



Conquering the complexity of Agent Building with Xircuits



Xpress AI

Hello. I'm Paul Dubs.

- Software developer for 20+ years
- ML practitioner for 10+ years
- CTO & Co-Founder of Xpress AI
 - Currently headquartered in Japan
 - Building a platform that allows everyone to create their own agents – in the cloud and on-prem.



What is an Agent?

agency:

“the capacity, condition or state of acting or exerting power”

agent:

“one that acts or exerts power”

Imagine something like
ChatGPT with Arms & Legs



Behold,
An Agent!



A meme featuring Rick Sanchez, Morty Smith, and Mr. Poopybutthole from the animated series Rick and Morty. Rick is on the left, looking surprised with his mouth open. Morty is in the center, also looking surprised. Mr. Poopybutthole is on the right, looking shocked with wide eyes and an open mouth. The background shows their workshop with various tools and a window looking out onto a landscape. The text "AGENT?" is at the top, and "THAT SOUNDS LIKE A CHATBOT WITH EXTRA STEPS" is at the bottom.

AGENT?

**THAT SOUNDS LIKE A
CHATBOT WITH EXTRA STEPS**

“Can you do X for me?”

Chatbot

“**No.** But here are some instructions for how you could try to do X yourself.”

Chatbot with RAG

“**No.** But here are some instructions, *I found*, for how you could try to do X yourself.”

Chatbot with Function Calling

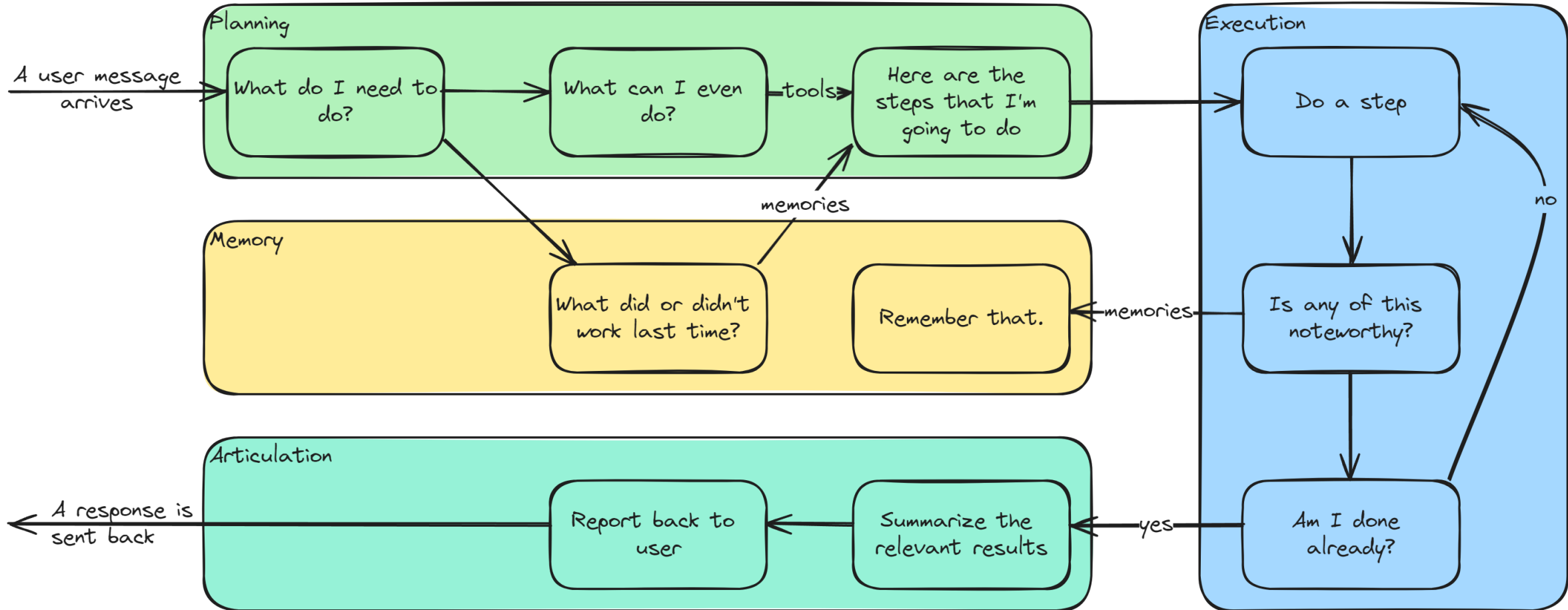
“call X(args)”

Agent

“Sure, let me get started on that.”

