# Tecnologie Cloud e Mobile

## Progetto - Orienteering


Stefano Gechele
Giorgio Chirico
Davide Revrena
Fabio Assolari

# 1) Aggiungere funzionalità **visualizzazione classifica** di categoria

Nella Homepage viene visualizzata la lista delle gare ottenute dalla chiamata al server, successivamente una lista di categorie della gara selezionata

Questo è possibile passando come parametro l'**id della gara** selezionata
"/list_classes?id=***65126464***"

Selezionando una categoria tra quelle visualizzate verrà inviata un'altra chiamata all'endpoint di AWS: "/results?id=***65126464***&class=***M30***"

# Esempio di risposta: https://156d5fv...amazonaws.com/**results?id=***65b64***&class=***DIRECT*

GET https://156d5fv16.execute-api.us-east-1.amazonaws.com/results?id=2be8b455-81d2-4088-838b-df177552ad9d&class=DIRECT

Params · Authorization · Headers (7) · Body · Pre-request Script · Tests · Settings

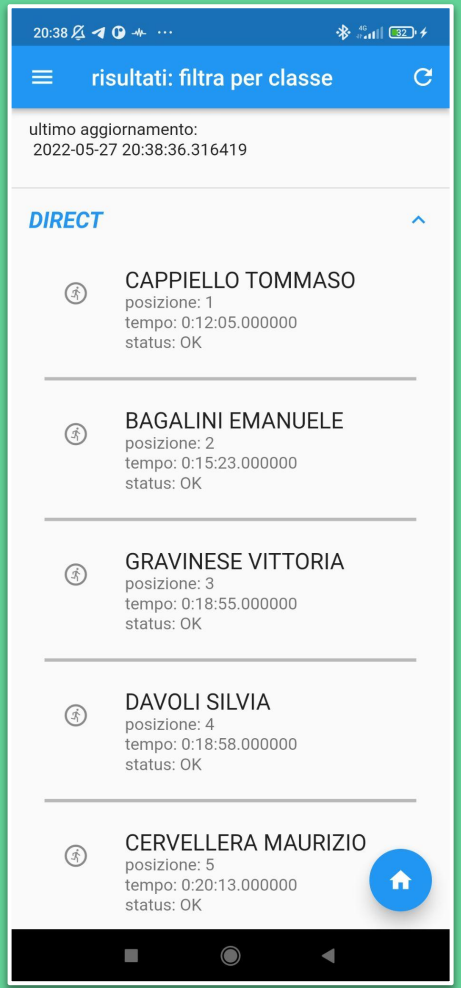
Query Params

| KEY | VALUE |
|---|---|
| id | 2be8b455-81d2-4088-838b-df177552ad9d |
| class | DIRECT |

Body · Cookies · Headers (10) · Test Results

Pretty · Raw · Preview · Visualize · JSON


```
[
    {
        "id": "VE16674",
        "name": "TOMMASO",
        "surname": "CAPPIELLO",
        "org": "G. S. OR. FOLGORE ",
        "position": "1",
        "time": "725",
        "status": "OK"
    },
    {
        "id": "EM4428",
        "name": "EMANUELE",
        "surname": "BAGALINI",
        "org": "IL MOSAICO",
        "position": "2",
        "time": "923",
        "status": "OK"
    },
    {
        "id": "PU3516",
        "name": "VITTORIA",
        "surname": "GRAVINESE",
        "org": "CORSISTA",
        "position": "3",
        "time": "1135",
        "status": "OK"
    },
    {
```

# 2) Visualizzazione dei **risultati di tutti i concorrenti** di un club

Selezionata la gara e scelto ‘Organizzazioni’ si giunge all’elenco dei club in gara, restituito dalla chiamata al server **‘/list_organizations’** , passandogli **‘raceid’.**

