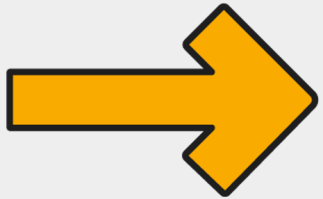
第二部分

我们今天的 Code Lab

今天我们要完成一个什么样的App?如何完成?



Gemini CodeLab

使用Gemini的API来实现一个简单的聊天应用界面。

基本功能

- 通过使用**Gemini云端API**，使用不同的模型，进行基本的问答
- 通过使用**MediaPipe提供的Android端的API**，使用Gemma 2 2B或者Gemma3 1B模型进行基本的问答

扩展功能

- 支持**图片**输入和**文档**输入
- 支持**基于会话**的聊天功能

Gemini CodeLab

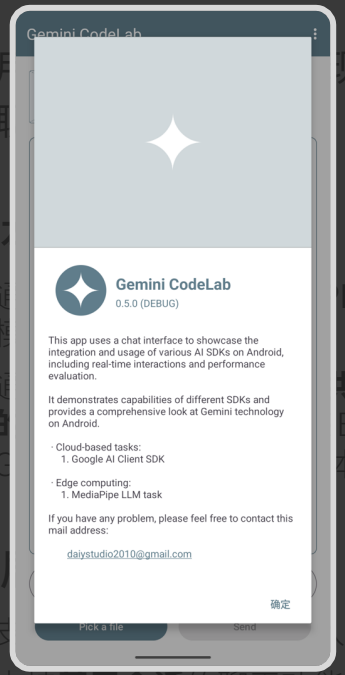
使用 现一个简单
的

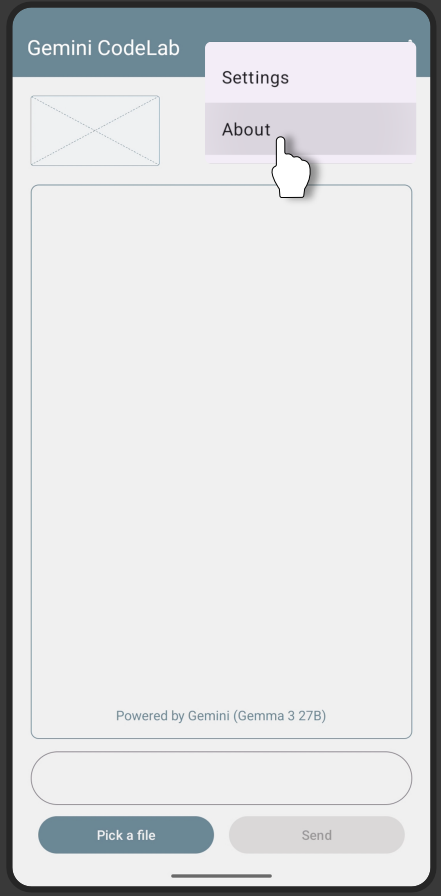
基

- 通 I，使用不同的
模
- 通 **的Android端**
的 B或者
G 本的问答

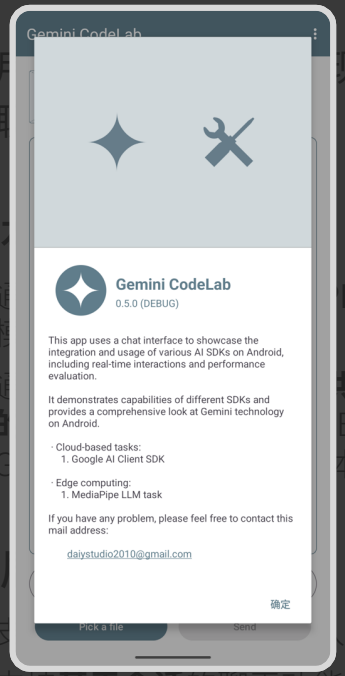
扩

- 支
- 支持基于会话的聊天功能





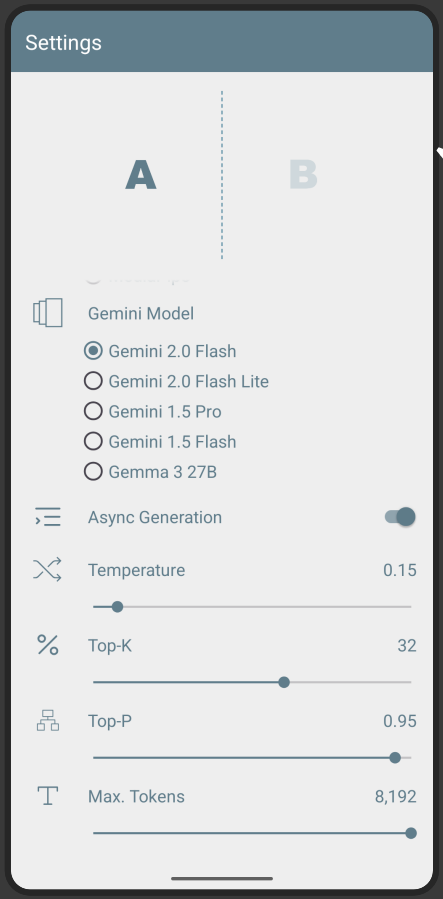
调试模式



点击顶部插画5次

更多设置选项

统计数据



Gemini CodeLab

使用Gemini的API来实现一个简单的聊天应用界面。

基本功能

- 通过使用**Gemini云端API**，使用不同的模型，进行基本的问答
- 通过使用**MediaPipe提供的Android端的API**，使用Gemma 2 2B或者Gemma3 1B模型进行基本的问答

