EasyVersity

Tomassini Danilo
Cappella Simone
Mannini Luca

Ingegneria Informatica e dell'Automazione

Settembre, 2019

Indice

1 Obbiettivi **3**

2 Funzionalità **3**
2.1 Orario 4
2.2 I miei appunti 4
2.3 Appunti condivisi 5
2.4 Impostazioni 6

3 Implementazioni Android studio **7**
3.1 Login e Registrazione 7
3.2 Orario 8
3.2.1 Fragment 9
3.3 I miei appunti 14
3.4 Appunti condivisi 14
3.5 Impostazioni 14

4 Implementazioni Xamarin **14**
4.1 Login e Registrazione 14
4.2 Orario 14
4.3 I miei appunti 14
4.4 Appunti condivisi 14
4.5 Impostazioni 14

1 Obbiettivi

EasyVersity rappresenta uno strumento di supporto per lo studente.
Permette di gestire:

- **Orario:** salva l'orario delle lezioni nella tua applicazione per consultarlo quando vuoi.
- **Archivio appunti locale:** da la possibilità di salvare appunti raggruppandoli per materia, indicando titolo e data si può contestualizzare al meglio l'appunto in questione.
- **Condivisione appunti:** rende possibile la condivisione ed il download degli appunti.
- **Impostazioni:** da qui si possono cambiare informazioni come username e password, eprendere visione di info "about us".

2 Funzionalità

Avviata l'applicazione ci si trova davani alla **First activity**; qui si può scegliere di effettuare il login o registrarsi. Una volta effettuato il login si può accedere al menù principale che prevede 4 scelte:

- **Orario.**
- **I miei appunti.**
- **Appunti condivisi.**
- **Impostazioni.**

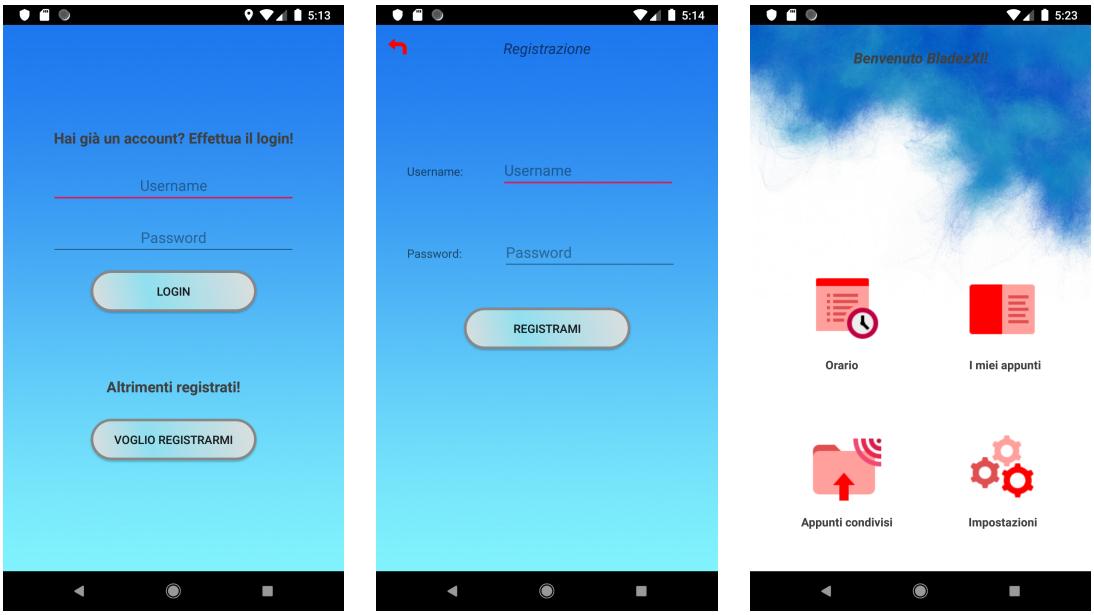

Figura 2.1: Schermata Login, Registrazione e Menu principale.

