

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
3.3.1	Aggiungi una materia	14
3.3.2	Vedi appunti	15
3.4	Appunti condivisi	19
3.5	Impostazioni	19
3.6	Database	19
4	Implementazioni Xamarin	19
4.1	Login e Registrazione	19
4.2	Orario	19
4.3	I miei appunti	19
4.4	Appunti condivisi	19
4.5	Impostazioni	19
4.6	Database	19

1 Obiettivi

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:** dà 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, prendere visione di info "about us".

2 Funzionalità

Avviata l'applicazione ci si trova davanti 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.**

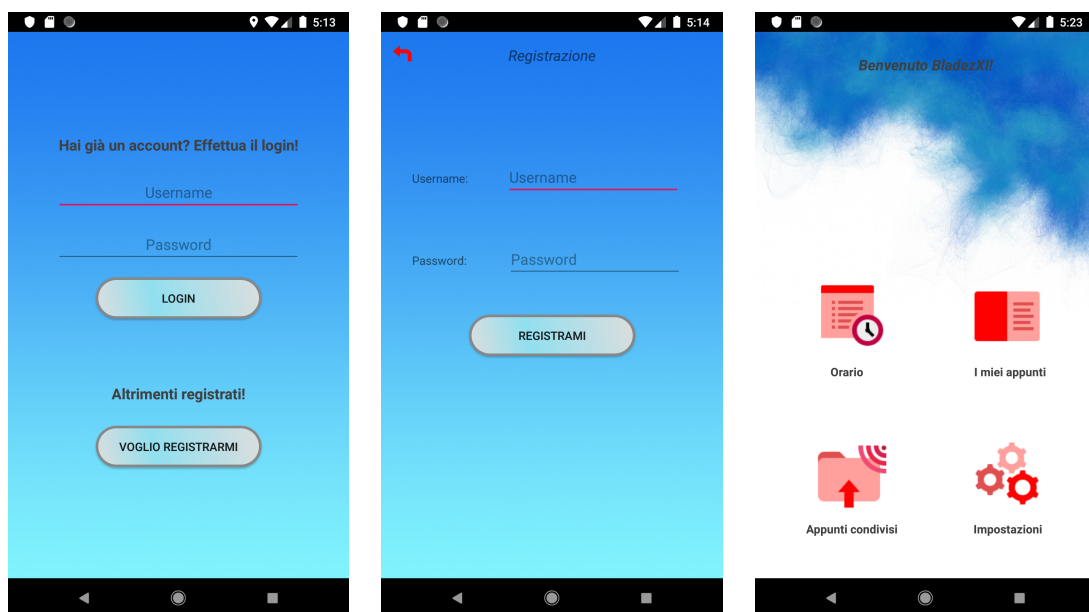


Figura 2.1: Schermata Login, Registrazione e Menu principale.

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.

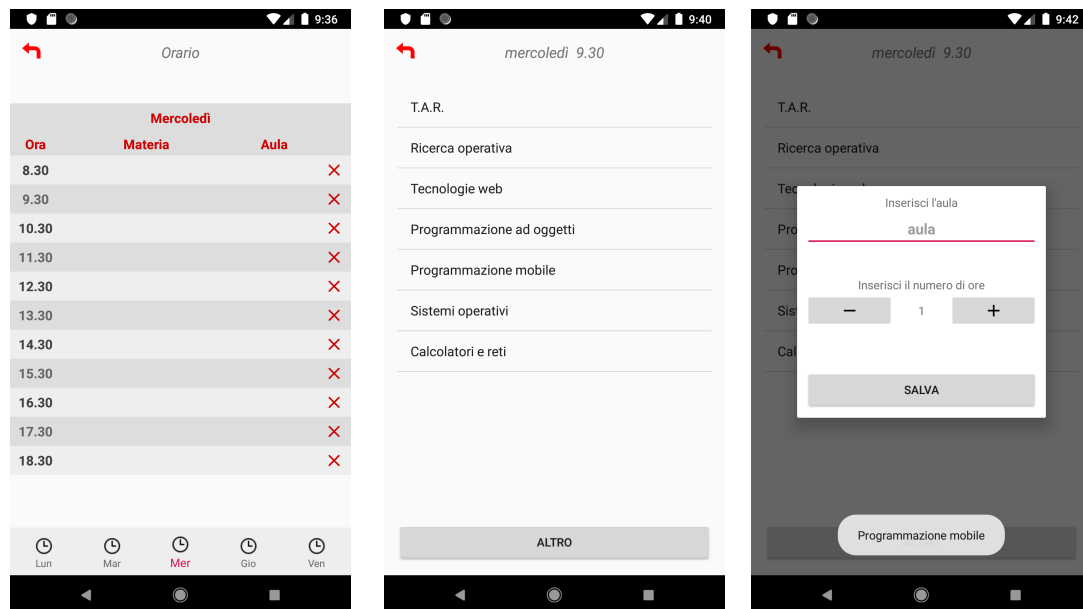


Figura 2.2: Orario.