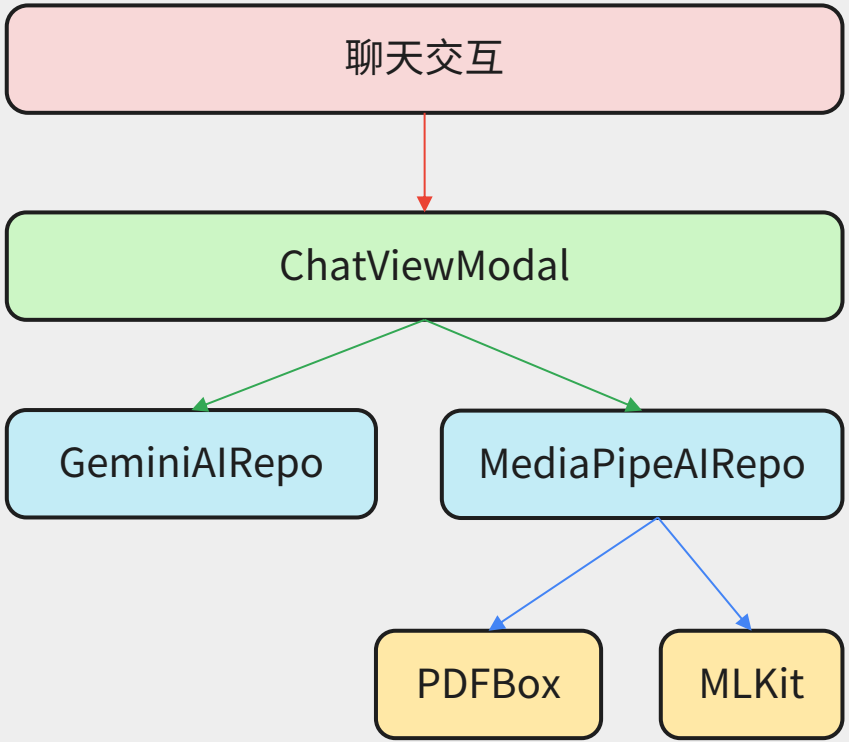
扩展功能

- 支持**图片**输入和**文档**输入
- 支持**基于会话**的聊天功能

程序架构

我们使用标准MVVM架构来程序的结构分层更加清晰，职责明确：

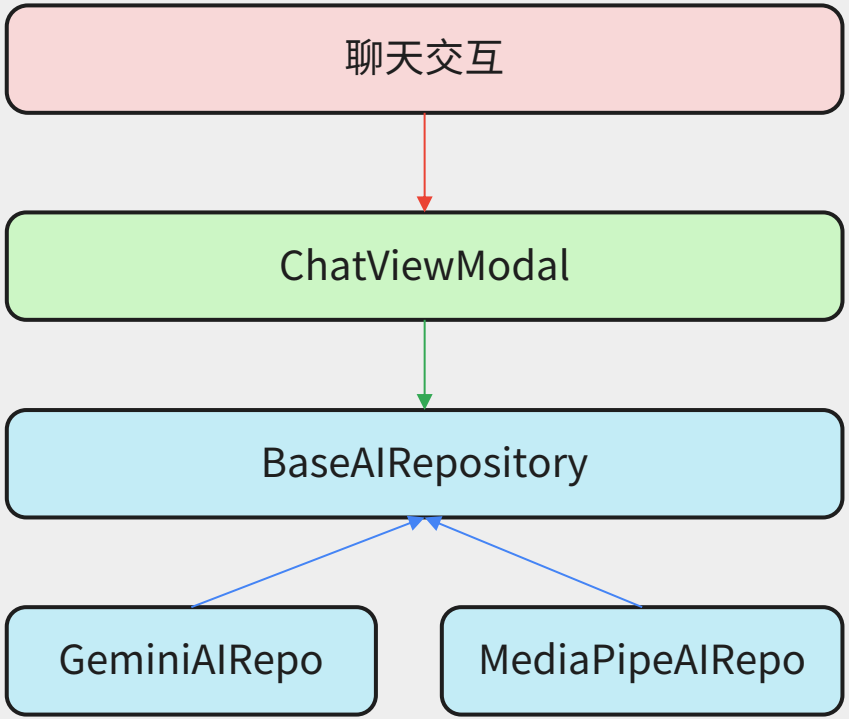
- **应用界面：**只负责展示，不处理数据逻辑
- **ViewModel：**处理上层应用所需的数据、状态和操作逻辑
