Le Tecnologie Assistive sono, o possono diventare, uno degli elementi più rilevanti per la realizzazione di una pedagogia veramente inclusiva: esse incidono profondamente sia in ambito educativo sia lavorativo, e determinano la possibilità di integrazione nella scuola e nella società. Esse comprendono ogni strumento, sistema o servizio che sostiene le persone disabili nella loro vita quotidiana, nell'educazione, nel lavoro, nel tempo libero.”⁵

Nel corso Tecnologie assistive per la didattica ho lavorato alla costruzione di un prototipo per bambini con disabilità sensoriali visive in età pre-scolare progettato insieme alla collega Martina Valeriani. Il progetto nasce per far fronte alla mancanza di strumenti e tecnologie d'assistenza per l'apprendimento di numeri e forme geometriche, che risulta essere difficile nei bambini ipovedenti e non vedenti.

Abbiamo così creato con la plastilina alcune forme geometriche e numeri tridimensionali con la trascrizione corrispettiva in braille impressa sulla superficie (vedi figura 2). L'aver messo il braille serve sia per facilitare l'apprendimento dell'alfabeto braille ai bambini con disabilità visive, sia può essere utile per avvicinare i bambini con altre disabilità o normodotati a una realtà a loro estranea ma molto vicina.

Le creazioni in plastilina sono state poi collegate con Makey Makey⁶ al computer e sul computer abbiamo programmato un breve script su Scratch⁷ che esegue una serie di funzioni: quando il bambino prende in mano un numero, parte l'audio che dice la parola che rappresenta il numero cosicché il bambino mentre scopre la tridimensionalità di quella forma può anche ascoltare come si dice e impararlo. Abbiamo scelto di creare noi l'audio, non solo per renderlo più familiare e meno estraneo, ma anche per far sì che un genitore o un insegnante possa a sua volta registrare le parole.

⁵ Besio S., Tecnologie assistive per la disabilità, Lecce, Pensa Multimedia, 2005

⁶ <https://makeymakey.com/>

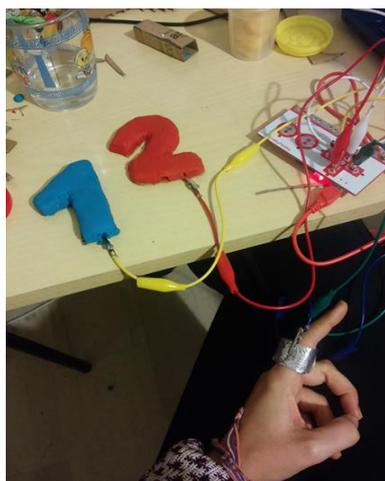
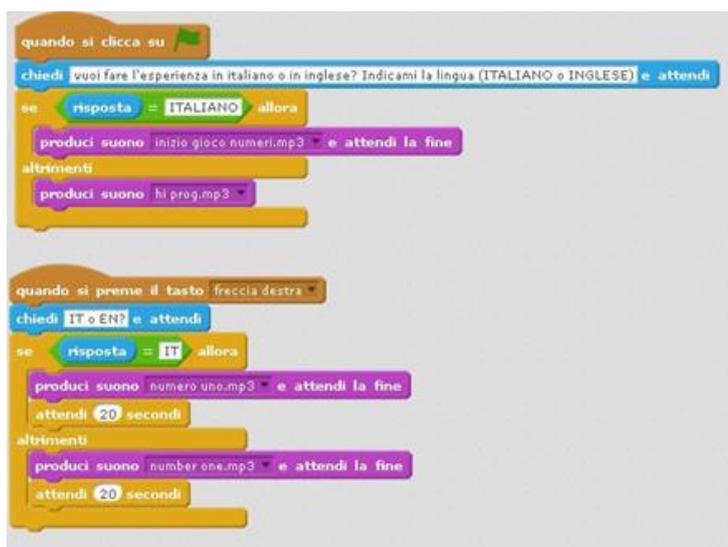
⁷ <https://scratch.mit.edu/>

Oltre all'accrescimento delle capacità spazio-cognitive, questo progetto aiuta infatti anche l'inclusione scolastica, poiché coinvolge sia bambini con disabilità sensoriali visive ma anche con disabilità cognitive o normodotati.