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.

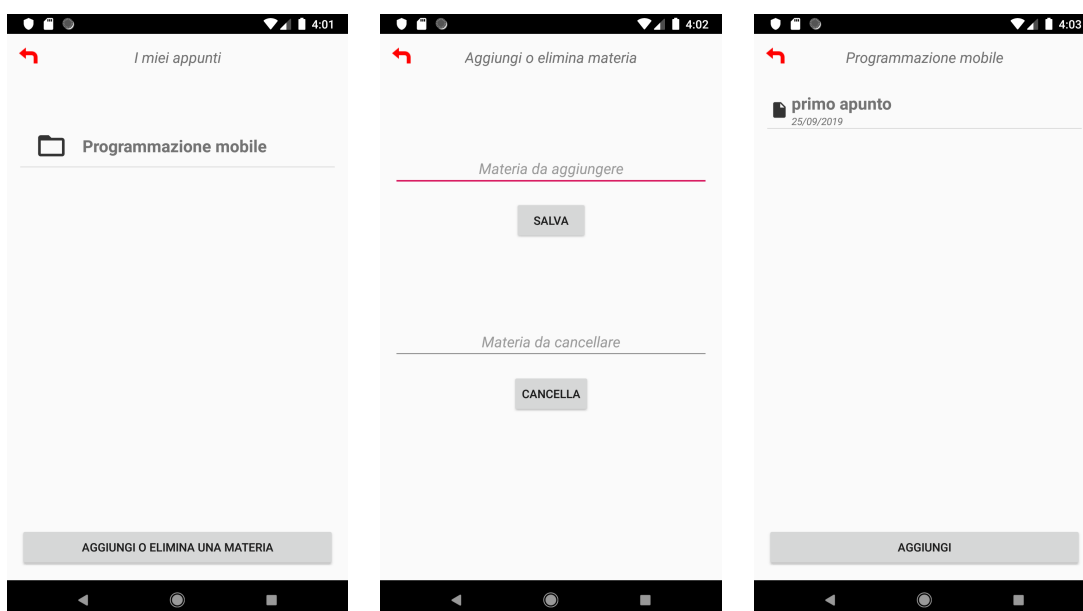


Figura 2.3: Scegli aggiungi o elimina una materia e scegli un appunto.

Scelta la materia accediamo ad una seconda lista contenente tutti gli appunti relativi ad essa; ogni elemento della lista 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.

