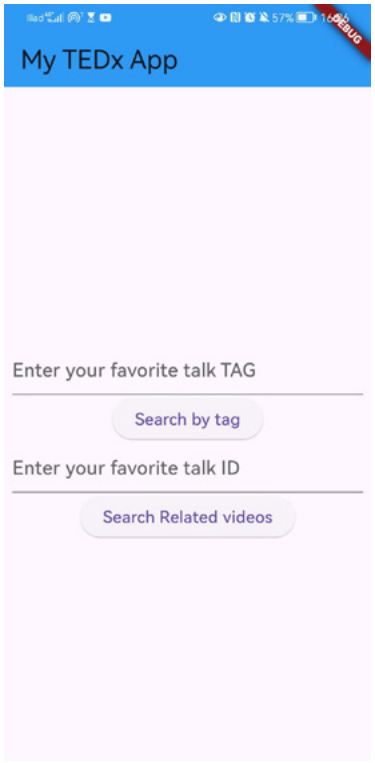
TEDx TOk

Progetto parte 4:
Flutter

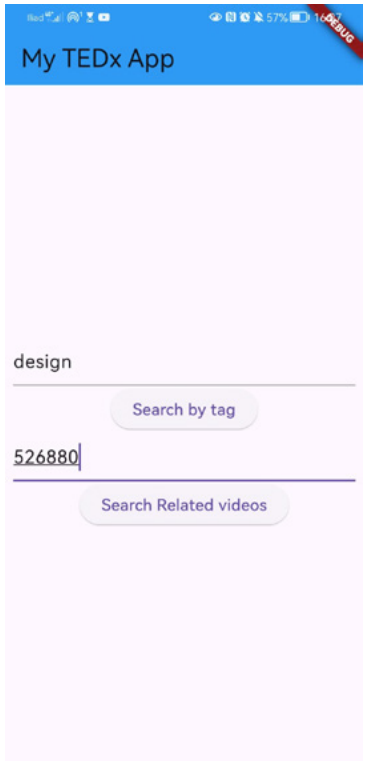
Mazzoleni Gabriele – 1079514
Masinari Gabriele - 1079692

