

# „DATENSCHUTZKONFORM MIT DEN EIGENEN DOKUMENTEN SPRECHEN“ SO SETZT MAN EIN RAG-SYSTEM AUF

---

## Übersicht

- » Einleitung
- » Was ist ein RAG?
- » Aufbau
- » Wieso ist diese Anwendung DSGVO-konform?
- » RAG aufsetzen: Schritt-für-Schritt-Anleitung
- » Ausblick



*This material is provided under  
the CC BY-NC-SA license.*

## Einleitung

Ein RAG-System (Retrieval Augmented Generation) macht Wissen, das oft über weite Ablagen verteilt ist, einfach zugänglich. Statt Handreichungen, Mails oder Protokolle zu durchsuchen, lassen sich organisationsinterne Daten damit gezielt einbinden und in natürlicher Sprache verfügbar machen. So können Mitarbeitende Fragen stellen und erhalten Antworten, die direkt aus dem Wissen der Organisation kommen. Das schafft Klarheit, spart Zeit und macht vorhandenes Wissen lebendig und nutzbar, ohne neue Strukturen oder Datensilos aufzubauen.

*Aber kostet sowas nicht richtig viel Geld und technisches Wissen? Und wie geht das überhaupt – vor allem wenn es DSGVO-konform sein muss?*

Mit dieser Handreichung möchten wir gemeinwohlorientierte Organisationen dabei unterstützen, ein solches RAG-System datenschutzkonform aufzubauen. Neben einer kurzen Einführung bietet sie auch eine Schritt-für-Schritt-Anleitung für den eigenen Einsatz.

Ein wenig technisches Know How wird nötig sein – aber nicht viel. Innerhalb von 1-2 Arbeitstagen sollte das System aufgesetzt sein. Die finanziellen Kosten liegen im beschriebenen Setup vom iac Berlin bei 50 € pro Monat. Dieser Wert kann allerdings nur als Orientierung dienen, da mehr Datenumfang oder andere Dienstleister zu anderen Kosten führen werden.

## Was ist ein RAG?

Ein RAG ist eine KI-Anwendung, bei der ein Sprachmodell (Large Language Model: LLM) mit einer Wissensquelle verknüpft wird – zum Beispiel einer speziellen Datensammlung einer Organisation.

Vor der Verknüpfung verfügt das Sprachmodell nur über das Wissen, mit dem es ursprünglich trainiert wurde, kennt aber noch keine spezifischen Informationen aus deiner Organisation. Durch die Anbindung an eine eigene (Vektor)Datenbank kann das Sprachmodell bei einer Anfrage dort nach relevanten Informationen suchen (Retrieval), mit dem eigenen Wissen verbinden (Augmented) und daraus dann eine Antwort generieren (Generation).

*Achtung: An dieser Stelle kommt die DSGVO in Spiel!*

Die Datensammlung, auf die das Sprachmodell durch das RAG zugreifen soll, sollte DSGVO-konform in der EU gehostet werden. Besonders bei einer Cloud-Lösung sollte man hier genau hinschauen. Auch das Sprachmodell sollte zur Verarbeitung der Anfrage nicht auf Rechenzentren in den USA zugreifen. Und die Verbindung zwischen Sprachmodell und Datensammlung sollte ebenfalls lokal passieren.

*Klingt kompliziert? Ist es nicht.*

Später erklären wir genau, was das heißt und wie es umgesetzt werden kann.

Hier ein einfaches Beispiel wie du dir ein RAG vorstellen kannst im Vergleich zu einem LLM wie ChatGPT, Gemini, oder Mistral:

### **>> Nur LLM ohne eigene Daten:**

Du fragst eine sehr belese Person, die das gesamte Internet kennt. Sie beantwortet deine Frage zu deiner Organisation aus öffentlich auffindbaren Quellen.

