



# Progetto PCTO E-Library



# Introduzione

Dopo la riapertura della biblioteca scolastica, la classe V A “Informatica e Telecomunicazioni” è stata incaricata a rendere digitalizzata la gestione della lista dei libri presenti nella biblioteca.

Dalla classe è stato realizzato un sito web dedicato alla gestione della biblioteca, nella fattispecie gestire le Prenotazioni della stanza.

Alcuni studenti hanno lavorato per creare progetti individuali per rendere il più moderna possibile la biblioteca.



# INTRODUZIONE

Per la completa gestione di una biblioteca scolastica abbiamo bisogno innanzitutto di una tabella per la catalogazione e il raccoglimento dei dati dei libri presenti in biblioteca. Per la gestione dei prestiti e della prenotazione della biblioteca dobbiamo avere una tabella in cui vengono inseriti i dati degli utenti che accedono alla piattaforma legata alla biblioteca, una tabella per avere dei ruoli all'interno della struttura (Alunno-Docente-Bibliotecario), ogni ruolo avrà dei propri permessi specifici, abbiamo poi bisogno di una tabella per la gestione dei prestiti e una per la gestione delle prenotazioni.

# Documentazione tecnica database e sito web

## MODELLO ER









# I progetti realizzati

## App

E' stata realizzata un'applicazione per la gestione dell'archivio dei libri e delle prenotazioni presenti nella biblioteca.

## Domotica

E' stato realizzato un progetto, tramite Arduino, che implementa alcune funzionalità all'interno della biblioteca.



Progetto E-Library

App



# Obiettivo del progetto: App

L'obiettivo del progetto dell'app E-Library è permettere una completa gestione della biblioteca, da una semplice app Android. Eseguendo tutte quelle che erano le funzioni fondamentali per un bibliotecario, come per esempio:

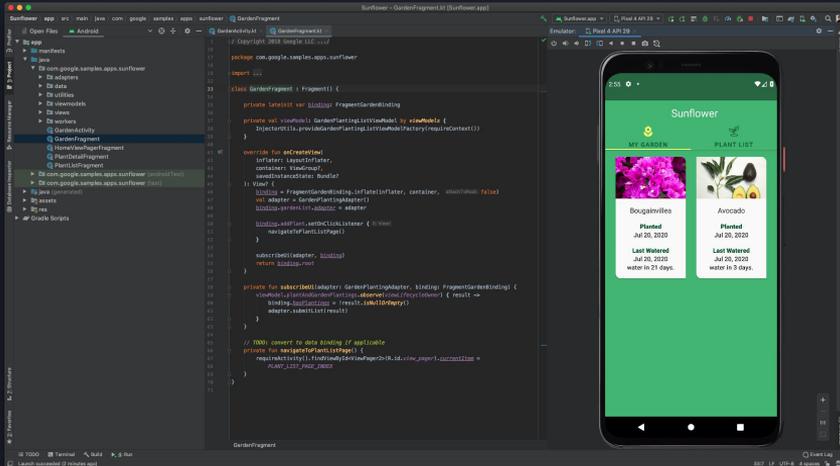
- Inserimento di un nuovo libro.
- Cancellazione di un libro.
- Visualizzazione dei libri presenti in biblioteca.
- Tenere conto dei prestiti dei libri.
- Inserire le prenotazioni.
- Visualizzare le prenotazioni.

Tutto ciò ovviamente associato ad un database MySQL e funzioni PHP e AlterVista è una piattaforma web Italiana dove è possibile aprire gratuitamente un blog che può anche essere utilizzato da Sito Web o come hosting di database.

# Android Studio



Android studio è un ambiente di sviluppo integrato (IDE, Integrated Development Environment) per la programmazione di applicazioni con Android. Include un editor di testi per la stesura assistita del codice sorgente, un editor visuale per il disegno dell'interfaccia grafica e un emulatore per il test di applicazioni su dispositivi virtuali.





# Specifiche dell'app

- Flutter è un framework(struttura da utilizzare come base per facilitare lo sviluppo software) open source sviluppato e supportato da Google. Gli sviluppatori frontend e full-stack utilizzano Flutter per sviluppare l'interfaccia utente (IU) di un'applicazione per più piattaforme con un'unica base di codice. Supporta lo sviluppo di applicazioni su sei piattaforme: iOS, Android, Web, Windows, MacOS e Linux.
- Flutter utilizza il linguaggio di programmazione open-source Dart, che è stato sviluppato anche da Google. Dart è ottimizzato per la creazione di UI e molti dei suoi punti di forza vengono utilizzati in Flutter.



# Specifiche più importanti di Flutter

- In Flutter, tutto è un widget. I widget sono i blocchi costitutivi dell'interfaccia utente e rappresentano tutto, dai pulsanti e dai layout ai temi e ai gestori di eventi. Questo approccio consente una grande flessibilità e componibilità.
- Con Flutter, è possibile scrivere un'unica base di codice per creare applicazioni che funzionano su diverse piattaforme, riducendo i costi di sviluppo e manutenzione.
- Flutter ha un vasto ecosistema di pacchetti e plugin disponibili attraverso il repository di pacchetti Dart, che facilita l'integrazione di funzionalità comuni come la connettività di rete, l'accesso al database, la gestione dello stato e altro.

