Elixir Chatbot Alchemy

Develop a self-hosted chatbot with Elixir, Ollama, Ecto, and Phoenix LiveView

Isaak Tsalicoglou

Elixir Chatbot Alchemy

Develop a self-hosted chatbot with Elixir, Ollama, Ecto, and Phoenix LiveView

Isaak Tsalicoglou

This book is available at https://leanpub.com/elixir-chatbot-alchemy

This version was published on 2025-06-11

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Also By Isaak Tsalicoglou

Phoenix Product Codex
Lo and Behold, X! – Above, Beyond, and Far Away
Northwind Elixir Traders
Lo and Behold, X! – Bifurcation and Deliverance
Lo and Behold, X! – Doldrums and Machinations
Lo and Behold, X! – Growing, Pains, and Awareness
The Incredible Story of Deft (2nd ed.)
Lo and Behold, X! – Denizens of the Ivory Fortress
The Cave

Contents

Before we begin

Elixir Chatbot Alchemy

Develop a self-hosted chatbot with Elixir, Ollama, Ecto, and Phoenix LiveView

by **Isaak Tsalicoglou**, Managing Director of **OVERBRING Labs**, in Athens, Greece

This is version 0.7.3 of the book, dated 2025-06-11.

> **Please note:** this is a *super-early* draft of this book, launched silently and without fanfare to put it out there while I'm working primarily on **Phoenix Product Codex**. Thus, updates will be infrequent until August 2025, when I expect my workload to allow focused time for developing **Elixir Chatbot Alchemy**.
>
> Also note that I reserve the right to down-scope the requirements on the target application scope of the chatbot, in case I deem it as too convoluted to implement sensibly in a manner that conveys the knowledge I want to convey about the topic.
>
> In case you are a super-early adopter, you have my thanks! I look forward to your feedback and anything you might want to see covered or have questions about. Feel free to contact me!

* * *

For more information, contact the author on ElixirForum, on LinkedIn, or by email.

About this book

This is a super-early draft, so this hasn't been written yet!

Software versions used

- **Elixir:** version 1.18.3, installed using the asdf version manager.
- **Phoenix:** version 1.7.21 from Hex.pm.
- **Ecto:** version 3.12.5 (will be pulled automatically by Phoenix).
- **Ecto SQLite3:** version 0.19.0 (will be pulled automatically by Phoenix).
- **SQLite:** version 3.40.1 from the `sqlite3` package for Debian 12.

To my knowledge, nothing above is specific to Debian 12 or Linux, and you should be able to work through this book on any popular operating system and CPU architecture of your choice. Should you face any issues or should anything not work as expected, kindly contact me so that I can publish a new release with corrections.

License of the code in this book

- **Use the original project's logo or name** without permission.

However, you must:

- **Keep the license notice** in all copies of the code.
- **State any changes** you make if you modify the code.
- **Include a copy of the license** when redistributing.

Other aspects:

- Regarding the patent grants aspects of Apache-2.0: the book does not include patented technology.
- The Apache-2.0 license **does not** extend to the text of the book.

Typographic conventions

This book contains a lot of code and IEx output. It would have ended up way longer without the following conventions, established in order to better utilize the space on every page:

- Commands shown in IEx have been written less for readability, and more with the goal of enabling you to copy and paste them into IEx.
- The lines of output of commands in IEx have been wrapped, and are displayed differently in code blocks, compared to how they will appear on screen. This is especially true for output related to changesets.
- The formatting of the code has also been made more concise / horizontal, instead of pretty with vertical pipelines, and is neither everywhere the same as what `mix format` will generate, not "best practice".
- Non-relevant IEx output and unchanged lines of code are replaced by an ellipsis (**...**).
- When mentioning a row/record/schema/module, its name is mentioned interchangeably with its struct; e.g., `%Chat{}` or `Chat`.

Preface

This is a super-early draft, so this hasn't been written yet!

