RAG basics

JUNE 2025

There are no magic! It is simple REST API!

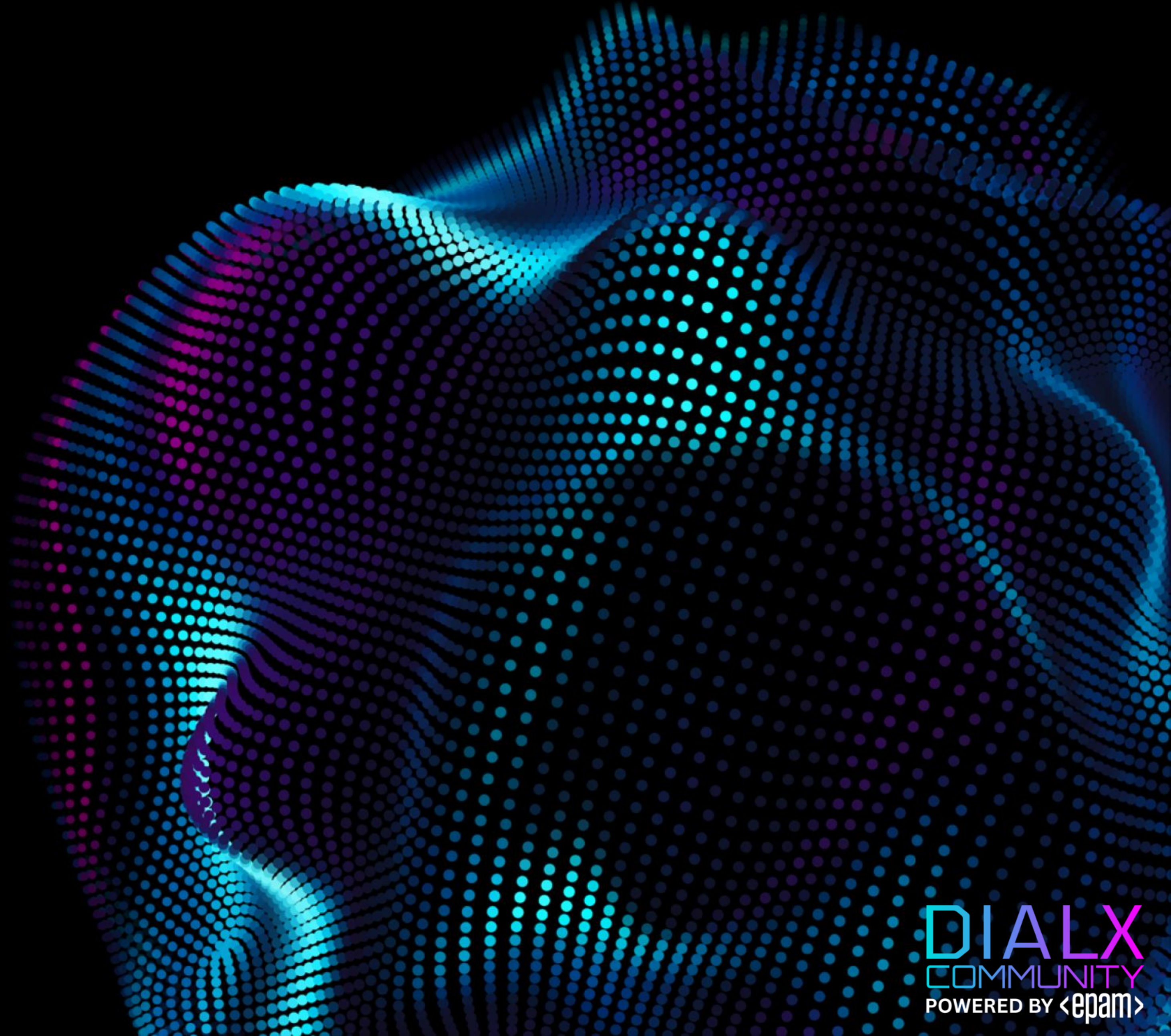
Before we start:

- Raise your hand and ask questions if you have any
- It is better to ask questions when you have
- Also, type them in chat
- We will need DIAL API key for this session

Agenda:

