Make Rusty AI

Learning AI architectures by implementing them

Jorge D. Ortiz-Fuentes

Canonical Examples

2026-03-04

A Canonical Examples production

Intro

Thanks!

BcnRust and the Rust Foundation for having me

Erik Sebastián de Erice for fixing some of my design mistakes

Agenda

- LLMs
- Chat REPL
- RAG
- Agents with tools
- Multi-agent

Goals

- Learn the basics behind AI applications
- Do it 100% in Rust (No Python here, move on)

No-goals

- Rust basics
- State of the art AI


LLMs

Complete with a word on 3 (1,2,3)

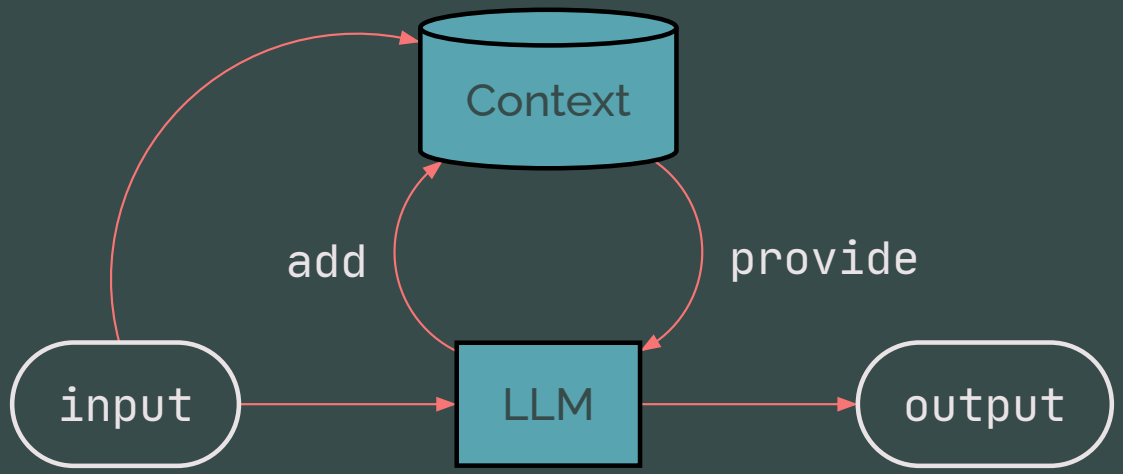
I love writing code in Rust because it is ...


Create your own crystal ball!

Let's code!