2.1 Orario

Aperto l'orario ci si trova di fronte alla tabella relativa al giorno corrente; con la bottom navbar si può scegliere il giorno su cui andare ad agire. Premendo nel campo **materia**, relativo all'ora di interesse si può inserire la materia in tabella, si accede, infatti, ad una lista predefinita di materie (questa può essere estesa scegliendo di inserirne una manualmente con il bottone "altro"). Scelta la materia viene visualizzata una finestra di dialogo che permette di inserire l'aula in cui si svolgerà la lezione e la durata della stessa.

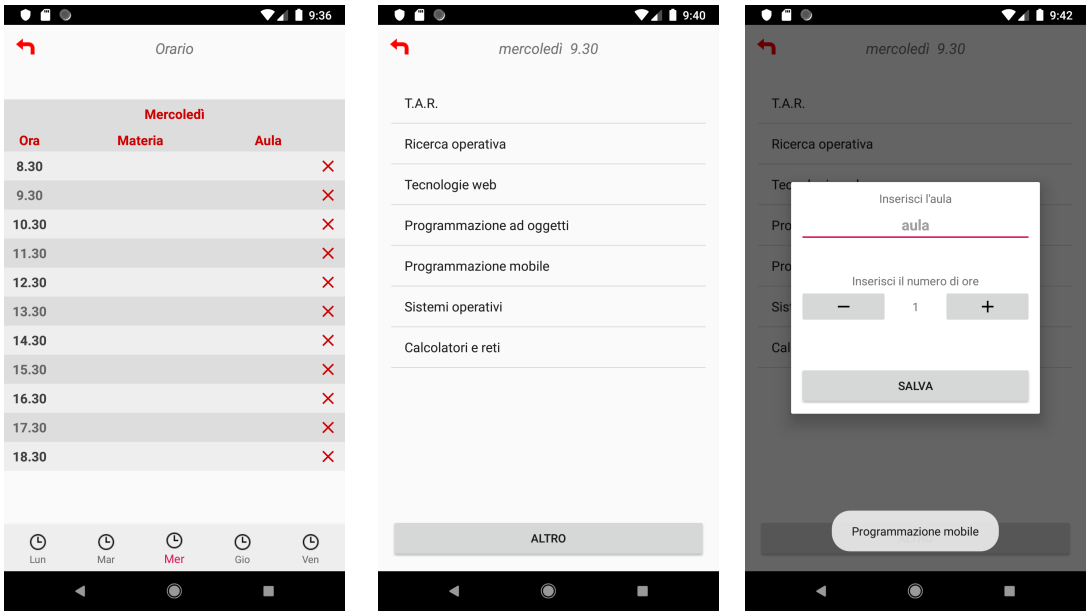

Figura 2.2: Orario.

Salvata la materia nell'orario questa viene salvata nel database e inserita in una lista nella sezione **i miei appunti**.

2.2 I miei appunti

In questa sezione abbiamo accesso alla lista di tutte le materie salvate nel database. Abbiamo la possibilità di aggiungerne altre a piacimento o eliminarle inserendo il nome esatto della materia.

