**Relazione Progetto Programmazione per Dispositivi Mobili**

**Gruppo**

Bortolotti Simone

Davide De Cenzo

Marignati Luca

**Applicazione**

E-Tutoring

Sommario

[Tecnologia: scelta implementativa 3](#_Toc74648693)

[Filesystem 4](#_Toc74648694)

[Pubspec.yaml 5](#_Toc74648695)

[Database MySQL 7](#_Toc74648696)

[Accesso 7](#_Toc74648697)

[Tabelle 7](#_Toc74648698)

[Vista del DB: tabelle e attributi 9](#_Toc74648699)

[Backend-ws 9](#_Toc74648700)

[WS implementati 10](#_Toc74648701)

# Tecnologia: scelta implementativa

L’applicazione è stata sviluppando mediante l’utilizzo di Flutter e del linguaggio Darti fruttandone i numerosi vantaggi che forniscono:

* **cross-platform e indipendenza dal sistema operativo** (un codice, più piattaforme): le applicazioni sviluppate in Flutter implementano uno strato intermedio definito Bridge che permette di comunicare con le API native dei vari dispositivi (Android e iOS). È, dunque, possibile scrivere il codice una sola volta e non differenziare per le diverse tipologie di SO;
* **performance simili a quelle native**: nonostante la presenza dello strato di Bridge che comunica con le API native, le performance non vengono impattante negativamente e sono simili a quelle delle App native;
* **hot reloading – impatto positivo sulle tempistiche di sviluppo**: la funzione hot reload aiuta a sperimentare rapidamente e facilmente, creare interfacce utente, aggiungere funzionalità e correggere bug più velocemente. Stimola il processo di sviluppo fornendo a uno sviluppatore un record del codice sorgente direttamente all’interno dell’applicazione funzionante. Aiuta a riflettere sulle modifiche apportate al codice in meno di 2-3 secondi e senza ripristinare lo stato dell’applicazione. Pertanto, Hot Reload consente di monitorare rapidamente lo sviluppo dell’applicazione. A seconda della complessità e della natura del progetto, è possibile risparmiare dal 20 al 50% in termini di tempo, che può essere speso per lo sviluppo di altre funzioni utili.
* **semplicità nell’implementazione della logica**: Flutter fornisce funzionalità del sistema operativo avanzate come coordinate GPS, raccolta dei dati dei sensori, gestione delle autorizzazioni, Bluetooth, credenziali e altre funzionalità in **plug-in pronti per l’uso supportati da Google**. Se la app che progettate di sviluppare fa affidamento su una funzionalità a livello di sistema operativo non disponibile come plug-in, Flutter può stabilire la comunicazione tra il suo linguaggio di programmazione Dart e il codice nativo utilizzando i canali della piattaforma.
* **riduzione dei costi di sviluppo**: con Flutter si riducono notevolmente i costi di sviluppo dell’app, visto che si utilizza lo stesso set di librerie per l’interfaccia utente e il medesimo framework. Flutter è in grado di gestire ogni singolo pixel del display, in questo modo si raggiungeranno altissimi livelli di personalizzazione e si potranno sviluppare interfacce utente con grafiche innovative e dettagli visivi di altissima qualità;
* **framework open source**: tempo si è creata una vera e propria community di sviluppatori, che pubblicano esempi di codice e che aiutano e supportano gli utenti nella creazione di nuove app cross-platform.

# Filesystem



Come si nota, il filesystemp rispecchia la struttura in un progetto sviluppato mediante Flutter. Il codice dell’applicazione è stato sviluppato in linguaggio di programmazione *Dart* e si trova nella cartella *lib*.

I file/cartelle principali sono i seguenti:

* cartella *assets*: contiene i file multimediali, font (OpenSans) e immagini (es. logo) utilizzati dall’applicazione;
* cartella *backend-ws*: contiene i Web Services che comunicano con il D:
  + sono stati sviluppati mediante linguaggio PHP e comunicano con un database MySql;
  + la logica dei WS è quella tipica: l’applicazione effettua una chiamata http (metodo GET o POST) in cui specifica eventuali parametri (es. username e password);
  + il relativo WS chiamato effettua una query interrogando il DB e fornendo, in output, in formato JSON la risposta all’applicazione (es. lista di tutti i corsi, lista di tutti i tutor);
  + per i dettagli visionare la sezione che ne descrive in maniera più approfondita la struttura;
* cartella *lib*: come detto, contiene il codice dell’applicazione vera e propria (in linguaggio Dart);
* file *pubspec.yaml*: specifica le dipendenze richieste dal progetto, ad esempio pacchetti particolari (e relative versioni), caratteri o file di immagine. Specifica anche altri requisiti, come le dipendenze dai pacchetti per sviluppatori (come pacchetti di test o mocking) o vincoli particolari sulla versione di Flutter SDK.