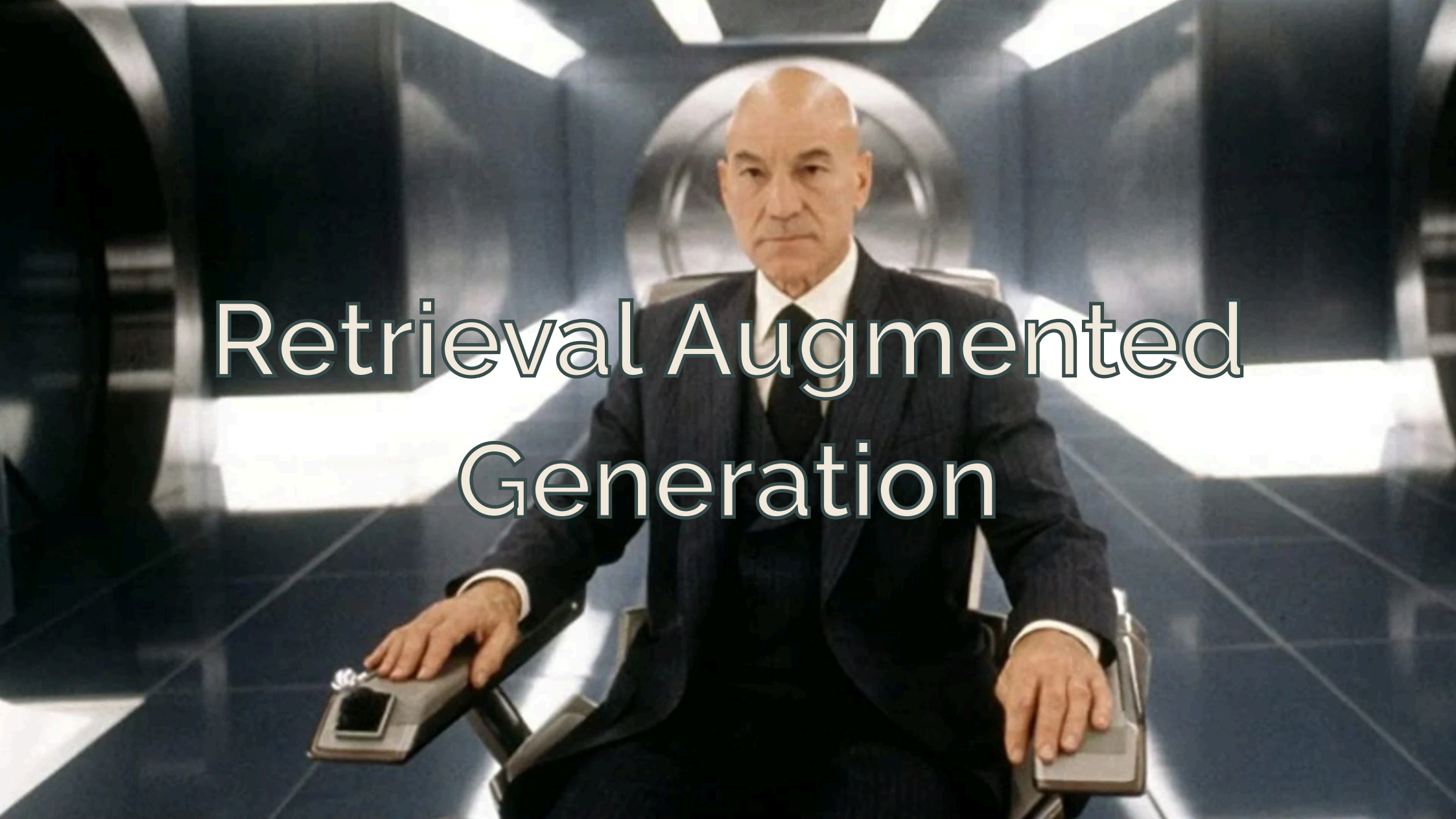

Chat REPL

Chat REPL



Context

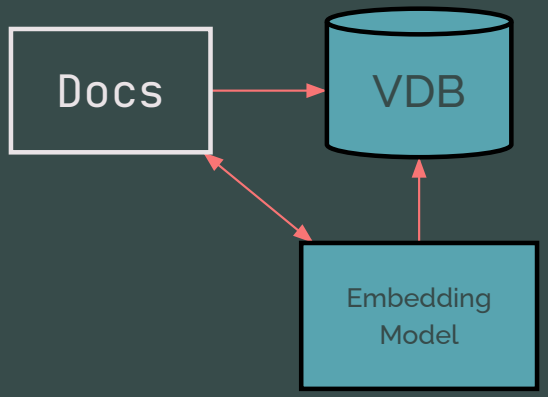
```
let mut messages = vec![
    ChatMessage::user().content("Hello. Call me Jorge.").build(),
    ChatMessage::assistant()
        .content("Hello, Jorge! How can I help you today.")
        .build(),
];

loop {
    match readline {
        Ok(line) => {
            messages.push(ChatMessage::user().content(line).build());
            chat_interaction(&llm, &mut messages).await;
        },
        _ => { } // Other cases
    }
}
```