## PRESENTAZIONE APP

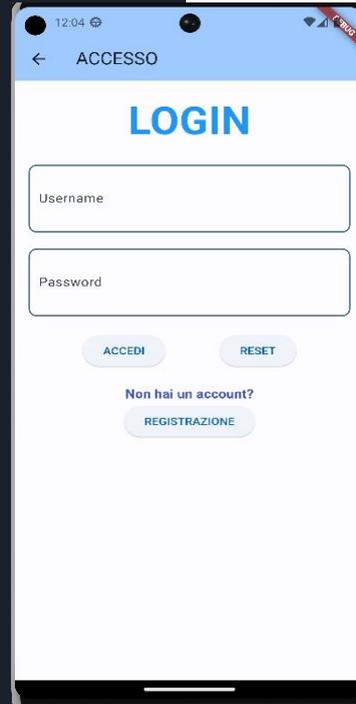
# HOME



Home principale dell'applicazione che permette l'accesso tramite bottone con possibilità di inserire form per la registrazione e accesso del personale.

## PRESENTAZIONE APP

# ACCESSO



Pagina di accesso dell'applicazione, questa pagina collegata ad un database permette l'accesso a tre diverse pagine in base al Ruolo(Studente, Docente, Bibliotecario)



# CODICE PHP PER L'ACCESSO

Nel codice PHP, JSON viene utilizzato per restituire risposte strutturate ai client, facilitando la comunicazione tra il server e l'interfaccia utente del web. Utilizzando JSON, il codice garantisce che le risposte siano facilmente interpretabili e utilizzabili dal client, migliorando l'esperienza dell'utente e la robustezza dell'applicazione.

```
1 <?php
2 // Avvia la sessione
3 session_start();
4
5 // Crea una nuova connessione al database MySQL
6 $conn = new mysqli("localhost", "root", "", "my_antoniomdcv");
7
8 // Verifica se ci sono errori di connessione al database
9 if ($conn->connect_errno) {
10     // Se c'è un errore, stampa un messaggio e termina lo script
11     echo "Impossibile connettersi al server: " . $conn->connect_error;
12     exit;
13 }
14
15 // Recupera i dati inviati tramite il form (metodo POST)
16 $Username = $_POST['Username'];
17 $Password_U = $_POST['Password_U'];
18
19 // Eseguì l'hash della password usando MD5 (Nota: MD5 non è considerato sicuro per la gestione delle password)
20 $Password_U = md5($Password_U);
21
22 // Definisce la query SQL per selezionare l'utente con il ruolo associato dal database
23 $sql = "SELECT utenti.ID_Utente, utenti.Username, utenti.Password_U, ruoli.ID_Ruolo
24     FROM utenti
25     INNER JOIN ruoli ON ruoli.ID_Ruolo = utenti.ID_Ruolo
26     WHERE utenti.Username='$Username' AND utenti.Password_U='$Password_U'";
27
28 // Eseguì la query SQL
29 $result = $conn->query($sql);
30
31 // Verifica se la query è stata eseguita correttamente
32 if (!$result) {
33     // Se c'è un errore nella query, restituisce un messaggio di errore in formato JSON e termina lo script
34     echo json_encode(["error" => "Errore nella query: " . $conn->error]);
35     exit;
36 }
37
38 // Conta il numero di righe restituite dalla query
39 $conta = $result->num_rows;
40
41 // Se non ci sono righe restituite (l'utente non è trovato)
42 if ($conta == 0) {
43     // Restituisce un messaggio di errore di accesso negato in formato JSON
44     echo json_encode(["error" => "Accesso Negato. Controlla le tue credenziali e riprova!"]);
45 } else {
46     // Se l'utente è trovato, recupera la riga con i dati dell'utente
47     $row = $result->fetch_assoc();
48     // Salva l'ID dell'utente nella sessione
49     $_SESSION['ID_Utente'] = $row["ID_Utente"];
50     // Restituisce un messaggio di successo con i dettagli dell'utente in formato JSON
51     echo json_encode(["success" => "Accesso Avvenuto!", "ID_Ruolo" => $row["ID_Ruolo"], "ID_Utente" => $row["ID_Utente"]]);
52 }
53
54 // Chiude la connessione al database
55 $conn->close();
56 ?>
```



# JSON

JSON (JavaScript Object Notation) è un formato di testo leggero utilizzato per la memorizzazione e lo scambio di dati. È facilmente leggibile sia da esseri umani che da macchine e viene spesso utilizzato per trasmettere dati tra un server e un client web.

Nel contesto del codice PHP, JSON viene utilizzato per restituire risposte strutturate al client.

Vediamo alcuni punti chiave del codice in relazione all'utilizzo di JSON:

1) Connessione e Verifica del Database:

Il codice inizia con l'avvio della sessione e la creazione di una connessione a un database MySQL. Se la connessione fallisce, viene restituito un messaggio di errore.

2) Recupero dei Dati dal Modulo:

I dati inviati tramite un modulo (username e password) vengono recuperati utilizzando `$_POST`.

3) Esecuzione della Query SQL:

Viene eseguita una query per verificare se l'utente esiste nel database con il ruolo associato.

4) Verifica del Risultato della Query:

Se la query ha esito negativo, viene restituito un messaggio di errore in formato JSON, altrimenti successo.

