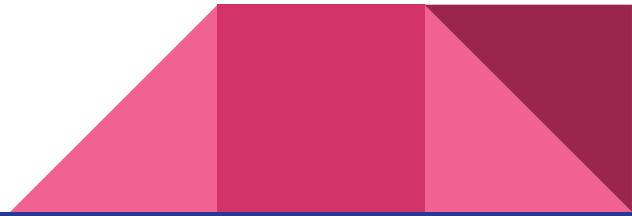
# Programmazione Mobile

**Nicola Noviello**
nicola.noviello@unimol.it

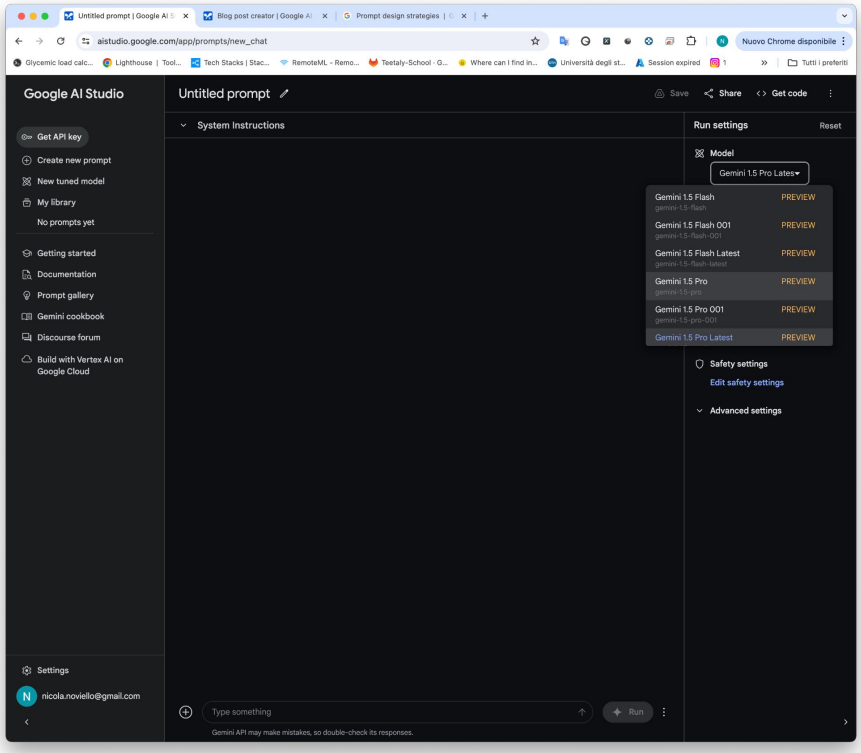
Corso di Laurea in Informatica
Dipartimento di Bioscienze e Territorio
Università degli Studi del Molise
Anno 2023/2024

# Lezione: Integrazione con Gemini

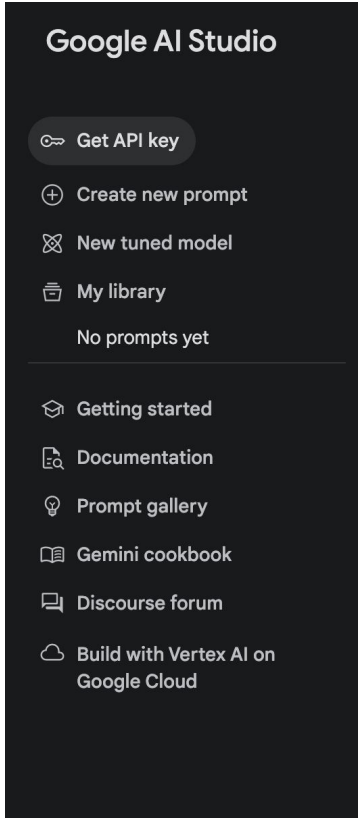
- Integrazione prompt testuale
- Integrazione prompt testuale con output JSON
- Integrazione prompt con immagini
- Chatbot AI