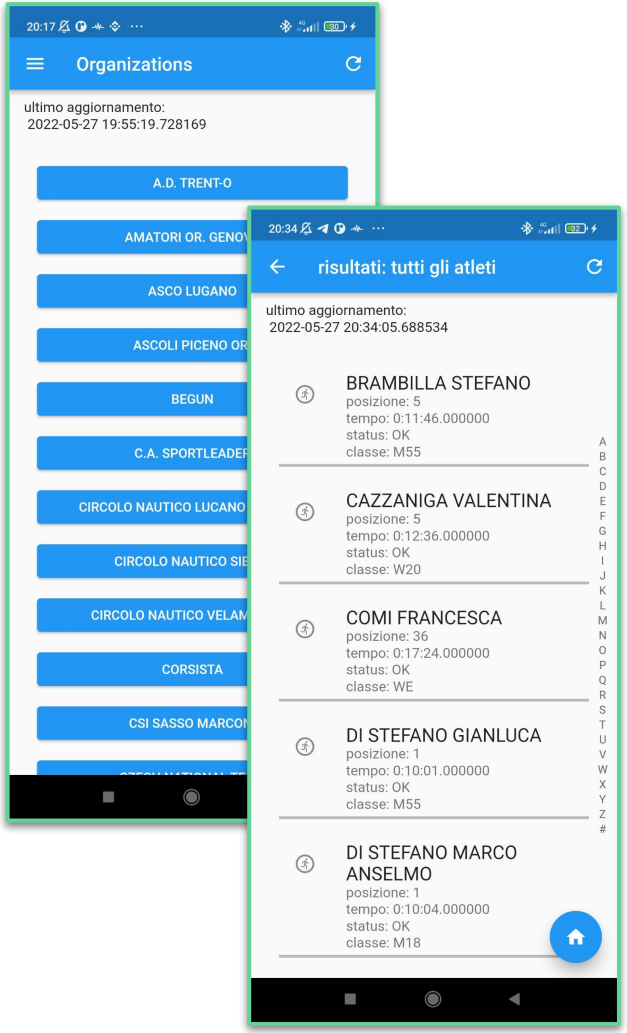
Selezionata l’organizzazione d’interesse, si possono trovare due differenti visualizzazioni:

- ‘*tutti gli atleti*’
- ‘*filtra per classe*’ : i concorrenti sono suddivisi per classe

Tali stati riportano i dati degli atleti (*name, surname, time, class, status*), i quali vengono ottenuti da AWS in questo modo:

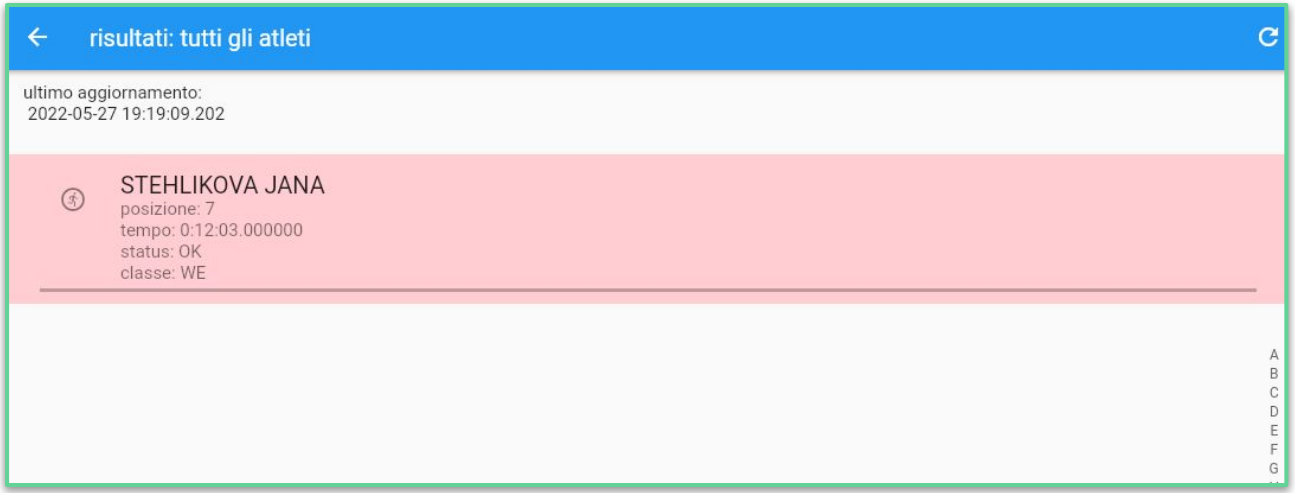
“/results?id=**raceid**&organisation=**org**”