# CODICE DART PER LA PAGINA ACCESSO PT. 1

```
1 import 'dart:convert'; // Importa la libreria per la codifica e decodifica JSON
2 import 'package:flutter/material.dart'; // Importa il framework Flutter per la creazione di interfacce utente
3 import 'package:http/http.dart' as http; // Importa la libreria HTTP per effettuare richieste al server
4
5 // Definizione della classe StatefulWidget 'Accesso'
6 class Accesso extends StatefulWidget {
7   const Accesso({Key? key}) : super(key: key);
8
9   @override
10  _AccessoState createState() => _AccessoState();
11 }
12
13 // Stato della classe 'Accesso'
14 class _AccessoState extends State<Accesso> {
15   final TextEditingController _username = TextEditingController(); // Controller per il campo 'Username'
16   final TextEditingController _password_U = TextEditingController(); // Controller per il campo 'Password'
17   bool _isLoading = false; // Booleano per gestire lo stato di caricamento
18
19   // Funzione per effettuare l'accesso
20   Future<void> _effettuaAccesso() async {
21     setState(() {
22       _isLoading = true; // Imposta lo stato di caricamento a vero per mostrare l'indicatore di caricamento
23     });
24
25     final String username = _username.text; // Ottiene il valore inserito nel campo 'Username'
26     final String password = _password_U.text; // Ottiene il valore inserito nel campo 'Password'
27
28     try {
29       // Effettua una richiesta POST al server con i dati dell'utente
30       final response = await http.post(
31         Uri.parse('http://ftp.antoniodcv.altervista.org/Accesso/Accesso.php'),
32         body: {'Username': username, 'Password_U': password}, // Invia i dati come corpo della richiesta
33       );
34
35       if (response.statusCode == 200) { // Controlla se la risposta del server è OK (200)
36         // Decodifica la risposta JSON del server
37         final data = json.decode(response.body);
38
39         if (data is Map<String, dynamic>) { // Verifica che i dati decodificati siano una mappa
40           if (data.containsKey('error')) { // Controlla se la risposta contiene un campo 'error'
41             // Mostra un messaggio di errore se la risposta contiene un errore
42             ScaffoldMessenger.of(context).showSnackBar(
43               SnackBar(
44                 content: Text(data['error']),
45                 backgroundColor: Colors.red,
46               ),
47             );
48           } else if (data.containsKey('success')) { // Controlla se la risposta contiene un campo 'success'
49             // Mostra un messaggio di successo se la risposta contiene 'success'
```

```
50
51 ScaffoldMessenger.of(context).showSnackBar(
52   SnackBar(
53     content: Text(data['success']),
54     backgroundColor: Colors.green,
55   ),
56 );
57
58 // Converte il valore del ruolo in un intero
59 final ruolo = int.tryParse(data['ID_Ruolo']); // Converte l'ID_Ruolo da stringa a intero
60 final idUtente = data['ID_Utente']; // Ottiene l'ID dell'utente dalla risposta
61
62 if (ruolo != null) { // Verifica che la conversione sia riuscita
63   // Naviga alla schermata appropriata in base al ruolo dell'utente
64   switch (ruolo) {
65     case 1: // Ruolo Studente
66       Navigator.pushReplacement(
67         context,
68         MaterialPageRoute<void>(
69           builder: (context) => Studente(idUtente: idUtente), // Naviga alla schermata Studente
70         ),
71       );
72       break;
73     case 2: // Ruolo Docente
74       Navigator.pushReplacement(
75         context,
76         MaterialPageRoute<void>(
77           builder: (context) => Docente(idUtente: idUtente), // Naviga alla schermata Docente
78         ),
79       );
80       break;
81     case 3: // Ruolo Libreria
82       Navigator.pushReplacement(
83         context,
84         MaterialPageRoute<void>(
85           builder: (context) => Libreria(idUtente: idUtente), // Naviga alla schermata Libreria
86         ),
87       );
88       break;
89     default: // Ruolo non supportato
90       print('Ruolo non supportato: $ruolo');
91   }
92 } else {
93   print('Il ruolo non è un intero: $ruolo'); // Messaggio di errore se il ruolo non è un intero
94 }
```

# CODICE DART PER LA PAGINA ACCESSO PT. 2

```
94     }
95   } else {
96     print("Risposta non valida: $data"); // Messaggio di errore se la risposta non è valida
97   }
98 } else {
99   // Mostra un messaggio di errore se lo stato della risposta non è 200
100  ScaffoldMessenger.of(context).showSnackBar(
101    SnackBar(
102      content: Text('Si è verificato un errore durante la richiesta al server'),
103      backgroundColor: Colors.red,
104    ),
105  );
106 }
107 } catch (e) {
108   // Gestisce eventuali eccezioni durante la richiesta HTTP
109   print("Errore durante la richiesta HTTP: $e");
110 } finally {
111   setState(() {
112     _isLoading = false; // Imposta lo stato di caricamento a falso per nascondere l'indicatore di caricamento
113   });
114 }
115 }
116
117 // Funzione per resettare i campi del modulo
118 void _resetForm() {
119   _Username.clear(); // Pulisce il campo 'Username'
120   _Password_U.clear(); // Pulisce il campo 'Password'
121 }
122
123 @override
124 Widget build(BuildContext context) {
125   return Scaffold(
126     appBar: AppBar(
127       backgroundColor: Theme.of(context).colorScheme.inversePrimary,
128       title: const Text("ACCESSO"),
129       leading: IconButton(
130         icon: Icon(Icons.arrow_back),
131         onPressed: () {
132           Navigator.pop(context); // Torna indietro alla schermata precedente
133         },
134       ),
135     ),
```

