Contents

Property Witch Developer Manual v3.1

AI-Powered Portuguese Real Estate Assistant

Version 3.1 | February 2025 | Property Witch Team

Table of Contents

---

1. Overview

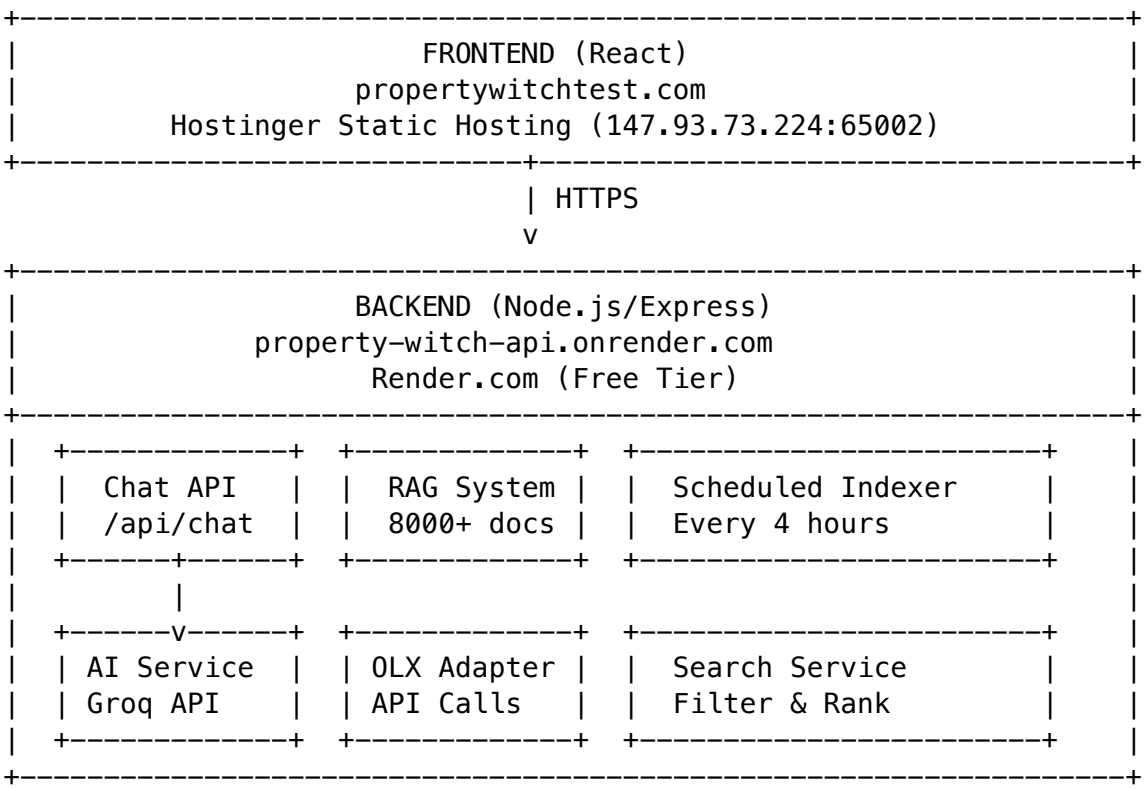
Property Witch is an AI-powered real estate search assistant focused on Portuguese properties. It uses natural language processing to understand user queries, fetches listings from OLX Portugal, and provides intelligent analysis of each property.

Key Capabilities

Feature	Description
Natural Language Search	“Find me a 2-bedroom apartment in Lisbon under 200k”
AI Intent Detection	Understands context, follow-ups, and refinements
Vision AI	Analyzes photos to detect pools, sea views, ruins, etc **NEW**
Smart Analysis	Detailed property analysis for <=10 results, brief for >10
Scheduled Indexing	Auto-indexes 8,000+ listings every 4 hours
Portugal Expertise	Understands “urbano” vs “rustico” land classifications

---

2. Architecture



Tech Stack

Layer	Technology
Frontend	React 18 + TypeScript + Vite
Backend	Node.js + Express + TypeScript
AI Provider	Groq API (llama-3.3-70b-versatile)
Data Source	OLX Portugal API
Hosting (Frontend)	Hostinger
Hosting (Backend)	Render.com