*Ergebnis:* Die Antworten können nützlich sein, sind aber abhängig von im Web verfügbaren Informationen, Zufallsfunden und beinhalten keine verlässlichen Belege aus deinem internen Wissen. Aufgrund dieser Unsicherheit besteht auch eine höhere Anfälligkeit für Halluzinationen – also Ungenauigkeiten oder schlicht erfundene Antworten.

### **>> LLM mit zur Verfügung gestellten Daten – z. B. via Copilot:**

Du gibst derselben Person ein paar deiner Daten in die Hand – wie auf ein paar Moderationskarten – und stellst dann die Frage. Sie bezieht genau diese Informationen zusätzlich in ihre Antwort ein.

*Ergebnis:* Die Antworten stützen sich stärker auf deine Inhalte, aber nur solange die Unterlagen Teil der aktuellen Konversation sind – man spricht hier von der Kontextfenster-Grenze, die keinen dauerhaften Speicher mitbringt. Zur Effizienz- und Kostenoptimierung werden Konversationen irgendwann neu gestartet. Und für jede neue Konversation musst du die relevanten Dateien wieder bereitstellen oder auf sie hinweisen.

### **>> RAG – also LLM mit Daten und Infrastruktur:**

Die imaginäre Person hat eine eigene, von dir kuratierte Mini-Bibliothek mit Dokumenten deiner Organisation. Bei jeder Frage sucht sie zuerst in diesen Regalen (Retrieval), bezieht sich bei der Erstellung der Antwort (Generation) nur darauf und liefert Quellenangaben gleich mit.

*Ergebnis:* Das Wissen in den Regalen ist einfach zu aktualisieren, die Antworten sind nachvollziehbar durch Zitate oder Links und damit weniger anfällig für Halluzinationen. Die Bibliothek muss allerdings aktuell gehalten werden. Wenn Dokumente veraltet oder in verschiedenen Versionen vorhanden sind, werden auch Antworten entsprechend inkorrekt. Diese administrative Aufgabe muss aktuell noch ein Mensch übernehmen.



This material is provided under the CC BY-NC-SA license.

## Aufbau

Für den datenschutzkonformen Aufbau braucht man generell folgende Bausteine:

**1.** Einen **Server**, der innerhalb der EU (idealerweise in Deutschland) steht.

Der Server sollte über mindestens 2 CPUs, 4 GB RAM und 50 GB Speicherplatz verfügen. (Stand: November 2025)

Es empfiehlt sich, einen Anbieter zu wählen, der AI-Hosting unterstützt, da solche Anbieter häufig bereits KI-Services wie LLMs bereitstellen. Dadurch entfällt ein Großteil der aufwändigen Einrichtung, die sonst manuell vorgenommen werden müsste.

*Tip:* Habt ihr eine Homepage? Dann fragt einfach mal beim Hosting-Service dieser Seite nach.

**2.** Ein **Web-Interface**, sprich eine **Chat-Anwendung**, die das gewählte LLM hosten und mit der gewählten (Vektor)Datenbank verbinden kann. Bitte behaltet im Kopf, dass Chat-Anwendung und LLM nicht das gleiche sind.

Auch hier bieten manche Server-Anbieter fertige Lösungen an, die sich leicht einrichten lassen und den technischen Aufwand stark verringern. Wenn die KI selber Text vektorisieren kann, vereinfacht das die folgende Arbeit enorm. Was das genau bedeutet, erklären wir später.

**3.** Ein **KI-Sprachmodell (LLM)**, das „Gehirn“ der Anwendung.

Es verarbeitet Anfragen, interpretiert deren Bedeutung und erzeugt daraufhin passende Antworten aus den Informationen in der Datenbank.