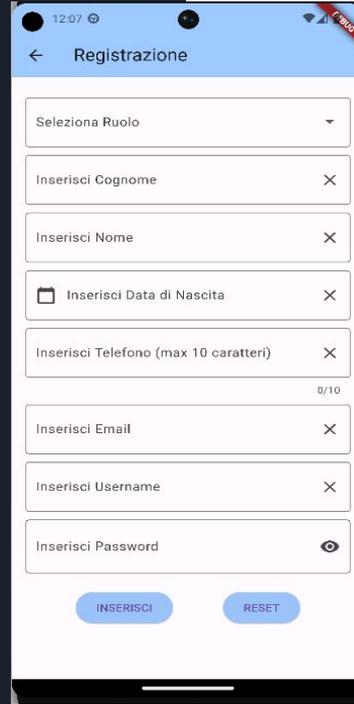
```
135   ),
136   body: _isLoading
137     ? Center(child: CircularProgressIndicator()) // Mostra un indicatore di caricamento durante il login
138     : SingleChildScrollView(
139       child: Padding(
140         padding: const EdgeInsets.all(16.0),
141         child: Column(
142           mainAxisAlignment: MainAxisAlignment.center,
143           children: [
144             Text(
145               'LOGIN',
146               style: TextStyle(
147                 fontSize: 50,
148                 color: Colors.blue,
149                 fontWeight: FontWeight.bold,
150               ),
151             ),
152             SizedBox(height: 20), // Spaziatura tra gli elementi
153             Container(
154               padding: EdgeInsets.all(10),
155               decoration: BoxDecoration(
156                 border: Border.all(
157                   color: Colors.blueGrey,
158                   width: 2.0,
159                 ),
160                 borderRadius: BorderRadius.circular(10), // Bordo arrotondato
161               ),
162               child: TextField(
163                 controller: _Username, // Campo di testo per l'username
164                 decoration: InputDecoration(
165                   labelText: 'Username',
166                   border: InputBorder.none, // Nessun bordo
167                 ),
168               ),
169             ),
170             SizedBox(height: 20), // Spaziatura tra gli elementi
171             Container(
172               padding: EdgeInsets.all(10),
173               decoration: BoxDecoration(
174                 border: Border.all(
175                   color: Colors.blueGrey,
176                   width: 2.0,
177                 ),
178                 borderRadius: BorderRadius.circular(10), // Bordo arrotondato
179             ),
```

# CODICE DART PER LA PAGINA ACCESSO PT.3

```
180 child: TextField(  
181   controller: _password_U, // Campo di testo per la password  
182   obscureText: true, // Nasconde il testo inserito  
183   decoration: InputDecoration(  
184     labelText: 'Password',  
185     border: InputBorder.none, // Nessun bordo  
186   ),  
187 ),  
188 ),  
189 SizedBox(height: 20), // Spaziatura tra gli elementi  
190 Row(  
191   mainAxisAlignment: MainAxisAlignment.spaceEvenly, // Distribuisce uniformemente i pulsanti  
192   children: [  
193     ElevatedButton(  
194       onPressed: _effettuaAccesso, // Chiama la funzione di accesso  
195       child: Text('ACCEDI'),  
196     ),  
197     ElevatedButton(  
198       onPressed: _resetForm, // Resetta il modulo  
199       child: Text('RESET'),  
200     ),  
201   ],  
202 ),  
203 SizedBox(height: 20), // Spaziatura tra gli elementi  
204 Text(  
205   'Non hai un account?',  
206   style: TextStyle(  
207     fontSize: 16,  
208     color: Colors.indigo,  
209     fontWeight: FontWeight.bold,  
210   ),  
211 ),  
212 ElevatedButton(  
213   onPressed: () {  
214     Navigator.push(  
215       context,  
216       MaterialPageRoute<void>(builder: (context) => Registrazione()), // Naviga alla schermata di registrazione  
217     );  
218   },  
219   child: Text('REGISTRAZIONE'),  
220 ),  
221 ],  
222 ),  
223 ),
```

# PRESENTAZIONE APP

# REGISTRAZIONE



The screenshot shows a mobile application interface for registration. At the top, the status bar displays the time 12:07, signal strength, and battery level. The app's title bar is blue with a back arrow and the text "Registrazione". Below the title bar, there are several input fields: "Seleziona Ruolo" (a dropdown menu), "Inserisci Cognome" (with a clear 'x' button), "Inserisci Nome" (with a clear 'x' button), "Inserisci Data di Nascita" (with a calendar icon and a clear 'x' button), "Inserisci Telefono (max 10 caratteri)" (with a clear 'x' button and a character count "0/10"), "Inserisci Email" (with a clear 'x' button), "Inserisci Username" (with a clear 'x' button), and "Inserisci Password" (with a clear 'x' button and a visibility toggle icon). At the bottom, there are two blue buttons: "INSERISCI" and "RESET".

Se l'utente non possiede un account può crearlo direttamente dalla Registrazione.

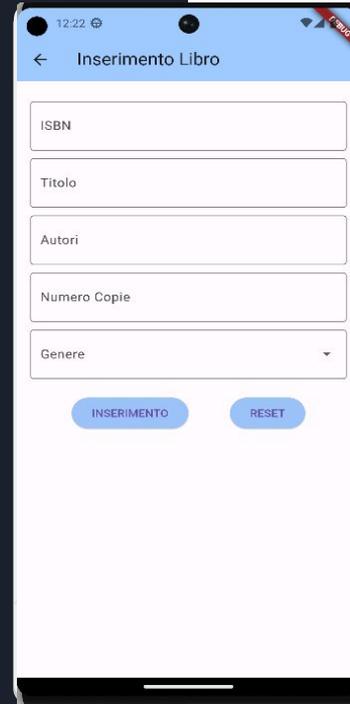
## PRESENTAZIONE APP

# BIBLIOTECARIO



Possibilità di scelta delle operazioni che si vogliono effettuare. Connesse a funzioni php hostate su Altervista dirette sul database.