---

3. Features

3.1 AI-Powered Intent Detection

The system uses AI to understand user intent, not hardcoded regex patterns:

```
// User says: "narrow down within 60m2 range"
// AI detects:
{
  intent: "pick_from_results",
  extractedFilters: { area: 60 },
  selectionCriteria: "filter by area closest to 60m2"
}
```

Supported Intents:

- `search` - New property search
- `refine_search` - Modify previous search
- `pick_from_results` - Select from current results
- `conversation` - General chat/questions
- `follow_up` - Questions about shown listings

3.2 Adaptive Analysis Depth

Results Count	Analysis Type	Description
<= 10 listings	**Detailed**	4-6 sentences covering price, location, features, pros/cons
> 10 listings	**Brief**	2-3 sentences with key match info

3.3 Portuguese Land Classification

Critical for construction queries:

- **Urbano** (Urban) - Can build
- **Rustico** (Rural) - Cannot build (agricultural only)

The AI automatically detects and warns about land types.

3.4 Auto-Expanding Input

The chat textarea automatically grows as you type, with a max height of 200px.

---

4. Setup & Installation

Prerequisites

- Node.js 18+
- npm or yarn
- Groq API key (free at console.groq.com)

Local Development

```
# Clone the repository
git clone https://github.com/eduartgeorgia/propertywitch.git
cd propertywitch

# Setup backend
cd server
cp .env.example .env
# Edit .env and add your GROQ_API_KEY
npm install
npm run dev

# In another terminal — setup frontend
cd web
npm install
npm run dev
```

Open http://localhost:5173

Environment Variables

```
# server/.env
PORT=3001
GROQ_API_KEY=gsk_xxxxxxxxxxxxx
NODE_ENV=development

# Optional: Ollama fallback
OLLAMA_URL=http://localhost:11434
```

---

5. Deployment

Backend (Render.com)

1. Connect GitHub repo to Render
2. Create new Web Service
3. Settings:
   - Build Command: `cd server && npm install && npm run build`
   - Start Command: `cd server && node dist/server.js`
   - Add env var: `GROQ_API_KEY`

Frontend (Hostinger)

```
# Build frontend
cd web
npm run build

# Deploy to Hostinger
sshpass -p 'YOUR_PASSWORD' scp -P 65002 -r dist/* \
  u805002786@147.93.73.224:~/domains/propertywitchtest.com/public_html/
```

---

6. API Reference

POST /api/chat

Main chat endpoint for all interactions.

Request:

```
{
  "message": "apartments in lisbon under 300k",
  "mode": "auto",
  "threadId": "uuid-thread-id",
  "userLocation": {
    "lat": 38.7223,
    "lng": -9.1393,
    "city": "Lisbon",
    "currency": "EUR"
  }
}
```

Response:

```
{
  "type": "search",
  "intentDetected": "search",
  "message": "I found 35 apartments in Lisbon...",
  "searchResult": {
    "listings": [...],
```

```
    "matchType": "exact",
    "appliedPriceRange": { "min": 0, "max": 300000 }
  },
  "threadId": "uuid-thread-id"
}
```

GET /api/chat/ai/health

Check AI backend status.

GET /api/rag/status

Get RAG system statistics (indexed listings count).

POST /api/rag/initialize

Manually trigger RAG reindexing.

GET /api/chat/vision/status

Get Vision AI service status.

Response:

```
{
  "available": true,
  "model": "llama-3.2-90b-vision-preview",
  "cacheSize": 42
}
```

POST /api/chat/vision/analyze-image

Analyze a single image for property features.

Request:

```
{
  "imageUrl": "https://example.com/property-photo.jpg"
}
```

Response:

```
{
  "success": true,
  "analysis": {
    "features": ["sea_view", "modern_architecture", "terrace"],
    "confidence": { "sea_view": 0.95, "modern_architecture": 0.88, "terrace": 0.92 },
    "description": "Modern white villa with ocean view and spacious terrace",
    "propertyCondition": "excellent",
    "architecturalStyle": "contemporary",
    "surroundings": "Coastal area with ocean views"
```

```
  },
  "summary": "Features: sea view, modern architecture, terrace"
}
```

POST /api/chat/vision/analyze-listing

Analyze multiple photos for a listing.