Chapter 1: Requirements, specifications, and deliberate decisions

The objective of this project is to **build a chatbot in Elixir** that uses the CPU for inference with open-weights models served by a self-hosted Ollama API. Here is a list of **must-have** requirements, as well as the rational behind it:

- The chatbot shall **be usable on any page of our web app**. Initially, we want to show the chatbot on our own Elixir web app. Ideally, users should be able to embed the chatbot on any website (see the nice-to-have requirements below).
- The chatbot shall **make it possible for the visitor of any given page of the web app to discuss the content of that specific page**. The idea behind the project is to allow a user to interrogate the page being viewed currently.
- The chatbot shall **have the necessary safeguards to prevent abuse as a general-purpose LLM**. We do not want visitors to use our self-hosted inference infrastructure to answer all kinds of questions, even those unrelated to the current page or all pages of our web app with textual content.
- The chatbot shall **helpfully decline to respond to questions that are unrelated to the contents of the web app's pages**.
- The chatbot shall **maintain the history of a conversation while the conversation is taking place**.
- The application shall **automatically generate a title for each conversation, and a list of tags that describe it**, after the conversation has ended.
- The application shall **provide an admin interface** for the operator of the web app.
- The admin interface shall **provide a list of recorded discussions with the chatbot**, and enable the operator to filter the list of conversations by page and by user, within a selected time period.
- The chatbot shall **generate its responses by utilizing one or more self-hosted LLMs**.
- The LLM utilized to generate responses shall **be capable of doing so at acceptable speeds of generating text**, i.e. at an adult's average reading speed or faster, **without relying on a GPU**.

We can also pose a list of **nice-to-have** requirements:

- The chatbot should **maintain the history of prior conversations** with a specific user regarding the currently visited page of the web app.
- The admin interface should **enable the operator to view reports** of the number of discussions per page and optionally per user or visitor, optionally within a selected time period.
- The admin interface should **enable the operator to view a summary of the discussions** per page, per user, and within a selected time period.
- The application should (if chosen by the user) **email the user with summaries and reports** of the chatbot's usage.
- The application should **automatically generate a title for each conversation**, as early as possible during the conversation, provided that there is enough context to allow the generation of a title that is accurately descriptive of the conversation.
- In the case that a question posed by the user to the chatbot does not pertain to the currently visited page of the web app, the application should **respond to the question based on a context that relates to the entire web page**.
- The chatbot should **respond in a language chosen by the user or inferred from the browser**.
- It should be possible to **embed the chatbot on any website**, beyond the Elixir web app.

We can now analyze the requirements into specifications. For some of those we have already made up our mind, and can reflect personal preference; for example:

- The application will be written in **Elixir** (obviously, otherwise what kind of alchemists would we be?)
- The web UI of the chatbot and the admin UI will be implemented with **Phoenix LiveView**.
- The styling of the web UI will be accomplished with the defaults of **Tailwind**, because we like sane defaults and this is not a book about hand-rolling CSS or using Bootstrap.

From our requirement related to utilizing self-hosted LLMs, we will be using **Ollama**. After you set it up, Ollama provides a REST API at

`http://localhost:11434` with endpoints that generate: a text completion, the next message of a chat (an existing list of prior messages), structured data (e.g. in JSON, which is useful for tagging), and embeddings (that we could use for RAG—TBD as we progress). There are currently two packages on hex.pm that we can use ollama with from within Elixir:

1. **ollamex:** this is a package that I published in early 2024.
2. **ollama:** this was first published 3 days after Ollamex, and has continued to grow both in scope and adoption since then.

Either package will work for our purpose. However, given the larger adoption of the second one, we'll go with that one.

Moreover, we can make deliberate decisions aiming to increase our speed and ease of implementation at this early stage, ideally without curtailing the optional implementation of our nice-to-have requirements or causing significant rework down the road. Those are the so-called "will" requirements:

- The application will **store data in an SQLite database** in a manner that does not prevent us from switching to a PostgreSQL database down the road.
- The LLM utilized will **have a number of parameters that makes it possible to run solely on the CPU** with the target acceptable speed of inference (or faster) while multiple conversations are taking place simultaneously across different pages or even website.
- The application will **use as few dependencies as possible**. This means that we'll keep Elixir dependencies to a minimum, and use Elixir's features as much as possible.

Chapter 2: Creating the Elixir application and making some decisions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Choosing an LLM suitable for CPU inference

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Taking the LLM for a spin with Elixir

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Understanding the Ollama API's response

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Chapter 3: Fundamental knowledge of LLMs, and a first chat

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Short conversations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Implementing a chat using `completion/2`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Context truncation, periodic summarization, and response caching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Failure modes of completion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Exceeding the context length

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Context compression through summarization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Chapter 4: GPU-based inference, parallel execution, and scaling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Summarizing the summaries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Sidequest: bigger models, better summaries (?)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Interrogating any text with context stuffing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

The `ask` function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Chunking the preamble

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/elixir-chatbot-alchemy.

