immoGenio

No. 01
März 2026

KI

KI-Mieter-Chatbot mit Tenant-Isolation: Tool-Calling, Row-Level-Security und der DSGVO-Pfad für Konversations-History

Mieter-Chatbot auf Basis von Claude mit strikter Tenant-Isolation, Tool-Calling auf eigenen Stammdaten und 90-Tage-Retention für Konversations-History.

AUTOR
ImmoGenio

VERÖFFENTLICHT
23. März 2026

ONLINE
www.immogenio.de/blog

— AUF EINEN BLICK

Inhalt

12 Kontakt

Sonntagabend, 21:14 Uhr — und niemand erreichbar

Ein Mieter sitzt am Sonntagabend mit seiner gerade angekommenen Nebenkostenabrechnung am Küchentisch. Die Position „Heizkostenabrechnung" liegt 280 Euro über der Vorjahressumme, der Beleg dazu ist nicht beigelegt, der Verteilerschlüssel wird mit einem Kürzel benannt, das er nicht zuordnen kann. Er greift zum Telefon und ruft die Notfallnummer der Hausverwaltung an — die zu Recht für Schadensmeldungen reserviert ist, nicht für Abrechnungsfragen. Die Mailbox antwortet. Der Mieter formuliert eine E-Mail, die am Montagmorgen in einem Stapel von 47 weiteren ungelesenen Nachrichten landet, drei Tage später beantwortet wird und ihm in der Zwischenzeit ein nagendes Gefühl hinterlässt, dass mit seiner Abrechnung etwas nicht stimmt.

Dasselbe Szenario in einer Verwaltung mit einem produktiv eingebundenen Mieter-Chatbot sieht anders aus. Der Mieter loggt sich in sein Self-Service-Portal ein, öffnet den Chat und tippt die Frage. Innerhalb von 30 Sekunden erhält er eine Antwort, die auf seinen konkreten Vertragsdaten beruht — der Verteilerschlüssel ist eine beheizbare Wohnfläche, der Heizkostenanteil ist aufgrund eines kalten Winters und gestiegener Gaspreise plausibel höher, der Abrechnungszeitraum entspricht dem Mietvertrag. Der Mieter ist nicht erleichtert, weil er recht hat, sondern weil er die Antwort versteht. Genau diese Art der Antwort — datenbasiert, mieterindividuell, zu jeder Tageszeit verfügbar — ist die Aufgabe eines Mieter-Chatbots, der seinen Namen verdient.

Warum „ChatGPT mit Wissensbasis" für Hausverwaltungen nicht reicht

Die naive Vorstellung eines KI-Chatbots in der Hausverwaltung läuft auf eine simple Architektur hinaus: ein Sprachmodell, das auf eine Wissensbasis zugreift, die mit Mietverträgen, Hausordnungen und FAQ-Texten gefüllt ist. Diese Architektur funktioniert in einer Anwaltskanzlei, die für jeden Mandanten eine eigene Instanz betreibt. Sie versagt in einer Hausverwaltung an vier strukturellen Punkten.

Eine Verwaltung betreut nicht einen Mandanten, sondern Dutzende — und in der SaaS-Variante des Anbieters teilen sich Hunderte Verwaltungen dieselbe technische Plattform. Eine Wissensbasis, die mit den Daten aller Mandanten gefüttert wird, ist eine offene Flanke: ein einziger Bug im Filter, eine vergessene Bedingung in einer Vector-Suche, und der Mieter der Verwaltung A bekommt Antworten, die auf den Daten der Verwaltung B basieren. Die DSGVO bezeichnet das in Art. 5 Abs. 1 lit. f als Verletzung der Vertraulichkeit; in der Praxis ist es das Ende der Geschäftsbeziehung.

Die Stammdaten einer Verwaltung ändern sich täglich. Ein Mieter zieht aus, eine Schadensmeldung wechselt den Status, ein Hausmeister wird ausgetauscht. Eine wöchentlich neu indizierte Vector-Datenbank ist hier eine Halluzinationsmaschine. Wenn der Chatbot dem Mieter mitteilt, sein letzter Schaden sei „in Bearbeitung", obwohl er seit gestern abgeschlossen ist, hat das System eine Information geliefert, die schlechter ist als gar keine.