Request:

```
{
  "listingId": "abc123",
  "photoUrls": ["url1", "url2", "url3"],
  "maxPhotos": 3
}
```

---

7. AI Analysis System

Analysis Flow

```
User Query
    |
    v
+--------------------+
| AI Intent Detection | <-- Groq llama-3.3-70b
+----------+----------+
           |
           v
+--------------------+
|   OLX API Search    | <-- Fetch listings
+----------+----------+
           |
           v
+--------------------+
| filterListingsByRelevance()  |
| - If <=10: Detailed prompt   |
| - If >10: Brief prompt       |
+----------+----------+
           |
           v
+--------------------+
| getRelevantListings()        |
| - Re-analyze if filtered     |
|   from many to few           |
+----------+----------+
           |
           v
```

```
+--------------------+
| pickBestListings()           | <-- For "narrow down" queries
| - Per-listing analysis       |
| - Updates aiReasoning field  |
+--------------------+
```

Key Functions

Function	Purpose
`detectIntent()`	AI-powered intent classification
`filterListingsByRelevance()`	Analyze listings with AI
`buildAnalysisPrompt()`	Create detailed or brief prompts
`analyzeListingsLocally()`	Fallback when AI unavailable
`pickBestListings()`	Handle “narrow down” / “choose X”
`getRelevantListings()`	Re-analyze when results shrink

Analysis Config

```
const AI_ANALYSIS_CONFIG = {
  maxListingsForAI: 20,        // Skip AI if more than this
  analysisTimeoutMs: 60000,    // 60 second timeout
  enableAIAnalysis: true,
  detailedAnalysisThreshold: 10 // Detailed if <= this
};
```

---

8. Vision AI (Image Analysis)

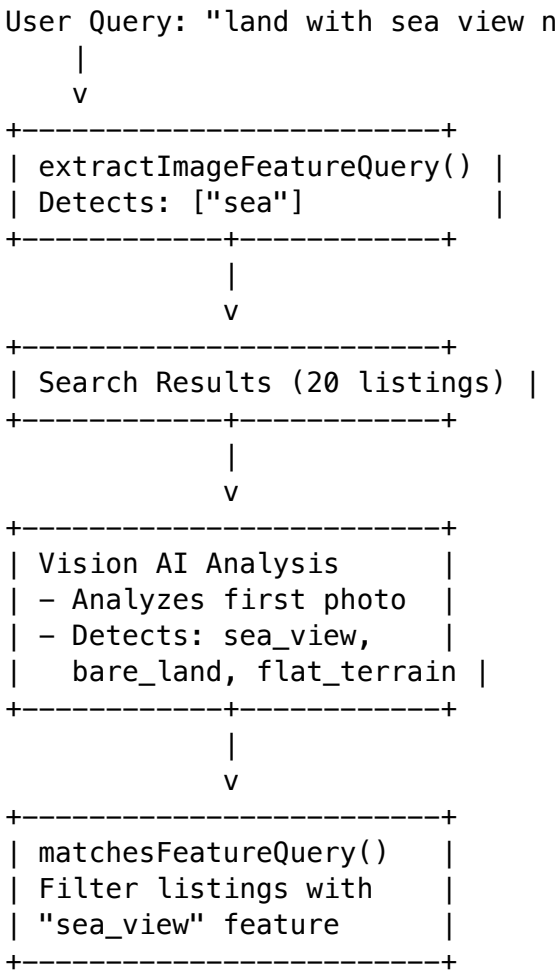
Property Witch includes computer vision capabilities to analyze property photos and identify visual features.

Overview

The Vision AI system uses Groq’s vision model (llama-3.2-90b-vision-preview) to analyze property images and detect features like:

Category	Detected Features
Water Features	Swimming pool, sea view, ocean view, waterfront, river view
Natural Surroundings	Forest, trees, garden, mountain view, vineyard, olive grove
Terrain	Bare land, flat terrain, sloped terrain, rocky terrain
Building Condition	Ruins, old building, needs renovation, modern, traditional
Amenities	Parking, garage, terrace, balcony, rooftop, solar panels
Location Type	Urban area, suburban, rural area, remote location

How It Works



Enabling Vision During Indexing

Set environment variable:

```
ENABLE_VISION_INDEXING=true
```

This will analyze the first photo of each new listing during scheduled indexing (limited to 50 per run to manage API costs).