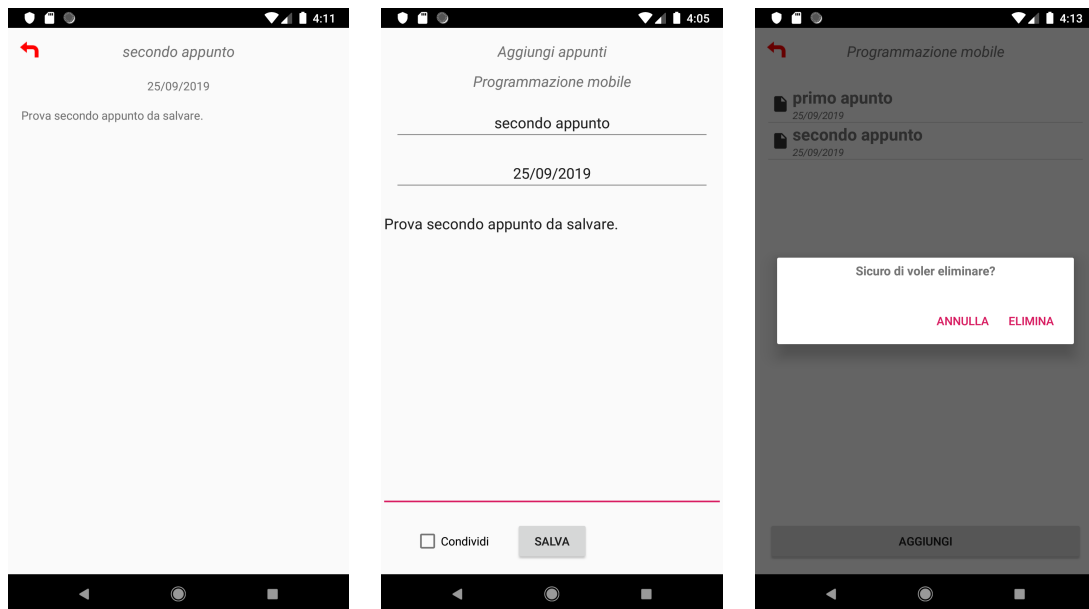


Figura 2.4: Visualizza, aggiungi o elimina un appunto.

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.

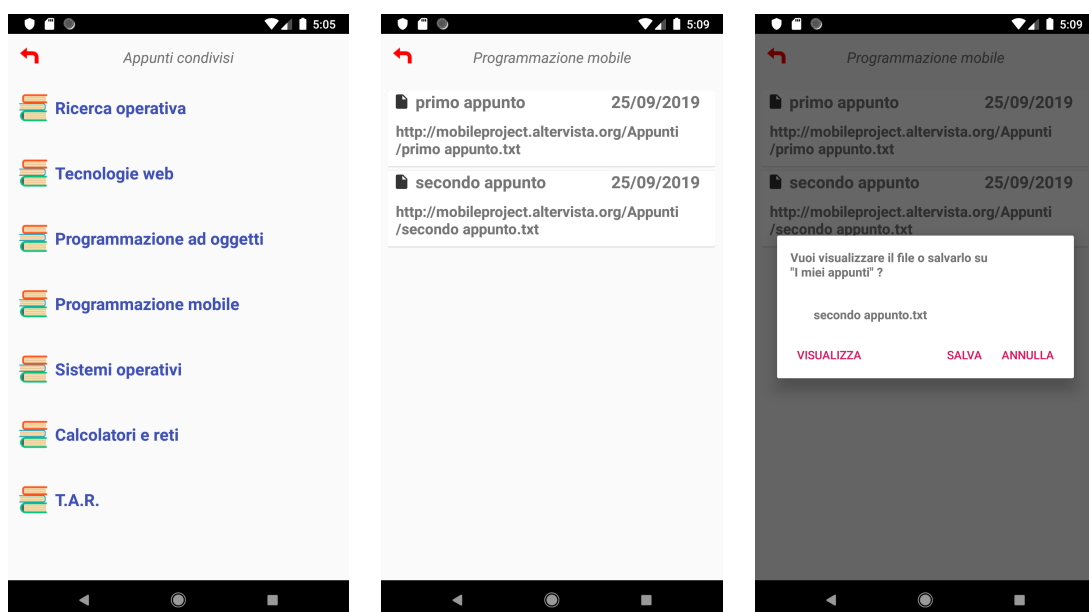


Figura 2.5: Appunti condivisi, scegli la materia, scegli l'appunto, visualizza o scarica.