# PRESENTAZIONE APP



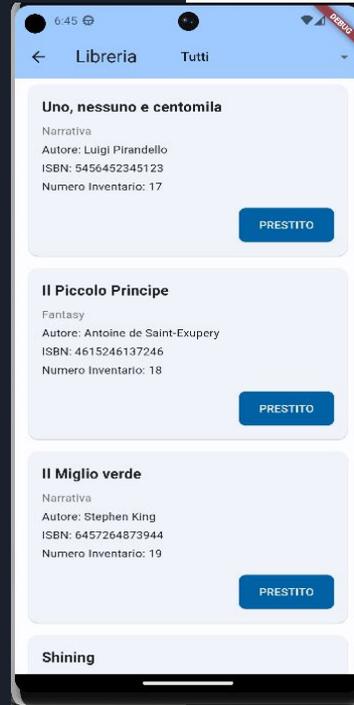
The screenshot shows a mobile application interface titled "Inserimento Libro". It features a light blue header with a back arrow and the title. Below the header are five input fields: "ISBN", "Titolo", "Autori", "Numero Copie", and "Genere" (a dropdown menu). At the bottom of the form are two blue buttons: "INSERIMENTO" and "RESET". The status bar at the top shows the time as 12:22 and various system icons.

Inserimento di un libro  
tramite interazione a  
php sul database MySQL  
con i vari campi.

INSERIMENTO

# PRESENTAZIONE APP

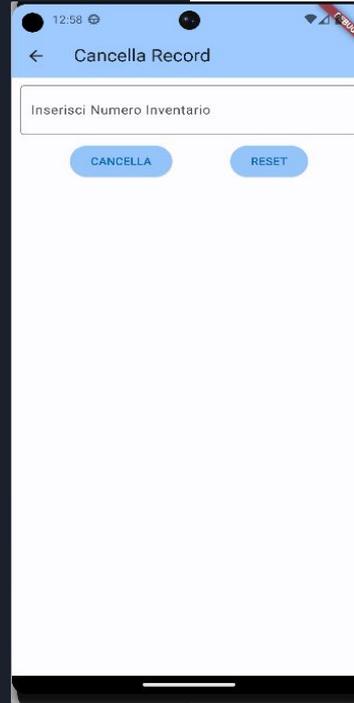
# LIBRERIA



Visualizzazione dei libri presenti sul database della biblioteca con possibilità di inserire dei filtri per visualizzare un genere specifico.

## PRESENTAZIONE APP

CANCELLAZIONE



Cancellazione di un libro  
tramite il proprio  
Numero Inventario.



# Pubblicazione

Dopo l'esecuzione dei test su diversi emulatori e dopo la correzione degli eventuali malfunzionamenti riscontrati, si predispone il pacchetto apk per distribuire l'applicazione sui dispositivi Android. La distribuzione dell'applicazione sui marketplace online richiede che il pacchetto venga firmato con un certificato digitale.





# **FINE PROGETTO APP**

Maiorino Antonio 5Ainf



Progetto E-Library

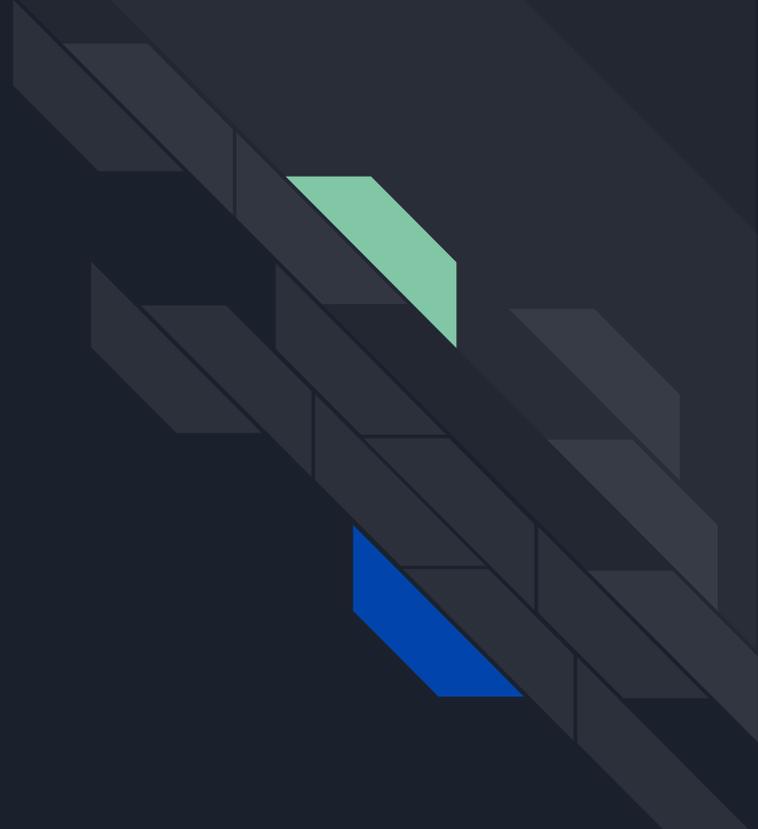
IoT con Arduino



# Introduzione a E-Library IoT con Arduino

Per una gestione completa della biblioteca abbiamo implementato alcuni sensori per rilevare alcuni dati ambientali all'interno della biblioteca, come:

- Umidità (tramite un DHT-11);
- Temperatura (tramite un DHT-11);
- Umidità del terreno (tramite un Moisture Sensor v1.2 - Sensore di umidità del terreno);
- Luminosità (tramite una fotoresistenza);