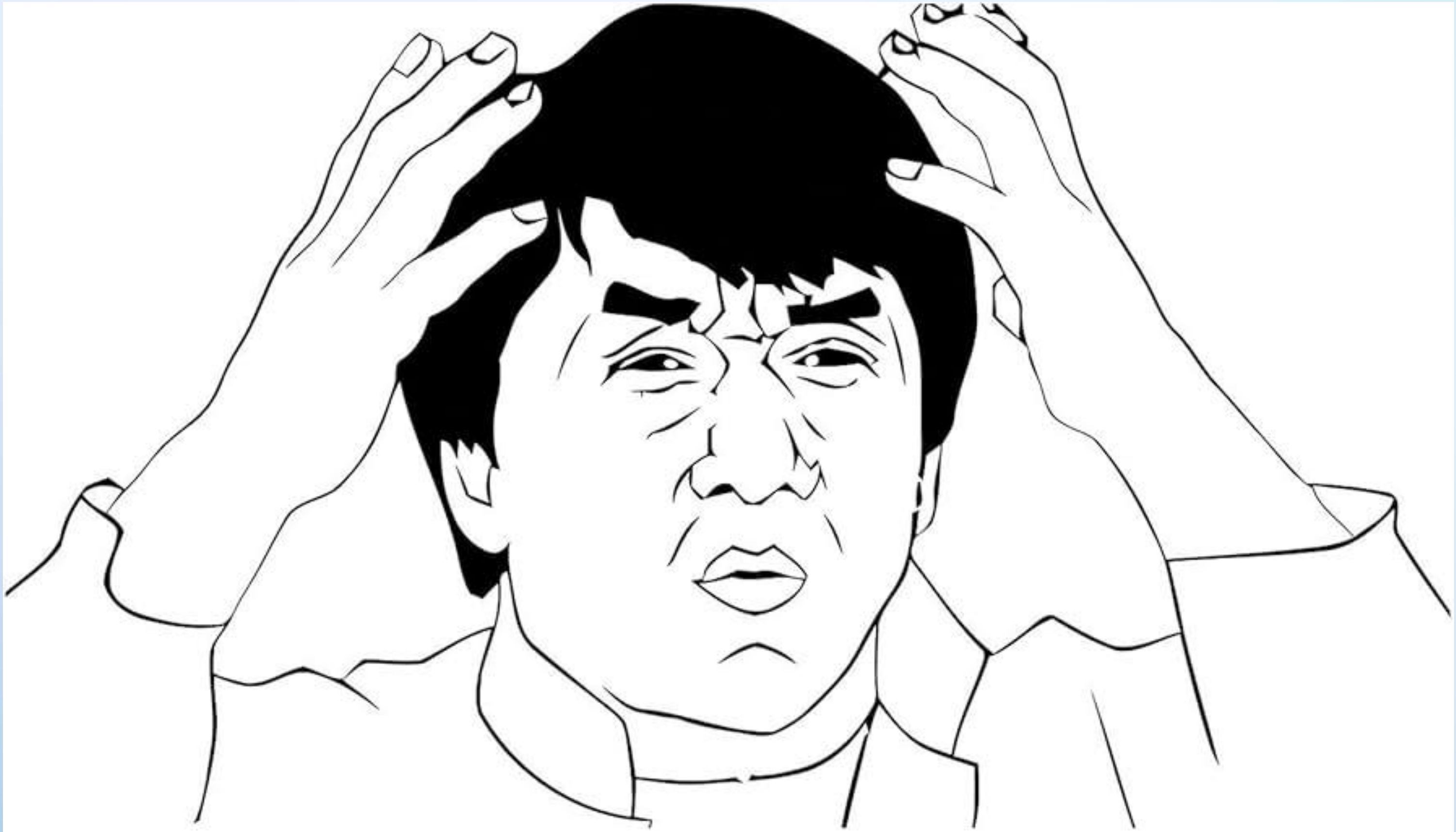
- Presentation:
  - About RAG Concept
  - Real life sample
  - Application architecture and main components
- Workshop:
  - Create MicrowaveRAG application

RAG Concept



DIALX
COMMUNITY
POWERED BY <epam>

RAG = Retrieval-Augmented Generation



What does it mean?

Retrieval-augmented generation (RAG) is a technique that enables Gen AI models to retrieve and incorporate new information.
It modifies interactions with a LLM so that the model responds to user queries with reference to a specified set of documents, using this information to supplement information from its pre-existing training data.
This allows LLMs to use domain-specific and/or updated information. Use cases include providing chatbot access to internal company data or generating responses based on authoritative sources.

R A G concept:

Retrieval

- The system searches through an external knowledge base (documents, databases, webpages, or vector stores) to find information relevant to the user query.

- Often, this is done using vector embeddings (semantic search) to find relevant documents based on similarity measures.

R A G concept:

Retrieval

Augmentation

The retrieved information is then used to extend or "augment" the context provided to the language model.