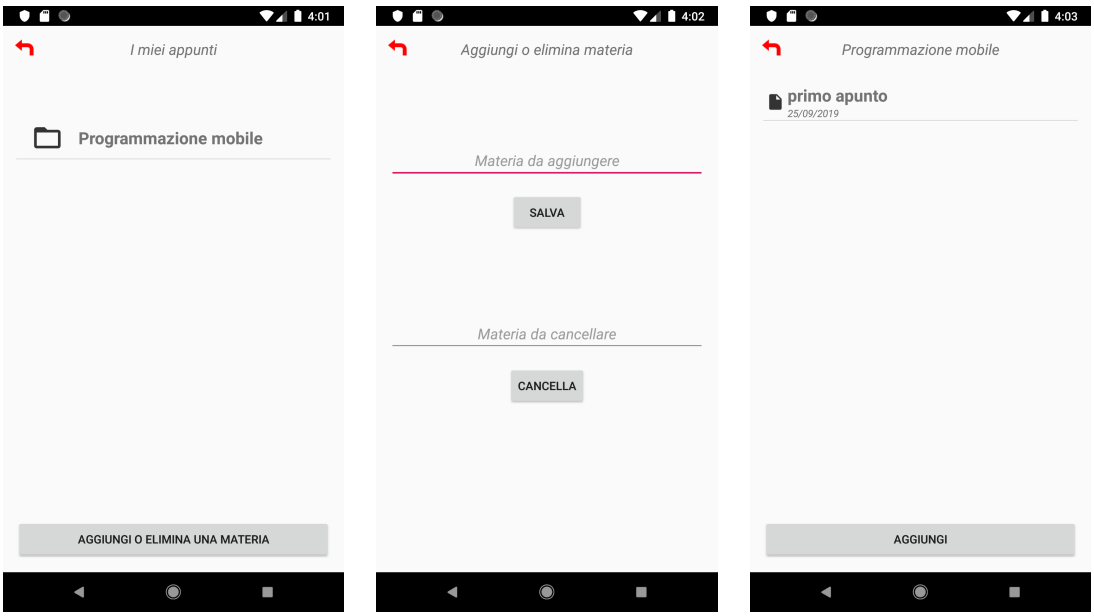

Figura 2.3: Scegli aggiungi o elimina una materia e scegli un appunto.

Scelta la materia accediamo ad una seconda lista contenente tutti gli appunti relativi ad essa; ogni elemento della lsita contiene titolo e data relativi ad ogni appunto. Da questa lista possiamo accedere e visualizzare l'annotazione vera e propria.

Gli appunti possono essere aggiunti con il bottone **"aggiungi"**, qui inseriremo titolo data e annotazione che vogliamo salvare; da questa schermata, inserendo la spunta nel campo **"Condividi"**, possono essere condivisi direttamente gli appunti con gli altri utenti.

Per eliminare un appunto basta tenere premuto l'elemento da eliminare, verrà attivata una finestra di dialogo da cui è possibile confermare la cancellazione.

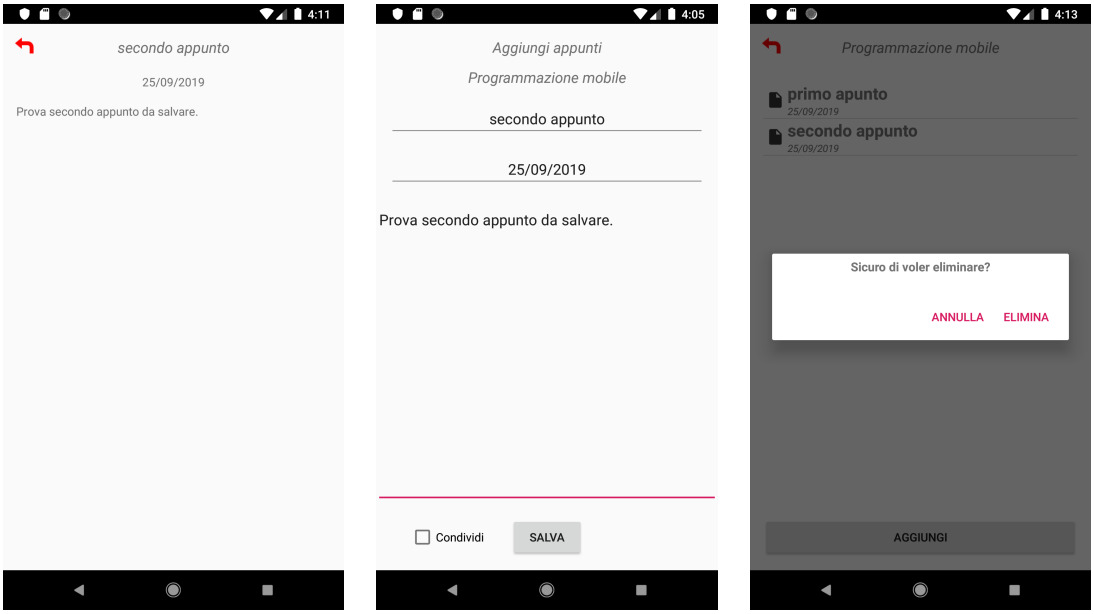

Figura 2.4: Visualizza, aggiungi o elimina un appunto.

2.3 Appunti condivisi

Nella sezione appunti condivisi possiamo scegliere una delle materie predefinite in lista per visualizzare gli appunti che sono stati condivisi ed eventualmente visualizzarli o scaricarli.