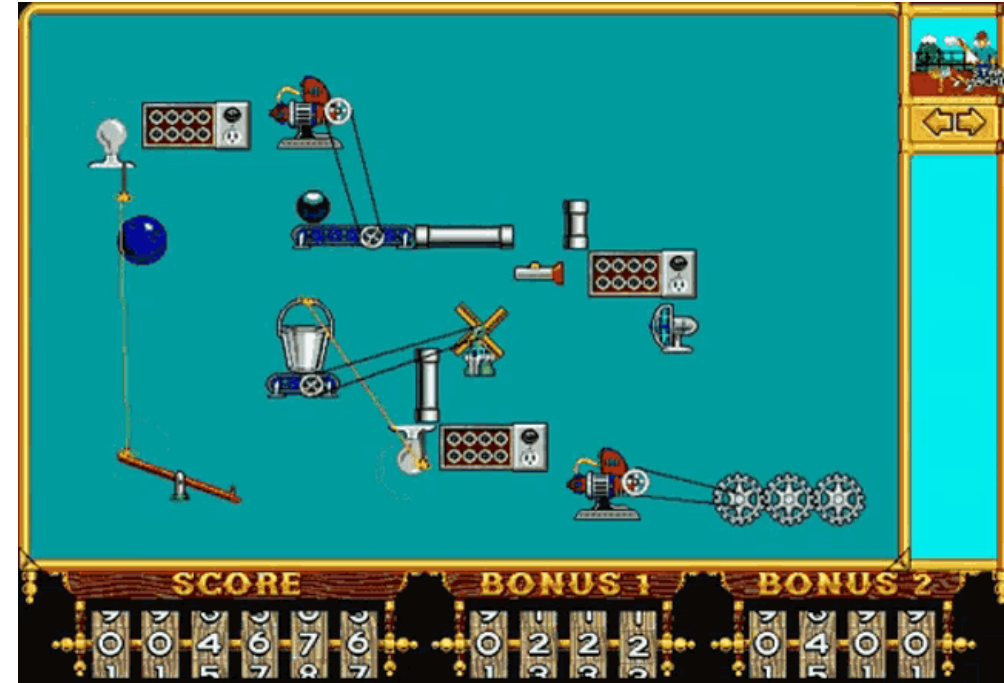


Agents: a lot more than function calling



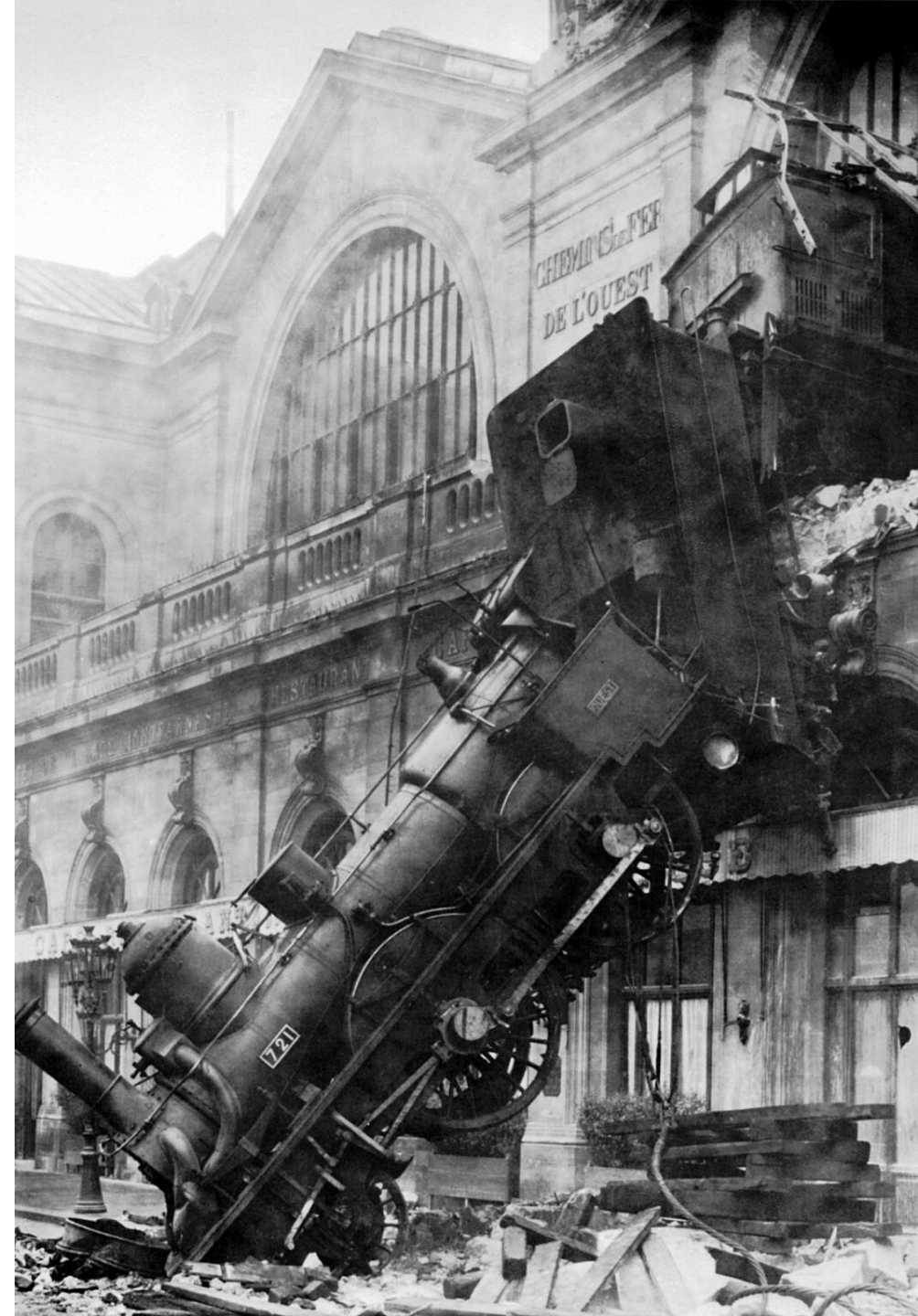
Inherent Complexity

- Infrastructure necessary to connect all those parts
- LLMs may be flaky
 - APIs may answer slowly (throttling, network issues)
 - May refuse requests
- Support for different models
 - specialized models for some tasks