«User input + Retrieved data»

R A G concept:

Retrieval

Augmentation

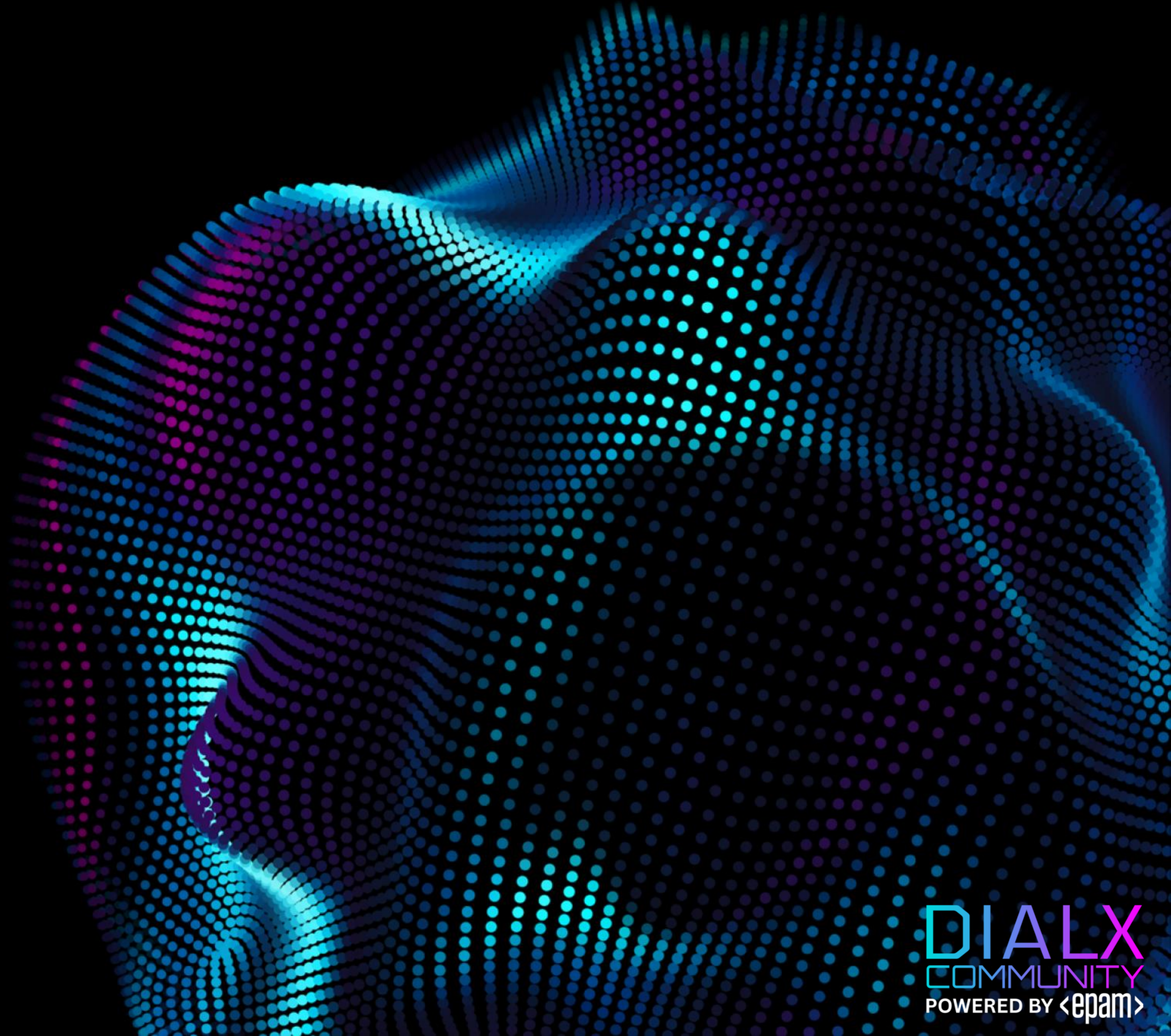
Generation

The LLM generates response based on the provided information (user input + retrieved data)

Retrieval-augmented generation (RAG) is a technique that helps us:

- Provide the most relevant context data based on user request
- Reduces hallucinations
- Enables up-to-date knowledge
- Enables domain specialization
- Reduces context window usage
- Reduces costs (not always*)