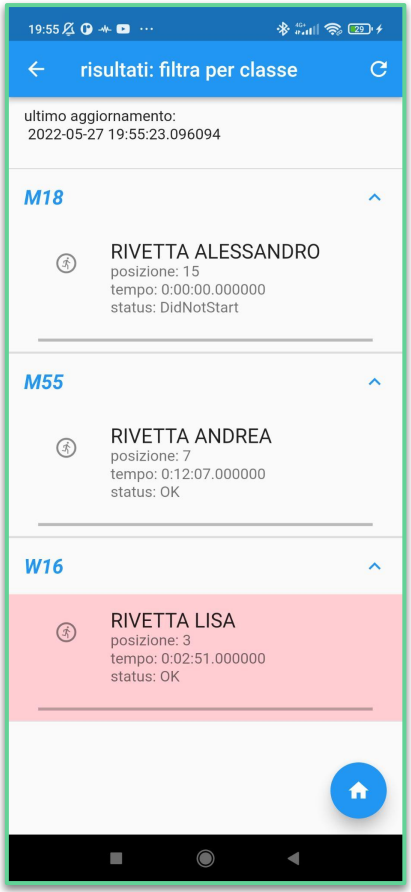
Vengono così restituiti tutti gli atleti appartenenti alla stessa organizzazione ‘**org**’, con i relativi posti in classifica nelle rispettive gare.



# 3) Evidenziare **aggiornamenti nella classifica** (Bonus)

Gli aggiornamenti vengono gestiti tramite la libreria JSON_Diff : estrapolando le differenze con il JSON vecchio siamo in grado di evidenziare con un background rosso gli atleti che hanno portato novità rispetto all'ultimo refresh.



# 4) Aggiungere pulsante/gesture per **ricaricare i dati** (Bonus)

Un “*IconButton*”, posto in alto a destra nella sezione action dell’Appbar, ricarica i dati visualizzati con la funzione: “_refreshData()” che esegue una chiamata con il metodo Get all’endpoint di AWS.