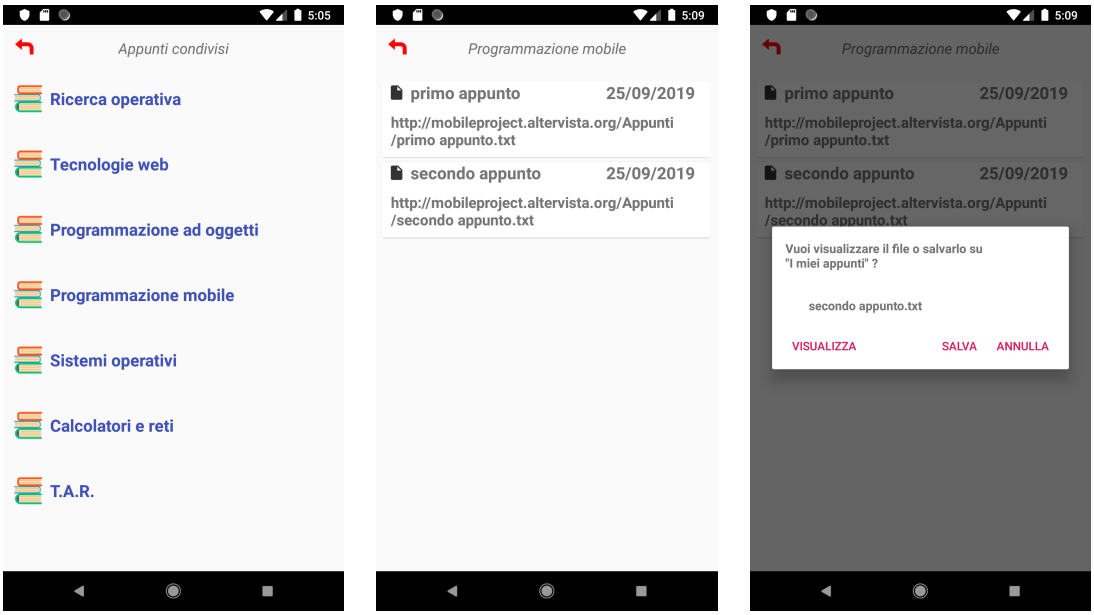

Figura 2.5: Appunti condivisi, scegli la materia, scegli l'appunto, visualizza o scarica.

2.4 Impostazioni

Dalle impostazioni possiamo banalmente modificare **username**, **password** e leggere informazioni **about us**.

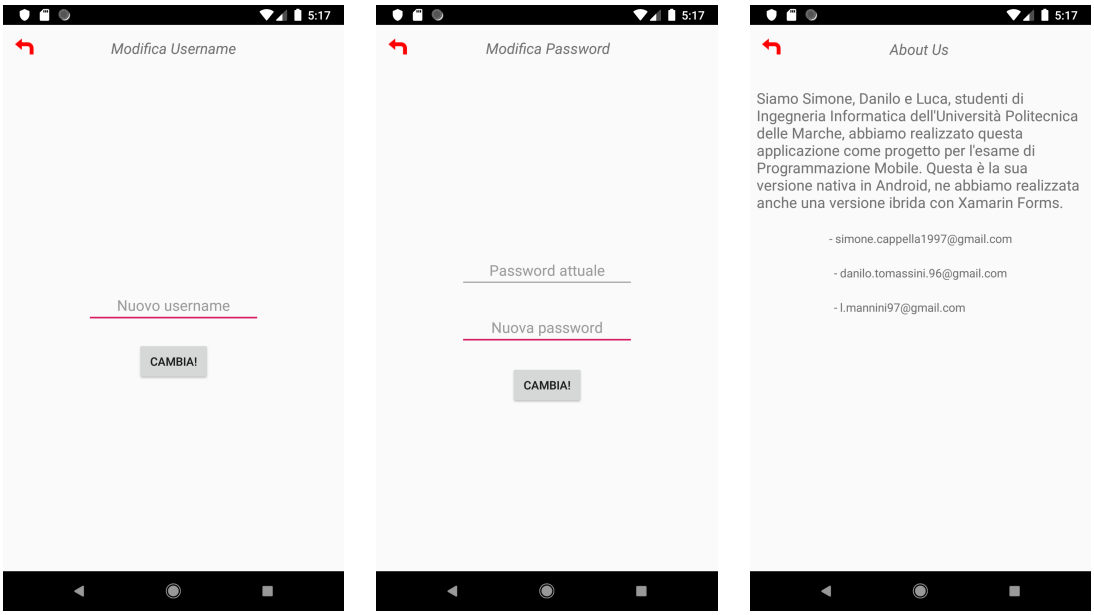

Figura 2.6: Azioni eseguibili nelle impostazioni.