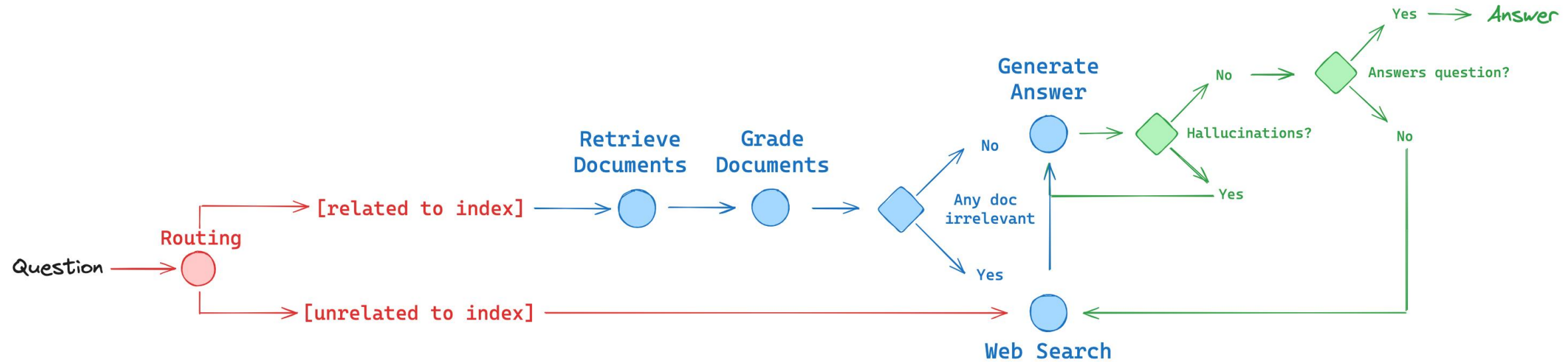


Accidental Complexity

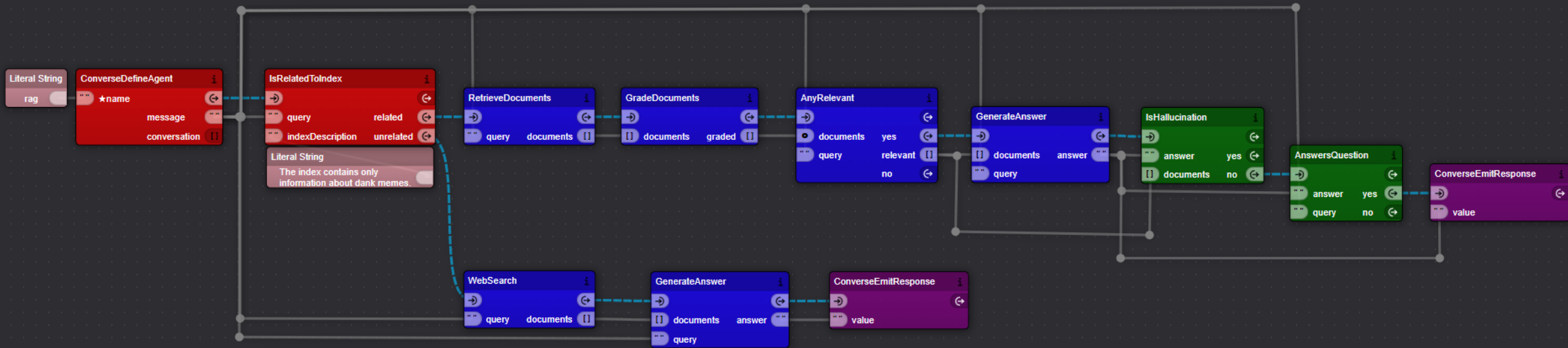
- Quick moving field
- Libraries that make for nice tutorials but hard to customize
 - many abstractions
 - steep learning curve beyond initial tutorials
 - Execution far removed from definition



Create a Diagram to deal with Complexity



Why not an executable Diagram?



Xircuits: Visual programming in JupyterLab

- Graphical programming approach to create executable diagrams
- Compiles to regular python
- Allows you to choose your own abstraction level
- Extendibility and modifiability are primary concerns
- Apache 2.0 License