Applicazione MyTEDX : Frame

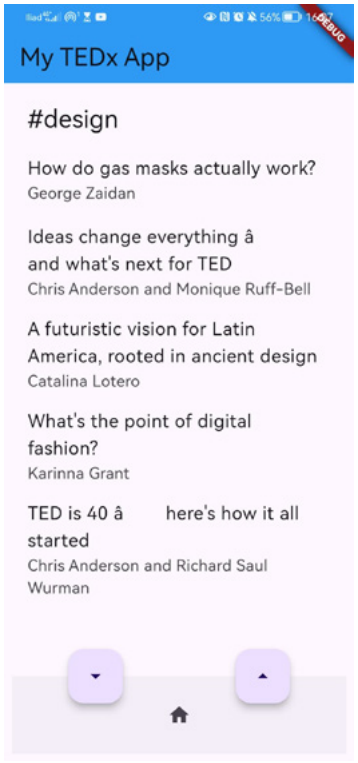
pagina principale



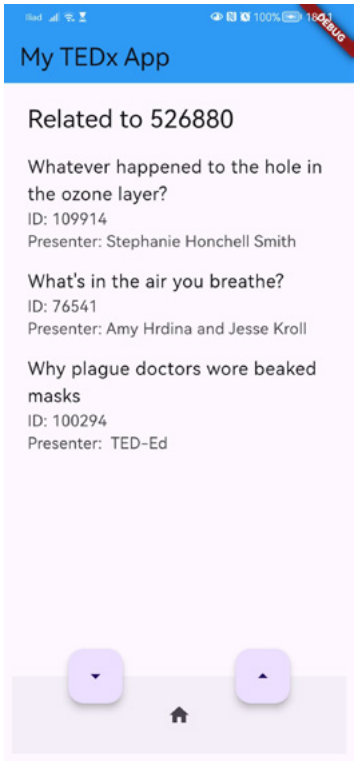
esempio di input



output ricerca sui tag



output ricerca sugli ID



Implementazione API: Get_Watch_Next_By_Idx

Questa applicazione, realizzata in Flutter, permette di inserire un tag o un ID, consentendo all'utente di visualizzare i talk corrispondenti (nel caso di ricerca per tag) o i talk correlati (nel caso di ricerca per ID).
È stata aggiunta una nuova icona che permette uno scorrimento completo dei risultati di ciascuna lista.

Codice Flutter

File talk_repository.dart
Definisce una funzione per inizializzare una lista vuota e una per andare a svolgere la nuova Lambda Function definendo il formato.

File relatedTalk.dart
Descrive il modello dati dei related talk.

```dart
class RelatedTalk{
  final String id;
  final String title;
  final String mainSpeaker;
  final String url;

  RelatedTalk.fromJSON(Map<String, dynamic> jsonMap) :
    id= jsonMap['related_video_ids'],
    title = jsonMap['related_video_title'],
    mainSpeaker = (jsonMap['related_presentedBy'] ?? ""),
    url = (jsonMap['related_url'] ?? "");
}
```

```dart
Future<List<RelatedTalk>> initEmptyRelList() async {

  Iterable list = json.decode("[]");
  var relTalks = list.map((model) => RelatedTalk.fromJSON(model)).toList();
  return relTalks;

}

Future<List<RelatedTalk>> getWatchNextByIDx(String id, int page) async {
  var url = Uri.parse('https://aoypr8ws59.execute-api.us-east-1.amazonaws.com/default/Get_Watch_Next_by_Idx');

  final http.Response response = await http.post(url,
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(<String, Object>{
      'id': id,
      'page': page,
      'doc_per_page': 5
    }),
  );
  if (response.statusCode == 200) {
    Iterable list = json.decode(response.body);
    var relTalks = list.map((model) => RelatedTalk.fromJSON(model)).toList();
    return relTalks;
  } else {
    throw Exception('Failed to load talks');
  }
}
```

Codice Flutter: main.dart

```
void _getRelatedByID() async {
  setState(() {
    init = false;
    showRelated=true;
    _relTalks = getWatchNextByIDx(_controllerB.text, page);
  });
}
  TextField( //widgets for TAG search
    controller: _controllerA,
    decoration:
        const InputDecoration(hintText: 'Enter your favorite talk TAG'),
  ), // TextField
  ElevatedButton(
    child: const Text('Search by tag'),
    onPressed: () {
      page = 1;
      _getTalksByTag();
    },
  ), // ElevatedButton
  TextField( //widgets for ID search for related videos
    controller: _controllerB,
    decoration:
        const InputDecoration(hintText: 'Enter your favorite talk ID'),
  ), // TextField
  ElevatedButton(
    child: const Text('Search Related videos'),
    onPressed: () {
      page = 1;
      _getRelatedByID();
    },
  ), // ElevatedButton
], // <Widget>[]
```

Definizione della funzione per chiamare l'API.

Elementi presenti all'interno della pagina principale con i relativi bottoni per richiamare le funzioni.

Codice Flutter: main.dart (2)

Creazione di un nuovo widget per creare la pagina di output dei related videos (non è inclusa la parte di codice relativa al pulsante home, che non si differenzia molto dal codice visto nella relativa lezione del corso)