Ein Mieter-Chatbot soll nicht nur informieren, sondern Aktionen auslösen — eine Schadensmeldung anlegen, einen Rückruf vereinbaren, einen Hausmeister-Termin anfragen. Eine reine Frage-Antwort-Architektur ohne Schreib-Zugriff bleibt auf einer Service-Stufe stehen, die den Telefonanruf nicht ersetzt.

Schließlich verlangt die DSGVO eine Architektur, die jede Verarbeitung personenbezogener Daten belegen und löschen kann. Wer eine Wissensbasis mit Mieterdaten betreibt, hat ein Auskunfts-Problem nach Art. 15 DSGVO und ein Lösch-Problem nach Art. 17 DSGVO — denn Embeddings sind nicht ohne Weiteres mit dem Ursprungsdatensatz verknüpfbar. Eine Mieter-Frage nach „Alle meine Daten löschen" ist in einer Embedding-Pipeline ein technisches Großprojekt; in einer Architektur ohne Embeddings ist sie eine `DELETE`-Anweisung.

Tool-Calling statt RAG — die Architektur, die genau diese Probleme löst

Die ImmoGenio-Plattform verzichtet bewusst auf eine Retrieval-Augmented-Generation-Pipeline (RAG) mit Vector-Datenbank. Stattdessen kommt **Tool-Calling** zum Einsatz: Das Sprachmodell — produktiv Anthropic Claude Sonnet — bekommt vom Backend eine Liste von Funktionen mit definierter Signatur. Wenn der Mieter eine Frage stellt, entscheidet das Modell, welche dieser Funktionen es mit welchen Parametern aufrufen muss, um die Antwort zu liefern. Das Backend führt den Aufruf aus, gibt das strukturierte Ergebnis zurück, und das Modell formuliert die Antwort daraus.

Vier produktive Tools sind in der ersten Iteration freigeschaltet:

`getTenantContext` liefert die Stammdaten der Verwaltung — den Verwalter-Namen, die Sprechzeiten, die Notrufnummern, die hinterlegten Hausordnungs-Auszüge. Dieser Tool-Call ist die Grundlage jedes Gesprächs und liefert dem Sprachmodell den Rahmen, in dem es operieren darf.

`getMieterInfo` gibt die zum eingeloggten Mieter gehörenden Vertragsdaten zurück — Einheit, Mietbeginn, Kaltmiete, Nebenkostenvorauszahlung, zuständiger Hausmeister-Bereich, aktive Mietvertrags-Klauseln. Diese Daten kommen direkt aus der Verwaltungs-Datenbank — keine zwischengespeicherte Kopie, kein Embedding, kein Snapshot, der älter als eine Sekunde sein kann.

`getAnfrageStatus` liefert die laufenden und kürzlich abgeschlossenen Anfragen des Mieters mit aktuellem Status, letzter Aktivität und nächster geplanter Maßnahme. Wer wissen will, wann sein gemeldeter Wasserschaden bearbeitet wird, bekommt die Antwort, die auch der Sachbearbeiter im Backoffice sieht.

`createSchadensmeldung` ist das einzige Schreib-Tool der ersten Iteration. Der Mieter beschreibt einen Schaden im Chat, das Modell extrahiert die Pflichtfelder — Art des Schadens, Raum, Dringlichkeit, freie Beschreibung — und ruft die Backend-Funktion auf, die im Hintergrund eine vollwertige Anfrage in der Verwaltungs-Software anlegt. Diese Anfrage durchläuft denselben Status-Flow wie eine telefonisch oder per E-Mail eingegangene Schadensmeldung.

Alle Tool-Calls laufen mit der **Mieter-Session** — also mit denselben Rechten, die der Mieter im Self-Service-Portal hätte. Es gibt keinen privilegierten Service-Account, der „im Namen" des Mieters Daten abruft. Das Sprachmodell sieht keine Datenbank, kennt keine Tabellen, hat keine Vorstellung von SQL. Es sieht ausschließlich die strukturierten JSON-Antworten der vier Funktionen — und die sind durch die Permission-Logik des Backends bereits auf das eingegrenzt, was dieser eine Mieter sehen darf.