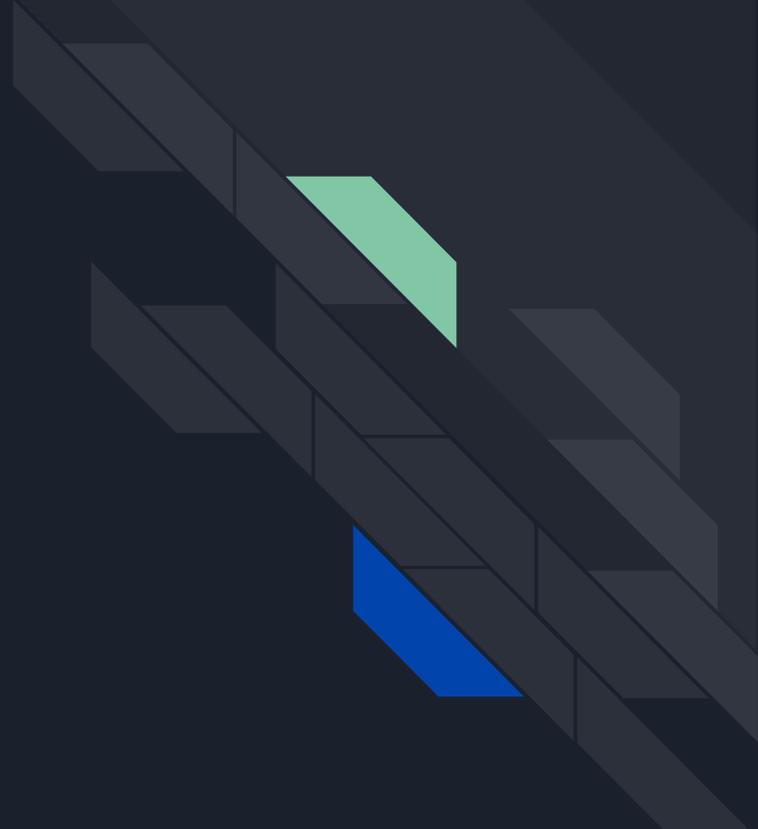


# Introduzione a E-Library IoT con Arduino

Oltre ai dati ambientali, per controllare gli accessi, e` stato implementato un sensore RFID.

Ogni professore al momento dell'ingresso e dell'uscita, dalla biblioteca, dovrà avvicinare la propria card RFID al lettore.

E` stata implementata anche il controllo di una lampadina, controllabile in app (nel progetto presentato la lampadina e` sostituita da un led).



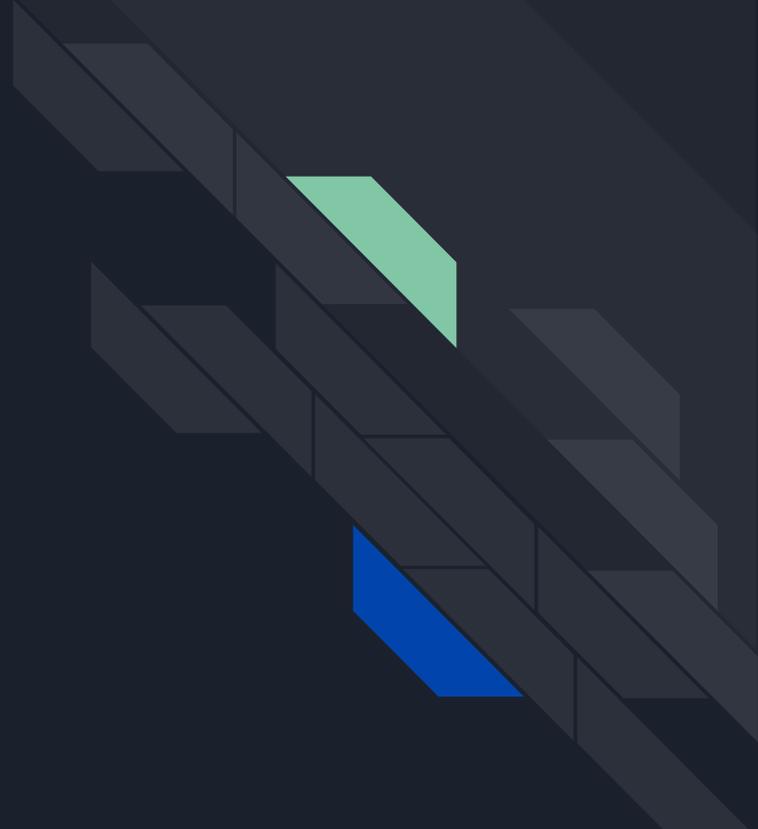


# Introduzione a E-Library IoT con Arduino

Tutti questi parametri vengono gestiti da una scheda Arduino Uno R3.

I dati saranno visualizzati in tempo reale tramite un LCD e tramite un app Android.

La lampadina sarà controllabile tramite uno switch presente sull'app.