2.1 Scratch

Scratch è un ambiente di sviluppo gratuito con un linguaggio di programmazione a blocchi. Si basa sul concetto "drag and drop" di questi blocchi che possono essere spostati graficamente dall'utente. La sua interfaccia molto intuitiva e accessibile si presta bene all'insegnamento facilitando sia gli insegnanti che bambini o studenti alle prime armi con la programmazione. E' un progetto del Lifelong Kindergarten Group dei Media Lab del MIT.⁸

Nella figura 1, un esempio dal nostro script. Infatti è possibile scegliere anche la lingua, italiano o inglese.



2.2 Makey Makey

Il progetto Makey Makey è iniziato attraverso una campagna su Kickstarter⁹ dai due inventori di Scratch e ha raggiunto oltre 500,000\$. Si tratta di un "kit per l'invenzione che incoraggia le persone a trovare modi creativi per interagire con i loro computer usando oggetti quotidiani al posto di tastiera e mouse"¹⁰. Nella figura 2 sono presenti i numeri collegati al Makey Makey e il Makey Makey collegato al dito che quando tocca i numeri crea un output.

⁸ <http://www.di.unito.it/~capecchi/scratch.pdf>

⁹ <https://www.kickstarter.com/>

¹⁰ Libera traduzione da

http://www.iisrocetticerulli.gov.it/Documenti/Formazione_Docenti/Formazione_Ambito_5_Teramo_2/2017_06_29_Making_e_Tinkering.pdf

3. Kotlin

Kotlin è un linguaggio di programmazione nato nel 2011 fortemente tipizzato ed ispirato a linguaggi come Java, Go, F#. Al Google I/O¹¹ di maggio 2017 è stato annunciato essere il 'first-class language for Android'¹². In questo anno il numero di applicazioni scritte in Kotlin sul Play Store è aumentato notevolmente, è infatti cresciuto di sei volte¹³.

3.1 Perché Kotlin

Scegliere Kotlin è stata una sfida perché nel team di cui facevo parte nessuno aveva mai lavorato con questo linguaggio ma proprio per le motivazioni elencate di seguito e grazie alla professionalità delle persone con cui mi sono trovata a lavorare, è stata fatta una ricerca mirata nel linguaggio riuscendo così a sfruttarne le migliori rispetto a Java per Android. Ecco alcune dei miglioramenti e caratteristiche che abbiamo potuto sfruttare e notare:

- È un linguaggio conciso ma allo stesso tempo espressivo.
- Il punto e virgola ';', simbolo per eccellenza di fine riga in Java così come in altri linguaggi, è considerato superfluo.
- Per richiamare le View dal codice non è più necessaria la funzione findViewById()¹⁴, è sufficiente quindi scrivere l'id della View cui si vuol far riferimento.
- E' possibile estendere le ViewGroup grazie alla funzione inflate¹⁵.

¹¹ <https://www.theverge.com/2017/5/17/15654988/google-jet-brains-kotlin-programming-language-android-development-io-2017>

¹² <https://android-developers.googleblog.com/2017/11/update-on-kotlin-for-android.html>

¹³ <https://www.infoq.com/news/2018/06/kotlin-news-android>

¹⁴ `import kotlinx.android.synthetic.main.content_main.*`

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
        // No need to call findViewById(R.id.textView) as TextView  
        textView.text = "Kotlin for Android rocks!"  
  