3 Implementazioni Android studio

3.1 Login e Registrazione

Per avviare il login, ovviamente, è necessario riempire le **EditText** relative a username e password; i relativi valori verranno assegnati alle variabili **login_name** e **login_pass**.

La connessione al database viene instaurata indicando **method**, stringa inizializzata in precedenza a **"login"**, dati relativi ad username e password e l'url contenente il percorso fino alla funzione di login contenuta nel server.

Se la funzione del server da esito negativo non verrà effettuato il login e verrà stampato un **Toast**, per mezzo della funzione **show**. Altrimenti verrà stampato un messaggio di benvenuto e verrà lanciata la **MainActivity**.

```
\\FirstActivity\\
[...]
public void userLogin(View view) {
[...]
try {
  Sring url = "http://mobileproject.altervista.org/login.php";
  auth = supportTask.execute(method, login_name, login_pass, url).get();
  } catch (ExecutionException e)
  {
    e.printStackTrace();
  }catch (InterruptedException a)
  {
    a.printStackTrace();
  }
  if(auth.equals("Login Success"))
  {
    launchMainActivity(view);
    show("Benvenuto "+ login_name + "!");
    finish();
  }else
  {
    show("Dati errati. Riprova.");
  }
}
[...]
```

La registrazione viene gestita in modo simile.

In un **if** viene verificato in modo veloce se le condizioni su username e password sono verificate, in caso contrario viene lanciato un avviso.

Il metodo con cui si instaura la connessione al server è lo stesso, cambiano solo gli argomenti, **url** e **method**.

Una volta che la registrazione è andata a buon fine l'attività viene interrotta e si torna alla schermata di login.

```
\\Register\\
[...]
public void userRegister() {
[...]
if (register_name.length() >= 3 && register_pass.length() >= 3)
{
  try {
    String url = "http://mobileproject.altervista.org/register.php";
    auth= supportTask.execute(method, register_name, register_pass, url).get();
  } catch (ExecutionException e) {
    e.printStackTrace();
  } catch (InterruptedException e) {
    e.printStackTrace();
  }
  if (auth.equals("Registrazione avvenuta con successo!")) {
    show("Registrazione effettuata con successo, accedi!");
    finish();
  } else if (auth.equals("Username in uso")) {
    show("Username gia' in uso, prova con uno diverso!");
```

```
  }
}
else
{
show(I campi devono contenere almeno 3 caratteri.);
}
[...]
```

3.2 Orario

Avviata l'activity **orario** vengono catturate le dimensioni del display, calcolata la dimensione del **fragment** contenente la tabella dell'orario e fatto partire uno switch sul giorno corrente.

```
\\Clock\\
[...]
protected void onCreate(Bundle savedInstanceState) {
[...]
Calendar calendar = Calendar.getInstance();
int day = calendar.get(Calendar.DAY_OF_WEEK);
Fragment fragment = null;
switch (day){
  case Calendar.MONDAY:
    fragment = new lun_fragment();
    nav.setSelectedItemId(R.id.lun);
    break;
  case Calendar.TUESDAY:
    fragment = new mar_fragment();
    nav.setSelectedItemId(R.id.mar);
    break;
  case Calendar.WEDNESDAY:
[...]
}
loadFragment(fragment);
[...]
```

Questo switch permette di caricare il fragment relativo al giorno corrente (es. se oggi è lunedì viene caricato il fragment relativo a lunedì); viene, infatti, inizializzato l'oggetto fragment e viene selezionato l'elemento sulla navbar.

```
\\Clock\\
[...]
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
Fragment fragment = null;
switch(item.getItemId())
{
  case R.id.lun:
    next = 0;
    fragment = new lun_fragment();
    break;
  case R.id.mar:
    next = 1;
    fragment = new mar_fragment();
    break;
  case R.id.mer:
    next = 2;
    fragment = new mer_fragment();
    break;
[...]
}
return loadFragment(fragment);
[...]
```

Allo stesso modo la funzione **onNavigationItemSelected**, associa in base allitem della navbar cliccato un'istanza del fragment all'oggetto fragment e nuovamente viene lanciata la funzione **loadFragment**.