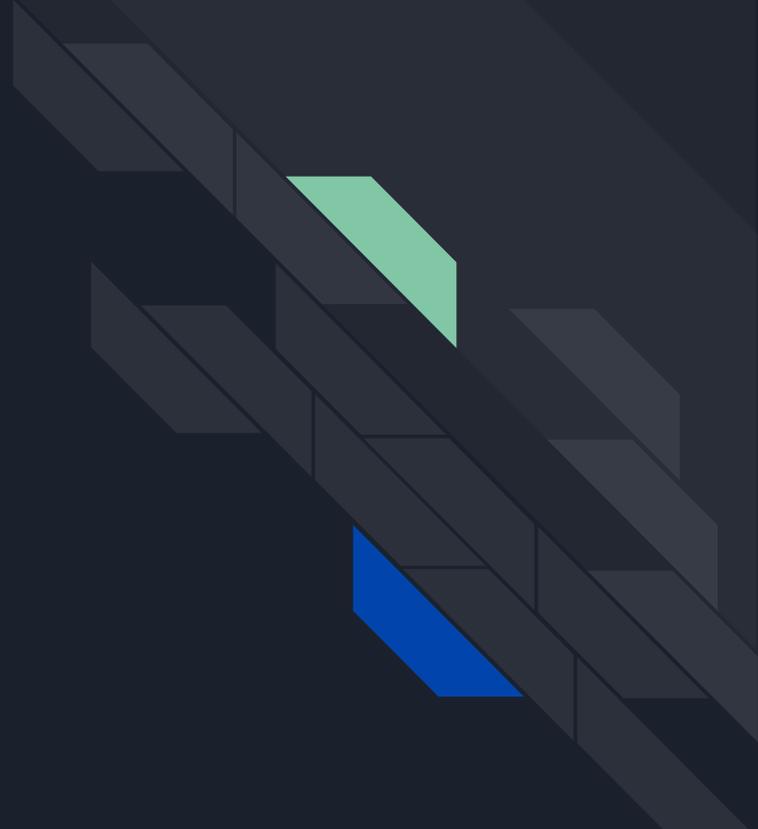




# Specifiche circuitali

I materiali utilizzati sono i seguenti:

- Arduino Uno R3;
- DHT11 - Sensore di temperatura e umidità;
- MFRC522 - Modulo RFID;
- Alcune card RFID a 13.56MHz;
- Modulo Bluetooth HC-06;
- Moisture Sensor v1.2 - Sensore di umidità del terreno;
- Fotorisistenza (LDR) - Sensore di luminosità;
- LiquidCrystal\_I2C - LCD 1602 con interfaccia I2C;
- LED;
- Resistenza 220 $\Omega$ ;
- Resistenza - Per la LDR (generalmente 10k $\Omega$ );
- Cavi di collegamento - Jumper wires;
- Breadboard - Per il collegamento dei componenti.



# Specifiche circuitali

Di seguito è descritto come collegare i vari componenti all'Arduino Uno:

## **DHT11:**

VCC -> 5V  
GND -> GND  
DATA -> Pin 2

## **MFRC522:**

SDA -> Pin 10  
SCK -> Pin 13  
MOSI -> Pin 11  
MISO -> Pin 12  
IRQ -> Non collegato  
GND -> GND  
RST -> Pin 9  
3.3V -> 3.3V

## **Modulo Bluetooth (HC-06):**

VCC -> 5V  
GND -> GND  
TXD -> Pin 11 (RX)  
RXD -> Pin 12 (TX)

## **Moisture Sensor v1.2:**

VCC -> 5V  
GND -> GND  
A0 -> Pin A0

## **Fotoreistenza (LDR):**

Un lato del LDR -> 5V  
Altro lato del LDR -> Pin A1 e un'estremità della resistenza da 10kΩ  
Altra estremità della resistenza da 10kΩ -> GND

## **LCD 1602 con I2C:**

SDA -> A4  
SCL -> A5  
VCC -> 5V  
GND -> GN

## **LED:**

Anodo (lato lungo) -> Pin 13 (attraverso una resistenza da 220Ω)  
Catodo (lato corto) -> GND

# Codice Arduino C++:

```
#include <DHT.h>
#include <SoftwareSerial.h>
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define DHTPIN 2
#define DHTTYPE DHT11
#define LEDPIN 13
#define SOIL_MOISTURE_PIN A0
#define LDR_PIN A1
#define SS_PIN 10
#define RST_PIN 9

DHT dht(DHTPIN, DHTTYPE);
MFRC522 mfrc522(SS_PIN, RST_PIN);
SoftwareSerial BTSerial(10, 11);
LiquidCrystal_I2C lcd(0x27, 16, 2);

String lastCard = "";
bool isProfessorPresent = false;
unsigned long previousMillis = 0;
const long interval = 300;
int position = 0;

unsigned long currentMillis;
float hum;
float temp;

int hum_ter_raw;
int lumino_raw;

int hum_ter;
int lumino;
```

```
void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600);

  pinMode(LEDPIN, OUTPUT);
  digitalWrite(LEDPIN, LOW);

  dht.begin();
  SPI.begin();
  mfrc522.PCD_Init();
  lcd.init();
  lcd.backlight();
  Serial.println("Perfavore appoggia la carta al lettore..");
}

void loop() {

  currentMillis = millis();
  hum = dht.readHumidity();
  temp = dht.readTemperature();
  hum_ter_raw = analogRead(SOIL_MOISTURE_PIN);
  lumino_raw = analogRead(LDR_PIN);

  if (isnan(hum) || isnan(temp)) {
    Serial.println("Lettura DHT fallita!");
    return;
  }

  hum_ter = map(hum_ter_raw, 0, 1023, 0, 100); // Converti in
percentuale
  lumino = map(lumino_raw, 0, 1023, 0, 100); // Converti in
percentuale
  hum = map(hum, 0, 100, 0, 100); // Converti in percentuale
```

```
String line1 = "Temperatura:" + String(temp) + "C"+ "Luminosita':" +
String(lumino) + "% "; //Scritta che scorre sulla prima riga
String line2 = "Umidita` terreno:" + "% "+ String(hum_ter) +
"Umidita`." + String(hum) + "% "; //Scritta che scorre sulla seconda
riga

//Passaggio dati all'app
BTSerial.print("Temperatura: ");
BTSerial.print(temp);
BTSerial.print("Umidita` ");
BTSerial.print(hum);
BTSerial.print("Umidita` del terreno: ");
BTSerial.print(hum_ter);
BTSerial.print("Luminosita` ");
BTSerial.println(lumino);
BTSerial.println("%");

if (BTSerial.available()) {
  char command = BTSerial.read();
  if (command == '1') {
    digitalWrite(LEDPIN, HIGH);
  } else if (command == '0') {
    digitalWrite(LEDPIN, LOW);
  }
}

if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial())
{
  String cardID = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    cardID += String(mfrc522.uid.uidByte[i], HEX);
  }
  cardID.toUpperCase();

  String professor = getProfessorName(cardID);
  if (professor != "") {
    lcd.clear();
    if (isProfessorPresent) {
      lcd.print("Arrivederci: ");
      lcd.setCursor(0, 1);
      lcd.print(professor);
      delay(8000);
      isProfessorPresent = false;
    }
  }
}
```