    }  
}
```

¹⁵ `// Extend ViewGroup class with inflate function
fun ViewGroup.inflate(layoutRes: Int): View {`

- È un linguaggio JVM e quindi è 100% interoperabile con Java.
- Come Android, è un progetto open source Apache 2.0.
- Molte applicazioni tra le più conosciute sono state costruite in Kotlin: Evernote, Netflix, Slack, Udacity, Wordpress, Expedia, New York Times...

3.2 App in Kotlin

Come descritto nel capitolo 1, l'applicazione è stata creata per essere un supporto quotidiano nella vita dei genitori con figli in età prescolare frequentanti nidi e scuole dell'infanzia (asili nido e scuole materne). Qualora il genitore non segnalasse l'assenza del figlio tramite l'applicazione, gli verrebbe notificata l'assenza in modo da prevenire l'eventualità di lasciare il figlio in auto dove purtroppo può rischiare la vita.

Nei prossimi paragrafi mi concentrerò su alcuni aspetti fondamentali per l'avvio della produzione della maggioranza delle applicazioni con riferimenti a quella su cui ho lavorato durante lo stage: creazione del layout da codice (3.2.1), salvataggio nella cartella values (3.2.2), controllo dei campi (3.2.3), launch screen (3.2.4), WebView (3.2.5), DatePicker(3.2.6), Dialog(3.2.7).

3.2.1 Creazione del layout da codice

Il layout delle applicazioni Android viene comunemente creato dall'apposita sezione: o in XML o nella sezione design. A volte però è necessario creare il layout da codice (in questo caso da Kotlin) se per esempio i dati da mostrare sono collegati al server/database e quindi non prefissati.

Vista la necessità di creare dei bottoni dinamici per la lista dei possibili figli collegati all'account del genitore, il layout di questa activity¹⁶ è stato scritto da codice. Per ogni figlio è stato creato un bottone con sopra, graficamente parlando, l'icona di un bambino, il nome e cognome del figlio, e infine l'icona di una freccia. Ecco di seguito il codice commentato con tutti i settaggi.

//l'index fa riferimento al numero del bambino nell'array della lista dei figli del genitore

```

return LayoutInflater.from(context).inflate(layoutRes, this, false)
}

=====

// Call inflate directly on the ViewGroup instance
override fun onCreateView(parent: ViewGroup, viewType: Int): ViewHolder {
    val v = parent.inflate(R.layout.view_item)
    return ViewHolder(v)
}

```

¹⁶ <https://developer.android.com/reference/android/app/Activity>

```

fun imageChildCreate(index: Int){
    //inizializzazione e dichiarazione dell'immagine
    imageChild = ImageView(this)
    //settaggio della larghezza e altezza
    imageChild.layoutParams = RelativeLayout.LayoutParams(100, 95)
    //settaggio di quale immagine da estrarre dalle risorse
    imageChild.setImageResource(child_image)
    //settaggio del background
    imageChild.setBackground(ContextCompat.getDrawable(this, child_girl))
    //settaggio del padding
    imageChild.setPadding(0, 3, 0, 3)
    //settaggio dei margini
    var paramImageChild = imageChild.layoutParams as RelativeLayout.LayoutParams
    if (index != 0){
        //se non è il primo figlio allora il margine Top viene incrementato
        marginTop = 32+(95+16)*index
    }
    paramImageChild.setMargins(40, marginTop, 0, 0)
    //cambiamento del colore dell'immagine
    imageChild.setColorFilter(Color.WHITE)
    //aggiunta dell'immagine al ViewGroup
    relative_children_list.addView(imageChild)
}

```

Per motivi di spazio non sono state inserite le funzione per creare il bottone e quella per creare la freccia, comunque simili alla funzione per l’inserimento dell’immagine del bambino.

3.2.2 Salvataggio nella cartella *values*

La cartella *values* contiene la definizione dei valori di base in XML, come le stringhe e i colori (*value resource files*). Questa *resource folder* si trova in *res/value*. È da far notare che questa cartella non è una particolarità di Kotlin, ma di Android Studio.

È molto utile per poter organizzare il lavoro e, in caso di modifiche ad un colore o ad una stringa, è sufficiente cambiare nel file quel valore.

Nel file dei colors.xml è possibile trovare

```
<color name="colorDrawerBackground">#3c85da</color>
```

ove cambiando il valore del colore, questo stesso verrà cambiato ovunque viene richiamato.