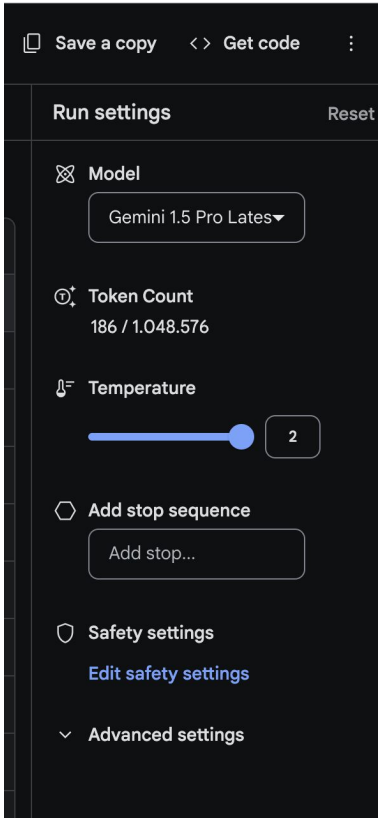


# https://aistudio.google.com/app/prompts/new_chat

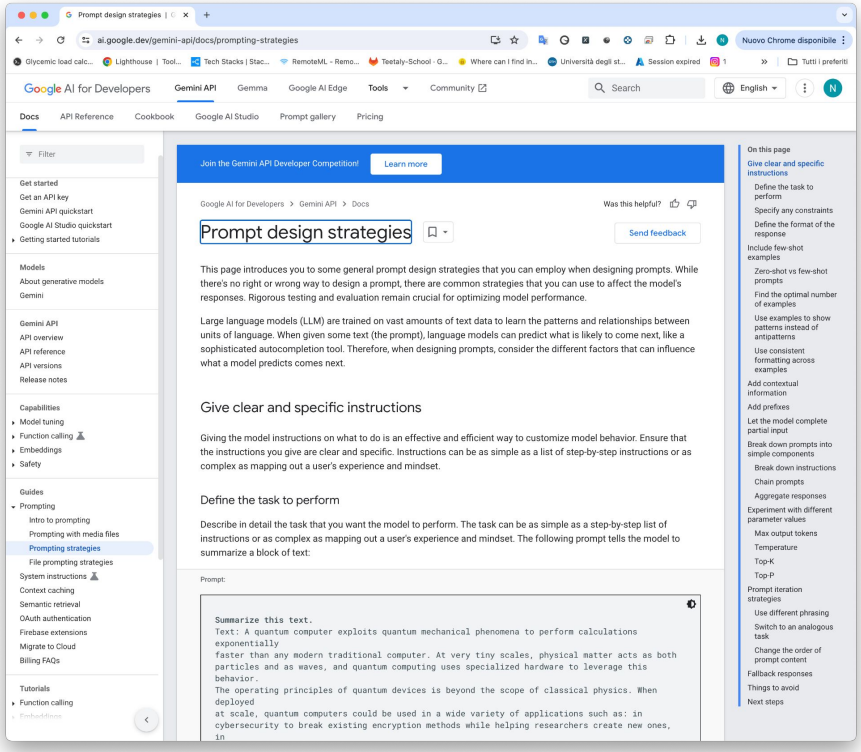


https://aistudio.google.com/app/prompts/new_chat

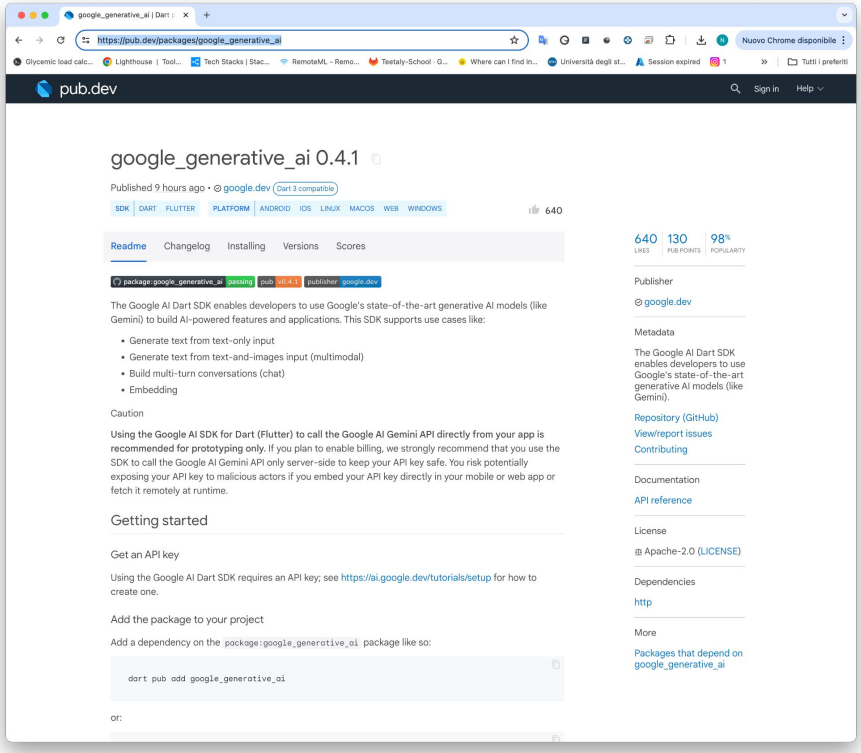


# https://ai.google.dev/gemini-api/docs/prompting-strategies



# https://pub.dev/packages/google_generative_ai



# Testo semplice

```dart
import 'package:google_generative_ai/google_generative_ai.dart';

const apiKey = 'xxx';

void main() async {
 final model = GenerativeModel(model: 'gemini-pro', apiKey: apiKey);

 final prompt =
     'Sei un professore di Programmazione Mobile che usa Flutter a lezione. Fai una domanda d\esame agli
studenti.';
 final content = [Content.text(prompt)];
 final response = await model.generateContent(content);

 print(response.text);
}
```

# Testo semplice (output)

Restarted application in 147ms.