# Pubspec.yaml

Specifica le dipendenze richieste dal progetto:

* flutter
* **flutter\_localizations**

<https://flutter.dev/docs/development/accessibility-and-localization/internationalization>

Per impostazione predefinita, Flutter fornisce solo localizzazioni in inglese americano. Per aggiungere il supporto per altre lingue, un'applicazione deve specificare proprietà MaterialApp aggiuntive e includere un pacchetto chiamato flutter\_localizations. A partire da novembre 2020, questo pacchetto supporta 78 lingue.

* **intl**: ^0.17.0

<https://pub.dev/packages/intl>

Fornisce funzionalità di internazionalizzazione e localizzazione, inclusa la traduzione dei messaggi, i plurali e i generi, la formattazione e l'analisi di data/numero e il testo bidirezionale.

* font\_awesome\_flutter: ^8.8.1

<https://pub.dev/packages/font_awesome_flutter>

Il pacchetto di icone Font Awesome disponibile come set di icone Flutter.

* scrollable\_positioned\_list: ^0.1.7

<https://pub.dev/packages/scrollable_positioned_list>

Elemento di View che consente di scorrere fino a un elemento specifico nell'elenco e di determinare quali elementi sono attualmente visibili.

Un ScrollablePositionedList funziona in modo molto simile alla versione builder di ListView, tranne per il fatto che l'elenco può essere fatto scorrere a un elemento specifico.

* **http: ^0.13.3**

<https://pub.dev/packages/http>

**Questo pacchetto contiene un insieme di funzioni e classi di alto livello che semplificano l'utilizzo delle risorse HTTP. È multipiattaforma e supporta dispositivi mobili, desktop e browser.**

* **flutter\_secure\_storage: ^4.2.0**

<https://pub.dev/packages/flutter_secure_storage>

È un plug-in Flutter per archiviare i dati in un archivio sicuro:

* + per iOS viene utilizzato Keychain (<https://developer.apple.com/documentation/security/keychain_services#//apple_ref/doc/uid/TP30000897-CH203-TP1>);
  + per Android, viene utilizzata la crittografia AES: la chiave segreta AES viene crittografata con RSA e viene archiviata nel **KeyStore** (<https://developer.android.com/training/articles/keystore>);
  + per Linux viene utilizzata libsecret (<https://wiki.gnome.org/Projects/Libsecret>);

Nota: KeyStore è stato introdotto in Android 4.3 (livello API 18). Il plugin non funzionerebbe per le versioni precedenti.

* move\_to\_background: ^1.0.2

<https://pub.dev/packages/move_to_background>

Plugin Flutter per l'invio di applicazioni mobili in background. Supporta iOS e Android.

* mockito: ^5.0.9

<https://pub.dev/packages/mockito>

* email\_validator: ^2.0.1

<https://pub.dev/packages/email_validator>

Plug-in per convalidare gli indirizzi e-mail senza utilizzare RegEx. Può essere utilizzato anche per convalidare le e-mail all'interno delle app Flutter

* table\_calendar: ^3.0.0

<https://pub.dev/packages/table_calendar>

Widget calendario altamente personalizzabile e ricco di funzionalità per Flutter.

* provider: ^4.3.3

<https://pub.dev/packages/provider>

# Database MySQL

Accesso:

mysql.aruba.it

Username: Sql1558195

Password: \*\*\*\*\*\*\*\*\*\*\*

Tabelle:

* **user**: contiene le informazione utilizzate per il login all’applicazione (username, passowrd, email). La password, ovviamente, non viene salvata in chiaro ma viene mantenuta in forma cryptata (md5).

**Esempio**:



* **user\_attribute**: contiene gli attributi dell’utente (es. nome, cognome, numero di matricola, città di residenza ecc.).

**Chiavi esterne**:

* + **user\_id**: si riferisce all’utente, tramite l’identificativo (attributo id), della tabella user;
  + **role\_id**: si riferisce al ruolo dell’utente (Studente o Tutor) tramite l’identificativo role\_id della tabella role;
  + **degree\_path\_id**: si riferisce al curriculum scelto in fase di registrazione dell’utente. L’attributo in questione si riferisce all’attributo degree\_path\_id della tabella degree\_path (chiave esterna);
  + **degree\_id**: si rifersice al corso di laurea a cui l’utente è iscritto.