2.4 Impostazioni

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

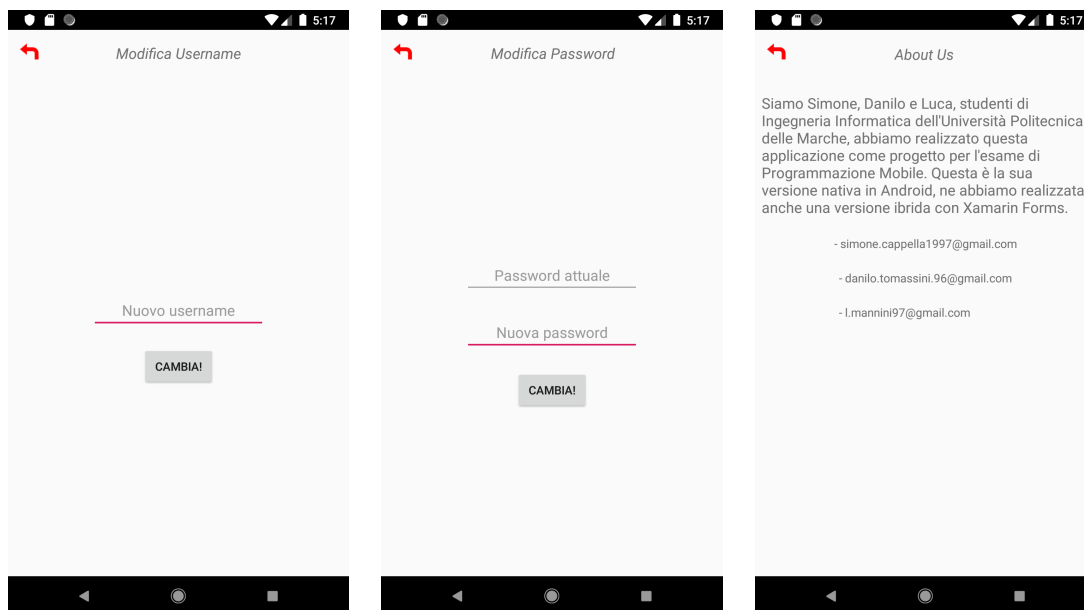


Figura 2.6: Azioni eseguibili nelle impostazioni.

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**.

```

1  \\FirstActivity\\
2  [...]
3  public void userLogin(View view) {
4  [...]
5  try {
6      String url = "http://mobileproject.altervista.org/login.php";
7      auth = supportTask.execute(method, login_name, login_pass, url).get();
8  } catch (ExecutionException e)
9  {
10     e.printStackTrace();
11 } catch (InterruptedException a)
12 {
13     a.printStackTrace();
14 }
15 if(auth.equals("Login Success"))
16 {
17     launchMainActivity(view);
18     show("Benvenuto " + login_name + "!");
19     finish();
20 } else
21 {
22     show("Dati errati. Riprova.");
23 }
24 }
25 [...]
```

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.

```

1  \\Register\\
2  [...]
3  public void userRegister() {
4  [...]
5  if (register_name.length() >= 3 && register_pass.length() >= 3)
6  {
7      try {
8          String url = "http://mobileproject.altervista.org/register.php";
9          auth= supportTask.execute(method, register_name, register_pass, url).get();
10     } catch (ExecutionException e) {
11         e.printStackTrace();
12     } catch (InterruptedException e) {
13         e.printStackTrace();
14     }
15     if (auth.equals("Registrazione avvenuta con successo!")) {
16         show("Registrazione effettuata con successo, accedi!");
17         finish();
18     } else if (auth.equals("Username in uso")) {
19         show("Username gia' in uso, prova con uno diverso!");
20     }
21 }
22 }
```

```

20     }
21 }
22 else
23 {
24 show(I campi devono contenere almeno 3 caratteri.);
25 }
26 [...]

```

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.

```

1  \\Clock\\
2  [...]
3  protected void onCreate(Bundle savedInstanceState) {
4  [...]
5  Calendar calendar = Calendar.getInstance();
6  int day = calendar.get(Calendar.DAY_OF_WEEK);
7  Fragment fragment = null;
8  switch (day){
9      case Calendar.MONDAY:
10     fragment = new lun_fragment();
11     nav.setSelectedItemId(R.id.lun);
12     break;
13     case Calendar.TUESDAY:
14     fragment = new mar_fragment();
15     nav.setSelectedItemId(R.id.mar);
16     break;
17     case Calendar.WEDNESDAY:
18     [...]
19     }
20 loadFragment(fragment);
21 [...]

```

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.

```

1  \\Clock\\
2  [...]
3  public boolean onNavigationItemSelected(@NonNull MenuItem item) {
4  Fragment fragment = null;
5  switch(item.getItemId())
6  {
7      case R.id.lun:
8          next = 0;
9          fragment = new lun_fragment();
10         break;
11     case R.id.mar:
12         next = 1;
13         fragment = new mar_fragment();
14         break;
15     case R.id.mer:
16         next = 2;
17         fragment = new mer_fragment();
18         break;
19     [...]
20     }
21 return loadFragment(fragment);
22 [...]

```

Allo stesso modo la funzione **onNavigationItemSelected**, associa in base all'item 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, andandolo 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.

```

1  \\lun_fragment\\
2  [...]
3  public View onCreateView(LayoutInflater inflater,
4  @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
5  [...]
6  SalvaOrario sa = new SalvaOrario();
7  [...]
8  txtMat1 = v.findViewById(R.id.materia1);
9  txtMat1.setText(sa.getMateria("lun_1", getActivity()));
10 txtMat1.setOnClickListener(this);
11 txtAula1 = v.findViewById(R.id.aula1);
12 txtAula1.setText(sa.getAula("lun_1", getActivity()));
13 txtOra1 = v.findViewById(R.id.ora1);
14
15 txtMat2 = v.findViewById(R.id.materia2);
16 txtMat2.setText(sa.getMateria("lun_2", getActivity()));
17 txtMat2.setOnClickListener(this);
18 txtAula2 = v.findViewById(R.id.aula2);
19 txtAula2.setText(sa.getAula("lun_2", getActivity()));
20 txtOra2 = v.findViewById(R.id.ora2);
21 [...]
```

