Building J.A.R.V.I.S. with OutSystems:

AI Orchestration with Workbench and RAG

AI assistants are everywhere. But most of them hit a wall the moment you ask them to handle real-world business tasks that cut across multiple systems. That's where J.A.R.V.I.S. comes in.

At OutSystems, we set out to build a smart assistant that doesn't just chat but works alongside developers and tech leads. Our goal was to orchestrate multiple AI agents to tackle real productivity challenges, while keeping things modular, scalable, and transparent.

This is the story of how we built J.A.R.V.I.S. with OutSystems, why we made the choices we did, and what we learned along the way.

Why J.A.R.V.I.S.?

Yes, the name comes from Iron Man. And yes, we knew we'd be compared to Tony Stark's omniscient assistant. But J.A.R.V.I.S. was more than a fun acronym.

We wanted an assistant that could:

- Understand context.
- Route requests intelligently.
- Use the right "expert" for the job.
- Delivery answers are grounded in real company data.

The Marvel nod gave us a memorable name, but the design goal was serious: a modular AI assistant with orchestrated agents that feels practical, not sci-fi.

High-Level System Overview

At its core, J.A.R.V.I.S. is a collection of specialized AI agents coordinated by a decision-making layer. OutSystems handles orchestration, integrations, and UI, while large language models (LLMs) provide reasoning and natural language understanding.

The flow looks like this:

1. User request enters the system.
2. Decision Maker evaluates intent and selects the right agent(s).
3. Agent(s) executes tasks, calling external systems or OutSystems apps.
4. Response is composed and sent back to the user.

This modularity lets us swap, upgrade, or extend agents without rewriting the whole system.

Meet the Agents

Each agent has one job. That keeps things simple, reusable, and easier to debug. Here's the lineup:

- Decision Maker: Routes requests to the right agent. Think of it as air traffic control.
- Calendar Agent: Reads the calendar, finds free slots, and suggests times.
- Email Agent: Retrieves emails and summarizes long threads so you don't waste 20 minutes scrolling.
- Mailman Agent: Writes and sends emails with the right tone and context.
- Meetings Agent: Schedules meetings and confirms availability.
- Project Manager Agent: Connects with an OutSystems app to perform calculations and access project data.

By breaking tasks into specialized roles, we avoid creating a bloated "do-everything" model that fails at the details. Each agent is lightweight, focused, and reusable in other contexts.

RAG: Why It Matters

Large language models are powerful, but they hallucinate. No developer wants an assistant that confidently makes things up. That's where Retrieval-Augmented Generation (RAG) saves the day.

Here's how we implemented RAG in J.A.R.V.I.S.:

- Break documents into chunks.
- Embed chunks into vectors.
- Store them in a vector database (we used Qdrant).
- Convert the user's query into an embedding.
- Run a similarity search.
- Feed the retrieved chunks plus the query into the LLM for the final answer.

This pipeline keeps responses grounded in your company's data. Instead of guessing, J.A.R.V.I.S. pulls relevant knowledge and ties it directly into the answer.

For OutSystems developers, this means you can plug in your own data—project documentation, design specs, process manuals—and trust the assistant to stay fact-based.

Challenges and Lessons Learned

No system build is complete without bumps in the road. Here are the main challenges we faced, and what we'd recommend to others.

Token Usage and LLM Costs

- LLMs burn tokens fast.
- Monitor usage and design prompts carefully.
- Anticipate costs before scaling production.

Start Modular

- Break down problems with small, reusable agents.
- Avoid the temptation to build a monolithic assistant.
- Modularity makes debugging and iteration much easier.

Intelligent Orchestration

- Orchestration is more than routing.
- Sequence and manage workflows for complex tasks.
- Plan for agents that need to work together, not just one at a time.

Human-in-the-Loop

- Automation doesn't mean full autonomy.
- Keep humans in the loop for feedback and validation.
- For tasks like sending client emails, you want to review before action.

AI Will Surprise You

- Models don't always behave as expected.
- Build logging and monitoring into every layer.
- Treat AI as a partner that sometimes goes off-script.

Why This Matters for OutSystems Developers and Enterprises

For developers and tech leads, J.A.R.V.I.S. demonstrates how OutSystems and AI orchestration combine to deliver real productivity gains.

Instead of juggling calendars, parsing emails, or manually updating project data, you offload that work to specialized agents. OutSystems takes care of the orchestration and UI, while the LLM handles reasoning and language.

The result is not just proof of concept. It's a practical model for how teams can build assistants that extend their workflows, save hours of repetitive work, and ground outputs in reliable data.

Takeaway

J.A.R.V.I.S. showed us one thing clearly: OutSystems plus AI orchestration is not about flashy demos. It's about building assistants that do actual work, reduce friction, and free up humans for higher-value tasks.

The future of productivity is not a single all-knowing model. It's a network of specialized agents, orchestrated intelligently, and plugged into your business systems.

If you're leading an OutSystems team, now is the time to think about how modular AI agents can support your workflows. Start small. Build reusable pieces. Keep humans in the loop. And let OutSystems handle the orchestration, so your assistant feels less like sci-fi, and more like a reliable teammate.