**Esempio**:



* **role**: contiene i ruoli che gli utenti possono assumere nell’applicativo;



* **course**: contiene le informazioni relative ai corsi (es. nome, CFU, tipo di insegnamento, SSD ecc.);



* **degree**: contiene le informazioni riguardante i corsi di laurea.



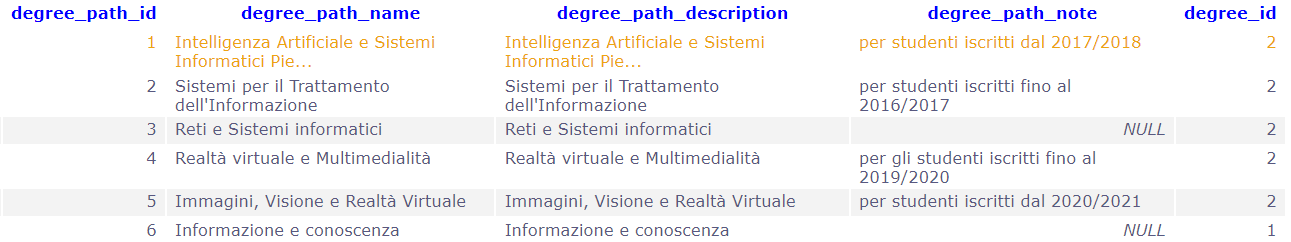
**Chiave esterna**:

* + **degree\_type\_id**: indica il tipo di corso di laurea (es. Triennale o Magistrale);
* **degree\_type**: indica il tipo di corso di laurea.



* **degree\_path**: contiene le informazioni riguardanti i curriculum.

Es. il percorso/curriculum “Realtà virtuale e Multimedialità”, tramite chiave esterna degree\_id, è associato al corso di laurea 2 (ovvero Corso di Laurea in Informatica Magistrale).



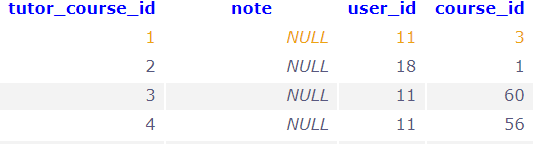
* **course\_path\_degree**: tabella ternaria che mette insieme le informazioni riguardanti i corsi, le lauree e i percorsi/curriculum.



Es. Analisi Matematica (course\_id = 1) è un corso relativia alla Laurea in Informatica (degree\_id = 1) del curriculum Informazione e conoscenza (degree\_path\_id = 6).

* **tutor\_course**: contiene le informazioni riguardanti i corsi insegnati dai tutor.

Es. il tutor con user\_id = 11 (ovvero Paolo Rossi) effettua tutoraggio del corso con course\_id = 3 (ovvero Agenti Intelligenti).



* **tutor\_time\_slot**: contiene le informazioni riguardanti le disponibiiltà dei tutor (giorni e orari). L’attributo user\_id è la chiave esterna che si riferisce all’utente (in questo caso il tutor).