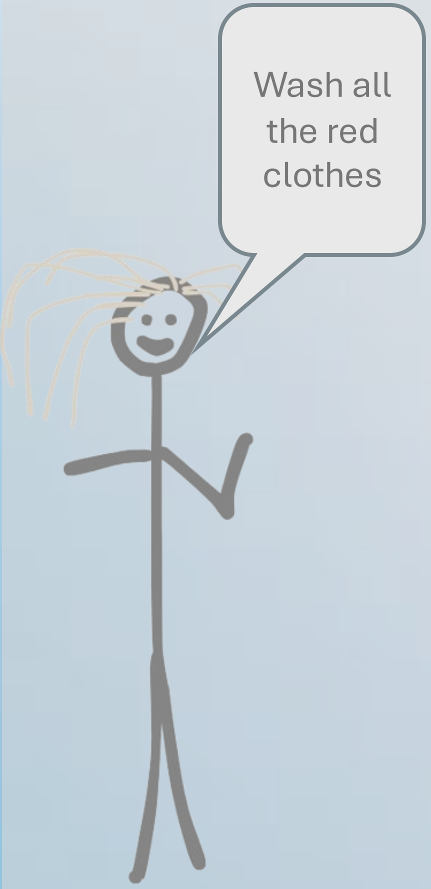
Sample



DIALX
COMMUNITY
POWERED BY <epam>

User request

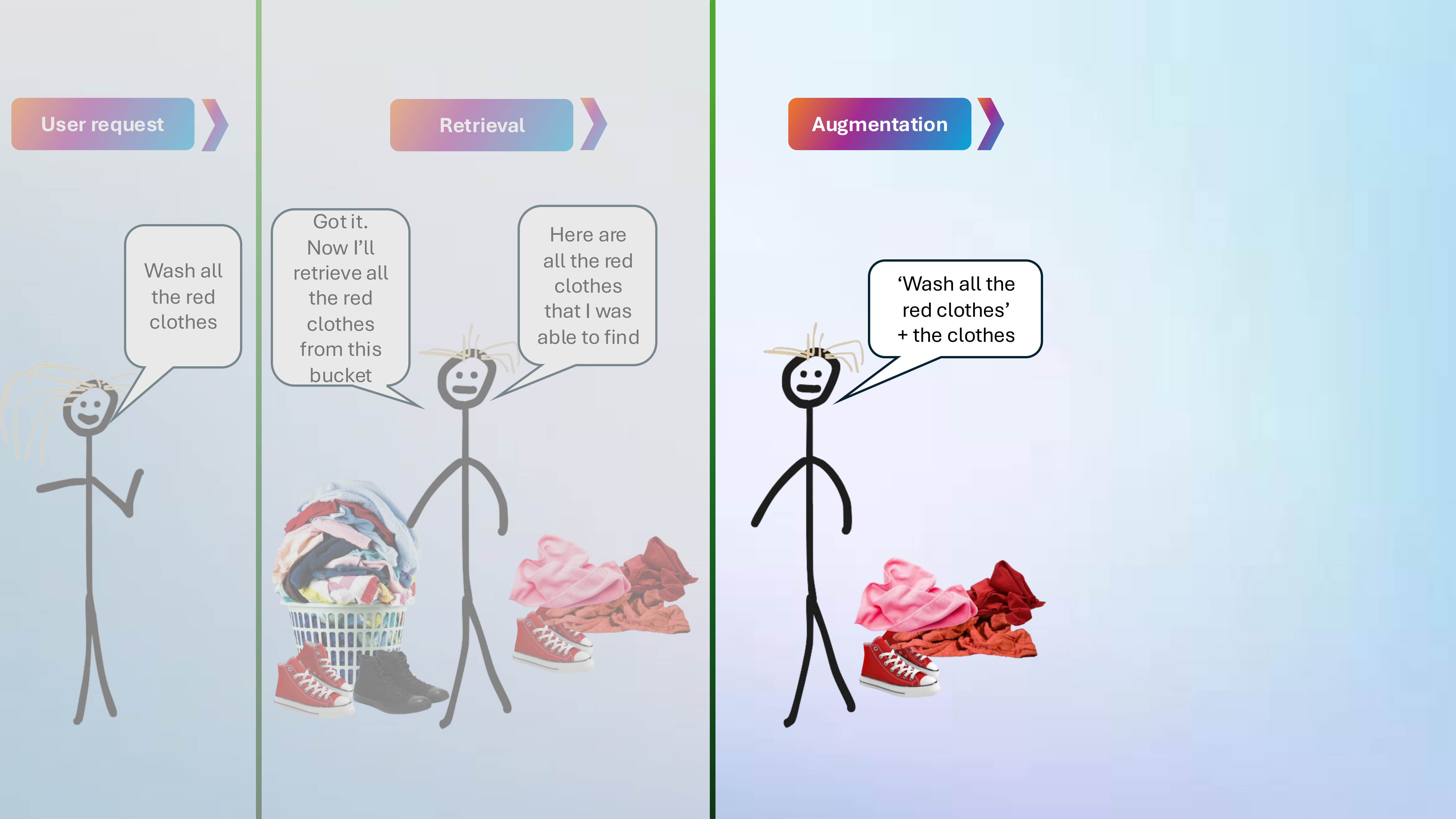