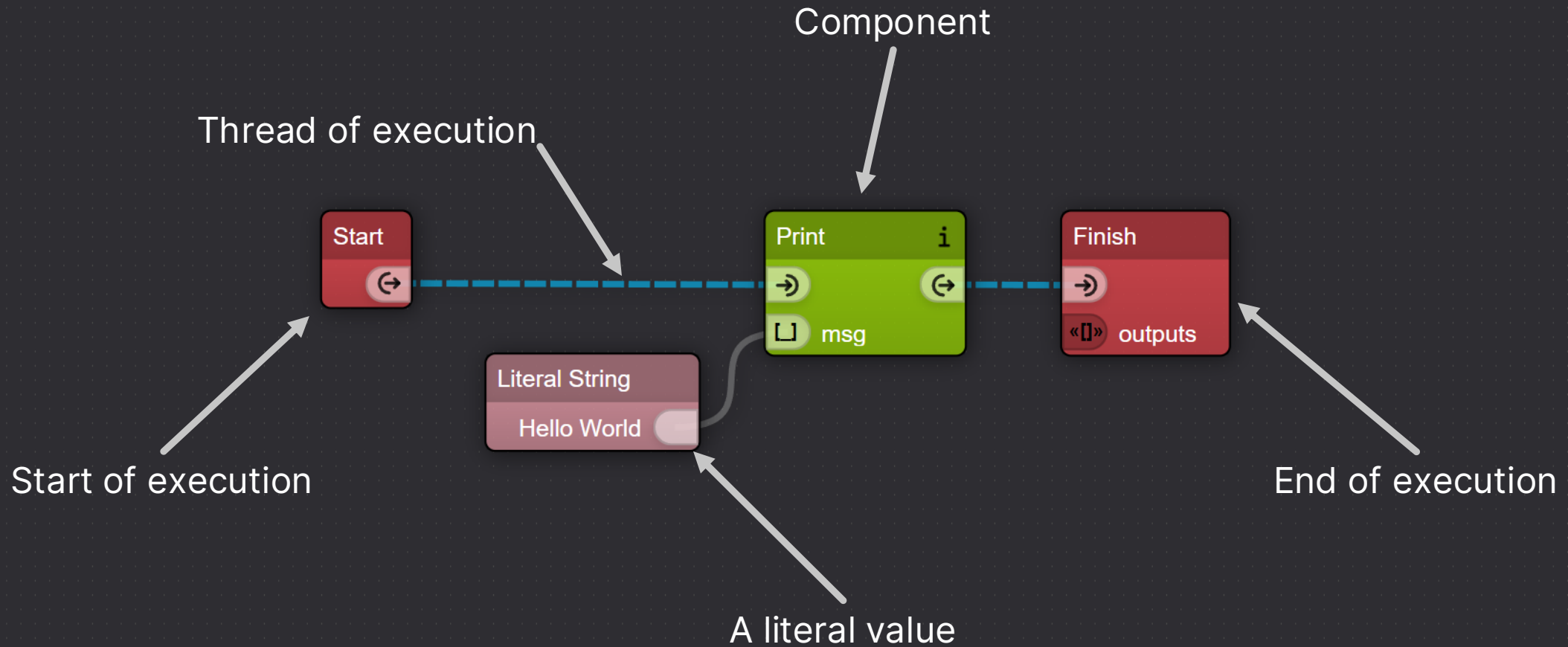


-
- The top screenshot shows the Unreal Engine Blueprints system. It features a 'Begin Overlap (Box)' event, a 'Get Player Pawn' node, a 'Cast To Character' node, a 'Branch' node with 'Condition' and 'True/False' paths, and a 'Launch Character' node. The 'Launch Character' node has a 'Target' input and 'Launch Velocity' (X: 0.0, Y: 0.0, Z: 0.0) inputs. The 'Cast To Character' node has an 'Object' input and 'Cast Failed' and 'As Character' outputs.
- The middle screenshot shows the Blender Material Editor. It displays a material graph with nodes for 'Image' (forstest_ground_03_diff_2k.png), 'Bright/Contrast', 'Hue Saturation Value', 'ColorRamp', 'Separate RGB', 'Bump', and 'Principled BSDF'. The 'Principled BSDF' node is configured with various parameters like 'GGX', 'Base Color', 'Subsurface', 'Metallic', 'Specular', 'Roughness', 'Anisotropic', 'Anisotropic Rotation', 'Sheen', 'Screen Tint', 'Clearcoat', 'Clearcoat Roughness', 'IOR', 'Transmission', 'Transmission Roughness', 'Emission', and 'Alpha'.
- The bottom screenshot shows the Davinci Resolve Fusion page. It displays a complex node graph for video compositing. Key nodes include 'Medialin3 (MI)', 'Medialin1 (MI)', 'DeltaKeyer1 (DK)', 'ImagePlane3D4 (3Im)', 'ImagePlane3D1 (3Im)', 'Displace3D1 (3Di)', 'Merge3D1 (3Mg)', 'ImagePlane3D2 (3Im)', 'Camera3D1 (3Cm)', 'Polygon1 (Ply)', 'BrightnessContrast1', 'ZDepthMatteGroup', and 'Renderer3D'. The graph shows a sequence of operations from input to final rendering.

Blender

Davinci Resolve

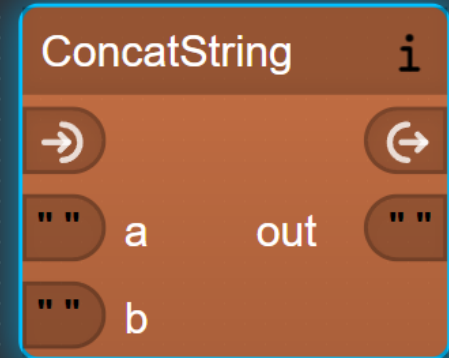
Hello World



Custom Components

```
@xai_component
class ConcatString(Component):
    a: InArg[str]
    b: InArg[str]
    out: OutArg[str]

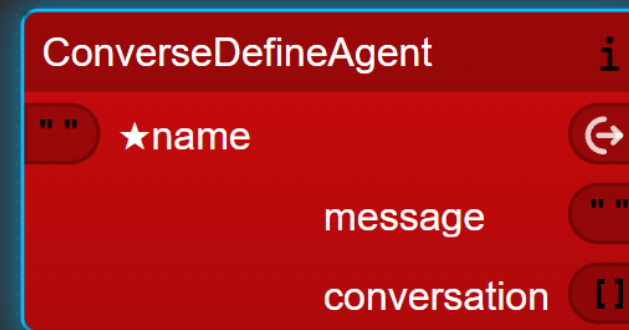
    def execute(self, ctx) -> None:
        self.out.value = "%s%s" % (self.a.value, self.b.value)
```