- **数据仓库：**只负责和数据相关的逻辑，直接和AI引擎直接交互



程序架构

我们使用标准MVVM架构来程序的结构分层更加清晰，职责明确：

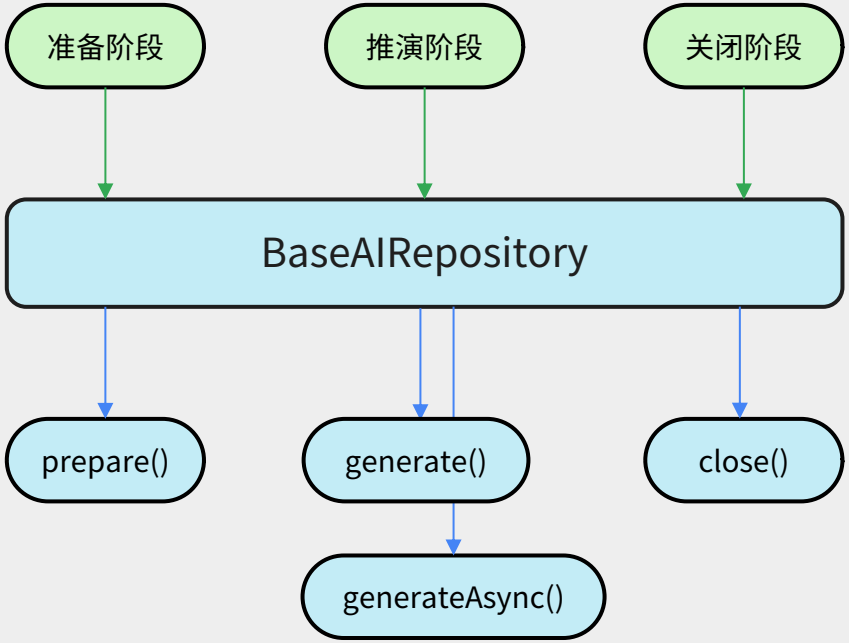
- **应用界面：**只负责数据展示，不处理任何相关的逻辑
- **ViewModel：**处理上层应用所需的数据、状态和操作逻辑
- **数据仓库：**只负责和数据相关的逻辑，直接和AI引擎直接交互