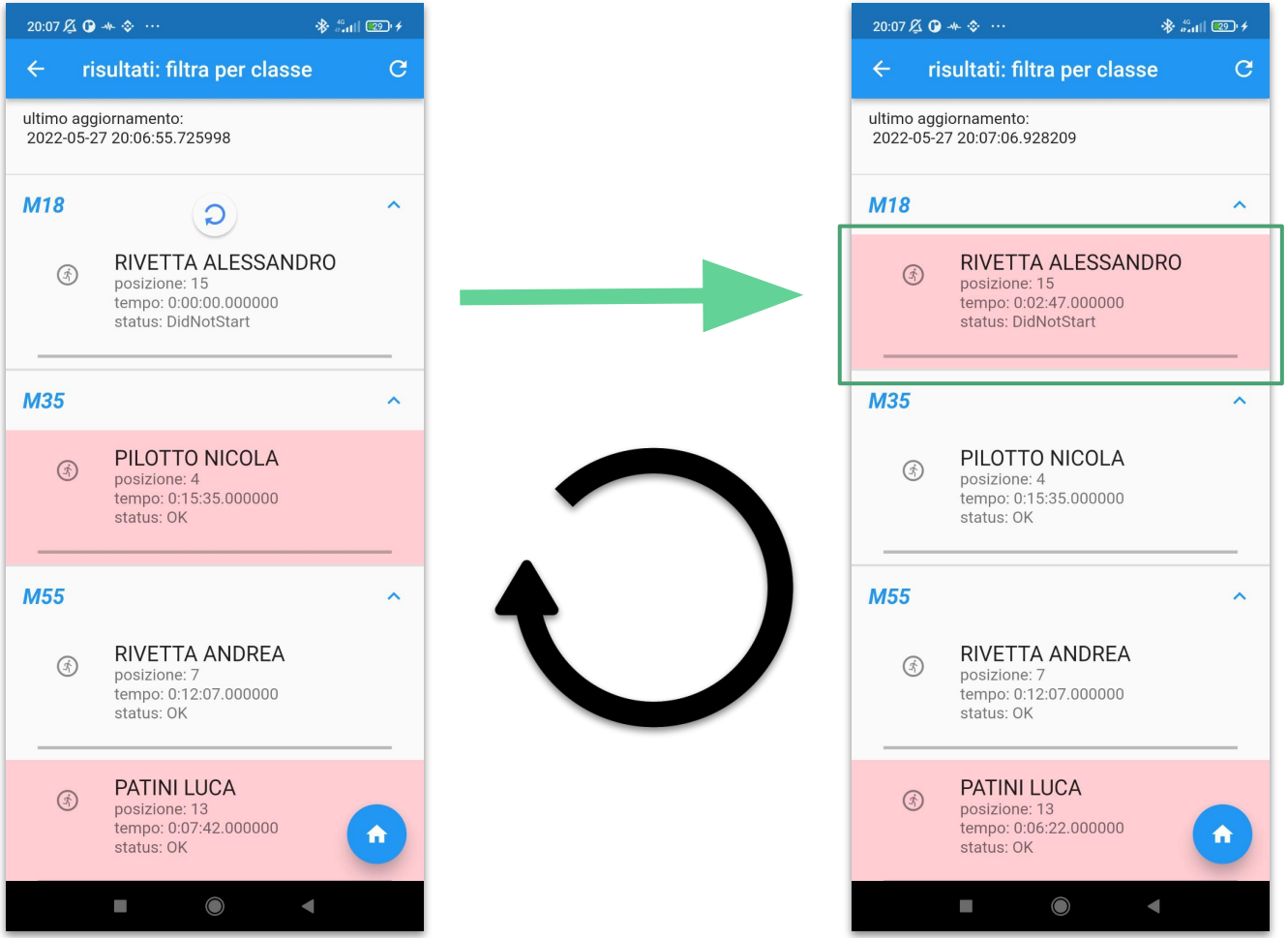
```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Available races'),
      actions: <Widget>[
        IconButton(
            icon: const Icon(Icons.refresh),
            tooltip: 'Refresh',
            onPressed: () {
              _refreshData();
            }), // IconButton
      ], // <Widget>[]
    ), // AppBar
```

```dart
Future<void> _refreshData() async {
  setState(() {
    futureRaces = fetchRaces();
  });
}
```

```dart
Future<List<Map<String, dynamic>>> fetchRaces() async {
  final response = await http.get(Uri.parse('$apiUrl/list_races'));

  if (response.statusCode == 200) {
    // If the server did return a 200 OK response,
    // then parse the JSON.
    return List<Map<String, dynamic>>.from(jsonDecode(response.body));
  } else {
    // If the server did not return a 200 OK response,
    // then throw an exception.
    throw Exception('Failed to load classes');
  }
}
```

# funzionalità **scroll-to-refresh** su dispositivi mobile



# 5) Gestione della **griglia di partenza** (Bonus)

La griglia di partenza viene gestita da una “MaterialPageRoute”, che si occupa di presentare i dati ottenuti dall’ endpoint **“/list_start”**.

L’ endpoint riceve come parametri ID e Classe restituendo un oggetto con vari campi  (*name,surname,org,time*).

La griglia di partenza può essere caricata sul Cloud tramite lo stesso token utilizzato per caricare i risultati richiamando un endpoint diverso ovvero **“/uploadstart”**.
(*Delle Lambda functions aggiuntive sono state create per questa funzionalità*)