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 **inserisciSalva0()**; (la variabile *inc* serve successivamente per inserire in un colpo solo più ore della stessa materia).

```

1  \\lun_fragment\\
2  [...]
3  public void onClick (View v) {
4  switch (v.getId())
5  {
6      case R.id.material1:
7          n = 1;
8          ora = txtOra1.getText().toString();
9          v.startAnimation(buttonClick);
10         launchList();
11         break;
12     case R.id.materia2:
13         n = 2;
14         ora = txtOra2.getText().toString();
15         v.startAnimation(buttonClick);
16         launchList();
17         break;
18     [...]
19     case R.id.edit1:
20         n = 1;
21         v.startAnimation(buttonClick);
22         materia = null;
23         aula = null;
24         inc = 1;
25         inserisciSalva();
26         break;
27     case R.id.edit2:
28         n = 2;
29         v.startAnimation(buttonClick);
30         materia = null;
31         aula = null;
32         inc = 1;
33         inserisciSalva();
34         break;
35     [...]

```

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, *"lunedì"* + *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.

```

1  \\lun_fragment\\
2  [...]
3  public static final int REQUEST_CODE = 0000;
4  [...]
5  public void launchList() {
6      Intent intent = new Intent (getActivity(), List.class);
7      intent.putExtra("giorno_ora", "lunedì" + ora);
8      startActivityForResult(intent, REQUEST_CODE);
9  }

```

L'activity **List** genera la lista 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 istanziati *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 provvede al salvataggio. Un listener sul bottone *btnSalva* fa sì 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'activity 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.

```

1  \\List\\
2  [...]
3  Intent intent = getIntent();
4  String giorno_ora = intent.getStringExtra("giorno_ora");
5  [...]
6  myDialog = new Dialog(this);
7  final ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
8  android.R.layout.simple_list_item_1, android.R.id.text1, listItem);
9  listMaterie.setAdapter(adapter);
10 [...]
11 listMaterie.setOnItemClickListener(new AdapterView.OnItemClickListener() {
12     @Override
13     public void onItemClick(AdapterView<?> parent, View view, int position,
14         long id) {
15         value = adapter.getItem(position);
16     [...]
17     myDialog.show();
18     [...]
19     btnMeno.setOnClickListener(new View.OnClickListener() {
20         @Override
21         public void onClick(View v) {
22             i--;
23             textContatore.setText(Integer.toString(i));
24         }
25     }
26     [...]
27     btnPiu.setOnClickListener(new View.OnClickListener() {
28         @Override
29         public void onClick(View v) {
30             i++;
31             textContatore.setText(Integer.toString(i));
32         }
33     }
34     [...]
35     btnSalva.setOnClickListener(new View.OnClickListener() {
36         @Override
37         public void onClick(View v) {
38             [...]
39             Intent intent = new Intent();
40             intent = intent.putExtra("mat", valore);
41             setResult(Activity.RESULT_OK, intent);
42             finish();
43             myDialog.dismiss();
44         }
45     [...]

```

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**.

```

1  \\lun_fragment\\
2  [...]
3  public void onActivityResult(int requestCode, int resultCode, Intent data){
4      super.onActivityResult(requestCode, resultCode, data);
5      if ((requestCode == REQUEST_CODE) && (resultCode == Activity.RESULT_OK)) {
6          String[] res = data.getStringArrayExtra("mat");
7          materia = res[0];
8          aula = res[1];
9          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 già 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 **SalvaOrario**, 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* contiene 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.