Example Queries with Vision

Query	Detected Features
“house with pool”	[“pool”]
“land with sea view”	[“sea”]
“property with vineyard”	[“vineyard”]
“modern apartment”	[“modern”]
“old house to renovate”	[“old”, “renovation”]
“rural land with forest”	[“rural”, “forest”]

Feature Synonyms

The system understands synonyms: - “pool” -> swimming_pool - “ocean”, “beach”, “sea” -> sea_view, ocean_view, waterfront - “trees”, “green” -> forest, trees, garden - “fixer upper” -> needs_renovation

---

9. Scheduled Indexer

Automatically indexes listings every 4 hours.

Coverage

Cities (10): - Lisbon, Porto, Faro, Braga, Coimbra - Setubal, Aveiro, Leiria, Evora, Funchal

Query Types (9): - Apartments, Houses, Land, Rooms - Commercial, Farms, Garages, Offices, Warehouses

Filters

- **Max Age:** 3 months (filters out old listings)
- **Deduplication:** By listing ID

Stats

Current index: **~8,500 listings**

Manual Trigger

```
curl -X POST https://property-witch-api.onrender.com/api/rag/initialize
```

---

10. Configuration

Frontend Config (web/src/App.tsx)

```
const API_URL = "https://property-witch-api.onrender.com";
```

CORS Config (server/src/index.ts)

```
const corsOptions = {
  origin: [
    "http://localhost:5173",
    "https://propertywitchtest.com",
    "https://www.propertywitchtest.com"
  ]
};
```

---

11. Troubleshooting

“Waking up server…”

Cause: Render free tier sleeps after 15 min inactivity.

Solution: The frontend has retry logic with exponential backoff.

AI Analysis Too Short

Check: 1. Is the listing count <= 10? (triggers detailed mode) 2. Is `pickBestListings()` being used? (for “narrow down” queries) 3. Check server logs: `[AI Analysis] Mode: DETAILED/BRIEF`

“fetch failed” Errors

Cause: OLX API rate limiting or network issues.

Solution: Built-in retry with exponential backoff (up to 3 retries).

Input Not Clearing

Fixed in v3.0: Input clears immediately on submit, not in `finally` block.

m2 vs EUR Confusion

Fixed in v3.0: AI distinguishes area (m2) from price (EUR) contextually.

Vision AI Not Working

Check: 1. Is `GROQ_API_KEY` configured? 2. Check `/api/chat/vision/status` endpoint 3. Check server logs for `[Vision]` messages

---

12. Project Structure

```
aipa/
|-- server/
|   |-- src/
|   |   |-- index.ts           # Express app entry
|   |   |-- routes/
|   |   |   |-- chat.ts        # Main chat endpoint + Vision APIs
|   |   |   +-- rag.ts         # RAG management
|   |   |-- services/
|   |   |   |-- aiService.ts   # AI analysis (MAIN LOGIC)
|   |   |   |-- visionService.ts # Vision AI (NEW)
|   |   |   |-- searchService.ts
|   |   |   |-- scheduledIndexer.ts
|   |   |   +-- rag/           # RAG system
```

```
|   |   +-- adapters/
|   |       +-- olxAdapter.ts  # OLX API integration
|   +-- data/
|       +-- rag/               # Indexed listings JSON
|-- web/
|   |-- src/
|   |   |-- App.tsx            # Main React component
|   |   |-- styles.css         # Tailwind + custom styles
|   |   +-- types.ts
|   +-- index.html
+-- README.md
```

---

13. Version History

Version	Date	Changes
3.1	Feb 2025	Vision AI (image recognition), detect pools/sea views/ruins/etc
3.0	Feb 2025	AI intent detection, adaptive analysis, pickBestListings fix
2.0	Jan 2025	Scheduled indexer, RAG system, Groq integration
1.0	Dec 2024	Initial release, OLX adapter, basic search

Contributing

1. Fork the repository
2. Create a feature branch
3. Make changes
4. Test locally
5. Submit a pull request

License

MIT License - See LICENSE file for details.

Property Witch Team | propertywitchtest.com