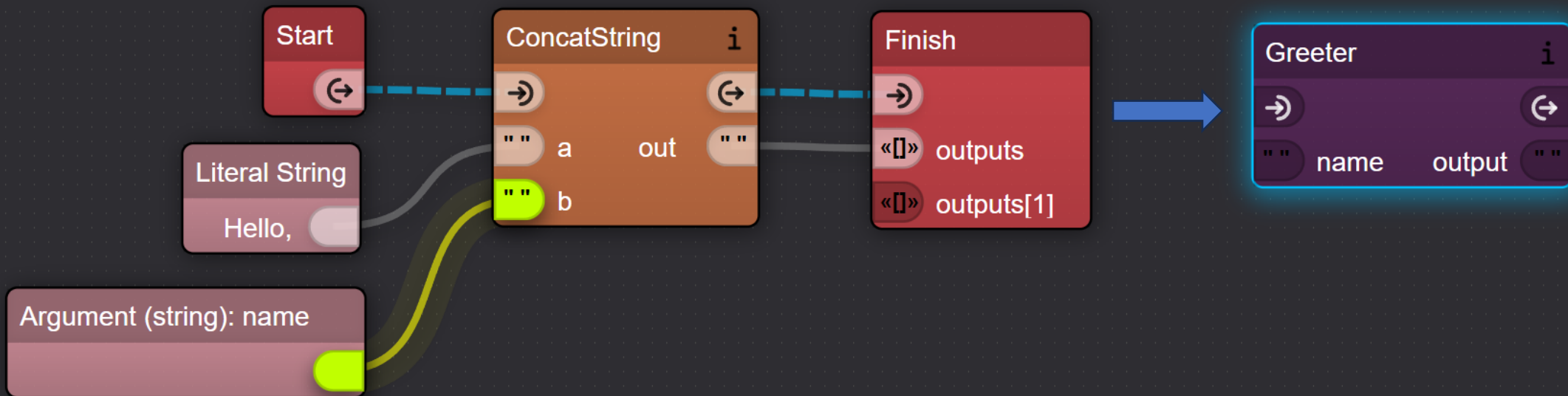
Advanced Components

```
@xai_component(type='Start', color='red')
class ConverseDefineAgent(Component):
    name: InCompArg[str]
    message: OutArg[str]
    conversation: OutArg[list]

    def init(self, ctx):
        ctx.setdefault(CONVERSE_AGENTS_KEY, {})[self.name.value] = self
```



No-Code Components



Batteries included

streamlit
mqtt
tensorflow discord
pycaret keras
rabbitmq multion
rpa anthropic
flask openai
hfagent agents
geminis sklearn
google gsread
xgboost mongodb gdrive
vectors stable diffusion
pytorch
actors slack spark

How to draw an owl

1.



2.