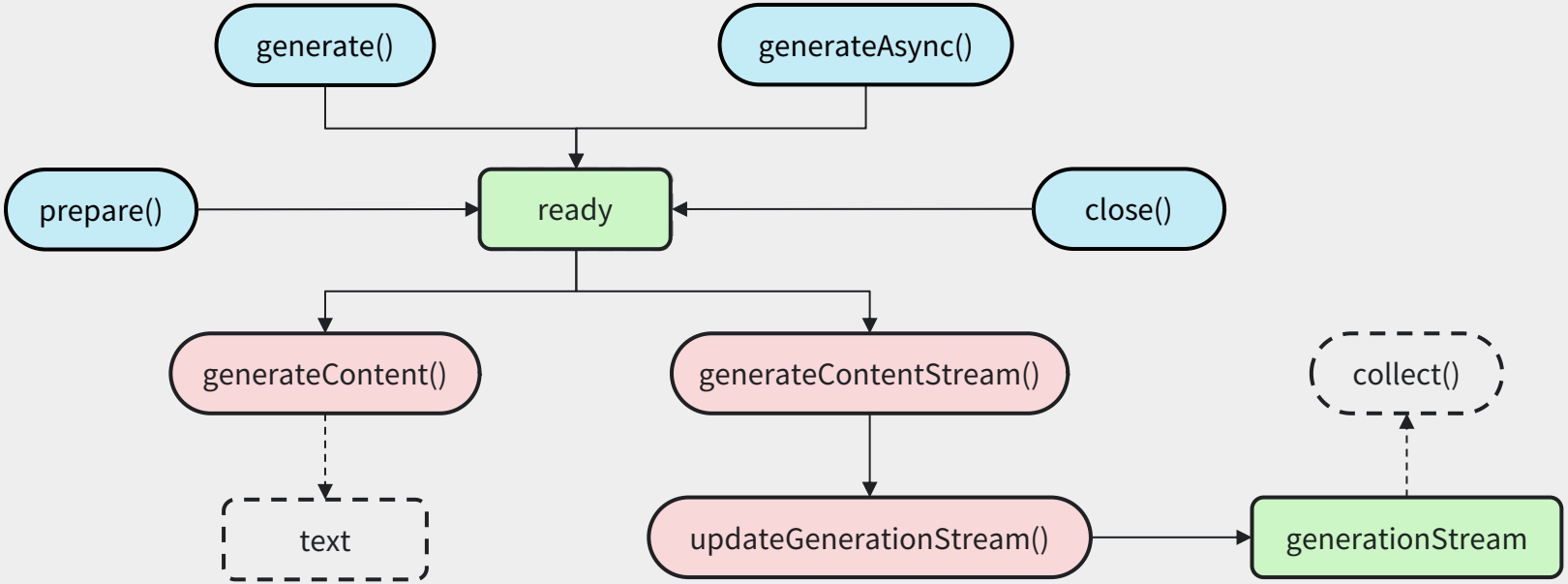
基本流程

一个完整的AI引擎的使用流程，主要分为以下几个步骤：

- **准备阶段：**AI引擎初始化，加载相应的模型资源，设置相关的参数
- **推演阶段：**将用户输入的文字，媒体资源封装，传给AI模型，将模型输出处理后返回给用户
- **关闭阶段：**停止相关的操作，释放引擎占用的系统资源



BaseAIRepository



BaseAIRepository

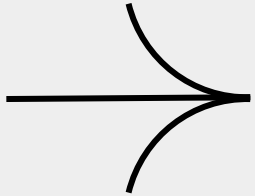
继承的类需要根据自己的情况，使用不同的API去实现这四个方法：

prepare()

generateContent()

generateContentStream()

close()



GeminiAIRepository

GenerativeModel()

generateContent()

generateContentStream()

MediaPipeAIRepository

LlmInference()　LlmInferenceSession()

generateResponse()

generateResponseAsync()

session.close()　inference.close()