Retrieval Augmented
Generation

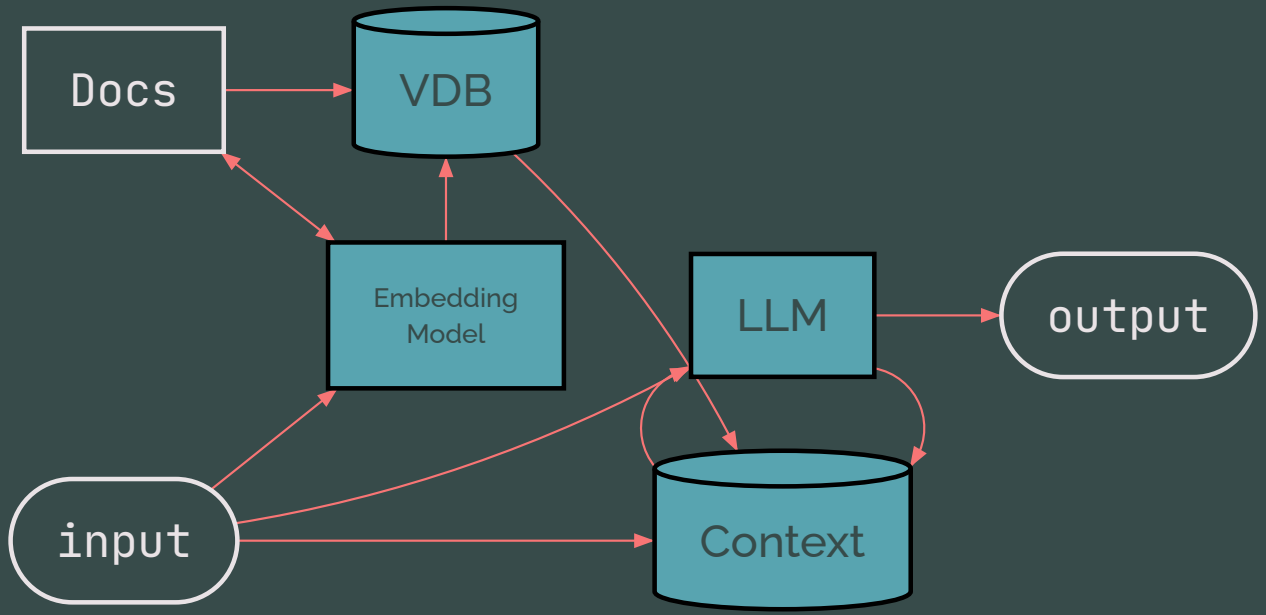
Retrieval Augmented Generation

- When: Newer or non-public information
- How:
  1. Use vector search with the prompt to retrieve related info.
  2. Use the new information together with the prompt.

RAG



RAG



RAG

```rust
let emb_client = client.embedding_model("nomic-embed-text");

let docs = fetch_notes();
let embeddings = EmbeddingsBuilder::new(emb_client.clone())
    .documents(docs)?
    .build()
    .await?;
let vector_store = InMemoryVectorStore::from_documents(embeddings);
let index = vector_store.index(emb_client);

let agent = client
    .agent("qwen3-coder:30b")
    .preamble(AGENT_RULES)
    .dynamic_context(1, index)
    .build();

let response = agent.prompt(line).await?;
```