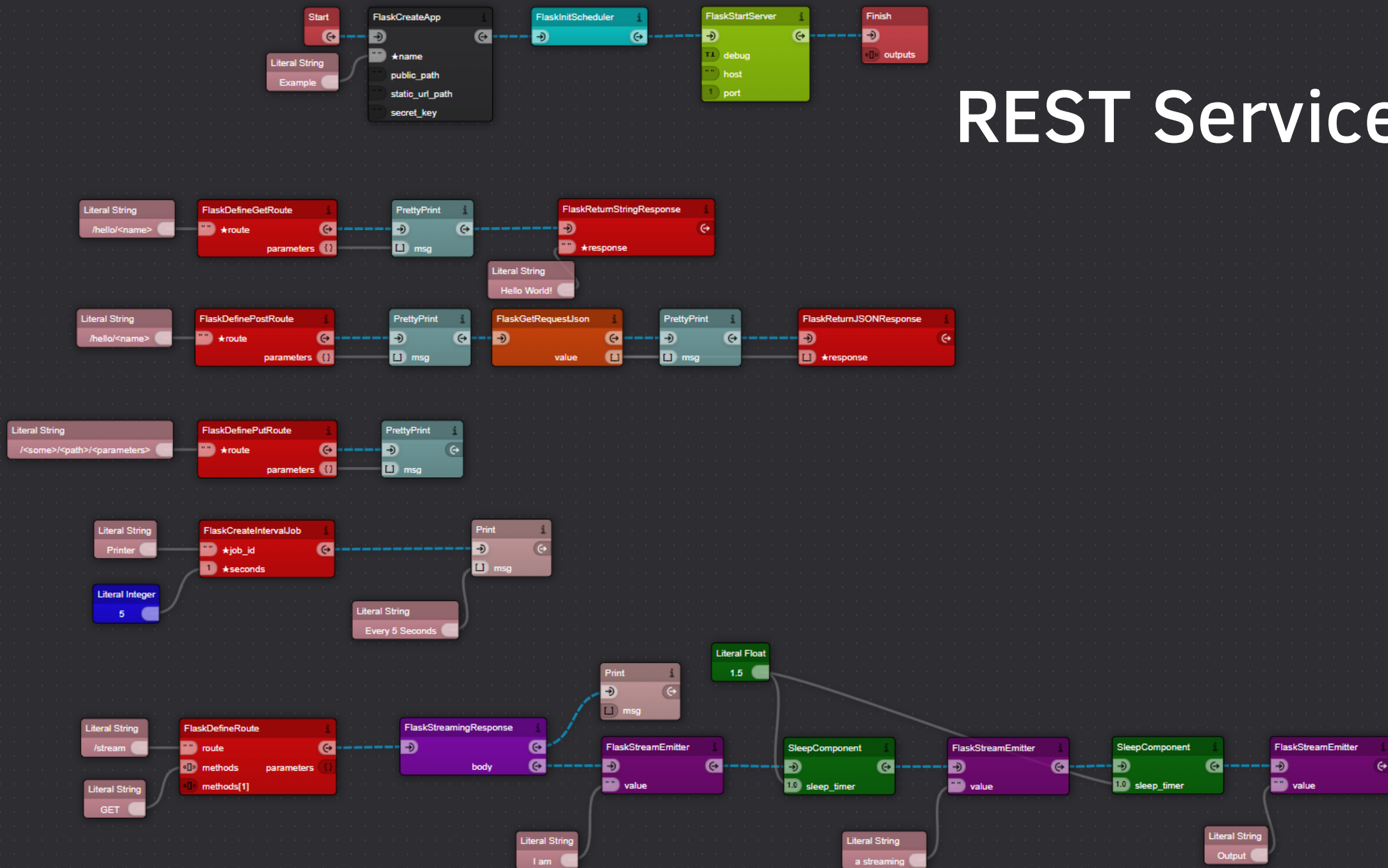


1. Draw some circles

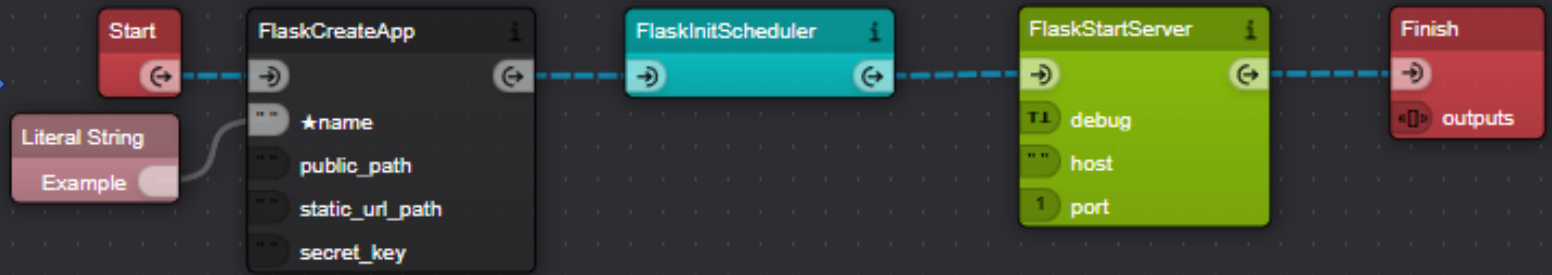
2. Draw the rest of the fine owl



REST Service

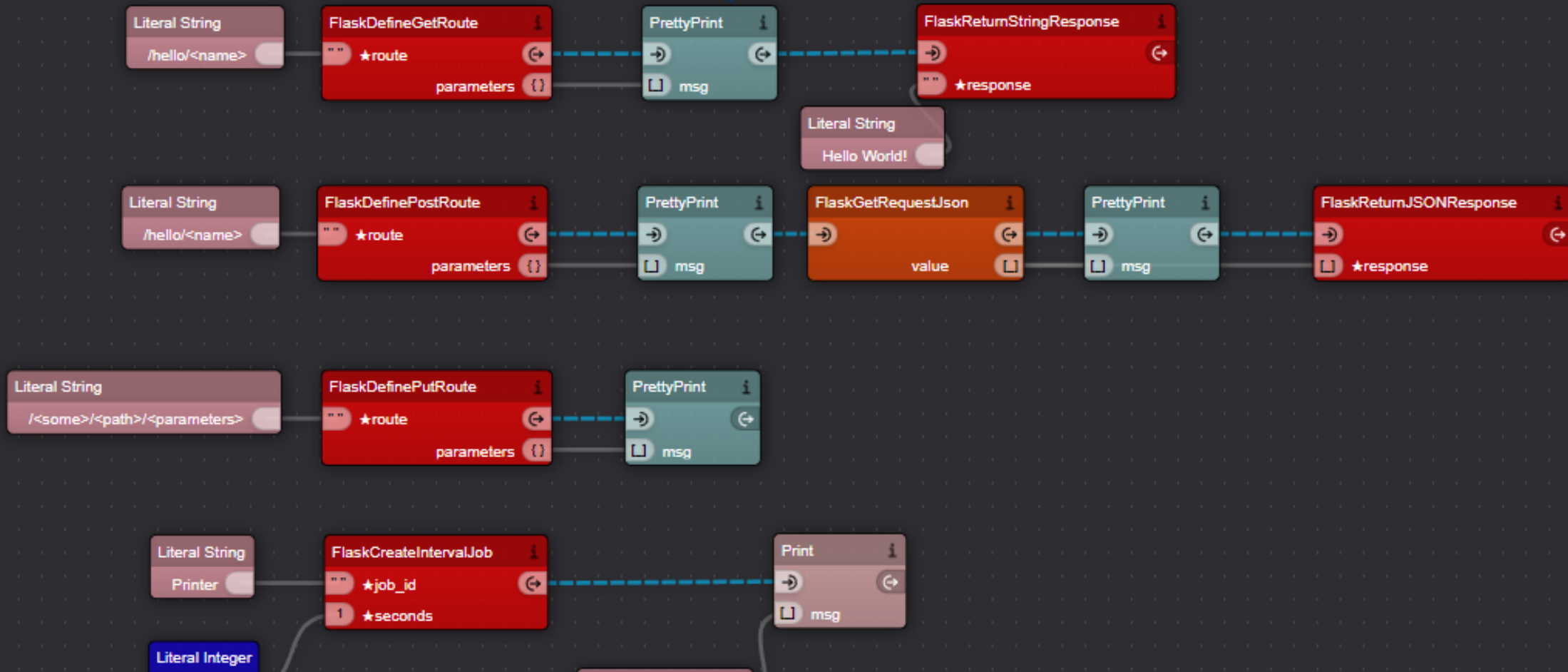


Create a server
on the main thread

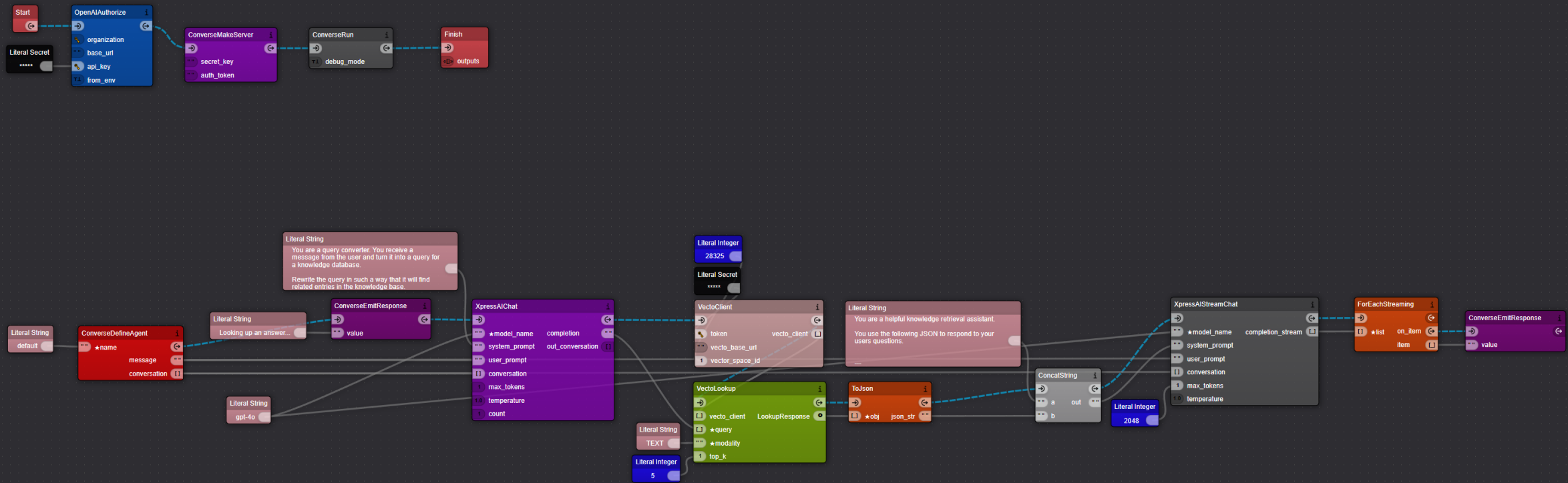


Define routes