```
<string name="login_title">Accesso</string>
```

Questa invece è un esempio di stringa col suo nome e il suo valore nel file strings.xml .

3.2.3 Controllo dei campi

Il controllo dei campi inseriti dall'utente nei form è molto importante perché se non eseguito correttamente può causare degli errori nella connessione col server o anche più concretamente se un'email è stata inserita sbagliata poi non potrà essere più ricontattata quella persona.

Innanzitutto è necessario controllare che le EditText¹⁷ non siano vuote:

```
fun checkFields() : Boolean{
    if (!checkField(edit_name_reg) || !checkField(edit_surname_reg) ||
        !checkEmail(edit_email)|| !checkPhone(edit_phone_1) ||
        !checkField(edit_username_reg) || !checkField(edit_password_reg)){
        return false
    }
    return true
}

fun checkField( editText: EditText) : Boolean{
    if (editText.text.toString() == ""){
        editText.setBackgroundResource(R.drawable.background_edit_error)
        editText.clearFocus()
        showToast("Registrazione impossibile")
        return false
    }
    return true
}
```

Con la funzione checkField, se il testo di una EditText è vuoto, gli viene settato uno sfondo col contorno rosso (il drawable background_edit_error) e gli viene tolto il focus. Con la funzione checkFields si controlla ogni EditText richiamando checkField. Per controllare l'inserimento dell'email e del telefono è necessario un ulteriore controllo per assicurarsi che vengano inserite email valide e numero di telefono con effettivamente solo numeri. Vengono quindi usate le espressioni regolari¹⁸:

```
var emailPattern : String = "[a-zA-Z0-9.!#$%&'*/=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9]{0,61}[a-zA-Z0-9])?(?:\\.[a-zA-Z0-9](?:[a-zA-Z0-9]{0,61}[a-zA-Z0-9])?)$"
var phonePattern : String = "[0-9]+"
```

¹⁷ <https://developer.android.com/reference/android/widget/EditText>

¹⁸ <https://medium.com/factory-mind/regex-tutorial-a-simple-cheatsheet-by-examples-649dc1c3f285>

```

fun checkEmail ( editText: EditText) : Boolean{
    if (matches(this.emailPattern, editText.text)){
        return true
    } else{
        editText.setBackgroundResource(R.drawable.background_edit_error)
        editText.clearFocus()
        showToast("Email errata")
        return false
    }
}

```

Infine, dopo che l'utente ha cliccato sul bottone per inviare i dati, nel caso in cui non risultino validi o vuoti e quindi i bordi dell'EditText diventino rossi (background_edit_error), cliccandoci nuovamente sopra per scrivere il testo giusto viene richiamata la callback onFocusChange¹⁹ in modo da reimpostare lo sfondo normale dell'EditText (il drawable background_edit_text).

```

override fun onFocusChange(v: View?, hasFocus: Boolean) {
    if (hasFocus){
        v?.setBackgroundResource(R.drawable.background_edit_text)
    }
}

```

3.2.4 Launch screen

Il launch screen, o splash screen, è la prima schermata che l'utente visualizza all'avvio dell'applicazione²⁰. Nelle linee guida del material design²¹, Google incoraggia l'inserimento del launch screen nelle applicazioni Android. Vi sono due tipi consigliati: il primo è un placeholder UI, dove è presentato il layout dell'applicazione vuoto, in attesa di essere riempito col contenuto, l'altro è lo splash screen brandizzato col logo dell'applicazione.

Lo splash screen brandizzato è quello che è stato scelto per questa applicazione. Nella cartella res/drawable si crea il layout dello splash screen

¹⁹ <https://developer.android.com/reference/android/view/View.OnFocusChangeListener>

²⁰ https://www.wlsdevelop.com/index.php/it/blog?option=com_content&view=article&id=19

²¹ <https://material.io/design/communication/launch-screen.html#branded-launch>

```

<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
  <item
    android:drawable="@color/colorDrawerBackground"/>

  <item>
    <bitmap android:src="@drawable/logo_splash"
      android:gravity="center"></bitmap>
  </item>
</layer-list>

```

Il layer-list è un drawable che contiene un array di drawable²². Ogni drawable è rappresentato dall'item.²³ Nel primo item viene inserito il colore di sfondo, nel secondo si crea il figlio bitmap nel quale viene fatto il collegamento con il drawable del logo e se ne setta la posizione all'interno dello schermo con gravity. Nel file styles.xml definiamo lo stile dello splashscreen richiamando il drawable appena creato.