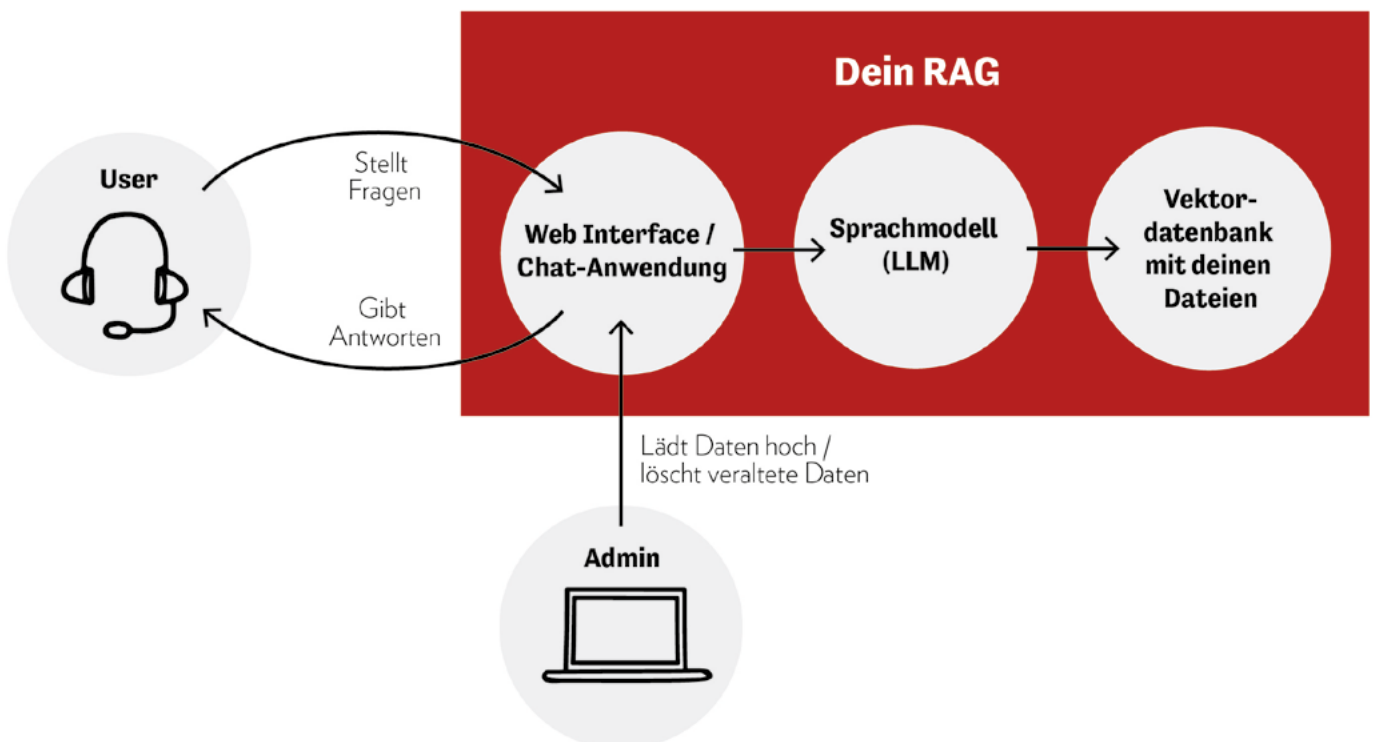
Im Kontext aktueller Souveränitätsdebatten empfiehlt es sich, ein EU-basiertes LLM wie Mistral zu nutzen. Außerdem gibt es unterschiedliche „Modell-Größen“. Ein schmales Modell wie Mistral Small ist ein eher verbrauchsarmes Sprachmodell, das für viele Zwecke vollkommen ausreicht und weniger Energie verbraucht als die ganz großen Modelle.

**4.** Eine **Datenbank** bestehend aus euren Daten – idealerweise in vektorisierter Form. Was das heißt, erklären wir noch.

Zusammengesetzt könnt ihr euch das in etwa so vorstellen, wie in der Grafik unten dargestellt.

Minimal komplexer wird die Logik aus Sicht der Administration. Es können kontinuierlich Daten zum Wissensstand des Systems hinzugefügt werden. Veraltete Daten sollten natürlich entfernt werden, da sonst die Antworten auch veraltet sind. Die administrative Pflege findet im Web Interface statt – siehe Grafik.