Agents
WALL·E

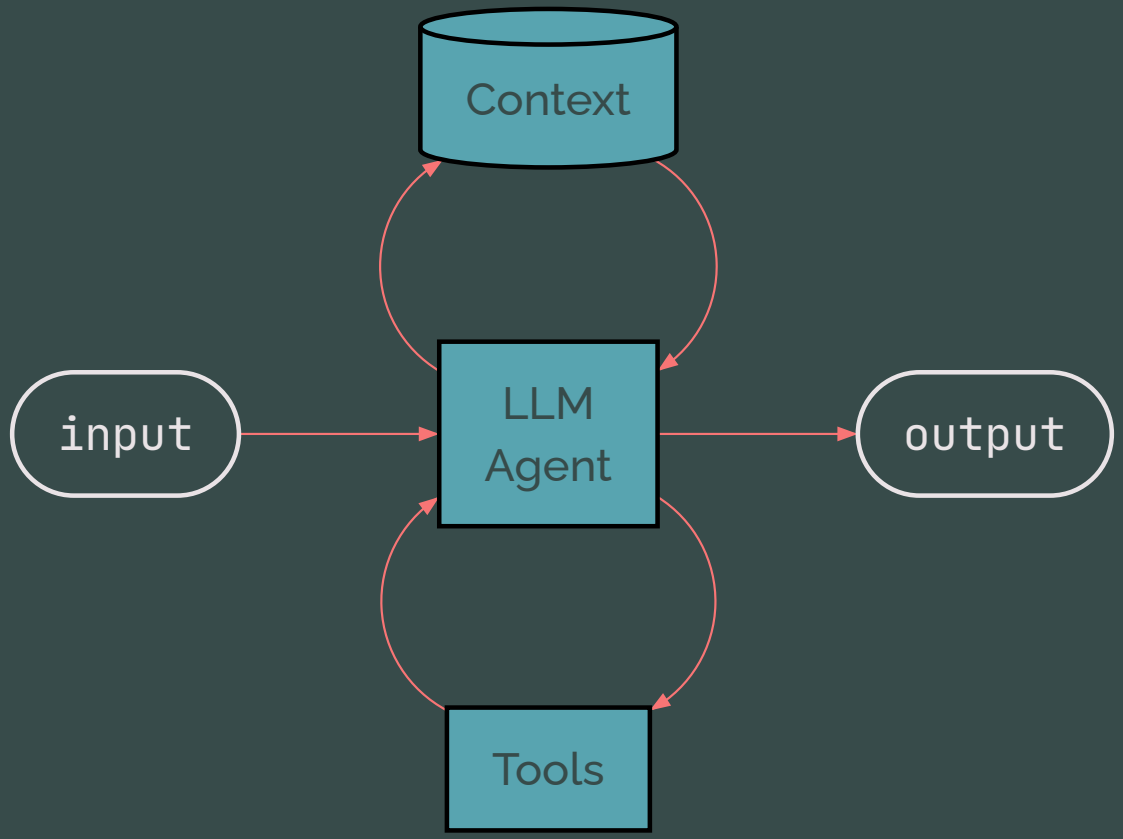
Agents

- Actually do the work

  Tools Register functions with description and structured I/O

  MCP Structured services that offer tools, resources, & prompts

Agent



Agent

```
loop {
    let compl_resp = agent.completion(&input, chat.to_owned())
        .await?.send().await?;
    if let AssistantContent::Text(rsp) = compl_resp.choice.first() {
        let response = rsp.text();
        let next_input: String = match parse_action(response) {
            Ok(act_req) => {
                let result = match act_req.tool_name.as_str() {
                    "list_files" => { list_files() }
                    "read_file" => {
                        act_req.args.get("file_name").map_or(json!
({"error": "No file_name."}), |f| read_file(f))
                    }
                    _ => { } // Other tools
                };
                if finish { agent_rsp = result.to_string(); }
                iter += 1;
                if iter >= MAX_ITER { bail!("Too long"); }
                result.to_string()
            }
```

```
                    Err(err) => { } // Deal with error
                };
                chat.push(Message::user(input));
                chat.push(Message::assistant(response));
                input = next_input;
                if finish { return Ok(agent_rsp); }
            }
        }
```


Multi-agents

Multi-agents

- Use several agents:
  - Different personas / speciallized
  - Different tools
  - Different responsibilities
- Orchestrate
  - A2A

Summary

- You don't need to be an ML expert to write AI apps.
- Architectures. Use

  Context + Loop memory

  RAG New/Additional information

  MCP/Tools AI takes actions

  Multi-agent Get the combined action of different **personas**

Thank You

Questions?

@jdortiz

github.com/jdortiz