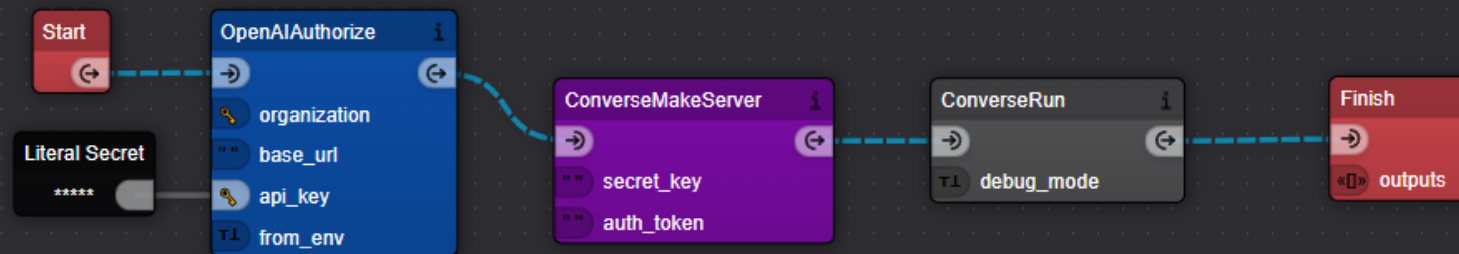
Define implementation



Retrieval Augmented Generation (RAG)



Create a server
on the main thread



Authorize to OpenAI

Tell the user you're working on it

Literal String

You are a query converter. You receive a message from the user and turn it into a query for a knowledge database.