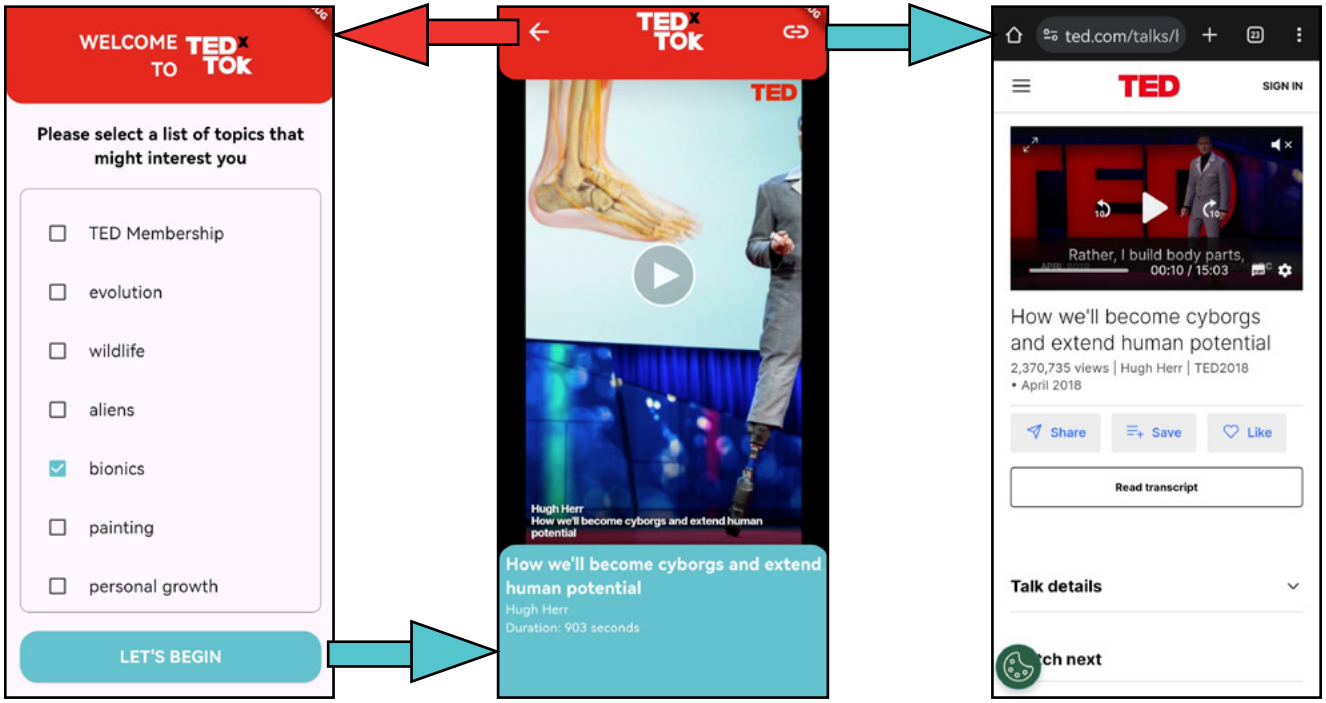
```
Widget _buildRelatedTalks(){ //ID PER PROVA 526880
        return FutureBuilder<List<RelatedTalk>>(
                future: _relTalks,
                builder: (context, snapshot) {
                  if (snapshot.hasData) {
                    return Scaffold(
                        appBar: AppBar(
                          title: Text("Related to ${_controllerB.text}"),
                        ), // AppBar
                        body: ListView.builder(
                          itemCount: snapshot.data!.length,
                          itemBuilder: (context, index) {
                            return GestureDetector(
                                  child: ListTile(
                                  subtitle: Column(
                                crossAxisAlignment:
                                    CrossAxisAlignment.start, // align to the left
                                children: [
                                  Text('ID: ${snapshot.data![index].id}'),
                                  Text('Presenter: ${snapshot.data![index].mainSpeaker}'),
                                ],
                              ), // Column
                                  title: Text(snapshot.data![index].title)), // ListTile

                            ); // GestureDetector
                          },// itemBuilder
                        ), // ListView.builder
```

```
floatingActionButtonLocation:
    FloatingActionButtonLocation.centerDocked,
floatingActionButton: Row(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: [
    FloatingActionButton(
      child: const Icon(Icons.arrow_drop_down),
      onPressed: () {
      if (snapshot.data!.length >= 5) {
       page = page + 1;
       _getRelatedByID();
      }
    },), // FloatingActionButton
    FloatingActionButton(
      child: const Icon(Icons.arrow_drop_up),
      onPressed: () {
      if (page > 1) {
       page = page - 1;
       _getRelatedByID();
      }
    },), // FloatingActionButton
  ]
), // Row
```