La funzione **loadFragment** carica il fragment richiesto e gestisce lo scorrimento a destra o sinistra dipendentemente dalla posizione del fragment corrente rispetto a quello scelto.

3.2.1 Fragment

Prendiamo come riferimento il fragment relativo al **lunedì**, gli altri saranno implementati in modo analogo.

All'interno del fragment vengono nuovamente catturate le dimensioni del display e impostate le dimensioni delle varie caselle della tabella, rispettivamente:

- **Ora:** l'ora di inizio della lezione.
- **Materia:** la materia inserita.
- **Aula:** l'aula nel quale si terrà la lezione.
- **X:** edit per cancellare la riga.

La tabella viene inizializzata nella funzione **onCreate**, qui, ogni elemento della tabella viene assegnato ad una variabile e successivamente "riempito" con il testo salvato, andadolo a recuperare con l'oggetto **sa**, della classe **SalvaOrario**, passando la chiave $\{lun\_1, lun\_2, ..., lun\_11\}$ per quanto riguarda la tabella di lunedì, per la tabella di martedì le chiavi saranno costruite come $\{mar\_1, ..., mar\_11\}$; la stessa logica vale per gli altri giorni; in tal modo possiamo recuperare in modo distinto le 11 materie salvate con la relativa aula per i diversi giorni.

```
\\lun_fragment\\
[...]
public View onCreateView(LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
[...]
SalvaOrario sa = new SalvaOrario();
[...]
txtMat1 = v.findViewById(R.id.materia1);
txtMat1.setText(sa.getMateria("lun_1", getActivity()));
txtMat1.setOnClickListener(this);
txtAula1 = v.findViewById(R.id.aula1);
txtAula1.setText(sa.getAula("lun_1", getActivity()));
txtOra1 = v.findViewById(R.id.ora1);

txtMat2 = v.findViewById(R.id.materia2);
txtMat2.setText(sa.getMateria("lun_2", getActivity()));
txtMat2.setOnClickListener(this);
txtAula2 = v.findViewById(R.id.aula2);
txtAula2.setText(sa.getAula("lun_2", getActivity()));
txtOra2 = v.findViewById(R.id.ora2);
[...]
```

Nella funzione **onClick** viene gestita la scelta di inserire una materia nell'orario o eliminarne una dallo stesso. C'è, infatti, uno switch che gestisce il "click" nelle diverse sezioni della tabella. Abbiamo due diversi tipi di azione:

- **Aggiungere una materia:** per aggiungere una materia si seleziona un campo materia relativo all'ora in cui vogliamo inserirlo, questo attiverà l'elemento corrispondente nello switch che imposterà la variabile *n*, utilizzata successivamente per costruire la chiave e salvare nella classe SalvaOrario, la variabile *ora*, presa direttamente dalla view e chiama la funzione **launchList()**.
- **Elimina una materia:** se viene selezionata la "X" sulla destra della tabella viene semplicemente "pulita" la riga corrispondente, infatti, nello switch, captato l'*edit* corrispondente, viene impostata di nuovo la variabile *n* per essere usata come chiave e le voci *materia* e *aula* vengono messe a *null*, viene lanciata, poi, la funzione **inserisciSalva()**; (la variabile *inc* serve successivamente per inserire in un colpo solo più ore della stessa materia).

```
\\lun_fragment\\
[...]
public void onClick (View v) {
switch (v.getId())
{
  case R.id.materia1:
    n = 1;
    ora = txtOra1.getText().toString();
    v.startAnimation(buttonClick);
    launchList();
    break;
  case R.id.materia2:
    n = 2;
    ora = txtOra2.getText().toString();
    v.startAnimation(buttonClick);
    launchList();
    break;
[...]
  case R.id.edit1:
    n = 1;
    v.startAnimation(buttonClick);
    materia = null;
    aula = null;
    inc = 1;
    inserisciSalva();
    break;
  case R.id.edit2:
    n = 2;
    v.startAnimation(buttonClick);
    materia = null;
    aula = null;
    inc = 1;
    inserisciSalva();
    break;
[...]
```