Der Unterschied zur RAG-Architektur ist fundamental: Es gibt keine Möglichkeit, dass das Sprachmodell „halluziniert", weil es keinen freien Lesezugriff auf einen Wissens-Korpus hat. Es muss eine Funktion aufrufen, um an Daten zu kommen. Wenn die gewünschte Information nicht über eine der vier Funktionen verfügbar ist, kann das Modell die Frage schlicht nicht beantworten — und gibt eine ehrliche Rückmeldung anstelle einer ausgedachten Antwort.

Tenant-Isolation durch Row-Level-Security

Tool-Calling schützt vor Halluzination, aber nicht vor der zweiten Klasse von Risiken: dem Daten-Crossover zwischen Verwaltungen. Ein Mieter der Verwaltung A darf unter keinen Umständen Daten von Mietern der Verwaltung B sehen — auch nicht, wenn er es durch geschickte Prompt-Formulierung versucht. Diese Garantie kommt nicht aus dem Sprachmodell, sondern aus der Datenbank.

Die ImmoGenio-Plattform betreibt alle Multi-Tenant-Tabellen mit `FORCE ROW LEVEL SECURITY` in PostgreSQL. Die Mieter-Session — etabliert beim Login ins Self-Service-Portal — enthält neben der Mieter-ID auch die `tenant_id` der zugehörigen Verwaltung. Jeder Request an das Backend, einschließlich aller Tool-Calls des Chatbots, läuft innerhalb einer Datenbank-Transaktion, in der dieser Tenant-Kontext per `SET LOCAL app.current_tenant` gesetzt wird. Die zugehörige `POLICY` auf jeder tenant-scoped Tabelle prüft bei jeder Lese- und Schreib-Operation, ob die Zeile zum aktuellen Tenant gehört.

Wenn ein Mieter im Chat schreibt „Zeige mir alle Mieter in meinem Haus" oder „Welche anderen Verwaltungen nutzt diese Software", passiert technisch Folgendes: Das Sprachmodell könnte versucht sein, eine Tool-Funktion zu suchen, die mehr Daten liefert als der angemeldete Mieter sehen darf. Eine solche Funktion existiert nicht. Selbst wenn das Modell die Funktion `getMieterInfo` zweimal aufrufen würde — die Datenbank-Policy gibt nur die Zeile zurück, deren `tenant_id` und `mieter_id` zur Session passen. Es gibt keine Tabellen-Scan-Funktion, keinen freien Query-Endpoint, keinen Pfad, über den ein Prompt zu einer Cross-Tenant-Abfrage führen könnte.

Dieselbe Isolation gilt für die Konversations-History selbst. Die Tabellen `chatbot_conversations` und `chatbot_messages` sind mit `FORCE ROW LEVEL SECURITY` versehen und an `tenant_id` plus `mieter_id` gebunden. Ein Mieter sieht ausschließlich seinen eigenen Chatverlauf — nicht den anderer Mieter derselben Verwaltung, und schon gar nicht den anderer Verwaltungen.

Die technische Tiefe dieser Architektur — `FORCE` versus `ENABLE`, das `BYPASSRLS`-Attribut des App-Users, die saubere Behandlung von Read-only-Pools — ist im Row-Level-Security-Beitrag ausführlich beschrieben. Für den Mieter-Chatbot ist die Konsequenz: Auch ein erfolgreicher Prompt-Injection-Angriff ändert nichts an der Sichtbarkeit der Daten. Der Angreifer kann das Sprachmodell vielleicht dazu bringen, in einem ungewöhnlichen Stil zu antworten — er kann es nicht dazu bringen, Daten zu sehen, die die Datenbank-Policy ihm verweigert.

Praxis-Workflow: vom Login bis zur Schadensmeldung

Ein konkreter Ablauf zeigt das Zusammenspiel der Komponenten. Der Mieter loggt sich am Sonntagabend ins Self-Service-Portal ein. Die Authentifizierung etabliert eine Session mit `mieter_id` und `tenant_id`. Er öffnet das eingebettete Chat-Widget und stellt seine erste Frage: „Wann kommt meine Nebenkostenabrechnung 2025?"