*Achtung:* Dies ist natürlich nur eine schematische Darstellung. Versichert euch am besten vorher, ob die KI die im Web Interface hochgeladenen Daten selber vektorisieren kann. Falls nicht, wird der Prozess etwas komplexer, da es dann eine sogenannte Embedding Pipeline braucht. Die Vektorisierung der Daten per se ist absolut zu empfehlen, da dies die Performance der Suche (Retrieval) deutlich schneller und effektiver macht. Was Vektoren sind, erklären wir noch genauer.



Wir haben uns für das Open Source Tool Open WebUI entschieden und damit sehr gute Erfahrungen gemacht. Es ermöglicht uns, Daten z. B. als DOC hochzuladen und übernimmt auch die Vektorisierung. Die fertigen Vektoren werden in unserem Setup in OpenSearch gespeichert, was sozusagen das Gedächtnis des RAG ist und schon im Angebot unseres Server-Providers enthalten war. OpenSearch ist ein System, das Daten wie in einem gut sortierten Nachschlagewerk speichert, damit man später schnell und gezielt darauf zugreifen kann.

## Wieso ist diese Anwendung DSGVO-konform?

Bevor wir hier ins Detail gehen, müssen wir kurz den Begriff „Container“ erklären, der folgend öfter auftauchen wird. Ein Container ist wie eine kleine, abgeschlossene Box zu verstehen, in der ein Programm läuft – zusammen mit allem, was es zum Funktionieren braucht. Dieser Container befindet sich auf eurem Server.

Was diese Anwendung DSGVO-konform macht, sind die bewusst gewählten Bausteine. Für das iac Berlin sieht dies folgendermaßen aus:

- 1.** Der Server, der die Bausteine und unsere hochgeladenen Daten enthält/hostet steht in Deutschland.  
» Es gibt keinen Datentransfer ins Ausland.
  - 2.** Die Chat-Anwendung ist eine Open-Source-Anwendung, die innerhalb des Servers in einem Container gehostet wird. Sie wird also nicht über eine API/eine Schnittstelle zu einem anderen System angesprochen.  
» Es gibt keinen Datentransfer nach außen.
- Auch Updates der Anwendung werden innerhalb des Containers ausgeführt. Das heißt, der Datenfluss geht nur in Richtung unseres Systems. Nicht andersrum.
- 3.** Das Sprachmodell wird im selben Container wie die Chat-Anwendung gehostet und nicht über eine API/Schnittstelle.  
» Es gibt keinen Datentransfer nach außen und alle Chats werden lokal gespeichert.
  - 4.** Die Datenbank wird in einem anderen Container innerhalb desselben Servers gehostet. Auch hier wird keine API/Schnittstelle zu einem externen Service genutzt.  
» Es gibt keinen Datentransfer nach außen.

Die komplette Anwendung unterliegt also unserer Kontrolle – mit Ausnahme des gemieteten Servers. Deshalb ist es zentral, dass dieser in der EU und idealerweise in Deutschland steht.

Wichtig ist es, dass ihr sowohl die Chat-Anwendung, als auch die Vektordatenbank mit einem starken Passwort schützt und dies nicht außerhalb der Organisation weitergebt. Für die User könnt ihr ein eigenes User-Konto für die Chat-Anwendung erstellen,

das separat vom Admin-Konto läuft. Dadurch ist die dahinter liegende Datenbank nicht für Alle zugänglich.

Trotz des sicheren Aufbaus könnt ihr zusätzliche Risiko-Minimierung betreiben, indem ihr keine unnötigen oder hochsensiblen Daten hochladet. Eine zusätzliche Orientierung über Risiken und Risikostufen bietet u.a. die Bundesnetzagentur ([https://www.bundesnetzagentur.de/DE/Fachthemen/Digitales/KI/2\\_Risiko/start\\_risiko.html](https://www.bundesnetzagentur.de/DE/Fachthemen/Digitales/KI/2_Risiko/start_risiko.html)).