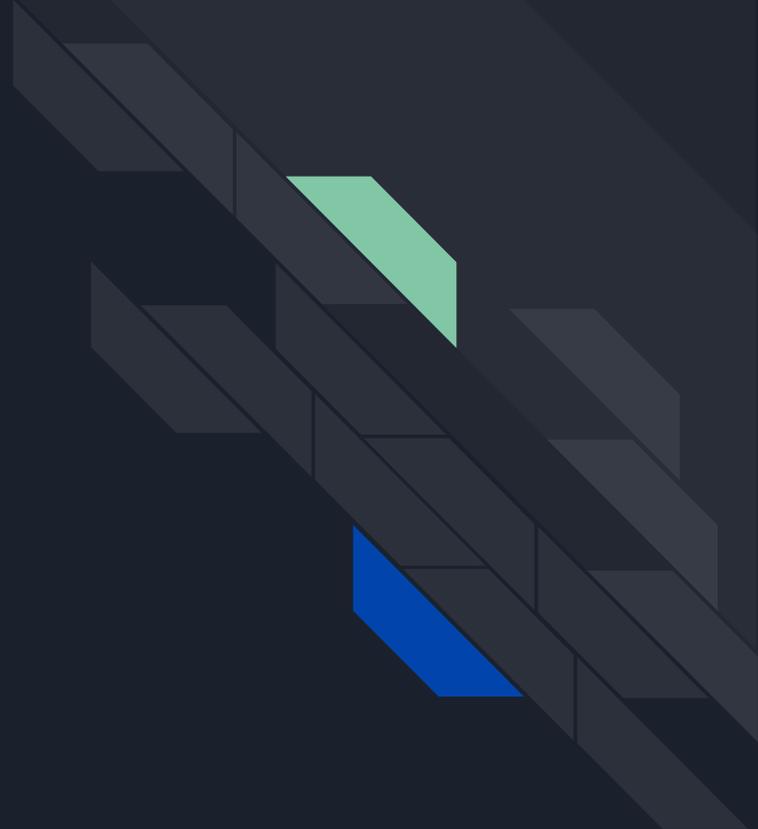


# Specifiche dell'app

Per lo sviluppo dell'app Android è stato utilizzato App Inventor, che permette di gestire i dati provenienti da una scheda Arduino.

Per comunicare con l'app, Arduino utilizza un modulo Bluetooth.

Sull'app è possibile visualizzare in tempo reale i dati ambientali inviati da Arduino, è possibile gestire una lampadina ed è possibile vedere quale professore è presente in quel momento in biblioteca.





# Specifiche dell'app

Ingrandimento  
del codice a  
blocchi, diviso  
in due parti:

(Prima parte)

```
when Connetti Bluetooth -> BeforePicking
do
  set Connetti Bluetooth -> Elements -> to BluetoothClient1 -> AddressesAndNames

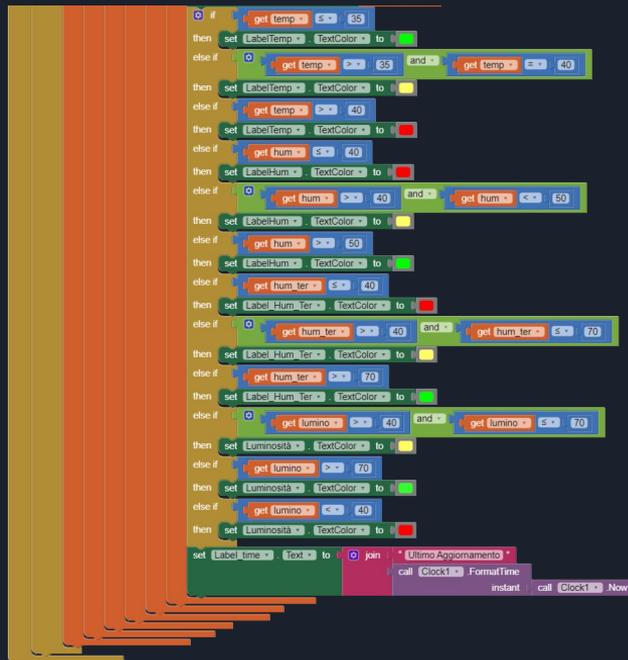
when Connetti Bluetooth -> AfterPicking
do
  set Connetti Bluetooth -> Selection -> to call BluetoothClient1 -> Connect
  address Connetti Bluetooth -> Selection ->
  set Connetti Bluetooth -> Text -> to * Connesso *
```

```
when Controllo_Led -> Changed
do
  if BluetoothClient1 -> IsConnected ->
  then
    if Controllo_Led -> On ->
    then
      call BluetoothClient1 -> SendText
      text * 1 *
    else
      call BluetoothClient1 -> SendText
      text * 0 *
```

# Specifiche dell'app

Ingrandimento  
del codice a  
blocchi, diviso  
in due parti:

(Seconda parte)





# Specifiche dell'app

L'interfaccia dell'app e` la seguente:





# **FINE PROGETTO ARDUINO IoT**

Spisso Luigi 5Ainf



Grazie per  
l'attenzione

E-Library Biblioteca scolastica, presso l'IIS  
"Della Corte Vanvitelli" di Cava de Tirreni  
(SA).