User request

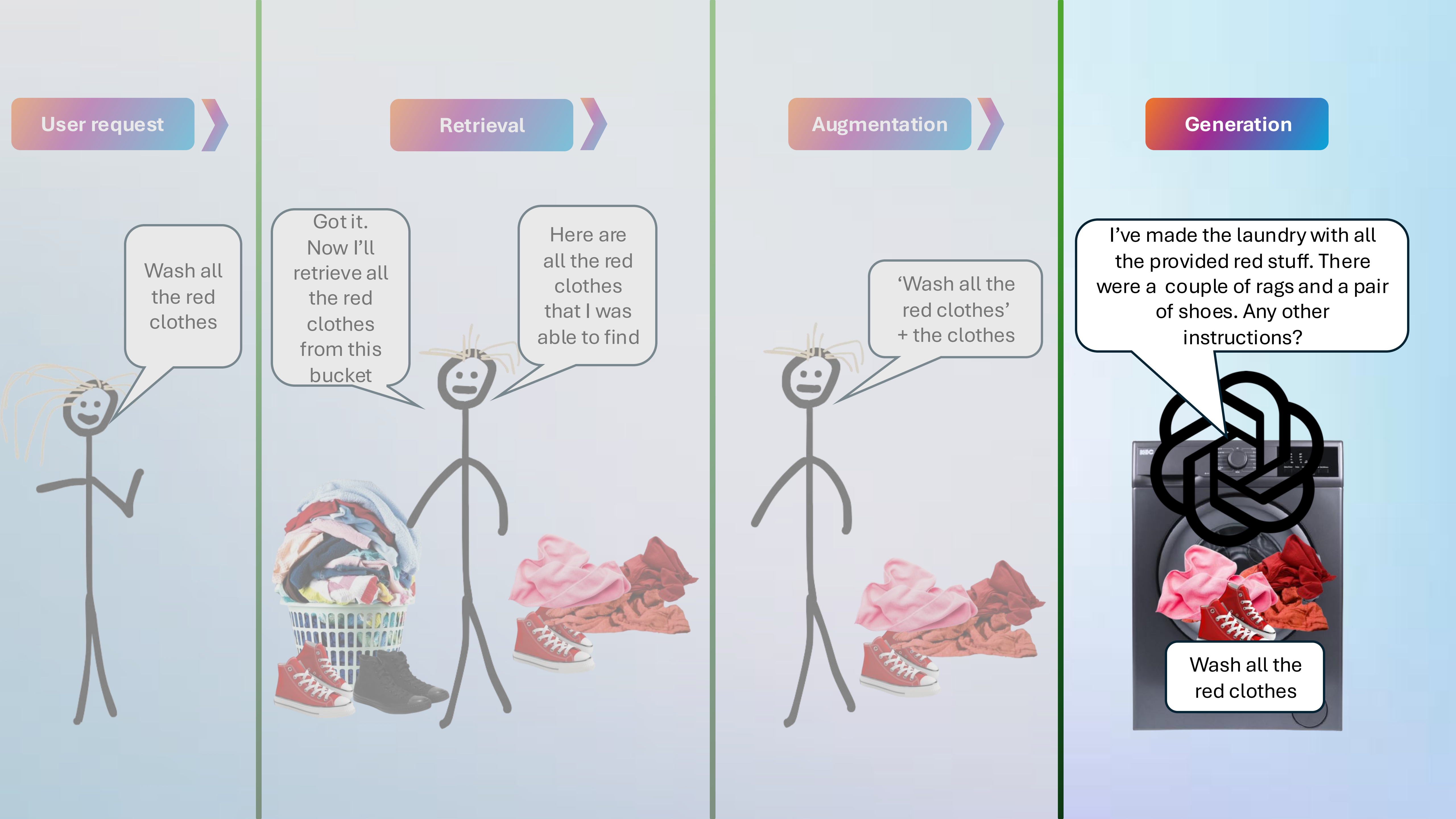


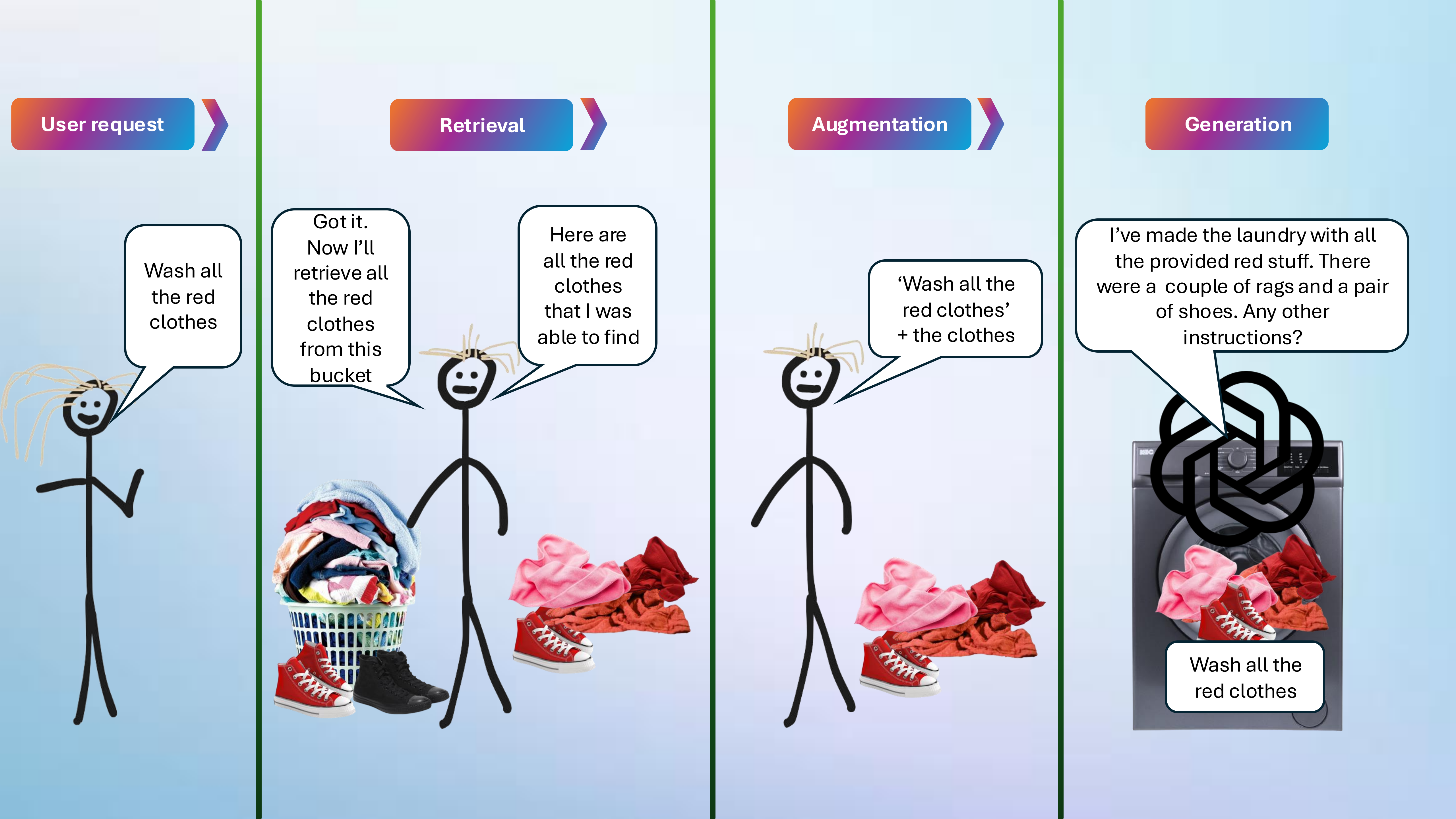
Retrieval

User request