```

<item name="android:windowBackground">@drawable/background_splash_logo</item>

```

Nell'AndroidManifest.xml si imposta l'activity dello splash screen come la prima schermata che apparirà all'avvio dell'applicazione:

```

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

```

3.2.5 WebView

La WebView è una View²⁴ che mostra pagine web²⁵. Usa il WebKit per mostrare le pagine web e include vari metodi per navigare nella cronologia, zoomare e altri strumenti. Per fare in modo che l'applicazione abbia accesso ad internet è necessario inserire nell'AndroidManifest.xml questa stringa:

```

<uses-permission android:name="android.permission.INTERNET" />

```

Di default la WebView non abilita Javascript e gli errori nelle pagine web sono ignorati. Per includere la WebView nell'activity è necessario includere una WebView nel layout.

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

```

²² <https://developer.android.com/guide/topics/resources/drawable-resource>

²³ <https://developer.android.com/guide/topics/resources/drawable-resource#LayerList>

²⁴ <https://developer.android.com/reference/android/view/View>

²⁵ <https://developer.android.com/reference/android/webkit/WebView>

```

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=".activities.WebViewActivity">

<WebView
    android:id="@+id/web_view"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>

```

Vi sono molte possibili customizzazioni per la WebView: come client è stato scelto di usare la sottoclasse `WebViewClient`. Verrà chiamata quando succede qualcosa che impatta il rendering del contenuto, per esempio per la gestione di errori. Per quanto riguarda i `WebSettings`²⁶, è stato abilitato Javascript.

```
webSettings.javaScriptEnabled = true
```

3.2.6 DatePicker

Il `DatePicker`²⁷ fornisce un widget per selezionare una specifica data. Questo può essere molto utile nelle applicazioni di viaggi per definire un lasso temporale ma anche per inserire una data singola in un form. In questa applicazione è stata usata per inserire la data di nascita al momento della registrazione del genitore o quando si inserisce un nuovo figlio.

Per cominciare viene inserita una `TextView`²⁸ all'interno del layout. Viene quindi settato l'id e tutti gli altri parametri (altezza, larghezza, margini, sfondo, font del testo, posizione del testo all'interno del campo, padding, il testo che deve comparire, il colore e la grandezza del testo, i constraints) per renderla più simile possibile alle altre `EditText` del form. Il testo fa riferimento alla stringa:

```
<string name="select_birt_date">Seleziona la data di nascita</string>
```

²⁶ <https://developer.android.com/reference/android/webkit/WebSettings>

²⁷ <https://developer.android.com/reference/android/widget/DatePicker>

²⁸ <https://developer.android.com/reference/android/widget/TextView>

```

<TextView
    android:id="@+id/tv_select_date_reg"
    android:layout_width="0dp"
    android:layout_height="35dp"
    android:layout_marginTop="8dp"
    android:background="@drawable/background_edit_text"
    android:ems="10"
    android:fontFamily="@font/helvetica_neue_light"
    android:gravity="center_vertical"
    android:paddingLeft="20dp"
    android:text="@string/select_birt_date"
    android:textColor="@color/colorStringDatePicker"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/edit_password_reg" />

```

Successivamente nell'activity in cui vogliamo inserire il datePicker, è stata creata la funzione datePicker() in cui si crea il datePicker. Nella variabile calendar si salva l'istanza del calendario²⁹ e poi con la funzione get si assegna all'anno, mese e giorno una variabile. Il datePicker viene così creato col costruttore in cui è presente il contesto dell'activity, il tema, il listener e le tre variabili che formeranno la data. Con la callback onDataSet³⁰ si salva l'anno, il mese e il giorno in modo poi da settarli come testo dell'EditText.

```

override fun onDataSet(view: DatePicker?, year: Int, month: Int, dayOfMonth: Int) {
    showLog("datepicker:" + year + dayOfMonth)
    var yearDP : String = year.toString()
    var monthDP : String = month.toString()
    var dayDP : String = dayOfMonth.toString()
    chosenDate = dayDP+"-"+monthDP+"-"+yearDP
    tv_select_date_reg.text=resources.getString(R.string.date_birth_child)+" " +chosenDate
}

fun datePicker(){
    val calendar = Calendar.getInstance()
    val year = calendar.get(Calendar.YEAR)
    val month = calendar.get(Calendar.MONTH)
    val day = calendar.get(Calendar.DAY_OF_MONTH)

    var datePicker = DatePickerDialog( context: this, R.style.PickerTheme, listener: this, year, month, day )
    datePicker.show()
}