GeminiAIRepository

准备阶段

创建一个GenerativeModel的实例，用来做后续的推演。

- 首先要去**AI Studio**里申请一个**API Key**，然后放到项目代码指定的位置

推演阶段

使用同步生成或者流式生成API生成内容。

- 可以使用统一的方法将用户的各种输入组装成提示词内容

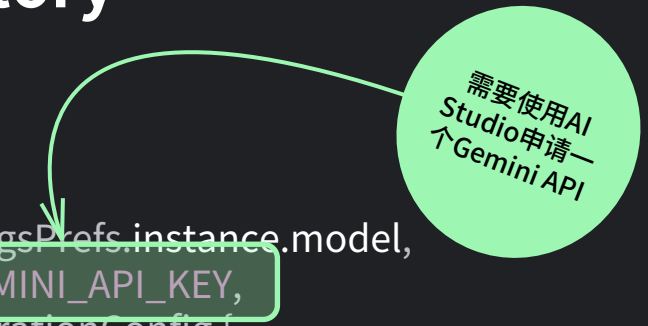
关闭阶段

关闭模型相关的实例，释放系统资源。

- 云端API不需要释放任何本地资源

GeminiAIRepository

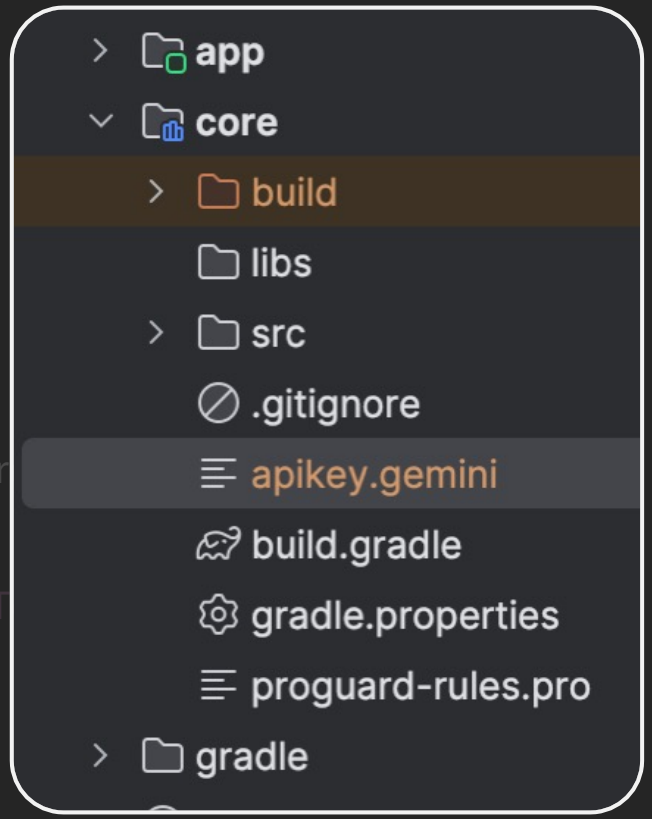
```
override fun prepare() {

    model = GenerativeModel(
        modelName = AppSettingsPrefs.instance.model,
        apiKey = BuildConfig.GEMINI_API_KEY,
        generationConfig = generationConfig {
            temperature = AppSettingsPrefs.instance.temperature
            topK = AppSettingsPrefs.instance.topK
            topP = AppSettingsPrefs.instance.topP
            maxOutputTokens = AppSettingsPrefs.instance.maxTokens
        },
    )

    setReady(true)
}
```



GeminiAIRepository

```
override fun prepare() {

    mode
        mo
        ap
        ge
                                                    peratur
            t
            topP = AppSettingsPrefs.instance.topP
            maxOutputTokens = AppSettingsPrefs.instance.maxT
        },
    )

    setReady(true)
}
```

API Key需要放在core目录下的apikey.gemini文件中。格式是：
API_KEY=xxxxx-xxxxxxxxxxxxxxxxxxxxx



GeminiAIRepository

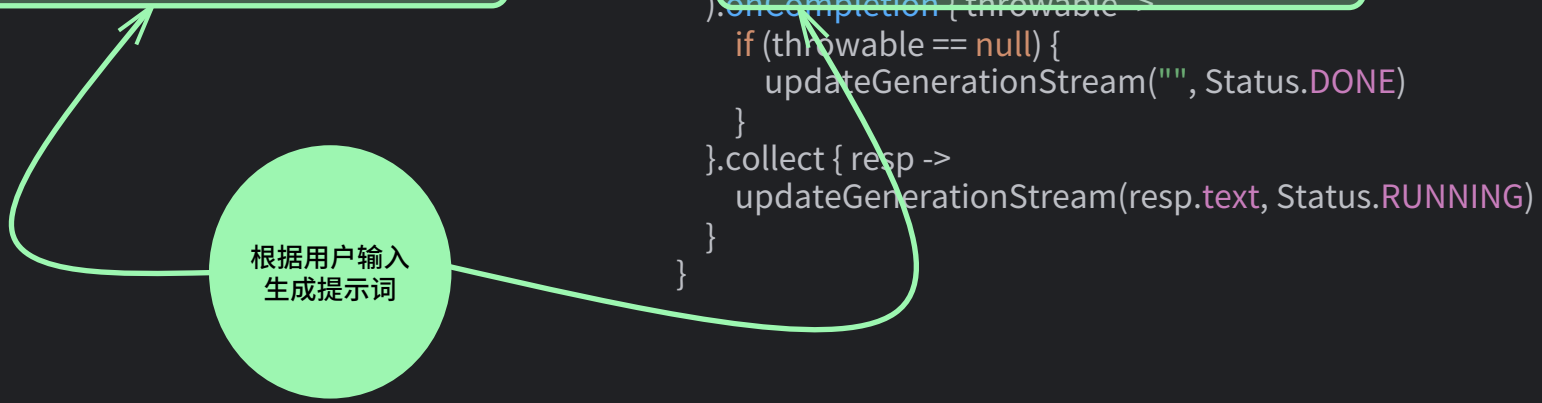
```
override fun prepare() {

    model = GenerativeModel(
        modelName = AppSettingsPrefs.instance.model,
        apiKey = BuildConfig.GEMINI_API_KEY,
        generationConfig = generationConfig {
            temperature = AppSettingsPrefs.instance.temperature
            topK = AppSettingsPrefs.inst        topK
            topP = AppSettingsPrefs.i           P
            maxOutputTokens = App               .instance.maxTokens
        },
    )

    setReady(true)
}
```

不要忘记调用 setReady()

GeminiAIRepository

```
override suspend fun generateContent(
    prompt: String,
    fileUri: String?,
    mimeType: String?
): String? {
    return model.generateContent(
        buildContent(prompt, fileUri, mimeType)
    ).text
}
```

```
override suspend fun generateContentStream(
    prompt: String,
    fileUri: String?,
    mimeType: String?
) {
    model.generateContentStream(
        buildContent(prompt, fileUri, mimeType)
    ).onCompletion { throwable ->
        if (throwable == null) {
            updateGenerationStream("", Status.DONE)
        }
    }.collect { resp ->
        updateGenerationStream(resp.text, Status.RUNNING)
    }
}
```



GeminiAIRepository


```
buildContent(prompt, fileUri, mimeType) {
    return content {
      if (mimeType.contains( “image” )) {
        val bitmap = ...
        bitmap?.let { image(it) }
      } else {
        var stream: InputStream? = null
        try {
          stream = ...
          stream?.let {
            blob(mimeType, stream.readBytes())
          }
        } catch (e: Exception) {
        } finally {
          stream?.close()
        }
      }

      text(prompt)
    }
}
```