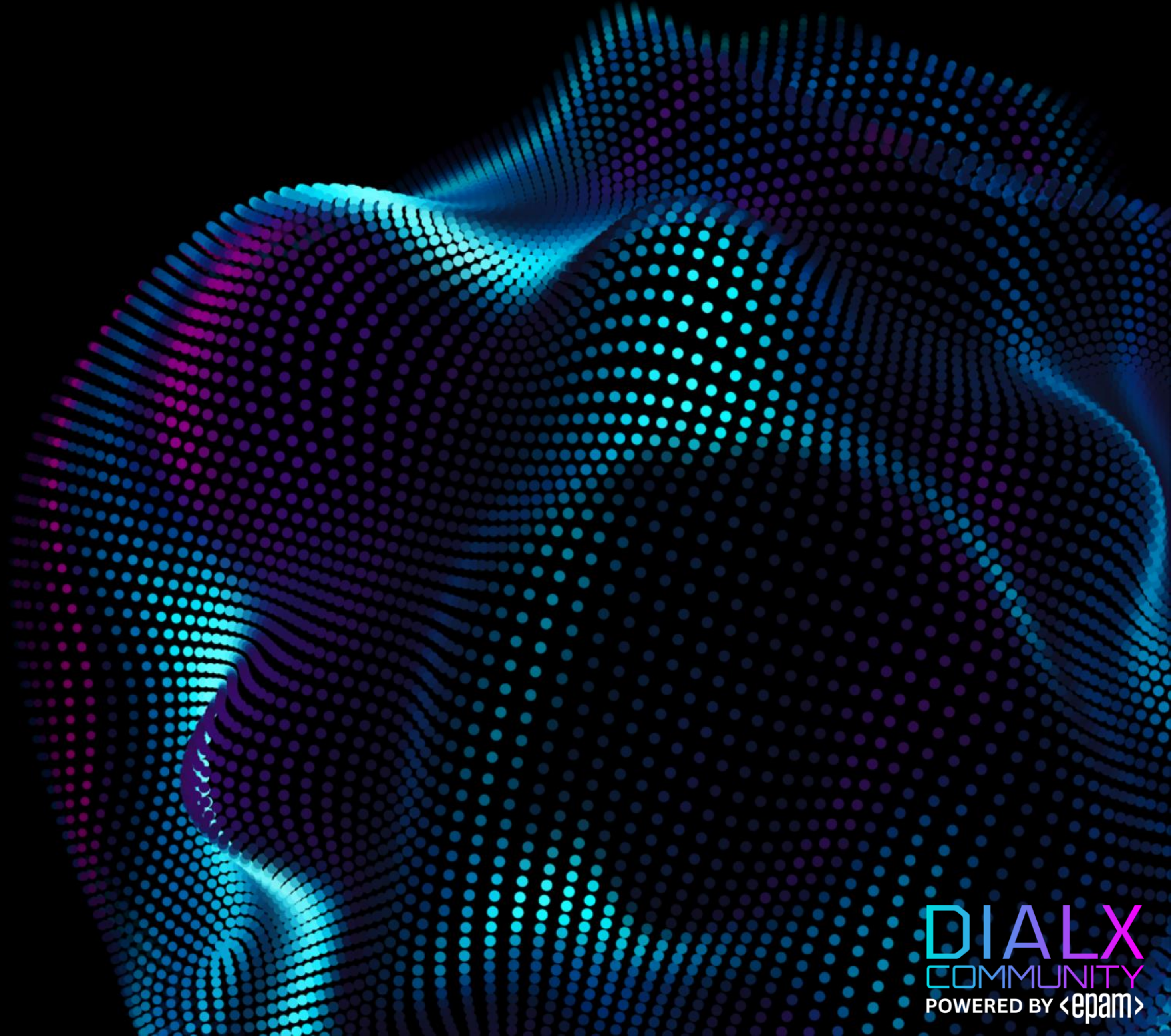
Retrieval


User request
Wash all the red clothes
Retrieval
Got it. Now I'll retrieve all the red clothes from this bucket
Here are all the red clothes that I was able to find
Augmentation
'Wash all the red clothes' + the clothes