TedxTok app: funzionalità

La nostra applicazione si compone di due pagine principali: la prima consente all'utente di selezionare i tag di suo interesse; una volta premuto il bottone di avvio, la seconda pagina richiama la nostra API per recuperare i talk correlati ai tag selezionati e li visualizza. È possibile navigare tra tutti i video scorrendo sulla pagina, visualizzarli e accedere al sito premendo l'icona in alto a destra.



Codice Flutter TedTok

A destra: funzione che permette dato un URL di TED di creare una pagina HTML per la visualizzazione del video.

Sotto: funzione che permette la chiamata dell'API

```dart
import 'dart:convert';
import 'package:tedxtok/Models/Talk.dart';
import 'package:http/http.dart' as http;

Future<List<Talk>> initEmptyList() async {

  Iterable list = json.decode("[]");
  var talks = list.map((model) => Talk.fromJSON(model)).toList();
  return talks;

}

Future<List<Talk>> getTalkstByTagList(List tags) async {
  var url = Uri.parse('https://kz0253u01m.execute-api.us-east-1.amazonaws.com/default/Get_Talk_List_By_Tags');

  final http.Response response = await http.post(url,
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(<String, Object>{
      'tags': tags,
    }),
  );
  if (response.statusCode == 200) {
    Iterable list = json.decode(response.body);
    var talks = list.map((model) => Talk.fromJSON(model)).toList();
    return talks;
  } else {
    throw Exception('Failed to load talks');
  }

}
```

```dart
String buildHtmlVideoPage(String videoUrl) {
  // Estrae l'ID del talk dall'URL
  final talkId = videoUrl.split('/').last;
  final embedUrl = 'https://embed.ted.com/talks/$talkId';

  return '''
    <!DOCTYPE html>
    <html lang="en">
    <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <style>
        body, html {
          margin: 0;
          padding: 0;
          height: 100%;
          overflow: hidden;
          display: flex;
          justify-content: center;
          align-items: center;
          background-color: black;
        }
        iframe {
          width: 100%;
          height: 100%;
        }
      </style>
    </head>
    <body>
      <iframe src="$embedUrl" frameborder="0" allowfullscreen></iframe>
    </body>
    </html>
  ''';
}
```

Codice Flutter TedTok

main.dart

```
SizedBox(height: 20),
// List of topics
Expanded(
  child: Padding(
    padding: const EdgeInsets.symmetric(horizontal: 16.0),
    child: Container(
      padding: EdgeInsets.all(16),
      decoration: BoxDecoration(
        border: Border.all(color: Colors.grey),
        borderRadius: BorderRadius.circular(sizes.smallRoundedCorner),
      ), // BoxDecoration
      child: ListView.builder(
        itemCount: topics.length,
        itemBuilder: (context, index) {
          return topicItem(
            topic: topics[index],
            isChecked: checkedTopics[index],
            onChanged: (bool? value) {
              setState(() {
                checkedTopics[index] = value ?? false;
              });
            },
          ); // topicItem
        },
      ), // ListView.builder
    ), // Container
  ), // Padding
), // Expanded
SizedBox(height: 20),
// Button

Padding(
  padding: const EdgeInsets.symmetric(horizontal: 16.0),
  child: ElevatedButton(
    style: ElevatedButton.styleFrom(
      backgroundColor: TedTokColors.tokBlue, // Azzurro TedTok
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(sizes.stdRoundedCorner),
      ), // RoundedRectangleBorder
      padding: EdgeInsets.symmetric(vertical: 15),
    ),
    onPressed: () {
      // Raccogli i tag selezionati
      List<String> selectedTopics = [];
      for (int i = 0; i < topics.length; i++) {
        if (checkedTopics[i]) {
          selectedTopics.add(topics[i]);
        }
      }
```

A destra: Creazione della LIstView contenente i tag e raccolta dei tag selezionati.