Das Sprachmodell empfängt die Frage zusammen mit dem System-Prompt, der die verfügbaren Tools beschreibt. Es entscheidet, dass `getMieterInfo` und `getAnfrageStatus` relevant sind, und ruft beide auf. Das Backend führt die Aufrufe im Tenant-Kontext aus, die Policy filtert auf den korrekten Mieter, das Modell erhält die strukturierten Ergebnisse. Aus den `getMieterInfo`-Daten kennt es den Abrechnungszeitraum des Mietverhältnisses, aus den `getAnfrageStatus`-Daten sieht es, ob bereits eine Anfrage zur Abrechnung vorliegt. Die Antwort an den Mieter nennt das voraussichtliche Versanddatum gemäß § 556 Abs. 3 BGB — spätestens zum Ende des zwölften Monats nach Ende des Abrechnungszeitraums — und verweist auf den konkreten Stichtag, der für diesen Mietvertrag gilt.

Der Mieter fragt nach: „Bei mir tropft seit gestern Wasser aus der Küchenarmatur, was soll ich tun?" Das Modell erkennt einen Schadensmeldungs-Intent und initiiert den Pfad zu `createSchadensmeldung`. Vor dem eigentlichen Schreib-Aufruf stellt das Modell die Rückfragen, die zum Anlegen einer Anfrage notwendig sind: Ist die Tropfquelle die Mischbatterie oder das Eckventil darunter, ist es ein langsames Tropfen oder ein deutlicher Strahl, kann das Wasser über Nacht aufgefangen werden. Die Antworten werden in den Tool-Aufruf gepackt, das Backend legt eine Anfrage mit Kategorie „Schaden", Priorität „normal" und freiem Beschreibungstext an. Der Mieter erhält die Bestätigung im Chat: „Ihre Schadensmeldung mit der Nummer 2026-0182 ist angelegt. Der zuständige Hausmeister-Bereich ist über die Eingabe informiert. Die geplante erste Sichtung erfolgt am nächsten Werktag bis 12 Uhr."

Am Montagmorgen sieht der Sachbearbeiter im Anfragen-Board die neu angelegte Anfrage neben den anderen 47 Eingängen — strukturiert, kategorisiert, mit voller Audit-Spur, die den Chatbot als Quelle nennt. Die Architektur der Anfragen-Pipeline ist im Anfrage-Management-Beitrag beschrieben — der Chatbot fügt sich als zusätzliche Quelle in dieselbe Pipeline ein, die auch E-Mail und Telefon bedient.

Konversations-History und 90-Tage-Retention

Jede Chat-Sitzung wird in zwei Tabellen persistiert: `chatbot_conversations` hält die Konversation als Ganzes mit Eröffnungszeitpunkt, Mieter-ID und Tenant-ID, `chatbot_messages` speichert jede einzelne Nachricht — sowohl die Eingaben des Mieters als auch die Antworten des Sprachmodells und die Zwischenergebnisse der Tool-Calls. Diese Persistenz ist notwendig, damit das Modell innerhalb einer Sitzung Kontext aufbauen kann („Wie war noch die Schadensmeldungs-Nummer?") und damit der Mieter beim nächsten Login die Konversation fortsetzen kann.

Die Retention ist bewusst kurz angesetzt: 90 Tage. Ein nächtlicher Cron-Job entfernt Konversationen und zugehörige Nachrichten, die älter als diese Frist sind. Diese Grenze ergibt sich aus Art. 5 Abs. 1 lit. e DSGVO, dem Grundsatz der Speicherbegrenzung — personenbezogene Daten dürfen nicht länger gespeichert werden, als es für den Zweck der Verarbeitung erforderlich ist. Der Zweck einer Chat-Sitzung ist die unmittelbare Beantwortung der Anfrage; nach 90 Tagen ist dieser Zweck regelmäßig erfüllt.