Coming soon

This book is currently a work in progress at a very early stage, and new chapters are being written and released infrequently depending on my workload.

For a higher-level overview of what's conceptually in scope **but not yet set in stone**, please refer to **Chapter 1**, which is freely available in the sample of the book.

The following is a tentative list of topics that are currently within the concept of this book:

- Using `chat/2` for an actual chat.
- Using system prompts for the LLM.
- Phoenix LiveView for the ongoing chat on a page.
- Storing chats in SQLite.
- Using a GenServer for tracking chat state.
- Implementing an admin UI with reporting.

Have something you would like to see covered and explored? Found a typo? Love this book? Hate it? Write me at *isaak@overbring.com* or contact me on LinkedIn.

About the Author

Isaak Tsalicoglou is a relatively recent aficionado of the Elixir programming language, its ecosystem, and its developer community. He discovered Elixir in 2021 and took the plunge, going *all-in* on Elixir in late 2022 after a serendipitous discussion with a great friend about the joys of functional programming. Since then, he's been applying his growing Elixir skill-set in the development of Apache-2.0-licensed Elixir packages, open-sourced web apps such as Changelogrex for the exploration of Linux kernel Changelogs, and of proprietary web apps such as Pragmata, for the management of spare parts for industrial equipment, and Bellweather, for the caching, analysis, and serving of meteorological data.

Isaak's budding fascination with Elixir, Ecto, Phoenix and Phoenix LiveView followed hot on the heels of his two prior years of experience developing, deploying and improving web apps and REST APIs with Django, Django Ninja, FastAPI, and backends with NocoDB for small-business use cases, such as the management of product information, inventory data, and commercial registry data, the generation of pricing recommendations and commercial offers, the historical analysis of invoices, and the tracking of order progress using QR codes.

Before that, parallel to and sometimes supporting his managerial roles and entrepreneurial activities, Isaak developed proprietary MVPs and tools in Python, such as NLP on web-scraped employee reviews, clustering and regression of web-scraped used-vehicle data, Machine Learning applications on hardware testing datasets of engineering design processes, post-processing and display of radar data, object detection and tagging of photos, and blob detection of aggregates in concrete slab cross-sections.

Even earlier, as a development engineer and R&D project manager, Isaak programmed extensively in Python, building numerous software tools for engineering design, FEM/CFD/numerical simulation, post-processing and analysis of turbines, shafts and other components. He also adapted, modularized and wrapped in-house C++ and FORTRAN tools for embarrassingly parallel computation on HPC infrastructure. In this role, he became skilled at technical writing, compiling numerous manuals on newly developed engineering

design processes and technical reports on mechanical component design and verification.

Besides Northwind Elixir Traders and Phoenix Product Codex, Isaak has also written extensively (articles and books) about all aspects of organizations and corporate professionals turning their knowledge about markets, customers, and technology into a competitive advantage through the judicious use of technology, vigorous collaboration across functions and locations, and an entrepreneurial mindset… but also writes about how hype, dogmatism, agency issues, culture clash and misaligned incentives prevent organizations from doing so.

Isaak studied Mechanical Engineering at ETH Zürich and also holds an MBA degree with Honors from IMD in Lausanne, Switzerland.

Through OVERBRING Labs, his consultancy and bespoke software development and product incubation and fractional development/management company in Athens, Greece, Isaak helps ambitious organizations, both domestically and abroad, to improve knowledge-work operations and to grow their business through both in-house and customer-facing digital means, by providing his services as a fractional manager of business, technology, and development projects, and as an expert in problem-solving, risk-management and QA related to software products and product-focused ventures. As part of this activity, Isaak is also the co-host of The Puzzle Podcast, a series of 10-minute episodes on the wicked problems and pieces of the "puzzle" of product innovation, development, and marketing.

Beyond that, Isaak is the General Manager of Breek.gr, the digital command-center for real-estate property management operations in Greece, a SaaS that he is in charge of incubating as Head of Software Development Projects of the Tethys family office.