## RAG aufsetzen: Schritt-für-Schritt-Anleitung

- 1.** Sobald ihr euch für einen Anbieter entschieden habt, mietet ihr einen passenden Server. Mindestanforderungen: 2 CPUs, 4 GB RAM und 50 GB verfügbarer Speicherplatz.
- 2.** Auf dem Server erstellt ihr einen ersten Container für das Sprachmodell. Bitte folgt hierfür den Anweisungen eures Server-Anbieters.

Wir haben uns für Mistral Small 3.2-24B Instruct entschieden. Mistral AI ist ein französisches Unternehmen, das seine KI-Sprachmodelle bereits unter EU-Datenschutzstandards trainiert und bereitstellt.

Wichtig sind bei der Erstellung vor allem die Umgebungsvariablen. Umgebungsvariablen sind Einstellungen, die man einem Programm während der Erstellung des Containers gibt. Sie sagen dem Programm unter anderem, wo sich die Datenbank befindet, welche User und welche Passwörter benutzt werden sollen, etc.

In unserem Fall wurde vom Server-Anbieter für das Sprachmodell direkt eine Chat-Anwendung angeboten, sodass wir beides in einem Container hosten konnten. Dies ist allerdings nicht Standard. Es kann also sein, dass ihr für die Chat-Anwendung einen zweiten Container erstellen müsst. Das ist aber keine technische Hürde und es gibt diverse Anleitungen dafür im Netz.

- 3.** Jetzt wird ein weiterer Container für die Datenbank erstellt. Auch hier ist es sinnvoll, sich zu informieren, welche Datenbanken vom Server-Anbieter unterstützt werden. Dann ist die Erstellung einfacher, da es auch hierfür wieder Dokumentationen gibt.

Wir haben uns für eine Vektordatenbank entschieden. Eine Vektordatenbank ist eine Datenbank, die Texte, Dokumente oder Bilder in Zahlen-Pakete umwandelt – sogenannte Vektoren. Diese Zahlen beschreiben die Bedeutung des Textes, d. h. die KI sucht nicht nach einzelnen Wörtern, sondern nach Bedeutung.

*Bsp.:* Ihr sucht nach dem Wort „Laptop“. Eine einfache Datenbank würde nach genau diesem Wort suchen. Eine Vektordatenbank versteht unter Laptop allerdings auch Notebook, Rechner, tragbarer Computer, etc. Dies kennt ihr bereits von Suchmaschinen im Internet. Wenn ihr dort nach etwas sucht, findet ihr es auch ohne den exakten Begriff einzugeben. Dies liegt an der gleichen Technologie: Vectoring.



*Achtung:* In unserem Fall vektorisiert die Chat-Anwendung Open WebUI mithilfe des Sprachmodells die Daten automatisch und speichert die vektorisierten Daten direkt in der Datenbank. Dies ist nicht immer der Fall, es kann also sein, dass ihr die Vektorisierung mithilfe eines weiteren Programms oder mit dem Programm der Vektordatenbank selbst einstellen müsst. Dafür braucht ihr dann die oben erwähnte Embedding Pipeline und solltet euch eine entsprechende Dokumentation im Internet ansehen.

**4.** Jetzt werden die Container miteinander verbunden. Bei uns ging dies über die erwähnten Umgebungsvariablen der Container. Keine Sorge, das klingt hier komplizierter als es in der Umsetzung tatsächlich ist.

**5.** Öffnet jetzt die Chat-Anwendung – also Open WebUI - im Browser und prüft ob:

- a. das Sprachmodell drauf läuft und
- b. hochgeladene Daten direkt vektorisiert in der Datenbank gespeichert werden. Falls sie das nicht sind, gibt es eine Fehlermeldung.

**6.** Wenn alles funktioniert, könnt ihr euren Wissensspeicher erstellen. In unserem Fall können Wissensspeicher wie z. B. „Onboarding“ oder „Richtlinien für Projekt-Administration“ innerhalb der Chat-Anwendung erstellt werden. Dort werden die Dokumente in Textform hochgeladen.