Diese kurze Retention schränkt die Beweisführungs-Möglichkeit ein. Wer drei Monate später wissen will, ob ein Mieter den Bot über eine bestimmte Information unterrichtet hatte, findet die Konversation nicht mehr. Diese Einschränkung ist gewollt — und sie ist die Antwort auf eine viel größere Frage: Die Aufbewahrungspflicht für die fachlich relevanten Informationen liegt nicht beim Chatprotokoll, sondern bei den daraus entstandenen Datensätzen. Eine über den Chatbot angelegte Schadensmeldung wird zu einer regulären Anfrage, die unter den allgemeinen Aufbewahrungsfristen — typischerweise sechs Jahre nach §

257 HGB für geschäftliche Korrespondenz, zehn Jahre für rechnungslegungsrelevante Vorgänge nach § 147 AO — gehalten wird. Die Anfrage zitiert in ihrem Audit-Trail den Chatbot als Quelle, behält aber alle inhaltlichen Informationen unabhängig von der ursprünglichen Konversation.

Die Konsequenz: Der Chat ist Dialog-Kanal, nicht Archiv. Was im Dialog entsteht und fachlich relevant ist, wird in den fachlichen Tabellen verewigt; was nur Dialog-Phatik ist, verschwindet nach 90 Tagen — und mit ihm das Datenschutz-Risiko einer überflüssigen Datenhaltung.

Compliance-Anker: DSGVO, Art. 22, SCCs

Der Mieter-Chatbot berührt mehrere Stellen der DSGVO, die nicht versehentlich übergangen werden dürfen.

Art. 5 DSGVO definiert die Grundprinzipien — Datenminimierung, Zweckbindung, Speicherbegrenzung. Die Architektur erfüllt diese Prinzipien strukturell: Tool-Calls liefern nur die Daten, die zur Beantwortung der konkreten Frage notwendig sind, und nicht mehr. Es gibt keinen vorausschauenden Daten-Pull „auf Vorrat". Die Retention von 90 Tagen für Konversationen setzt die Speicherbegrenzung um. Die Zweckbindung ist durch die Architektur klar: Daten aus Tool-Calls dienen der Beantwortung der konkreten Mieter-Frage und werden nicht in das Training eines Modells eingespeist.

Art. 22 DSGVO regelt automatisierte Einzelentscheidungen mit rechtlicher Wirkung. Hier ist eine klare Abgrenzung wichtig: Der Chatbot trifft **keine** rechtlich verbindliche Entscheidung. Er kündigt keine Verträge, beschließt keine Mietminderung, verweigert keine Erstattung. Er informiert den Mieter über bestehende Sachverhalte und legt strukturierte Anfragen an, die anschließend von einem menschlichen Sachbearbeiter bearbeitet werden. Diese Grenze ist nicht nur juristisch geboten, sondern architektonisch erzwungen — die vier produktiven Tools enthalten keinen Endpoint, der eine rechtsgestaltende Aktion ausführen könnte. Wer einen Bot baut, der eigenständig Mahnungen verschickt oder Verträge auflöst, fällt sofort in den Anwendungsbereich von Art. 22 DSGVO und braucht eine Rechtsgrundlage plus Widerspruchsrecht.

Art. 13 DSGVO verlangt eine Informationspflicht beim Erstkontakt. Die ImmoGenio-Implementierung blendet beim ersten Öffnen des Chats einen Hinweis ein, der die wesentlichen Punkte transparent macht: Verantwortlicher ist die jeweilige Verwaltung, technischer Dienstleister ist die Plattform mit Sprachmodell-Anbindung an Anthropic, Konversationen werden 90 Tage gespeichert, die Antworten basieren auf den Stammdaten der Verwaltung. Der Mieter bestätigt den Hinweis aktiv, bevor er die erste Nachricht senden kann.

Anthropic ist ein US-Unternehmen. Damit greift die Drittlands-Transfer-Regulierung der DSGVO. Die EU-US-Data-Boundary, die Anthropic seit 2024 anbietet, sorgt dafür, dass Anfragen aus europäischen Quellen in europäischen Rechenzentren verarbeitet werden — der Datenfluss verlässt die EU für die reine Inferenz nicht. Für die Vertragsbeziehung wird zusätzlich auf die Standardvertragsklauseln (SCCs) der EU-Kommission im Beschluss 2021/914 zurückgegriffen, die einen rechtlichen Rahmen für unvermeidbare Drittlands-Aspekte schaffen — etwa für Support-Zugriffe oder die organisatorische Vertragsbeziehung mit dem US-Konzern. Ein Auftragsverarbeitungsvertrag nach Art. 28 DSGVO mit Anthropic existiert und wird Verwaltungen auf Anforderung ausgehändigt.