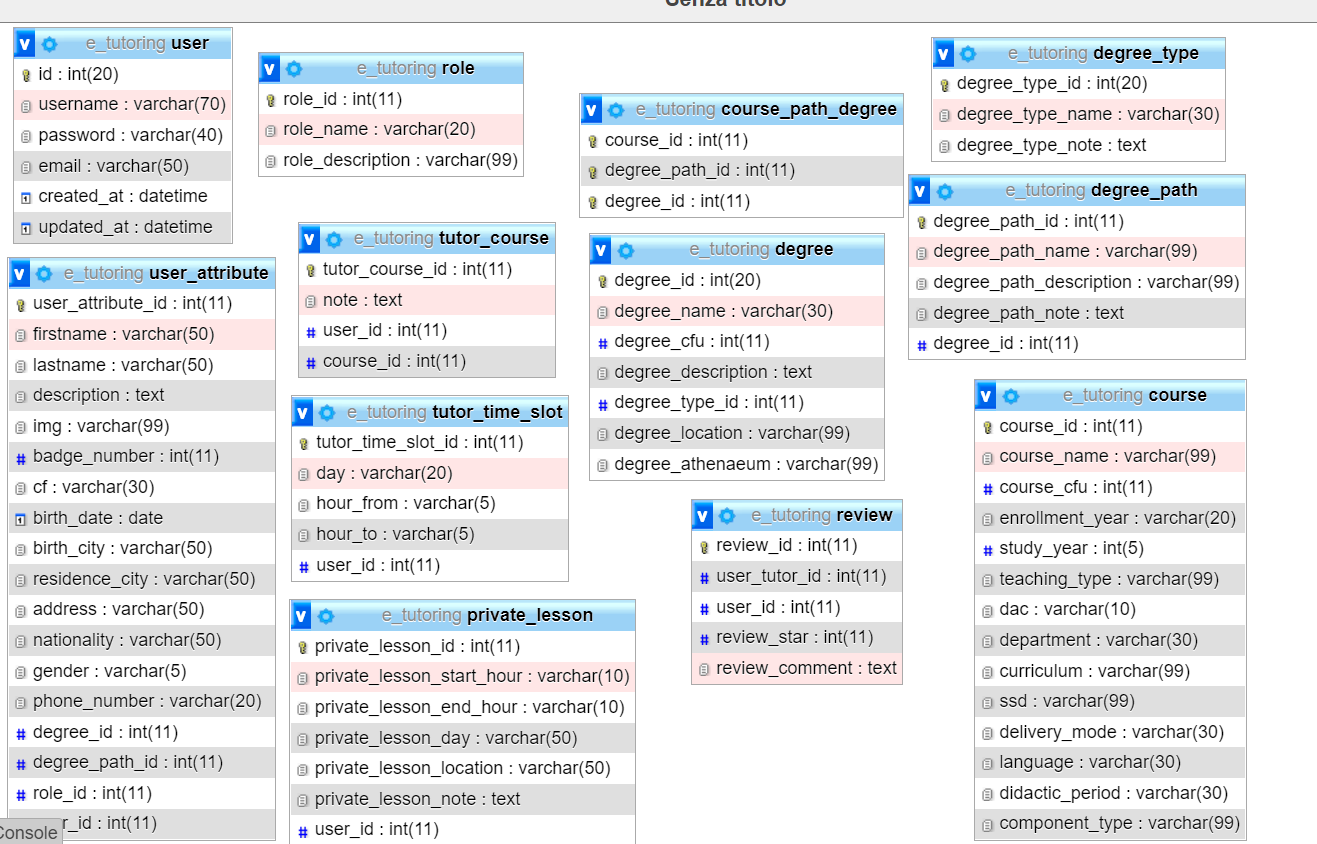


* **review**: contiene le informazioni riguardanti le recensioni che gli utente (identificati dall’attributo user\_id) effettuano sui tutor (identificati dall’attributo user\_tutor\_id).



* **private\_lesson**:

## Vista del DB: tabelle e attributi



# 

# Backend-ws

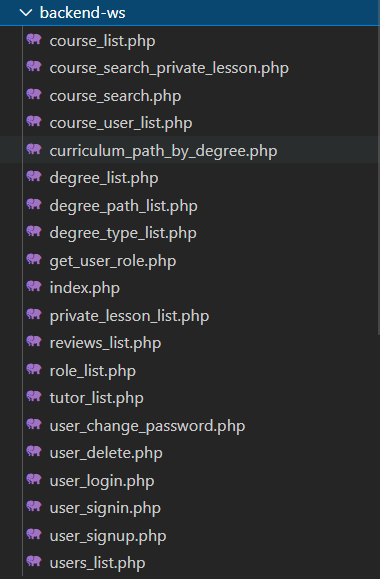
Per semplicità abbiamo provveduto a pubblicare in un dominio vero e proprio i WS sviluppati in modo da testare il reale funzionamento anche da dispositivo fisico e non solo mediante emulatore.

Dominio: <https://www.e-tutoring-app.it/>

**Sicurezza**: i dati esposti contengono delle informazioni riservate e, dunque, per accedere ai WS che comunicano con il database MySQL è stata predisposta una forma di autenticazione (**basic authentication**).



## WS implementati



* course\_list.php:
* course\_search\_private\_lesson.php:
* course\_search.php:
* course\_user\_list.php:
* curriculum\_path\_by\_degree.php
* degree\_list.php:
* degree\_path\_list.php:
* degree\_type\_list.php:
* private\_lesson\_list.php:
* reviews\_list.php:
* role\_list.php:
* tutor\_list.php:
* user\_change\_password.php:
* user\_delete.php:
* user\_login.php:
* user\_signup.php:
* users\_list.php: