T Developers

OpenAPI: Building an Android Parser and Tester App

Mario Bodemann

Telco made easy

5G SA$^{(standalone)}$ Testbed in Berlin

- Own 5G Core
- CAMARA APIs
- 4x times this year
- **Goal:** Evaluate API specs and implementation
  - Quality on Demand (QoD; Low Latency & High Throughput)

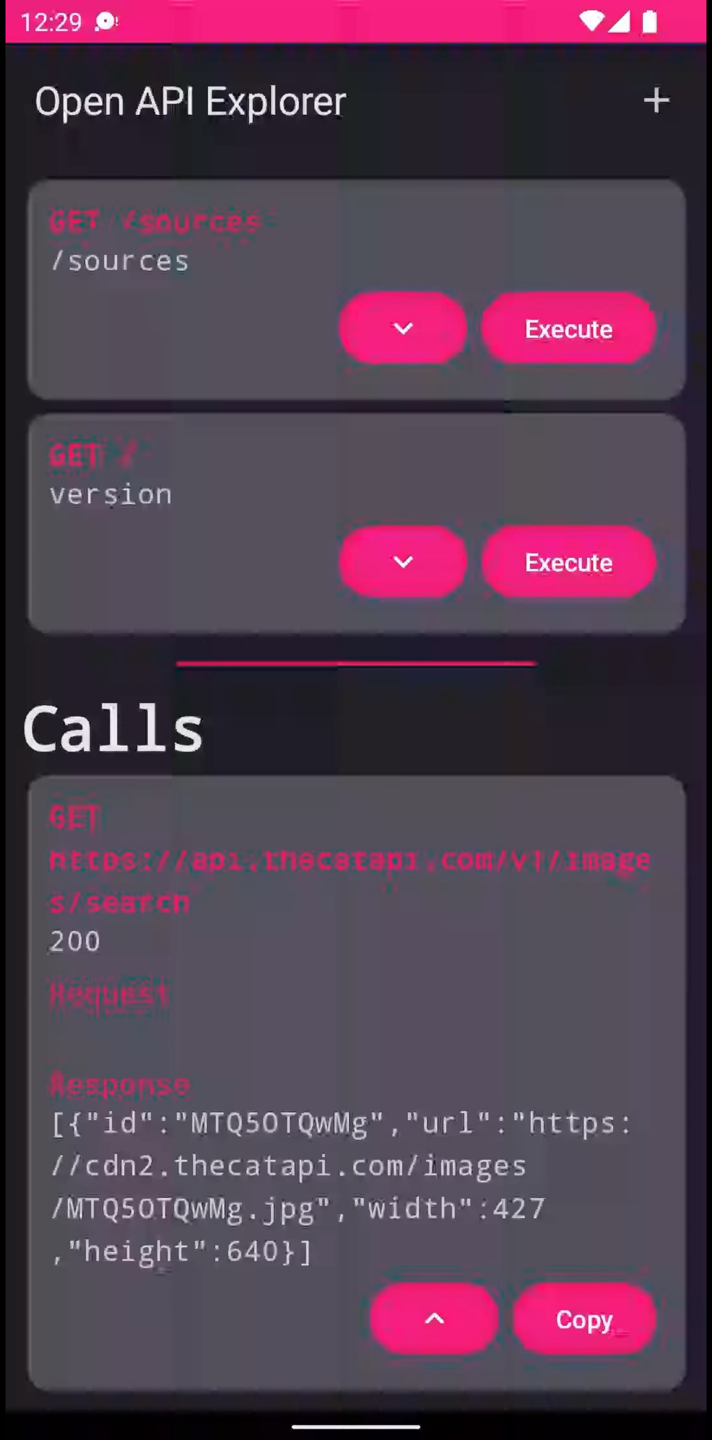
➡️ more at the booth and/or from Noel (“**Network APIs are coming - developers can directly interact with networks**”, Api Governance Track, tomorrow 16:40)

How to adapt to changes of the API?

- Rebuild the integration with every change?
- Never do changes?
- Automatically adapt to changes?

Base Use Cases

Different APIs



Open API Explorer

GET /sources
/sources

GET /
version

Calls

GET
https://api.thecatapi.com/v1/images/search
200

Request

Response

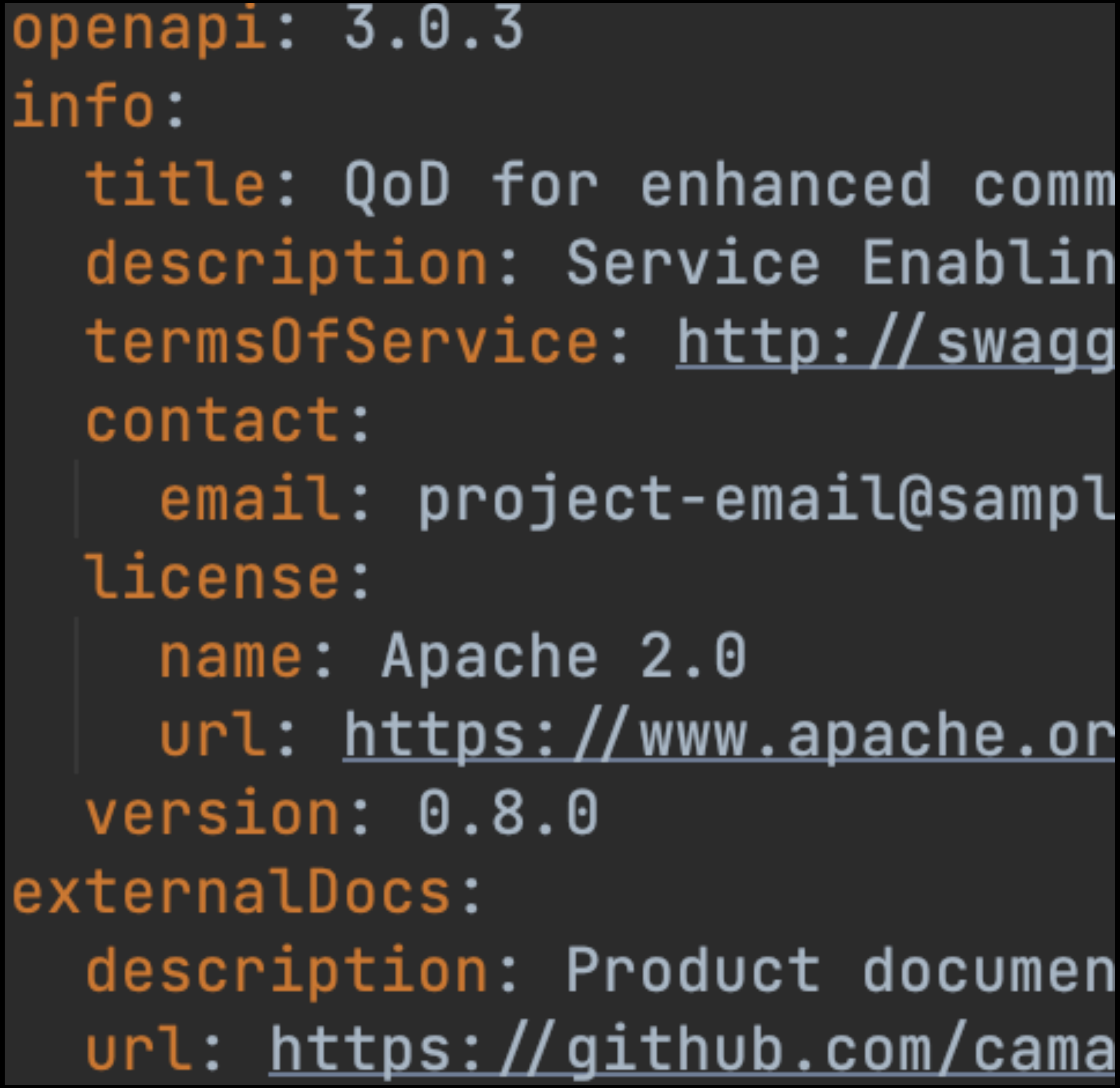
```
[{"id":"MTQ5OTQwMg","url":"https://cdn2.thecatapi.com/images/MTQ5OTQwMg.jpg","width":427,"height":640}]
```

Android Details

- Jetpack Compose
- Kotlin Serialization
- Retrofit
- OkHttp
- com.charleskorn.kaml for Kotlin YML parsing
  see https://github.com/charleskorn/kaml

Open API Spec

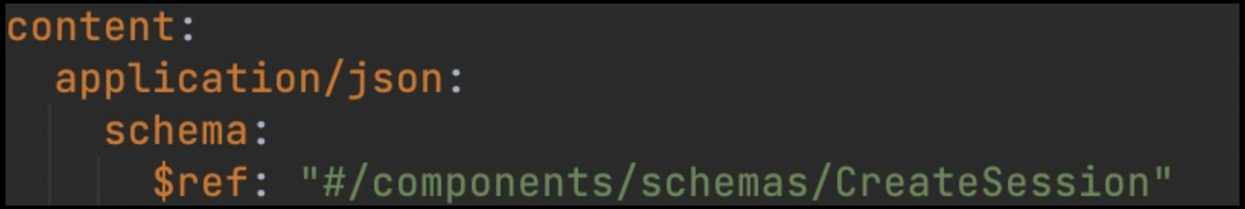
- Standardized
- Widespread
- YML & JSON
- Apparently easy
  - See other talks in this track ;)



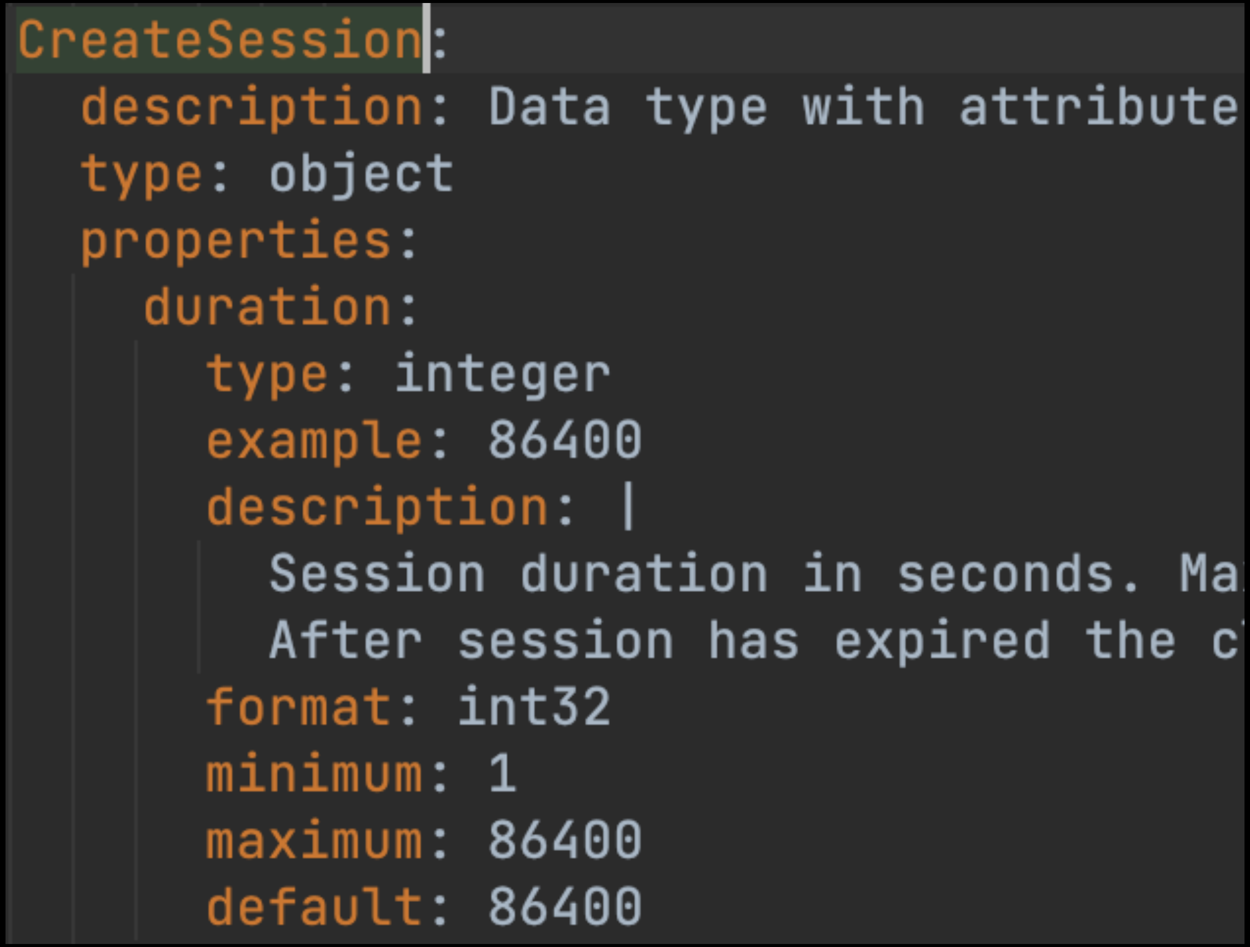
```
openapi: 3.0.3
info:
  title: QoD for enhanced comm
  description: Service Enablin
  termsOfService: http://swagg
  contact:
    email: project-email@sampl
  license:
    name: Apache 2.0
    url: https://www.apache.or
  version: 0.8.0
externalDocs:
  description: Product documen
  url: https://github.com/cama
```

References

- Can appear in well defined ...
- .. but unexpected places
- Naïve implementation needs manual parsing



```
content:
  application/json:
    schema:
      $ref: "#/components/schemas/CreateSession"
```

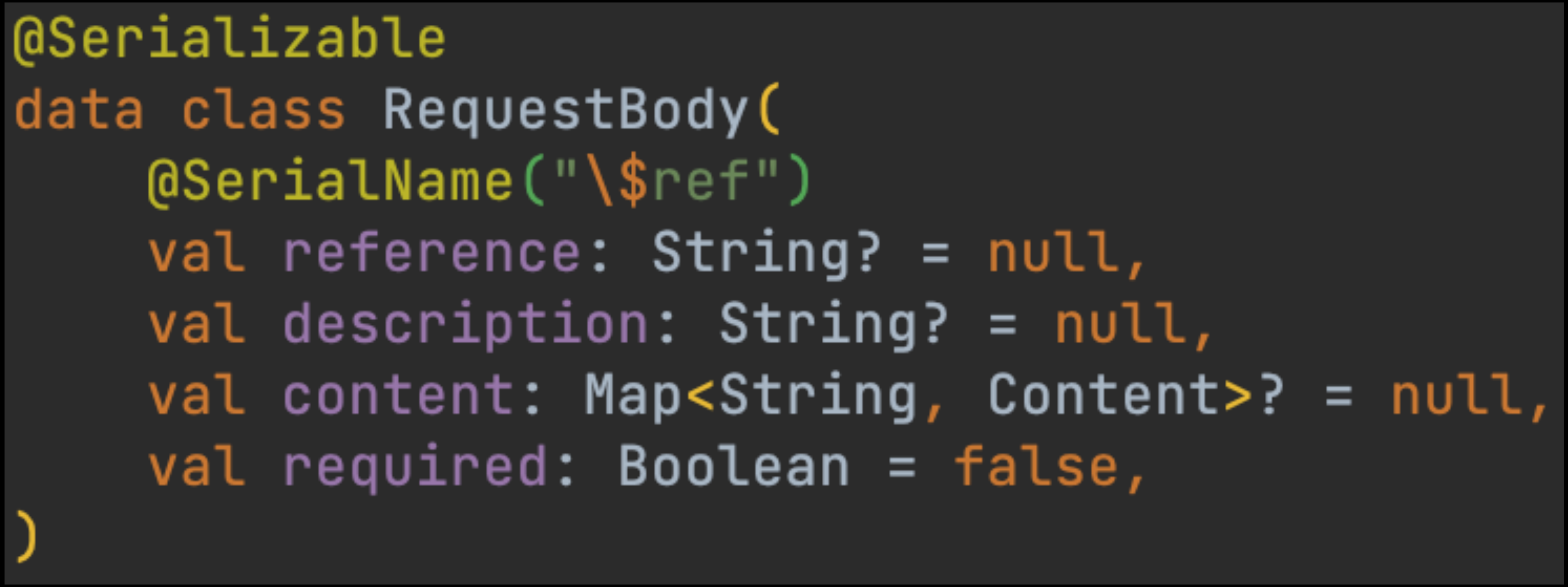


```
CreateSession:
  description: Data type with attribute
  type: object
  properties:
    duration:
      type: integer
      example: 86400
      description: |
        Session duration in seconds. Ma
        After session has expired the c
      format: int32
      minimum: 1
      maximum: 86400
      default: 86400
```

References Parsed

Method

- Parse once with KAML:

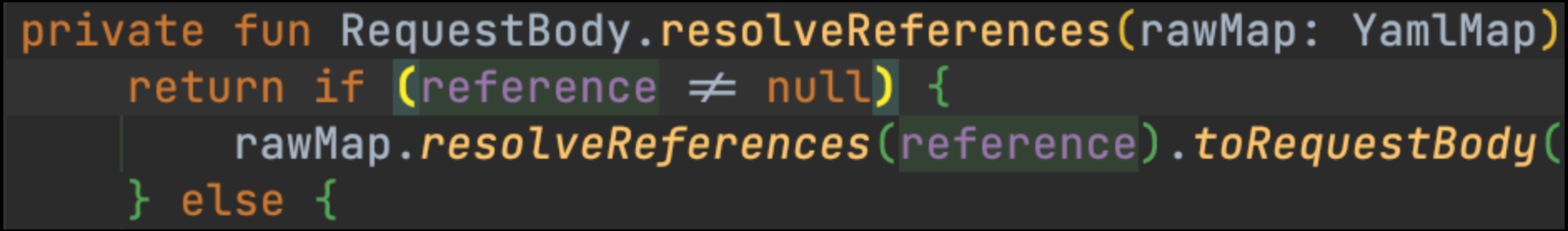


```kotlin
@Serializable
data class RequestBody(
    @SerialName("\$ref")
    val reference: String? = null,
    val description: String? = null,
    val content: Map<String, Content>? = null,
    val required: Boolean = false,
)
```

References Parsed 2.0

Method

- traverse all fields and check its **reference** field

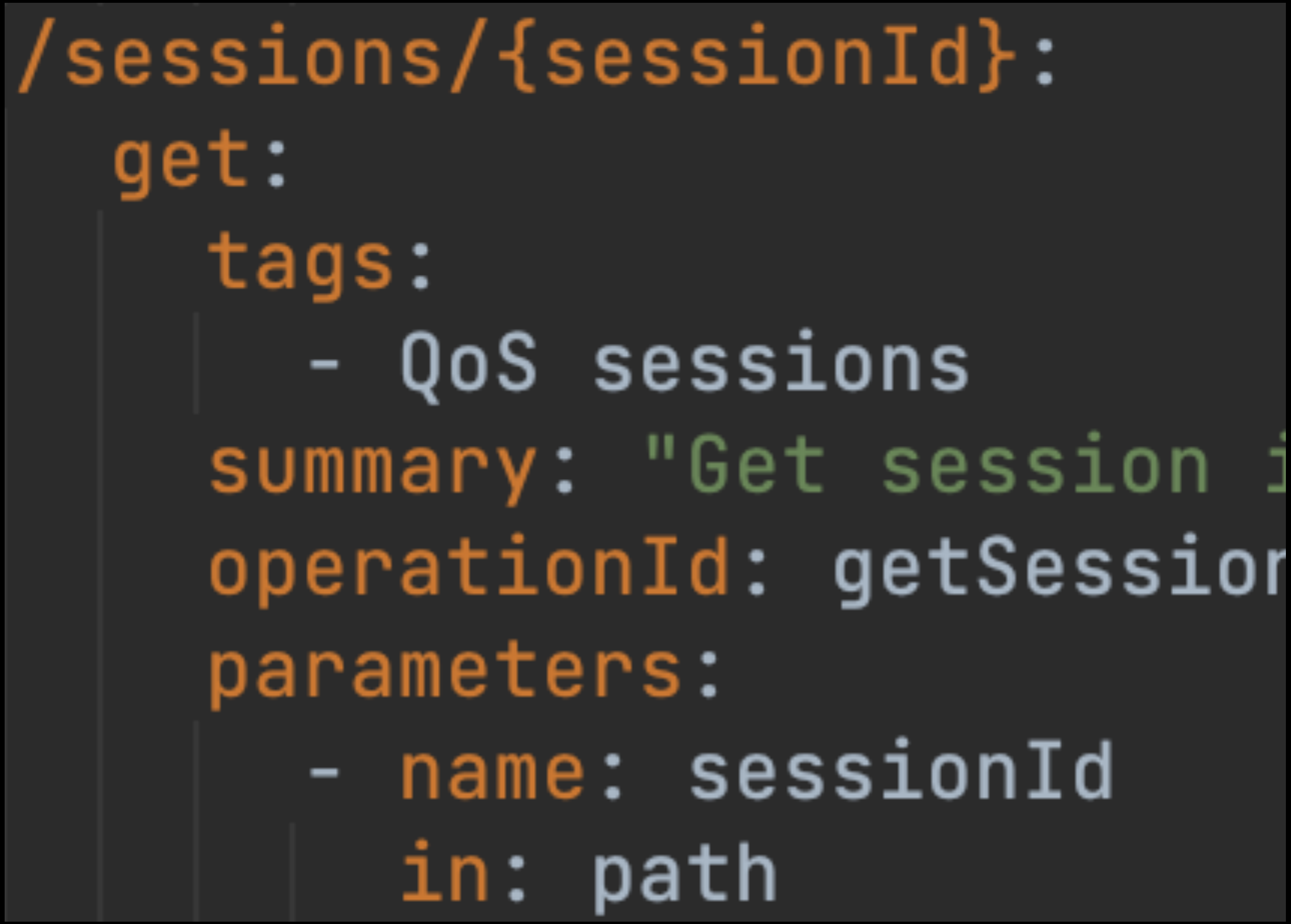


```
private fun RequestBody.resolveReferences(rawMap: YamlMap)
    return if (reference ≠ null) {
        rawMap.resolveReferences(reference).toRequestBody(
    } else {
```

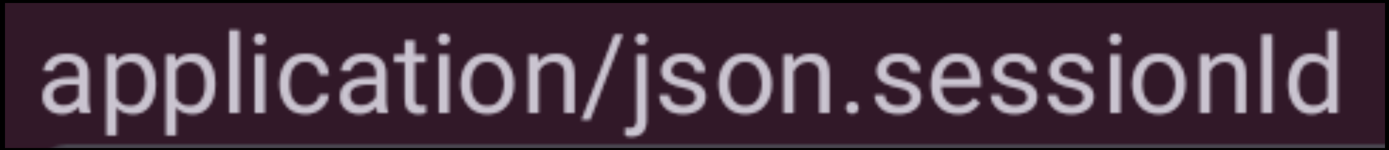
- Find it in references **components** section of specification
- Convert found component into **RequestBody** (or whatever contains the **reference** field)
- Continue down the line

User Input Generation

- User clicked on **execute**
- Task: find all parameter and variables
- Travesre api specification (server, and selected endpoint/method)
- Remember by user input requests by path
- Use that memory to build dialog



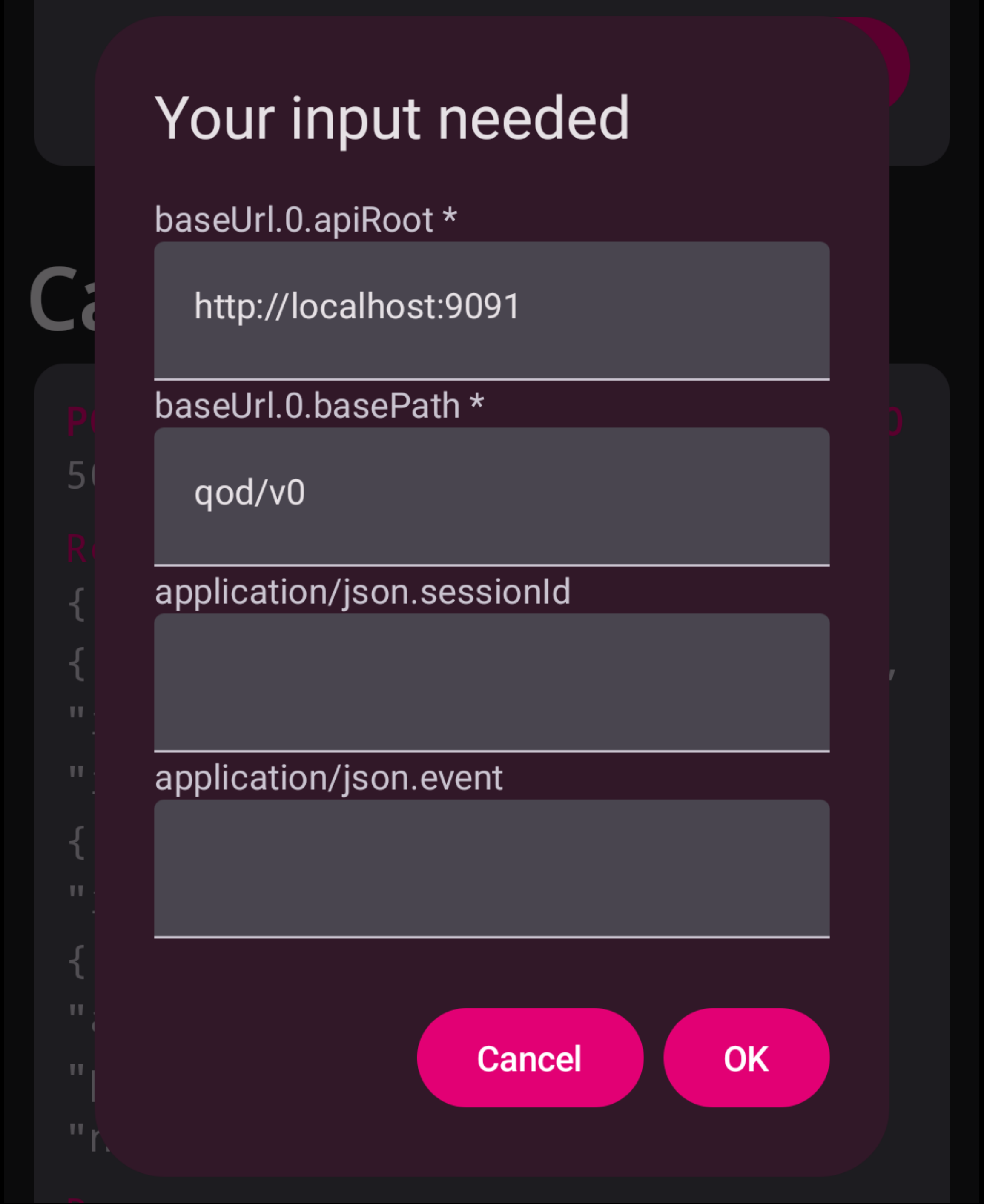
```
/sessions/{sessionId}:
  get:
    tags:
      - QoS sessions
    summary: "Get session i
    operationId: getSessior
    parameters:
      - name: sessionId
        in: path
```



application/json.sessionId

Dialog Building

- Build Dialog based on found variables and parameters
- For every parameter use its path and a default or saved value as input
- Once confirmed
    - Save user input to shared preferences
    - Iterate through input to build api call



Your input needed

baseUrl.0.apiRoot *

http://localhost:9091

baseUrl.0.basePath *

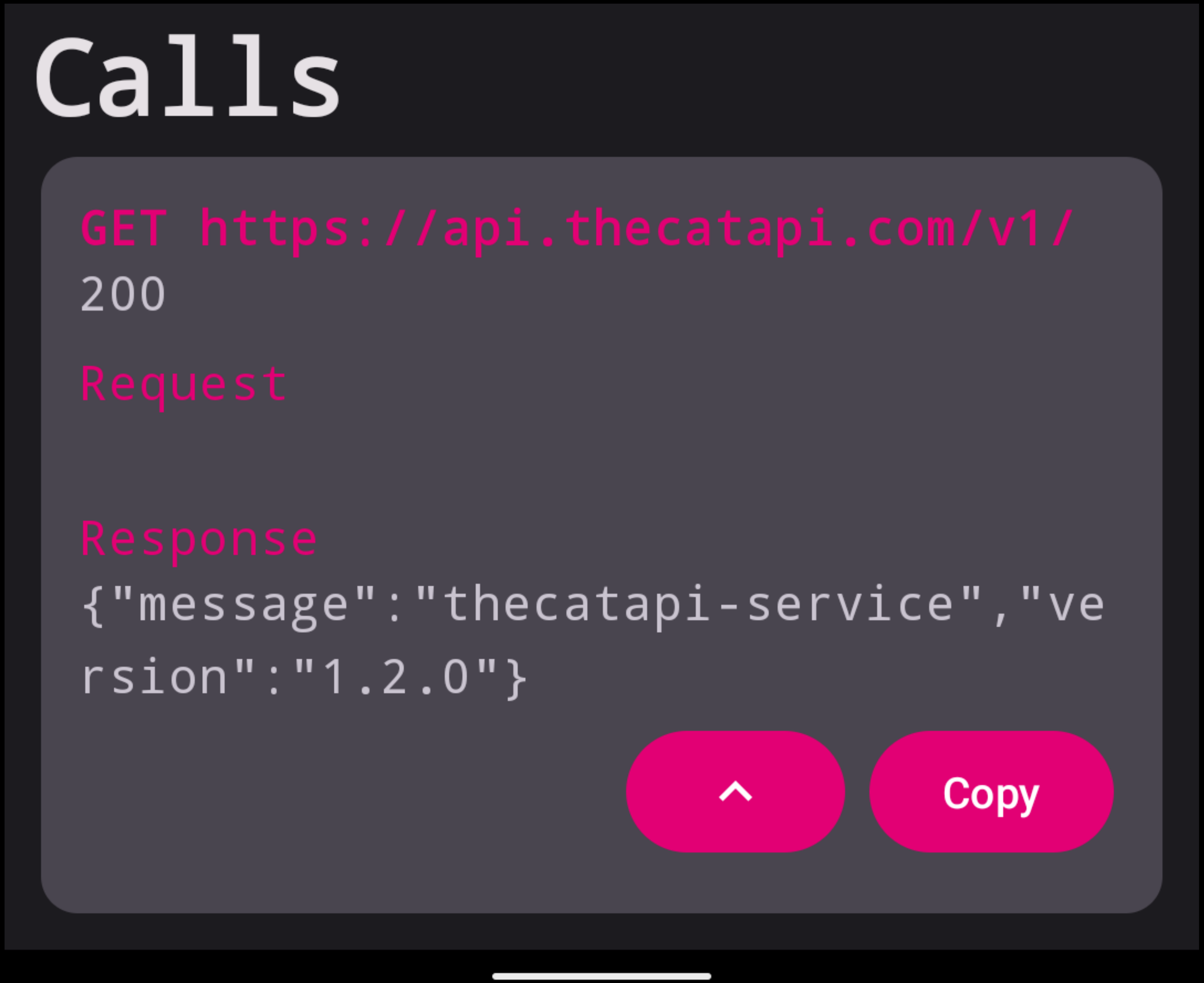
qod/v0

application/json.sessionId

application/json.event

Filing Data

- Traverse API Spec
- Build Body and Server
- Fill in any parameters or variable parts
- Execute call using okhttp
- And display result



Calls

GET https://api.thecatapi.com/v1/
200

Request

Response

```
{"message":"thecatapi-service","version":"1.2.0"}
```

!Success!

Changes of APIs are now easy, just update it in the app™

Next steps

- Resilience towards more places a reference can appear
- Better authorization handling
    - Oauth?
    - Auth Headers?
- Markdown support?
- Images?
- Repeation of calls?
- UI / UX?

Developers

Telco made easy

Who wants us to publish the code?

Stay tuned at https://github.com/dt-developers.