Eine Datenschutz-Folgenabschätzung nach Art. 35 DSGVO ist für den Einsatz eines KI-Chatbots mit Zugriff auf Mieterdaten nicht zwingend, aber empfohlen — die Schwellenwerte aus den Listen der Aufsichtsbehörden werden in der jetzigen Architektur nicht erreicht, weil weder Profiling stattfindet noch besondere Datenkategorien nach Art. 9 DSGVO verarbeitet werden. Wer auf Nummer sicher gehen will, dokumentiert eine schlanke DSFA mit Risiko-Bewertung und Maßnahmen — die Plattform liefert die nötigen Bausteine.

Grenzen der ersten Version

Die produktive Version des Chatbots hat bewusste Grenzen, die in späteren Iterationen adressiert werden — aber heute nicht überschritten werden.

Kein proaktiver Outbound. Der Chatbot wartet auf eine Mieter-Eingabe und antwortet darauf. Er sendet keine ungefragten Nachrichten, keine Erinnerungen, keine Marketing-Hinweise. Diese Selbstbeschränkung schützt vor der Frage, auf welcher Rechtsgrundlage ein unbestellter Bot-Kontakt erfolgen dürfte.

Kein Sprach-Kanal. Wer telefonisch fragen will, landet beim Voicebot, der eine eigene Architektur mit eigenen Compliance-Anforderungen hat. Die beiden Systeme nutzen dieselben Backend-Tools, aber jeweils einen kanalspezifischen Frontend-Layer.

Nur Deutsch. Multi-Sprachen-Support steht auf der Roadmap, ist aber in v1 nicht enthalten. Mieter mit anderer Erstsprache erhalten den Hinweis, dass die Verwaltung schriftlich erreichbar ist und gerne eine Übersetzungshilfe organisiert.

Keine Bildverarbeitung. Der Mieter kann kein Foto seines Wasserschadens hochladen. Bildverarbeitung in einem mandantengetrennten Setup verlangt eine eigene Architektur für Storage, Virenscan und Lösch-Workflow und kommt in einer späteren Iteration. Für v1 beschreibt der Mieter den Schaden im Text — was für die strukturierte Anlage einer Anfrage in der Regel ausreicht.

FAQ

Welches Sprachmodell wird verwendet?

Der produktive Default ist Anthropic Claude Sonnet — gewählt wegen der nativen Tool-Calling-Unterstützung, der konsistenten Antwortqualität auf deutsche Eingaben und der EU-Data-Boundary, die seit 2024 für europäische Kunden bereitsteht. Die Architektur ist provider-agnostisch ausgelegt; ein Wechsel auf ein alternatives Modell ist ohne Anpassung der Tool-Definitionen möglich.

Werden meine Chat-Daten in den USA gespeichert?

Die Konversations-History selbst liegt ausschließlich in der ImmoGenio-Datenbank in einem deutschen Rechenzentrum. Für die Inferenz — also die Verarbeitung einer Nachricht durch das Sprachmodell — wird die EU-Data-Boundary von Anthropic genutzt; europäische Anfragen werden in europäischen Rechenzentren verarbeitet. Die organisatorische Vertragsbeziehung mit Anthropic ist über Standardvertragsklauseln (Beschluss 2021/914 der EU-Kommission) und einen Auftragsverarbeitungsvertrag nach Art. 28 DSGVO abgesichert.

Kann der Chatbot meine Daten an einen anderen Mieter weitergeben?

Nein. Die technische Garantie dafür liegt nicht im Sprachmodell, sondern in der Datenbank: PostgreSQL Row-Level Security mit `FORCE`-Policies stellt sicher, dass jeder Tool-Call nur Zeilen sieht, die zur authentifizierten Mieter-Session passen. Selbst ein erfolgreicher Prompt-Injection-Versuch ändert nichts an dieser Garantie, weil das Modell keinen freien Datenbank-Zugriff hat — es kann nur die vorgegebenen Funktionen aufrufen, und die geben gefilterte Ergebnisse zurück.