A sinistra: Bottone che permette la navigazione alla seconda pagina, prendendo i tag selezionati dall'utente.

Per la versione demo, abbiamo limitato i tag selezionabili a 20

```
                // Naviga alla seconda pagina
                Navigator.push(
                  context,
                  MaterialPageRoute(
                    builder: (context) =>
                        SecondPage(selectedTags: selectedTopics),
                  ), // MaterialPageRoute
                );
              },
              child: const Center(
                child: Text(
                  "LET'S BEGIN",
                  style: fontStyles.buttonText,
                ), // Text
              ), // Center
            ), // ElevatedButton
          ), // Padding
          SizedBox(height: 20),
        ],
      ), // Column
    ); // SafeArea
  }
}
```

Codice Flutter TedTok

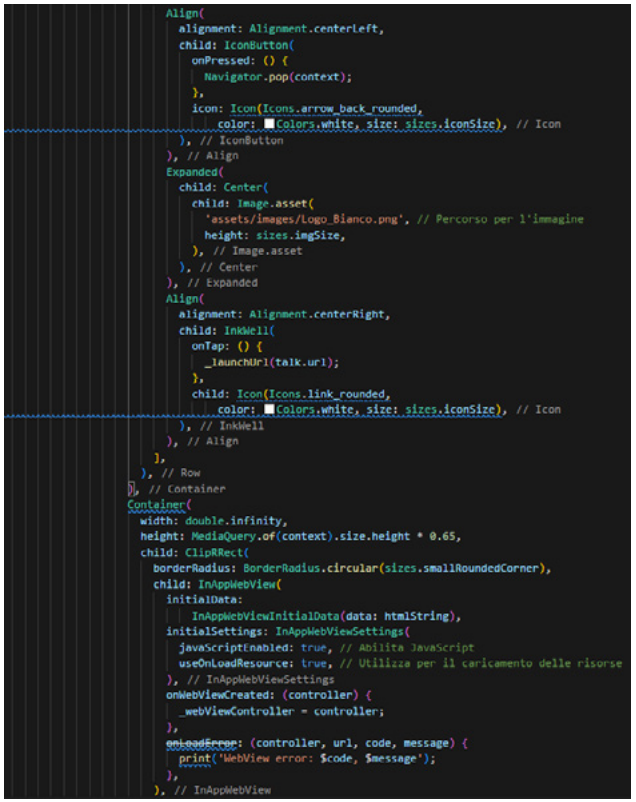
SecondPage.dart

Codice per lo scorrimento pagine e per la visualizzazione del video.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.black,
    body: FutureBuilder<List<Talk>>(
      future: _talks,
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return Center(
            child: CircularProgressIndicator(
              color: TedTokColors.tokBlue,
            ), // CircularProgressIndicator
          ); // Center
        } else if (snapshot.hasError) {
          return Center(
            child: Text('Error: ${snapshot.error}'
            ), // Text
          ); // Center
        } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
          return Center(
            child: Text('No talks available',style: fontStyles.ErrorText,),
          ); // Center
        } else {
          List<Talk> talks = snapshot.data!;
          return PageView.builder(
            scrollDirection:
                Axis.vertical, // Abilita lo scorrimento verticale
            itemCount: talks.length,
            itemBuilder: (context, index) {
              Talk talk = talks[index];
              String videoUrl = talk.url;
              String htmlString = buildHtmlVideoPage(videoUrl);

              return Column(
                children: [
                  //header
                  Container(
                    decoration: const BoxDecoration(
                      color: TedTokColors.tedRed, // Rosso TedTok
                      borderRadius: BorderRadius.only(
                        bottomLeft: Radius.circular(sizes.stdRoundedCorner),
                        bottomRight: Radius.circular(sizes.stdRoundedCorner),
                      ), // BorderRadius.only
                    ), // BoxDecoration
                    padding: EdgeInsets.all(20.0),
                    width: double.infinity,
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.spaceBetween,
                      children: [
```

Codice di impostazione degli widget dell'header e del body, con il video player.