```

Il testo fa riferimento alla stringa:

```

<string name="date_birth_child">Data di nascita: </string>

```

²⁹ <https://developer.android.com/reference/java/util/Calendar>

³⁰ <https://developer.android.com/reference/android/app/DatePickerDialog.OnDateSetListener>

Per evitare che vengano inviati dati vuoti al click del bottone, come per le EditText era stata creata la funzione CheckFields, in questo caso viene creata una stringa vuota (chosenDate) e, al momento del settaggio del testo della TextView, viene riempita anche la stringa vuota con la data, così non sarà più vuota. Viene quindi poi controllato con la funzione booleana checkDate se chosenDate è vuota: in caso positivo allora lo sfondo avrà un bordino rosso come nel caso delle TextField vuote. checkDate viene poi richiamata da checkFields insieme alle altre checkField già menzionate sopra.

```
fun checkDate() : Boolean{
    if (chosenDate== null){
        tv_select_date_reg.setBackgroundResource(R.drawable.background_edit_error)
        return false
    }
    return true
}
```

Nell'onCreate³¹ con la callback setOnClickListener³² viene impostato che, una volta cliccato sulla TextView, il datePicker si crea (con la funzione datePicker) e lo sfondo viene impostato come normale.

```
tv_select_date_reg.setOnClickListener(object : View.OnClickListener{
    override fun onClick(v: View?) {
        datePicker()
        tv_select_date_reg.setBackgroundResource(R.drawable.background_edit_text)
    }
})
```

3.2.7 Dialog

Un dialog³³ è una piccola finestra che chiede all'utente di prendere una decisione. Solitamente non riempie tutto lo schermo ed è usato per eventi modali che richiedono che l'utente faccia un'azione per procedere.

In questa applicazione era necessario creare due dialog per chiedere all'utente se fosse sicuro di completare l'operazione: nel caso di disiscrizione del bambino dalla scuola e nel caso dell'invio dell'assenza.

È stato creato quindi il layout con due bottoni ("sì", "no"). Successivamente nella BaseActivity, l'activity che contiene tutte le funzioni comuni alle activity, è stata creata la funzione

³¹ [https://developer.android.com/reference/android/app/Activity#onCreate\(android.os.Bundle,%20android.os.PersistableBundle\)](https://developer.android.com/reference/android/app/Activity#onCreate(android.os.Bundle,%20android.os.PersistableBundle))

³² [https://developer.android.com/reference/android/view/View#setOnClickListener\(android.view.View.OnClickListener\)](https://developer.android.com/reference/android/view/View#setOnClickListener(android.view.View.OnClickListener))

³³ <https://developer.android.com/guide/topics/ui/dialogs>

showAlertAreYouSure nella quale viene creato e settato il dialog. Grazie al builder³⁴ vengono settati i parametri, con il layoutInflater³⁵ viene settato il layout precedentemente creato, e poi viene creato e mostrato.

Con la callback setOnClickListener sul bottone del “sì” viene richiamata la funzione che mostra il progressDialog e quella che disabilita la view per evitare che mentre è mostrata la progressDialog accidentalmente lo user possa toccare lo schermo e dare l’input per altre azioni, infine viene fatto uno switch, che in Kotlin viene chiamato when³⁶, tra le due activity in cui verrà poi richiamata questa funzione: a seconda dell’activity in cui si trova l’utente, l’azione successiva al click sul bottone sarà differente. Infatti in un caso è per segnalare l’assenza e nell’altro caso per segnalare la disiscrizione dalla scuola. Dopo il click sul bottone, l>alert scompare grazie alla funzione hideAlertDialog.