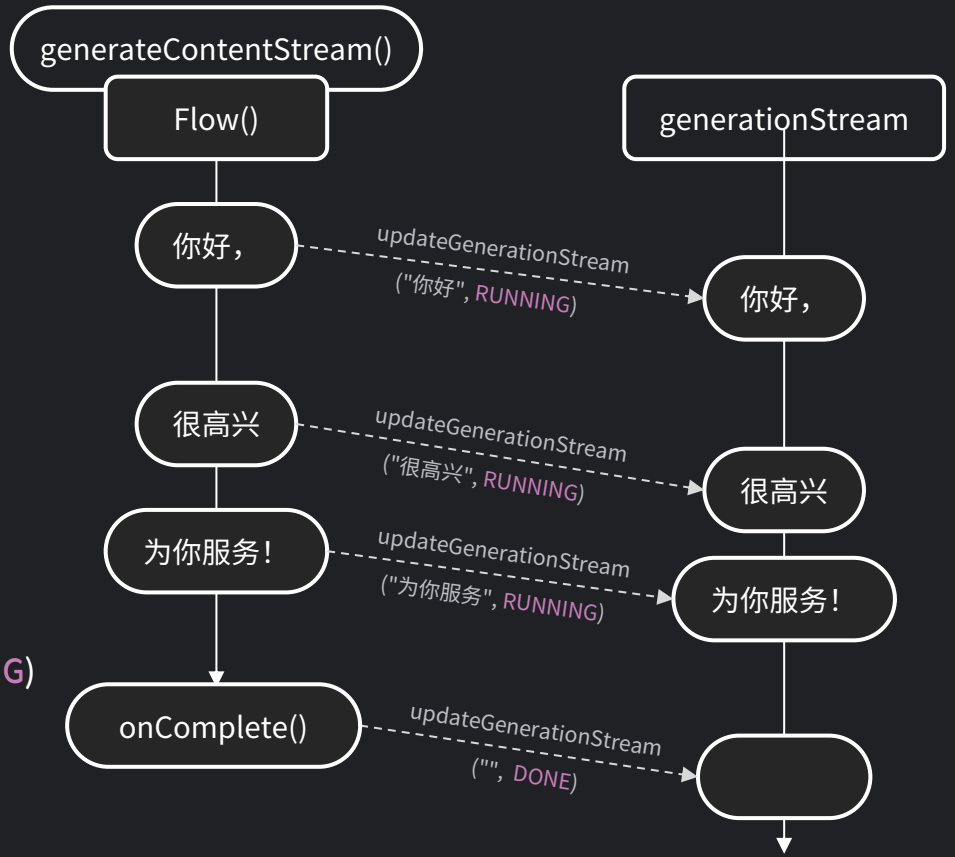
GeminiAIRepository

```
override suspend fun generateContent(
    prompt: String,
    fileUri: String?,
    mimeType: String?
): String? {
    return model.generateContent(
        buildContent(prompt, fileUri, mimeType)
    ).text
}
```

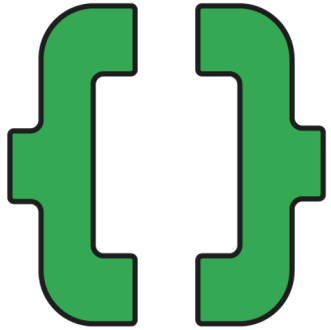
```
override suspend fun generateContentStream(
    prompt: String,
    fileUri: String?,
    mimeType: String?
) {
    model.generateContentStream(
        buildContent(prompt, fileUri, mimeType)
    ).onCompletion { throwable ->
        if (throwable == null) {
            updateGenerationStream("", Status.DONE)
        }
    }.collect { resp ->
        updateGenerationStream(resp.text, Status.RUNNING)
    }
}
```

GeminiAIRepository

```
override suspend fun generateContentStream(
    prompt: String,
    fileUri: String?,
    mimeType: String?
) {
    model.generateContentStream(
        buildContent(prompt, fileUri, mimeType)
    ).onCompletion { throwable ->
        if (throwable == null) {
            updateGenerationStream("", Status.DONE)
        }
    }.collect { resp ->
        updateGenerationStream(resp.text, Status.RUNNING)
    }
}
```



让我们开始吧！

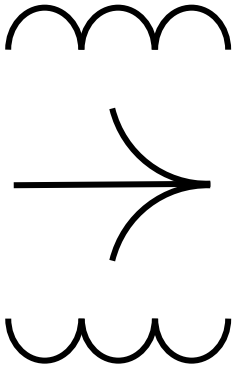


Code Lab of Gemini on Android

https://github.com/dailystudio/gemini-codelab-bwa-04-12

Gemini API

https://ai.google.dev/gemini-api/docs?hl=zh-cn



MediaPipe LLM

https://ai.google.dev/edge/mediapipe/solutions/genai/llm_inference/android

LLM Models

https://ai.google.dev/edge/mediapipe/solutions/genai/llm_inference/index - models

Google Developer Group

Editable Location

谢谢大家！

Build with AI