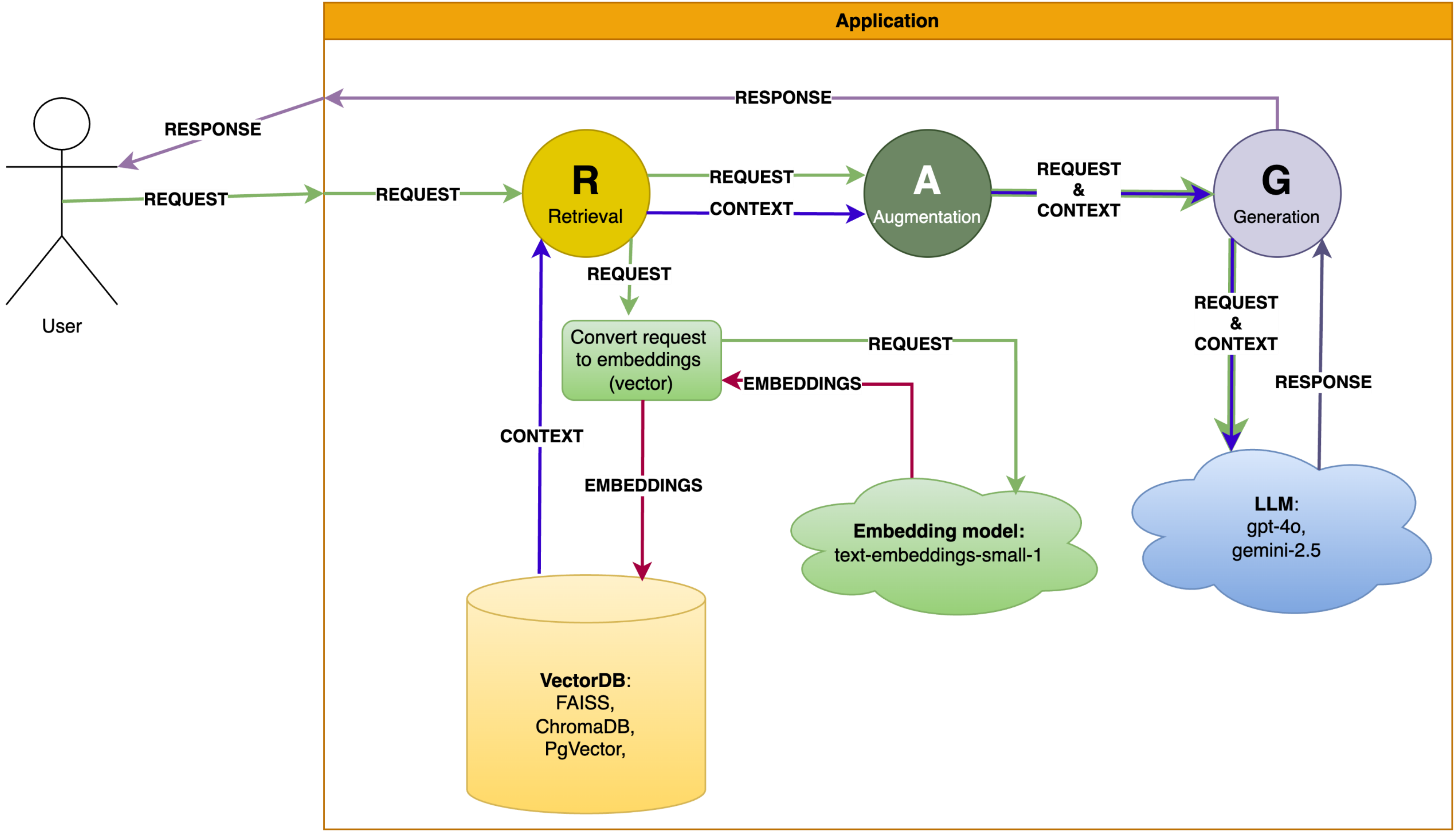

User request
Retrieval
Augmentation
Generation
Wash all the red clothes
Got it. Now I'll retrieve all the red clothes from this bucket
Here are all the red clothes that I was able to find
'Wash all the red clothes' + the clothes
I've made the laundry with all the provided red stuff. There were a couple of rags and a pair of shoes. Any other instructions?
Wash all the red clothes


User request
Wash all the red clothes
Retrieval
Got it. Now I'll retrieve all the red clothes from this bucket
Here are all the red clothes that I was able to find
Augmentation
'Wash all the red clothes' + the clothes
Generation
I've made the laundry with all the provided red stuff. There were a couple of rags and a pair of shoes. Any other instructions?
Wash all the red clothes

Application



DIALX COMMUNITY
POWERED BY <epam>



Application
RESPONSE
RESPONSE
REQUEST
REQUEST
User
R
Retrieval
REQUEST
CONTEXT
A
Augmentation
REQUEST
&
CONTEXT
G
Generation
REQUEST
Convert request to embeddings (vector)
REQUEST
EMBEDDINGS
CONTEXT
EMBEDDINGS
REQUEST
&
CONTEXT
RESPONSE
Embedding model:
text-embeddings-small-1
LLM:
gpt-4o,
gemini-2.5
VectorDB:
FAISS,
ChromaDB,
PgVector,

Client that is working with Embedding model. Via this client you will convert text chunks and user requests into embeddings. Embedding models:

- text-embedding-3-large-1
- text-embedding-3-smal-1

```
AzureOpenAIEmbeddings(
        deployment='text-embedding-3-large-1',
        azure_endpoint=DIAL_URL,
        api_key=SecretStr(API_KEY),
)
```

Client to work with LLM:

- gpt-4o
- gemini-2.5
- sonnet-3.7

```
AzureChatOpenAI(
        temperature=0.0,
        azure_deployment='gpt-4o-2024-08-06',
        azure_endpoint=DIAL_URL,
        api_key=SecretStr(API_KEY),
        api_version="2024-05-01-preview"
)
```

Via this API you will be able to load locally saved indexed FAISS Vector DB. This is needed to avoid creation new DB each time when Application is started

```
FAISS.load_local(
        folder_path='microwave_faiss_index',
        embeddings=self.embeddings,
        allow_dangerous_deserialization=True,
)
```

Will convert Chunks (documents) into embeddings and create FAISS Vector DB with them. Then you can locally save it.

```
FAISS.from_documents(chunks, self.embeddings)
```

Will convert Chunks (documents) into embeddings and create FAISS Vector DB with them. Then you can locally save it.

```
RecursiveCharacterTextSplitter(
        chunk_size=300,
        chunk_overlap=50,
        separators=["\n\n", "\n", "."]
)
```

Via this API you can make similarity search relevant context to user request (query).

- k – limit of results that we expect to get
- score_threshold – filter with similarity score. Range 0.0-1.0

```
vectorstore.similarity_search_with_relevance_scores(
        query,
        k=k,
        score_threshold=score
)
```

DIALX COMMUNITY POWERED BY <epam>

Join us:

Subscribe to WeAreCommuntiy

https://wearecommunity.io/communities/dial

Keep in touch with our latest updates. Here you find webinars, workshops and articles about DIALX features and products.

Subscribe to YouTube

https://www.youtube.com/@TeamDIALX

Here we publish videos about our newest products and features.

Join our Discord community

https://discord.gg/jvTCQv4E4q

✨AI DIALX Community✨ is the place where you can find help with your questions about DIALX, direct communication with DIALX team and contributors.

DIALX COMMUNITY

POWERED BY <epam>

Thank you!