Nel caso in cui l’utente clicchi sul “no”, nella callback viene richiamata solo la funzione per nascondere l’alert.

```
fun showAlertAreYouSure(title : String, message : String, viewGroup : ViewGroup, id : Int, activity: Activity){
    var builder : AlertDialog.Builder = AlertDialog.Builder( context: this)
    var view : View = inflater.inflate(R.layout.areyousure_dialog_layout, root: null)
    builder.setMessage(message)
        .setTitle(title)
        .setView(view)
    mAlertDialog = builder.create()
    mAlertDialog.show()
    showLog("alert dialog is showing?" + mAlertDialog.isShowing.toString())
    if (mAlertDialog.isShowing){
        view.button_yes_dialog.setOnClickListener(object : View.OnClickListener{
            override fun onClick(v: View?) {
                showProgressDialog()
                disableView(viewGroup)

                when (activity.javaClass){
                    AbsenceActivity::class.java -> {
                        mNetworkManager.absenceChild(id)
                    }
                    ChildManagerActivity::class.java -> {
                        mNetworkManager.disconnectFromSchool(id)
                    }
                }
                hideAlertDialog()
            }
        })
        view.button_no_dialog.setOnClickListener(object : View.OnClickListener{
            override fun onClick(v: View?) {
                hideAlertDialog()
            }
        })
    }
}
```

³⁴ <https://developer.android.com/reference/android/app/AlertDialog.Builder>

³⁵ <https://developer.android.com/reference/android/view/LayoutInflater>

³⁶ <https://antonioleiva.com/when-expression-kotlin/>

La funzione che nasconde l'alert richiama semplicemente la funzione dismiss³⁷.

```
fun hideAlertDialog() {  
    mAlertDialog.dismiss()  
}
```