```
                        Align(
                          alignment: Alignment.centerLeft,
                          child: IconButton(
                            onPressed: () {
                              Navigator.pop(context);
                            },
                            icon: Icon(Icons.arrow_back_rounded,
                                color: Colors.white, size: sizes.iconSize), // Icon
                          ), // IconButton
                        ), // Align
                        Expanded(
                          child: Center(
                            child: Image.asset(
                              'assets/images/Logo_Bianco.png', // Percorso per l'immagine
                              height: sizes.imgSize,
                            ), // Image.asset
                          ), // Center
                        ), // Expanded
                        Align(
                          alignment: Alignment.centerRight,
                          child: InkWell(
                            onTap: () {
                              _launchUrl(talk.url);
                            },
                            child: Icon(Icons.link_rounded,
                                color: Colors.white, size: sizes.iconSize), // Icon
                          ), // InkWell
                        ), // Align
                      ],
                    ), // Row
                  ), // Container
                  Container(
                    width: double.infinity,
                    height: MediaQuery.of(context).size.height * 0.65,
                    child: ClipRRect(
                      borderRadius: BorderRadius.circular(sizes.smallRoundedCorner),
                      child: InAppWebView(
                        initialData:
                            InAppWebViewInitialData(data: htmlString),
                        initialSettings: InAppWebViewSettings(
                          javaScriptEnabled: true, // Abilita JavaScript
                          useOnLoadResource: true, // Utilizza per il caricamento delle risorse
                        ), // InAppWebViewSettings
                        onWebViewCreated: (controller) {
                          _webViewController = controller;
                        },
                        onLoadError: (controller, url, code, message) {
                          print('WebView error: $code, $message');
                        },
                      ), // InAppWebView
```

Codice del footer, contenente le informazioni del talk

```
                    ), // ClipRRect
                  ), // Container
                  Expanded(
                    child: Container(
                      decoration: const BoxDecoration(
                        color: TedTokColors.tokBlue, // Blu TedTok
                        borderRadius: BorderRadius.only(
                          topLeft: Radius.circular(sizes.stdRoundedCorner),
                          topRight: Radius.circular(sizes.stdRoundedCorner),
                        ), // BorderRadius.only
                      ), // BoxDecoration
                      width: double.infinity,
                      padding: EdgeInsets.all(8.0),
                      child: Column(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                          Text(
                            talk.title,
                            style: fontStyles.TalkTitle,
                          ), // Text
                          Text(talk.mainSpeaker,
                              style: fontStyles.TalkSubitle), // Text
                          Text('Duration: ${talk.duration} seconds',
                              style: fontStyles.TalkSubitle), // Text
                        ],
                      ), // Column
                    ), // Container
                  ), // Expanded
                ],
              ); // Column
            },
          ); // PageView.builder
        }
      },
    ), // FutureBuilder
  ); // Scaffold
}
```

Criticità tecniche

- Al lancio dell'applicazione si potrebbe riscontrare una maggiore latenza.
- A volte la funzione Lambda restituisce errori a causa di problemi con il server, influenzando così l'usabilità dell'applicazione.

Possibili evoluzioni

- Inserimento di tutti i tag presenti nel database, con un'apposita casella di ricerca.
- Inserimento del ritaglio del video mostrando almeno il primo minuto.
- Implementazione di accesso tramite account.

TEDx TOk

Trello board

GitHub