Was passiert, wenn der Chatbot die Antwort nicht weiß?

Das Modell ist instruiert, eine ehrliche Rückmeldung zu geben, wenn die Frage nicht über die verfügbaren Tools beantwortbar ist. Anstelle einer ausgedachten Antwort erhält der Mieter einen Hinweis, dass die Anfrage in das normale Anfragen-System weitergeleitet wird oder dass er die Verwaltung direkt kontaktieren sollte. Diese Designentscheidung — lieber „weiß ich nicht" als eine plausibel klingende Falschauskunft — ist die Konsequenz aus der Tool-Calling-Architektur.

Wie unterscheidet sich der Chatbot vom Voicebot?

Der Voicebot bedient den telefonischen Kanal und arbeitet mit einer engeren Intent-Liste, einer Bandansage zur Einwilligung nach Art. 13 DSGVO und einer kürzeren Transkript-Retention. Der Chatbot bedient den textlichen Kanal im authentifizierten Self-Service-Portal

mit voller Mieter-Identifikation über die Session. Beide Systeme nutzen dieselben Backend-Tools, was Konsistenz in den Antworten sicherstellt — ein telefonisch erfragter Anfragen-Status entspricht dem im Chat erfragten.

Können wir den Chatbot in unsere Website einbetten?

Die produktive Variante läuft im authentifizierten Mieter-Portal. Eine öffentliche Einbettung — etwa auf der Verwalter-Webseite — ist architektonisch nicht vorgesehen, weil ohne authentifizierte Session keine Mieter-Identifikation möglich ist und damit kein Tool-Call die für eine sinnvolle Antwort nötigen Daten liefern könnte. Wer einen anonymen Vor-Kontakt-Bot sucht, betreibt einen separaten Frontend-Bot ohne Datenbank-Anbindung.

Verwandte Beiträge

- KI-Telefonassistent in der Hausverwaltung: Was ein DSGVO-konformer Voicebot leisten darf
- Row-Level Security in der Hausverwaltungs-Software: Warum die Mandantentrennung in die Datenbank gehört
- Vom E-Mail-Posteingang zum Ticket-System: Email-Triage, Inline-Messaging und der Status-Flow
- Mieter-Selbstservice-Portal: Schadensmeldung online aufnehmen und nachverfolgen

Wo wir stehen

Der ImmoGenio-Mieter-Chatbot ist produktiv im Einsatz und nutzt Anthropic Claude Sonnet als Default-Sprachmodell mit EU-Data-Boundary. Die vier Tool-Funktionen — `getTenantContext`, `getMieterInfo`, `getAnfrageStatus`, `createSchadensmeldung` — sind in der zentralen Service-Schicht implementiert und laufen unter denselben RLS-Policies wie alle anderen Backend-Aufrufe. Das Chat-Widget ist in das Mieter-Portal integriert; die Konversations-History wird in `chatbot_conversations` und `chatbot_messages` mit 90-Tage-Retention persistiert.

Auf der Roadmap stehen Multi-Sprachen-Support (zunächst Englisch, Türkisch und Polnisch als die häufigsten Mieter-Sprachen im deutschen Markt), eine Bildverarbeitungsfunktion für Schadensfotos mit gesondertem Storage- und Virenscan-Workflow sowie zwei zusätzliche Tools — `getDokumente` für den Zugriff auf hinterlegte PDF-Auskünfte und `requestRueckruf` für die strukturierte Rückruf-Anforderung. Outbound-Nachrichten und ein anonymer Vor-Kontakt-Bot bleiben außerhalb des Scopes.

Kontakt

Wenn Sie als Verwalter erwägen, einen Mieter-Chatbot in Ihr Self-Service-Portal aufzunehmen, und die Fragen zur Tenant-Isolation, zur Tool-Architektur oder zum DSGVO-Pfad mit uns konkretisieren möchten, freuen wir uns über Ihre Nachricht an kontakt@immogenio.-de.