Beachtet bitte das Format der Daten. Am besten eignet sich das DOCX-Format, da PDFs nicht immer richtig gelesen werden. Allgemein werden Bilder nicht erkannt und dementsprechend auch nicht vektorisiert. Hier ist es sinnvoll, wichtige Informationen von Bildern noch einmal als Text hinzuzufügen, damit sie im RAG ansprechbar sind.

**7.** Zuletzt müsst ihr noch die einzelnen KI-Agenten erstellen. Wenn es beispielsweise darum geht, einen Onboarding-Assistenten für neue Mitarbeitende zu erstellen, können wir in unserer Chat-Anwendung einen neuen Ordner mit dem Namen „Onboarding“ erstellen und einen System-Prompt anlegen. Ein System-Prompt ist eine Regel, an die sich euer KI-Agent bei Beantwortung jeder Frage halten muss und die entsprechend Orientierung und Leitplanken gibt. Er kann z. B. so aussehen:

*„Du bist ein KI-Assistent für neue Mitarbeitende des iac Berlin. Hilf ihnen anhand der hochgeladenen Richtlinien bei allen Fragen, die sie möglicherweise haben. Überprüfe immer alle bereitgestellten Richtlinien aus dem Wissensspeicher, um die Antwort zu finden, und gib die Quelle der Antwort an.“*

*Achtung:* Schaut euch am besten auch hierzu online weitere Beispiele an, da solche System-Prompts teilweise bis zu mehreren Seiten lang sind. Der kurze System-Prompt oben dient lediglich als Orientierung. Entsprechende Anleitungen findet ihr über jede Suchmaschine.

Wähle abschließend den Wissensspeicher – z. B. „Onboarding“ – aus, damit der Agent darauf zugreifen kann. Ab diesem Moment gehen alle Anfragen an das Agentensystem „Onboarding“ direkt zum Wissensspeicher „Onboarding“ und werden aus dem dort hinterlegten Wissen beantwortet.

**8.** Legt so viele Wissensspeicher und damit verbundene KI-Agenten an, wie ihr benötigt. Behaltet allerdings die Server-Auslastung im Auge und bucht zur Not einen größeren Server.

*Wichtig:* In unserem Setup müssen die Agenten für jeden User-Account einmal mit Namen und System-Prompt erstellt werden. Wer also einen User-Account fürs ganze Team anlegen möchte, macht dies nur 1x. Wer für jede Abteilung einen eigenen User-Account wünscht, macht diesen Schritt mehrmals.

## Ausblick

Ab jetzt wünschen wir euch viel Spaß beim Aufsetzen, Ausprobieren und Einsetzen eures eigenen RAG!

Und ja – wir sind uns bewusst, dass manche der Schritte technisch und anspruchsvoll erscheinen. Aber wir wissen aus eigener Erfahrung, dass man sich schnell einarbeitet und alles leichter versteht, wenn man die einzelnen Teile vor sich sieht. Dann ist z. B. sehr viel klarer, wie man einen Container anlegt, was Umgebungsvariablen sind oder wie ein Systemprompt erstellt wird.

Am Ende des Projektes werdet ihr nicht nur ein eigenes RAG entwickelt haben, sondern auch wertvolles Wissen, das euch im Umgang mit KI und Daten sehr helfen wird.

*Wir haben unser RAG und diese Dokumentation mit Unterstützung von zukunfft zwei und N3XTCODER im Rahmen der Civic Coding Initiative erstellt. Wir bedanken uns sehr für die tolle Zusammenarbeit, von der nun unter der Creative Commons Lizenz CC BY-NC-SA auch viele andere gemeinwohlorientierte Organisationen profitieren können.*

## IMPRESSUM

### iac Berlin

International Alumni Center gGmbH  
Linienstraße 65a  
10119 Berlin

Tel. +49 (0) 30 288 85 80 0  
www.iac-berlin.org  
info@iac-berlin.org



This material is provided under the CC BY-NC-SA license.