La finzione **launchList**, chiamata nel momento in cui si vuole aggiungere una materia all'orario non fa altro che lanciare un **intent esplicito**, questo ci permette di scambiare dati tra l'activity chiamante e la chiamata, infatti tra i parametri della funzione **putExtra** abbiamo una chiave, *giorno_ora* e un valore, *"lunedi'" + ora* (*ora* è stata assegnata in precedenza nello switch). Inoltre c'è bisogno di un *REQUEST_CODE* utilizzato come chiave di riconoscimento tra le activity.

```
\\lun_fragment\\
[...]
public static final int REQUEST_CODE = 0000;
[...]
public void launchList() {
  Intent intent = new Intent (getActivity(), List.class);
  intent.putExtra("giorno_ora", "lunedi'  " + ora);
  startActivityForResult(intent, REQUEST_CODE);
}
```

L'activity **List** genera la lsita predefinita delle materie da cui si può scegliere quella da inserire nella casella dell'orario scelta.

Qui i dati vengono recuperati proprio grazie alla chiave *giorno_ora*, successivamente vengono istaziati *myDialog* e *adapter*.

La funzione **setOnItemClickListener** chiamata per mezzo dell'oggetto *listMaterie* e la successiva **OnItemClick** permettono di captare quale materia della lista viene scelta grazie alla sua posizione nella lista stessa.

Scelta la materia viene aperta la finestra di dialogo, all'interno di questa vengono definiti diversi elementi come una editText per l'aula e due bottoni, −+, questi permettono di incrementare e decrementare le ore di lezione relative alla materia.

Una volta inserita l'aula e il numero di ore si provede al salvataggio. Un listener sul bottone *btnSalva* fa si che quando venga restituito il risultato all'activity chiamante con la variabile *intent*, questa contiene la materia scelta, l'aula e la conta delle ore, l'attivity termina con finish(). Una cosa

analoga avviene nel momento in cui si decide tramite l'apposito bottone di inserire una nuova materia non contenuta nella lista predefinita.

```
\\List\\
[...]
Intent intent = getIntent();
String giorno_ora = intent.getStringExtra("giorno_ora");
[...]
myDialog = new Dialog(this);
final ArrayAdapter<String>  adapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, android.R.id.text1, listItem);
listMaterie.setAdapter(adapter);
[...]
listMaterie.setOnItemClickListener(new AdapterView.OnItemClickListener() {
  @Override
  public void onItemClick(AdapterView<?> parent, View view, int position,
  long id) {
    value = adapter.getItem(position);
[...]
myDialog.show();
[...]
btnMeno.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    i--;
    textContatore.setText(Integer.toString(i));
  }
}
[...]
btnPiu.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
  i++;
  textContatore.setText(Integer.toString(i));
  }
}
[...]
btnSalva.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
[...]
    Intent intent = new Intent();
    intent = intent.putExtra("mat", valore);
    setResult(Activity.RESULT_OK, intent);
    finish();
    myDialog.dismiss();
  }
[...]
```

La funzione **onActivityResult** si occupa di reperire dall'activity Lsit i risultati ottenuti dalla scelta. L'array di stringhe *res* contiene ora nella posizione 0 la materia, nella posizione 1 l'aula e nella posizione 2 il numero di ore, che va "parsato" in un intero in quanto arrivava come una Stringa.

Viene considerato il caso in cui non si completa la scelta, in tal caso nell'activity Lista viene impostato il nome della materia a *back* e non viene eseguita nessun'altra azione, la routine si interrompe. Nel caso in cui la scelta è stata completata i dati raccolti andranno salvati, sono passati, dunque, alla funzione **salvaOrario**.

```
\\lun_fragment\\
[...]
public void onActivityResult(int requestCode, int resultCode, Intent data){
  super.onActivityResult(requestCode, resultCode, data);
  if ((requestCode == REQUEST_CODE) && (resultCode == Activity.RESULT_OK)) {
    String[] res = data.getStringArrayExtra("mat");
    materia = res[0];
    aula = res[1];
    inc = Integer.parseInt(res[2]);
```

```
10    }
11    if (materia.equals("back")){}
12    else{
13      salvaOrario(i, materia, aula);
14    }
15  }
16  [...]
```

La funzione **salvaOrario** va a definire un cursore che sarà popolato con l'elemento del database trovato dal metodo **searchM** (questo trova nel database l'elemento il cui campo materia ha lo stesso nome della materia inserita). Questo cursore permette di verificare, grazie all'*if* se la materia inserita è gia presente nel database contenente le materie, in caso positivo il vecchio elemento viene eliminato; successivamente si inserisce nel database delle materie il nuovo elemento e viene chiamata la funzione **inserisciSalva()**.