La funzione che crea la progressDialog è molto simile a quella per l>alertDialog. Il layout è invece diverso, infatti oltre alla TextView che avverte l'utente di attendere, viene richiamato un componente di sistema, ovvero quella della ProgressBar³⁸.

```
<ProgressBar  
    android:id="@+id/progressBar"  
    style="?android:attr/progressBarStyle"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="24dp"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

³⁷ <https://stackoverflow.com/questions/14853325/how-to-dismiss-alertdialog-in-android>

³⁸ <https://developer.android.com/reference/android/widget/ProgressBar>

4. Conclusioni

Lavorare a questi due progetti mi ha permesso di cogliere la connessione tra conoscenze informatiche teoriche e la loro applicazione concreta, quindi di “mettere degli occhiali” attraverso cui vedere gli aspetti più concreti che spero mi aspettino dopo la laurea, cioè un ruolo professionale. Oltre alle competenze teoriche, ricche in ambito accademico e non solo, ho verificato quanto sia importante consolidare anche quelle che vengono chiamate soft skills di cui parlerò più approfonditamente nel paragrafo 4.2.

Inoltre prima di ambire a un ruolo professionale è importante sapere come cercarlo. Durante il corso IFTS erano infatti previste ore di orientamento al lavoro che hanno permesso a me e ai miei compagni, provenienti da background diversi (da Master in Applicazioni Android a diploma in Ragioneria) e con prospettive future distanti, di affrontare nuove sfide. Affrontare un colloquio di lavoro, il test di selezione, l’inserimento in un team e, cosa più importante, come effettuare la ricerca attiva del lavoro (paragrafo 4.1) sono tutti passaggi che abbiamo concretamente vissuto già nella ricerca del tirocinio (240 ore in azienda).

4.1 Ricerca attiva del tirocinio

È stata la prima volta per me ed è stato molto stimolante, oltre ad essere una sfida. La ricerca attiva del lavoro è un processo dinamico che richiede organizzazione e metodo.³⁹

Innanzitutto è necessario identificare l’obiettivo che si vuole raggiungere, cioè a quale posizione si è interessati, al tipo di azienda e quali strumenti di ricerca possiamo utilizzare per arrivare alla meta. Inoltre è bene ogni qual volta che si manda il curriculum vitae e la lettera di presentazione prenderne nota: creare quindi un file su Excel è molto utile per monitorare le candidature⁴⁰. In questo modo non ho avuto problemi e ho potuto tenere tutto sotto controllo.

Grazie alla ricerca di questo tirocinio ho potuto fare tre colloqui in tre aziende: è stato molto interessante ed ha arricchito il mio bagaglio culturale. domande

4.2 Soft skills

Come sottolinea Almalaurea le soft skills *“Sono caratteristiche personali importanti in qualsiasi contesto lavorativo perché influenzano il modo in cui facciamo fronte di volta in volta alle richieste*

³⁹ Slides “La ricerca attiva del lavoro” dell’orientatore R. Taccini.

⁴⁰ Personalmente ho creato il file su Excel con queste colonne: nome azienda, temi e linguaggi di programmazione principali, posizione, lontananza dalla propria casa, nome della persona che si occupa delle Risorse Umane (se presente) e email per inviare la candidatura, data in cui ho inviato l’autocandidatura, data dell’eventuale risposta.

dell'ambiente lavorativo."⁴¹ Nel mondo del lavoro sono ormai diventate importanti quanto le hard skills. Lo stage mi ha permesso di acquisirne di nuove e di consolidarne altre.

Ho infatti acquisito più fiducia in me stessa, questo mi ha permesso di credere nelle mie possibilità, così ho inviato le candidature nelle aziende. Ho acquisito una maggiore flessibilità lavorando con persone con background completamente diversi dal mio e in situazioni diverse da quelle cui ero abituata. Ho aumentato la mia resistenza allo stress rimanendo focalizzata sugli obiettivi. Ho consolidato la mia capacità di organizzazione e pianificazione, ho appreso in maniera continuativa non solo durante le lezioni, ma anche durante lo stage dove per 8 ore al giorno ho visto argomenti e metodi nuovi per lavorare. Infine ho migliorato le mie competenze comunicative e la mia predisposizione al team working sia durante le lezioni che durante lo stage lavorando a stretto contatto con il mio tutor aziendale e i suoi collaboratori.

4.3 L'innovazione in un prototipo

Aver avuto la possibilità di progettare un prototipo, con gli strumenti acquisiti a lezione, per migliorare l'apprendimento dei bambini con disabilità visive, come ho già ripetuto, è stata un'occasione di arricchimento emotivo ma soprattutto uno stimolo per capire quale professione potrei un giorno fare. Il corso ha infatti riunito la mia passione per la tecnologia e l'innovazione con la possibilità di aiutare altre persone con disabilità, favorendo non solo l'inclusione ma anche e soprattutto l'apprendimento.

Riflettendo su questa esperienza ha significato per me la messa a punto di un percorso motivazionale.

4.4 La tecnologia al servizio delle comunità

Concludendo oltre agli aspetti già menzionati, le due esperienze che ho presentato in questa relazione sono legate per me intimamente in quanto mi hanno indirizzato verso un campo specifico: sviluppare applicazioni informatiche che abbiano un impatto significativo sul sociale e sul supporto della disabilità rappresenta oggi il mio obiettivo post-laurea. Forse una professione nascosta, forse un ruolo in un'amministrazione o in una startup, quello che mi preme sottolineare è che questo legame per me rappresenta una spinta motivazionale rilevante e spero quindi che si possa realizzare.

⁴¹ <https://www.almalaurea.it/info/aiuto/lau/manuale/soft-skill>

5 Sitografia ⁴²

<http://www.almalaurea.it/>

<https://antoniroleiva.com/>

<https://developer.android.com/>

<https://developer.android.com/kotlin/>

<https://fortyzone.it/launch-splash-screen-android/>

<https://it.quora.com/>

<https://kotlinlang.org/>

<https://kotlinlang.org/docs/reference/comparison-to-java.html>

<https://makeymakey.com/>

<https://material.io/>

<https://medium.com/>

<https://scratch.mit.edu/>

<https://stackoverflow.com/>

⁴² I siti nelle note a piè di pagina e nella sitografia sono tutti visitabili in data 4 settembre 2018.