```
1  \\SalvaOrario\\
2  [...]
3  public static void setMateria(String key, String value, Context context) {
4      SharedPreferences.Editor editor = preferences.edit();
5      editor.putString(key, value);
6      editor.commit();
7  }
8  public static String getMateria(String key, Context context)
9  {
10     return preferences.getString(key, null);
11 }
12 public static void setAula(String key, String value, Context context) {
13     SharedPreferences.Editor editor = preferences.edit();
14     editor.putString(key+"_A", value);
15     editor.commit();
16 }
17 public static String getAula(String key, Context context) {
18     return preferences.getString(key+ "_A", null);
19 }
```

3.3 I miei appunti

Avviata l'activity **i miei appunti** viene mostrata la lista delle materie salvate nell'applicazione, queste sono il risultato delle materie aggiunte nell'orario o eventualmente aggiunte nella sezione in questione.

All'avvio si definisce e istanzia un cursore che va ad estrarre tutti gli elementi dal database e con un *CursorAdapter* va a popolare la lista. Nuovamente viene utilizzato un *onItemClickListener* che permette di catturare l'elemento dalla lista, viene, dunque, chiamata l'activity **Notes_Page** con un intent passando il parametro *a*, stringa che contiene il nome della materia scelta.

```

1  \\MyNotes\\
2  [...]
3  protected void onCreate(Bundle savedInstanceState) {
4  [...]
5  cursor = dm.selectAll();
6  [...]
7  adapter = new SimpleCursorAdapter(this, R.layout.list_mat, cursor,
8      fromColumns, viewsList, 0);
9  listMat.setAdapter(adapter);
10 listMat.setOnItemClickListener(new AdapterView.OnItemClickListener() {
11     @Override
12     public void onItemClick(AdapterView<?> parent, View view,
13         int position, long id) {
14         Cursor c = adapter.getCursor();
15         String a = c.getString(2);
16         Intent i = new Intent(getApplicationContext(), Notes_Page.class);
17         i.putExtra("mat", a);
18         startActivity(i);
19     }
20 }
21 [...]
```

3.3.1 Aggiungi una materia

Possiamo aggiungere manualmente una nuova materia o, eventualmente, eliminarne una contenuta nel database attivando il relativo bottone. Viene lanciata con un intent esplicito l'activity **AggMaterie**; si utilizza un intent esplicito per permettere l'aggiornamento della lista una volta aggiunta o eliminata una materia dalla stessa. Nella funzione **onActivityResult**, infatti, appurata la validità del ritorno dall'activity si estrae il dato, *res* contiene il nome della materia aggiunta, e definito un nuovo cursore *nc* si applicano metodi quali **changeCursor** sull'adapter in modo da indicargli il nuovo cursore da utilizzare e la **notifyDataSetChanged**; in tal modo la lista verrà aggiornata.

```

1  \\MyNotes\\
2  [...]
3  public static final int REQUEST_CODE2 = 2222;
4  [...]
5  private void launchAggMat() {
6      Intent intent = new Intent(this, AggMaterie.class);
7      intent.putExtra("app", "agg materie");
8      startActivityForResult(intent, REQUEST_CODE2);
9  }
10 @Override
11 public void onActivityResult(int requestCode, int resultCode, Intent data){
12     super.onActivityResult(requestCode, resultCode, data);
13     if ((requestCode == REQUEST_CODE2) && (resultCode == Activity.RESULT_OK)) {
14         String res = data.getStringExtra("res");
15         Toast.makeText(getApplicationContext(), res, Toast.LENGTH_SHORT).show();
16     }
17     Cursor nc;
18     nc = dm.selectAll();
19     adapter.changeCursor(nc);
20     adapter.notifyDataSetChanged();
21 }
22 [...]
```

L'activity **AggMaterie** consiste di una *editText* e due bottoni, uno per salvare la materia inserita nella edit e una per eliminarla. Nella funzione **onCreate** dell'activity **AggMaterie** viene "raccolto" l'intent.

Le funzioni relative al salvataggio ed eliminazione sono implementate direttamente nella funzione **onClick**, con uno switch sull'elemento della vista cliccato.

- **Salva:** viene estratta la stringa e istanziato un cursore; tramite il cursore si va a cercare nel database un elemento il cui campo materia sia uguale alla materia inserita, nell'**if** viene dunque controllato tramite il metodo **getCount** sul cursore se tale elemento è stato trovato o meno, in caso positivo viene solo lanciato un avviso, altrimenti può essere salvata la nuova materia e terminata l'activity impostando il risultato dell'intent.
- **Elimina:** si estrae nuovamente la stringa dalla edit, si istanzia il cursore e nuovamente si procede alla ricerca di un elemento il cui campo materia corrisponda alla materia scelta, stavolta nel caso in cui venga trovata viene eliminata e terminata l'activity.