Le materie vengono inserite nel database delle materie per permettere alla sezione *mieiappunti* di rendere disponibili le materie inserite nell'orario.

```
1  \\lun_fragment\\
2  [...]
3  public void salvaOrario (String key, String materia, String aula) {
4    Cursor c;
5    c = dm.searchM(materia);
6    if (c.getCount() > 0) {
7      dm.delete(materia);
8    }
9    dm.insert(materia, ora, aula, key);
10   inserisciSalva();
11 [...]
```

La funzione **inserisciSalva** permette di salvare, nuovamente tramite i metodi della classe **Salva Orario**, materia e aula scelti e nuovamente di aggiornare i valori delle *txtMat* e *txtAula*.

Il **while** all'interno della funzione ripete il ciclo finchè la variabile *inc* che conteneva la conta delle ore non arriva a 0; la Stringa *q* viene costruita in modo da diventare una chiave per i metodi della classe SalvaOrario, infatti la variabile *i* contiele la stringa "lun_" e la variabile *n* contiene il numero dipendentemente dalla riga scelta dalla tabella. Le variabili *n* ed *inc* alla fine del ciclo vengono rispettivamente incrementata e decrementata, questo permette di salvare in un solo colpo stessa materia e stessa aula all'interno della tabella, infatti il numero dell'elemento della tabella incrementa finche non si esaurisce il numero di ore deciso nella finestra di dialogo nell'activity List.

Lo **switch** su *n* ha il compito di aggiornare immediatamente il valore della tabella nella posizione *n*.

```
1  \\lun_fragment\\
2  [...]
3  public void inserisciSalva(){
4    while (inc > 0){
5      String q = i + n;
6      sa.setMateria(q, materia, getActivity());
7      sa.setAula(q, aula, getActivity());
8      switch (n){
9        case 1:
10         txtMat1.setText(sa.getMateria(q, getActivity()));
11         txtAula1.setText(sa.getAula(q, getActivity()));
12         break;
13       case 2:
14         txtMat2.setText(sa.getMateria(q, getActivity()));
15         txtAula2.setText(sa.getAula(q, getActivity()));
16         break;
17 [...]
18   }
19   n++;
20   inc--;
21 }
22 [...]
```

La classe **SalvaOrario** gestisce salvataggio ed estrazione per mezzo delle **SharedPreferences** di materia e aula.

Al suo interno sono presenti, infatti, quattro metodi:

- **setMateria:** prende tra gli argomenti chiave e materia e va a salvare la materia con tale chiave.
- **getMateria:** va ad estrarre dai salvataggi la materia corrispondente alla chiave di ricerca.
- **setAula:** di nuovo prende tra gli argomenti chiave ed aula (in questo caso la chiave viene combinata con la stringa *"_A"* per caratterizzare le chiavi relative alle aule) salva, dunque, il nome dell'aula.
- **getAula:** estrae l'aula dai salvataggi per mezzo della chiave combinata con la stringa di cui sopra.

```
\\SalvaOrario\\
[...]
public static void setMateria(String key, String value,  Context context) {
  SharedPreferences.Editor editor = preferences.edit();
  editor.putString(key, value);
  editor.commit();
}
public static String getMateria(String key, Context context)
{
  return preferences.getString(key, null);
}
public static void setAula(String key, String value,  Context context) {
  SharedPreferences.Editor editor = preferences.edit();
  editor.putString(key+"_A", value);
  editor.commit();
}
public static String getAula(String key, Context context) {
  return preferences.getString(key+ "_A", null);
}
```

3.3 I miei appunti

3.4 Appunti condivisi

3.5 Impostazioni

4 Implementazioni Xamarin

4.1 Login e Registrazione

4.2 Orario

4.3 I miei appunti

4.4 Appunti condivisi

4.5 Impostazioni