I/flutter (17343): **Domanda d'esame:**

I/flutter (17343):

I/flutter (17343): Spiega in dettaglio l'architettura di Flutter e come il motore di rendering Skia gioca un ruolo cruciale nella creazione di interfacce utente reattive ed efficienti. Discuti i vantaggi e gli svantaggi dell'uso di Flutter rispetto ad altre framework per lo sviluppo di app mobili.

# Testo semplice con risposta json

```dart
import 'package:google_generative_ai/google_generative_ai.dart';

const apiKey = 'xxx';

void main() async {
 final generationConfig = GenerationConfig(
   maxOutputTokens: 200,
   temperature: 0.9,
   topP: 0.1,
   topK: 16,
   responseMimeType: "application/json",
 );
 final model = GenerativeModel(
   model: 'gemini-1.5-flash-latest',
   apiKey: apiKey,
   generationConfig: generationConfig,
 );
 final prompt =
     'Sei un professore di Programmazione Mobile che usa Flutter a lezione. Fai una domanda d\esame agli studenti.';
 final content = [Content.text(prompt)];
 final response = await model.generateContent(content);

 print(response.text);
}
```

# Testo semplice con risposta json (output)

Restarted application in 152ms.

I/flutter (17343): {"question": "Descrivi il processo di creazione di un widget personalizzato in Flutter, includendo esempi di come si possono utilizzare i parametri, gli stati e le proprietà per creare un widget flessibile e riutilizzabile. Spiega anche come si possono gestire gli eventi e le interazioni dell'utente all'interno del widget personalizzato."}

# Immagini

```dart
import 'dart:io';

import 'package:google_generative_ai/google_generative_ai.dart';

void main() async {
 // Access your API key as an environment variable (see "Set up your API key" above)
 final apiKey = 'xxx';
 if (apiKey == null) {
   print('No \$API_KEY environment variable');
   exit(1);
 }
 // The Gemini 1.5 models are versatile and work with both text-only and multimodal prompts
 final model =
     GenerativeModel(model: 'gemini-1.5-flash-latest', apiKey: apiKey);
 final (firstImage, secondImage) = await (
   File('assets/images/cat.jpg').readAsBytes(),
   File('assets/images/scones.jpg').readAsBytes()
 ).wait;
 final prompt = TextPart("What's different between these pictures?");
 final imageParts = [
   DataPart('image/jpeg', firstImage),
   DataPart('image/jpeg', secondImage),
 ];
 final response = await model.generateContent([
   Content.multi([prompt, ...imageParts])
 ]);
 print(response.text);
}
```

# Chat

```
class ChatWidget extends StatefulWidget {
 const ChatWidget({
   required this.apiKey,
   super.key,
 });

 final String apiKey;

 @override
 State<ChatWidget> createState() => _ChatWidgetState();
}

class _ChatWidgetState extends State<ChatWidget> {
 late final GenerativeModel _model;
 late final ChatSession _chat;
 final ScrollController _scrollController = ScrollController();
 final TextEditingController _textController = TextEditingController();
 final FocusNode _textFieldFocus = FocusNode();
 final List<({Image? image, String? text, bool fromUser})> _generatedContent =
     <({Image? image, String? text, bool fromUser})>[];
 bool _loading = false;

 @override
 void initState() {
   super.initState();
   _model = GenerativeModel(
     model: 'gemini-1.5-flash-latest',
     apiKey: widget.apiKey,
   );
   _chat = _model.startChat();
 }

 void _scrollDown() {
   WidgetsBinding.instance.addPostFrameCallback(
     (_) => _scrollController.animateTo(
       _scrollController.position.maxScrollExtent,
       duration: const Duration(
         milliseconds: 750,
       ),
       curve: Curves.easeOutCirc,
     ),
   );
 }
```

- Quello che vi interessa è questo, come viene definito il modello.
- Tutto il codice di esempio lo trovate nel progetto chat.zip della lezione 16

# Esercizio

TASSATIVAMENTE entro la fine della lezione, provate a creare una mini-applicazione che sfrutti le API di Gemini.

Inviate una mail con oggetto “Esercizio Finale” - Nome - Matricola a nicola.noviello@unimol.it

# Esercizio

Vi chiedo di procedere come descritto:

1. Provate il vostro prompt su AI Studio (https://aistudio.google.com/). Da prompt è gratis e avete infinite richieste, non serve nemmeno un'API Key
2. Disegnate l'interfaccia della vostra App, usando gli elementi di UI giusti per il funzionamento del prompt
3. Provate ad implementare, con il supporto della documentazione, del codice e del professore, quanto progettato

SE NON RIUSCITE AD IMPLEMENTARE L'APP  INVIATEMI IL LINK AL PROMPT ED IL DESIGN DELL'APP.

SE NON DISPONETE O NON VOLETE ATTIVARE LA KEY VE NE FORNISCO UNA IO O MANDATEMI IL PROGETTO SENZA KEY E LO PROVO IO