```

1  \\AggMaterie\\
2  [...]
3  protected void onCreate(Bundle savedInstanceState) {
4  [...]
5      Intent intent = getIntent();
6  }
7  [...]
8  @Override
9  public void onClick(View v) {
10     switch (v.getId()){
11     [...]
12         case R.id.btnInsert:
13             materia = editMateria.getText().toString();
14             Cursor c = dm.searchM(materia);
15             if(c.getCount() > 0){
16                 [...]
17             }else{
18                 dm.insert(materia, "ora", "aula", "code");
19                 Intent res = new Intent();
20                 res = res.putExtra("res", materia);
21                 setResult(Activity.RESULT_OK, res);
22                 finish();
23             }
24             break;
25         case R.id.btnDelete:
26             String materia = editMateria.getText().toString();
27             Cursor del = dm.searchM(materia);
28             if (del.getCount() > 0){
29                 dm.delete(materia);
30                 finish();
31             }
32             [...]
33             break;
34     }
35 }
36 [...]
```

3.3.2 Vedi appunti

Scelta la materia dalla lista visualizzata all'avvio di **MyNotes** viene lanciata, come visto, l'activity **Notes.Page**.

All'avvio viene subito estratto il dato passato dall'activity chiamante, si tratta del nome della materia scelta, successivamente vengono definiti un **cursore** che contiene gli elementi del database la cui materia coincide con quella ricevuta in *a* e un **adapter** che permette di riempire la lista con gli elementi del cursore, in particolare vengono messi nella lista titolo e data dell'appunto.

```

1  \\Notes_Page\\
2  [...]
3  protected void onCreate(Bundle savedInstanceState) {
```

```

4  [...]
5  Intent r = getIntent();
6  a = r.getStringExtra("mat");
7  [...]
8  cursor = da.searchM(a);
9  adapter = new SimpleCursorAdapter(this, R.layout.list_app_loc, cursor,
10     fromColumns, viewsList, 0);
11  listNote.setAdapter(adapter);
12  [...]

```

In tal caso abbiamo due possibili eventi sulla lista, un click singolo o un click prolungato (longClick).

- **Click:** la funzione **onItemClick** gestisce il click singolo sull'elemento, grazie alla posizione viene istanziato un nuovo cursore all'elemento in questione e ne vengono estratti come stringhe *materia*, *titolo*, *data* e *l'appunto vero e proprio*, viene, dunque, stampato a schermo un **Toast** contenente il titolo dell'appunto scelto e viene lanciata la funzione **launchVedi**.
- **LongClick:** il long click permette di eliminare l'appunto. Viene nuovamente istanziato un cursore attraverso il quale vado ad estrarre *code* e *titolo* dell'appunto selezionato.

Viene successivamente costruita una finestra di dialogo grazie alla classe **AlertDialog**. Nell'alert dialog devono essere settati i bottoni "positivo" e "negativo", rispettivamente grazie a **setPositiveButton** e **setNegativeButton**.

Settato il positive button al suo interno viene utilizzata una funzione **onClick** che al click fa partire il metodo **delete** della classe **DataAppLoc**, database per gli appunti, che va a cancellare l'appunto andandolo a cercare con la variabile *code*. Successivamente viene aggiornata la lista come nelle activity precedenti e costruita la stringa con il percorso del file questo viene cancellato.

Il set del negative button non implementa nessuna funzione, in tal caso la alert dialog viene chiusa.

```

1  \\Notes_Page\\
2  [...]
3  da = new DataAppLoc(this);
4  [...]
5  listNote.setOnItemClickListener(new AdapterView.OnItemClickListener() {
6      @Override
7      public void onItemClick(AdapterView<?> parent, View view, int position,
8          long id) {
9          Cursor c = adapter.getCursor();
10         code = c.getString(0);
11         String materia = c.getString(1);
12         String titolo = c.getString(3);
13         String data = c.getString(2);
14         String app = c.getString(4);
15         Toast.makeText(getApplicationContext(), code, Toast.LENGTH_SHORT).show();
16         launchVedi(materia, titolo, data, app);
17     }
18     @Override
19     public boolean onItemLongClick(AdapterView<?> parent, final View view,
20         final int position, long id) {
21         Cursor c = adapter.getCursor();
22         code = c.getString(0);
23         final String titolo = c.getString(3);
24         [...]
25         LayoutInflater inflater = LayoutInflater.from(context);
26         View mess = inflater.inflate(R.layout.messaggio_elimina, null);
27         AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(context);
28         alertDialogBuilder.setView(mess);
29         alertDialogBuilder.setPositiveButton("ELIMINA",
30             new DialogInterface.OnClickListener() {
31                 @Override
32                 public void onClick(DialogInterface dialog, int which) {
33                     da.delete(code);
34                     Cursor nc;

```