Rewrite the query in such a way that it will find related entries in the knowledge base.

Literal String

Looking up an answer...

ConverseEmitResponse

value

ConverseDefineAgent

name

message

conversation

Literal String

default

Literal String

gpt-4o

XpressAIChat

model_name

completion

system_prompt

out_conversation

user_prompt

conversation

max_tokens

temperature

count

Rephrase the query for a lookup in the knowledge base

Literal Integer

28325

Literal Secret

VectoClient

token

vecto_base_url

vector_space_id

VectoLookup

vecto_client

query

modality

top_k

Literal String

TEXT

Literal Integer

5



Lookup documents from
the knowledge base

Create prompt for the
actual answer

query for
all find

XpressAIChat

- ★model_name completion
- system_prompt out_conversation
- user_prompt
- conversation
- max_tokens
- temperature
- count

Literal Integer
28325

Literal Secret

VectoClient

- token vecto_client
- vecto_base_url
- vector_space_id

Literal String

You are a helpful knowledge retrieval assistant.

You use the following JSON to respond to your users questions.

__

VectoLookup

- vecto_client LookupResponse
- ★query
- ★modality
- top_k

Literal String
TEXT

Literal Integer
5

ToJson

- ★obj json_str

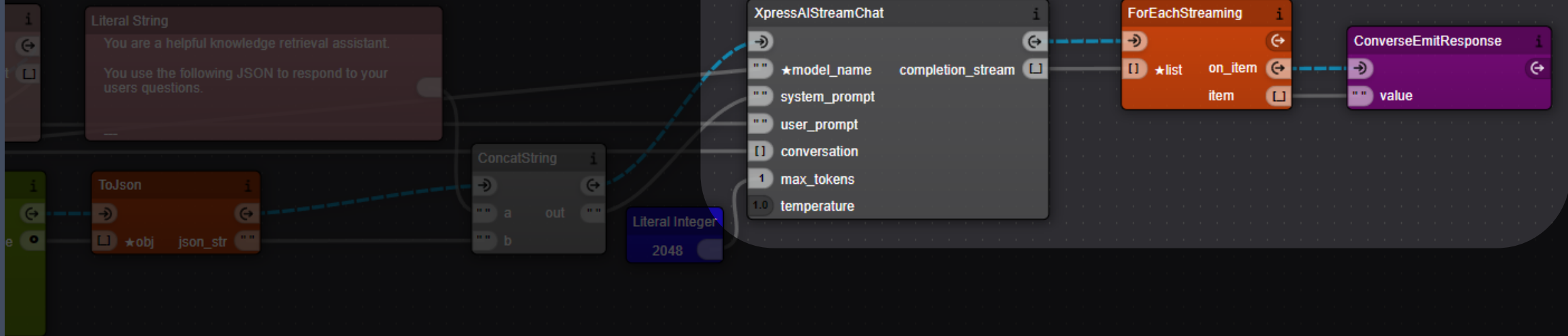
ConcatString

- a out
- b

Literal Integer
2048

XpressAIStre

- ★model_
- system_p
- user_prom
- conversat
- max_toke
- temperatu



Finally, stream out the answer





Demo: Agent in Action



Demo: Agent Implementation

Try it yourself

```
$ pip install xircuits
```

```
$ xircuits start
```

```
+ toml=0.10.2
+ toml=2.0.1
+ tornado=6.4.1
+ tqdm=4.66.4
+ traitlets=5.14.3
+ types-python-dateutil=2.9.0.20240316
+ typing-extensions=4.12.2
+ uri-template=1.3.0
+ urllib3=2.2.2
+ wcwidth=0.2.13
+ webcolors=24.6.0
+ webencodings=0.5.1
+ websocket-client=1.8.0
+ wrapt=1.16.0
+ xircuits=1.12.0
(mlcon2024)
pauld@Meshy MINGW64 /f/scratchPad/mlcon2024
$ xircuits start
```

```
>>>XIRCUIES
```

```
[I 2024-06-18 16:08:25.249 ServerApp] Extension pac
[I 2024-06-18 16:08:25.364 ServerApp] Extension pac
[I 2024-06-18 16:08:25.635 ServerApp] Extension pac
[I 2024-06-18 16:08:25.635 ServerApp] jupyter_lsp |
[I 2024-06-18 16:08:25.641 ServerApp] jupyter_serve
[I 2024-06-18 16:08:25.648 ServerApp] jupyterlab |
[I 2024-06-18 16:08:28.298 ServerApp] notebook_shim
[I 2024-06-18 16:08:28.299 ServerApp] xircuits | ex
[I 2024-06-18 16:08:28.404 ServerApp] notebook_shim
```

Questions?

Slides & Links to more materials:

<https://www.dubs.tech/mlcon2024>



SCAN ME

