Property Witch

AI Property Assistant

Developer Manual v2.0

February 2026

A comprehensive AI-powered property search assistant for Portugal with RAG knowledge base, multi-step agent reasoning, persistent chat threads, and intelligent property analysis.

Table of Contents

1. System Overview

Property Witch is an AI-powered property search assistant specialized for the Portuguese real estate market. It combines natural language processing, retrieval-augmented generation (RAG), and multi-step agent reasoning to help users find and understand property listings.

Key Features

- Natural language property search ("find apartments in Porto under 200k")
- RAG-powered knowledge base with Portuguese real estate laws and regulations
- Construction land intelligence - distinguishes buildable vs non-buildable land
- Persistent chat threads with conversation memory
- Multi-step agent reasoning for complex queries
- Pick/select from previous results ("pick 2 closest to center")
- Real-time property listings from OLX Portugal API
- PDF report generation for selected properties

Tech Stack

- Backend: Node.js + Express + TypeScript
- AI: Groq API (llama-3.3-70b-versatile)
- RAG: Custom TF-IDF embeddings + in-memory vector store
- Frontend: React 18 + Vite + TypeScript
- Hosting: Render.com (backend), Hostinger (frontend)

2. Architecture

Directory Structure

```
/server                 # Backend
  /src
    /routes             # API endpoints
      chat.ts           # Main chat endpoint
      threads.ts        # Thread management
    /services
      aiService.ts      # AI integration
      agentService.ts   # Multi-step agent
      threadService.ts  # Chat thread storage
      searchService.ts  # OLX search
      /rag              # RAG system
        knowledgeBase.ts  # Knowledge documents
        ragService.ts     # RAG operations
        vectorStore.ts    # Embeddings
/web                    # Frontend
  /src
    App.tsx             # Main React app
```

Data Flow

1. User sends message via chat interface
2. Backend detects intent (search/conversation/pick/refine)
3. For searches: parse query, search OLX, filter with AI, return results
4. For questions: retrieve RAG context, generate AI response
5. Store message in thread for conversation continuity

3. AI Services

Supported Backends

- Groq Cloud (Primary): llama-3.3-70b-versatile - fast, reliable
- Ollama (Local): llama3.3-thinking-claude - offline capable
- Claude API (Optional): claude-sonnet-4 - highest quality

Intent Detection

The system automatically detects user intent from messages:

- "search" - New property search query
- "conversation" - General chat/questions
- "follow_up" - Question about previous results
- "refine_search" - Modify previous search (e.g., "cheaper")
- "show_listings" - Display results again
- "pick_from_results" - Select specific listings ("pick 2 best")

Query Parsing

Natural language queries are parsed to extract:

- Property type (apartment, house, land)
- Price range (under/over/around X)
- Location (city, region)
- Features (bedrooms, area, amenities)

4. RAG Knowledge System

Overview

RAG (Retrieval-Augmented Generation) enhances AI responses with curated knowledge about Portuguese real estate. The system uses TF-IDF embeddings to find relevant context.

Knowledge Categories

- Buying Process: NIF, contracts, notary, registration
- Taxes: IMT, stamp duty, annual property taxes
- Regions: Algarve, Lisbon, Porto, Alentejo, Silver Coast
- Property Types: Apartments, houses, land, ruins
- Construction Land: Urban vs rural, permits, PDM
- Visas: D7 passive income, Golden Visa changes
- Financing: Mortgages, bank accounts, requirements

Knowledge Document Structure

```
{ id: "buying-process", title: "...", content: "...",
  category: "buying-process", tags: ["buying", "nif"] }
```

RAG Context Building

For each user query, the system:

- Generates TF-IDF embedding for the query
- Searches vector store for similar knowledge documents
- Retrieves top 3-5 most relevant documents
- Includes context in AI prompt for accurate responses

5. Chat Threads & Memory

Thread System

Chat threads provide persistent conversation memory across sessions. Each thread stores messages, search context, and results for continuity.

Thread Structure

```
ChatThread {
  id: string              // Unique identifier
  title: string           // Auto-generated from first message
  messages: Message[]     // Conversation history
  lastSearchContext: str  // Summary of last search
  lastSearchResults: []   // Actual listing data
}
```

API Endpoints

- POST /api/threads - Create new thread
- GET /api/threads - List all threads
- GET /api/threads/:id - Get thread with messages
- DELETE /api/threads/:id - Delete thread

Memory Features

- Conversation history (last 20 messages) sent to AI
- Search results stored for "pick X" queries
- Context maintained across page refreshes
- Thread sidebar for easy switching

6. Construction Land Intelligence

Portuguese Land Law

Not all land in Portugal can be built on. The AI understands:

Terreno Urbano (Urban Land) - CAN BUILD

- Classified for construction in PDM (municipal plan)
- Keywords: "urbano", "lote", "construção", "viabilidade"
- Price: typically €30-300/sqm

Terreno Rústico (Rural Land) - CANNOT BUILD

- Agricultural land, building not allowed
- Keywords: "rústico", "agrícola", "rural"
- Price: typically €1-15/sqm

AI Filtering

When user searches for "land for construction":

- ACCEPTS: listings with "urbano", "lote de terreno"
- REJECTS: listings with "rústico", "agrícola"
- WARNS: ambiguous listings, suggests verification

Key Documents to Check

- Caderneta Predial - Property registry (urbano vs rústico)
- PDM - Municipal zoning plan
- PIP - Pre-approval request (recommended before buying)

7. Agent System

Multi-Step Reasoning

The agent handles complex queries that require multiple steps:

- "Compare apartments in Porto vs Lisbon under 150k"
- "Find the cheapest land with sea view near Algarve"
- "What are the total costs for buying a 200k property?"

Agent Tools

- search_listings - Search OLX for properties
- get_knowledge - Retrieve RAG documents
- calculate - Perform calculations (taxes, costs)
- compare - Compare multiple properties/options

Agent Flow

1. Parse user query to understand goal
2. Plan sequence of tool calls
3. Execute tools, collecting results
4. Synthesize final answer from all data
5. Return comprehensive response

8. OLX API Integration

API Endpoint

```
https://www.olx.pt/api/v1/offers/?category_id=16
```

Search Parameters

- category_id=16 - All real estate
- filter_float_price:from/to - Price range
- query - Search text
- limit - Results per page (max 50)

Property Mapping

OLX responses are mapped to unified Listing format:

```
Listing { id, title, priceEur, locationLabel,
  beds, baths, areaSqm, photos, sourceUrl }
```

Filtering Strategy

1. Fetch broad results from OLX (category 16)
2. Apply price range filter from parsed query
3. Use AI to analyze relevance of each listing
4. Sort by relevance score
5. Return top matches with reasoning

9. Frontend (React)

Key Components

- App.tsx - Main application, state management
- Chat panel - Message display, input, auto-scroll
- Results grid - Property cards with images
- Thread sidebar - Conversation history
- Quick Look modal - Property details overlay

State Management

```
messages[]      - Chat messages
threads[]       - Available threads
currentThreadId - Active thread
searchResponse  - Current search results
isLoading       - Request in progress
```

Auto-Scroll

Chat automatically scrolls to show new messages:

```
const messagesEndRef = useRef<HTMLDivElement>(null);
useEffect(() => {
  messagesEndRef.current?.scrollIntoView({ behavior: "smooth" });
}, [messages]);
```

Build & Deploy

```
npm run build                          # Build
sshpass scp dist/* user@host:public_html/  # Deploy
```

10. Deployment

Backend - Render.com

- Auto-deploys from GitHub main branch
- Build: npm run build (esbuild)
- Start: node dist/server.js
- Environment: GROQ_API_KEY, PORT

Frontend - Hostinger

- SSH upload to public_html
- Static file hosting

```
sshpass -p "password" scp -P 65002 -r dist/*
  user@147.93.73.224:~/domains/site/public_html/
```

Environment Variables

```
GROQ_API_KEY    - Groq API key
PORT            - Server port (default 3001)
OLLAMA_URL      - Local Ollama URL (optional)
ANTHROPIC_API_KEY - Claude API key (optional)
```

11. API Reference

POST /api/chat

Main chat endpoint for all interactions

```
Request: { message, threadId?, userLocation, mode }
```

```
Response: { type, message, searchResult?, threadId }
```

GET /api/threads

List all chat threads

```
Response: { threads: ChatThread[] }
```

POST /api/threads

Create new thread

```
Response: { id, title, messages }
```

GET /api/ai/health

Check AI backend status

```
Response: { available, backend, model }
```

GET /api/rag/stats

Get RAG system statistics

```
Response: { knowledge: 20, listings: 8447, ... }
```

12. Troubleshooting

AI Not Responding

- Check GROQ_API_KEY environment variable
- Verify API endpoint: GET /api/ai/health
- Check Render logs for errors

No Search Results

- OLX API may be rate limited - wait and retry
- Check if category_id=16 is still valid
- Verify location/price filters are reasonable

Thread Memory Lost

- Threads stored in-memory (lost on restart)
- For production: implement database storage

"Pick X" Not Working

- Ensure threadId is sent with each request
- Search results must be stored in thread first
- Check console for "pick_from_results" intent

Construction Land Misidentified

- AI uses keywords: urbano, rústico, construção
- Price per sqm helps: urban > €30, rural < €15
- Advise users to verify with Caderneta Predial