```

35         nc = da.searchM(a);
36         adapter.changeCursor(nc);
37         adapter.notifyDataSetChanged();
38         Context context = getApplicationContext();
39         String folder = context.getFilesDir().getAbsolutePath()
40             + File.separator + "Appunti/";
41         File file = new File(folder + titolo + ".txt");
42         boolean deleted = file.delete();
43     }
44 }
45 alertDialogBuilder.setNegativeButton("ANNULLA",
46     new DialogInterface.OnClickListener() {
47         @Override
48         public void onClick(DialogInterface dialog, int which) {}
49     }
50     [...])

```

Come già detto nel caso di un singolo click su un elemento della lista viene chiamata la funzione **launchVedi**, questa crea un intent e un bundle contenente nome della materia cui si riferisce, titolo dell'appunto, data e annotazioni, questo bundle viene inserito nell'intent con il quale sarà chiamata l'activity **VediAppunti**.

```

1  \\Notes_Page\\
2  [...]
3  public void launchVedi(String materia, String titolo, String data, String app){
4      Intent i = new Intent(getApplicationContext(), VediAppunti.class);
5      Bundle bundle = new Bundle();
6      bundle.putString("materia", materia);
7      bundle.putString("titolo", titolo);
8      bundle.putString("data", data);
9      bundle.putString("app", app);
10     i.putExtra("data", bundle);
11     startActivity(i);
12 }

```

Nel caso in cui si voglia aggiungere un appunto va attivato il bottone **Aggiungi**, il cui evento è catturato nella funzione **onClick** della classe; questa banalmente chiama la funzione **launchAggiungi**.

La funzione in questione si occupa semplicemente di utilizzare un intent esplicito che va a lanciare l'activity **AggiungiAppuntiLoc**

```

1  \\Notes_Page\\
2  [...]
3  public void launchAggiungi(){
4      Intent intent = new Intent(this, AggiungiAppuntiLoc.class);
5      intent.putExtra("app", a);
6      startActivityForResult(intent, REQUEST_CODE1);
7  }
8  [...]

```

L'activity **AggiungiAppuntiLoc** si occupa di salvare gli appunti in locale ed eventualmente nella sezione per gli appunti condivisi.

Per quanto riguarda il salvataggio in locale, nella funzione **onClick** della classe nel caso in cui venga attivato il bottone *salva* vengono salvati i testi inseriti delle edit e i dati vengono salvati nel database relativo agli appunti, viene, quindi, restituito l'intent con la variabile *res* con una semplice stringa "Appuntisalvati".

```

1  \\AggiungiAppuntiLoc\\
2  [...]
3  public void onClick(View v) {
4      [...]
5      case R.id.btnSalvaApp:
6          final String titolo = editTitolo.getText().toString();
7          String data = editData.getText().toString();
8          String appunti = editApp.getText().toString();
9          da.insert(a, data, titolo, appunti);
10         String res = "Appunti salvati";
11         Intent intent = new Intent();

```

```
12     intent = intent.putExtra("res", res);
13     setResult(Activity.RESULT_OK, intent);
14     [...]
```

Terminata l'activity per salvare nuovi appunti si torna alla `Notes.Page`, qui la funzione **onActivityResult** come al solito controlla il risultato dell'activity chiamata con il `REQUEST_CODE` e il `RESULT_OK`, verificata la validità viene stampato il messaggio per la conferma del salvataggio e successivamente aggiornata la view della lista.

```
1 public void onActivityResult(int requestCode, int resultCode, Intent data){
2     [...]
3     if ((requestCode == REQUEST_CODE1) && (resultCode == Activity.RESULT_OK)) {
4         String res = data.getStringExtra("res");
5         Toast.makeText(getApplicationContext(), res, Toast.LENGTH_SHORT).show();
6     }
7     Cursor nc;
8     nc = da.searchM(a);
9     adapter.changeCursor(nc);
10    adapter.notifyDataSetChanged();
11 }
12 [...]
```

3.4 Appunti condivisi**3.5 Impostazioni****3.6 Database****4 Implementazioni Xamarin****4.1 Login e Registrazione****4.2 Orario****4.3 I miei appunti****4.4 Appunti condivisi****4.5 Impostazioni****4.6 Database**