21017 - EARS

Environment Adaptive Recommendation System

WP 4 – Platform Development

Deliverable 4.4: EARS Platform

Deliverable Type: Software

Classification: Public

Due Date: April 2026

HISTORY

Document Version	Date	Remarks
V1	15.06.2026	First Version
V2	23.06.2026	Second Version

Table of Contents

1 Introduction

This document describes the software of the EARS central integration platform: the application that brings the recommendation and analysis services of all consortium partners together behind one entry point. Where the earlier deliverables defined the system from different angles, the overall architecture and federated-learning design (D3.1), the data integration and API surface (D4.2), and the business layer and common services (D4.3), this document takes the software view. It explains how the platform is built, which components it is made of, how partner services interact at run time, how user access is unified, and how the system is deployed.

The aim of the platform is simple to state. Each partner already runs its own services in its own environment. Instead of asking every partner to change how they work, the central platform reaches those services and presents them through a single, consistent interface. A user or another application, talks to one place, and the platform takes care of routing each call to the right service and returning the result in a uniform shape.

The description below stays at the level of design and intent: what the software is meant to do and how its parts fit together, rather than a line-by-line account of the code.

2 Software Overview

The platform is a single web application built on ASP.NET Core (the web application framework of the .NET platform). It rests on two pillars.

The first pillar is the unified service surface. Every partner service is exposed under one host, following the same address pattern (/api/{country}/{partner}/...). This means a product recommendation call to Portugal and a clinical image analysis call to Spain are reached in exactly the same way, with the same conventions for requests, responses, and errors. Partners keep their services where they are; the surface is the common doorway in front of them.

The second pillar is the federated learning engine — federated learning meaning a way of training a shared model from many separate data owners without moving their raw data to one place. The engine accepts locally trained models from partner nodes, combines them into a single global model, evaluates the result, and serves recommendations from it. This is the part of the platform that turns many local efforts into one shared outcome. Several of the partner recommenders — Portugal's and Turkey's among them — are themselves built on federated learning, which is exactly what this engine is designed to bring together.

Around these two pillars sit a data store (a MySQL relational database that holds identities, models, and settings) and a web dashboard (a set of browser pages for the service catalogue, an integration map, and metrics). The heavier model computations are performed by Python scripts that the application calls when it needs them: .NET handles the orchestration and storage, and Python handles the machine-learning mathematics.

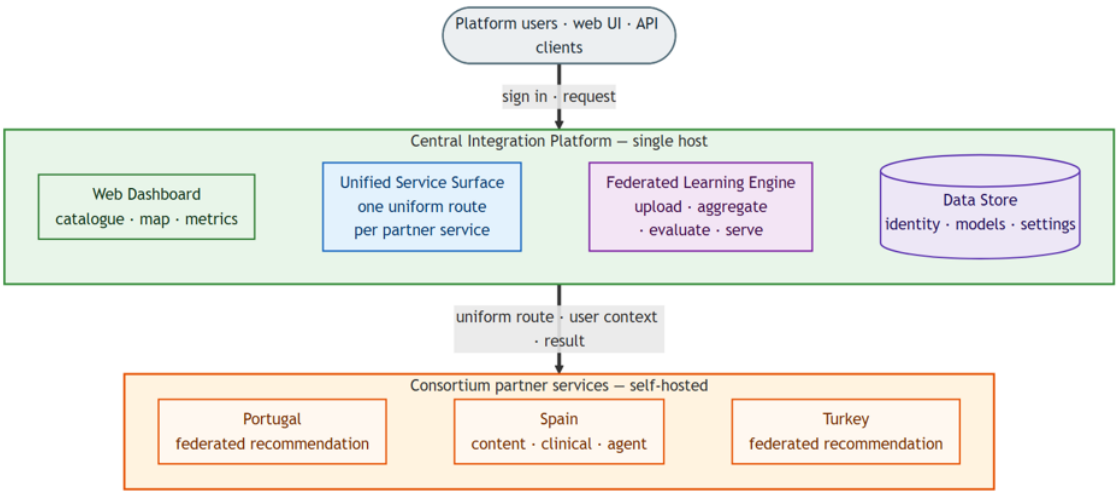

Figure 1. High-level software composition

3 System Components and Code Structure

The backend is organised in clear layers, each with a single responsibility.

3.1 Application layers

Controllers (API layer). These define the endpoints the outside world calls. They are grouped by purpose: partner service controllers (one group per country), federated-learning controllers (model upload, aggregation, prediction), and identity controllers (sign-in and platform listing).

Business services and workers (business layer). These hold the actual logic: the model aggregator, the identity/token service, the simulation orchestrator, and a background worker that performs aggregation on a schedule without anyone having to trigger it by hand.

Data layer. A single data context — built on Entity Framework Core, the data-access library of .NET — maps the application's objects to database tables.

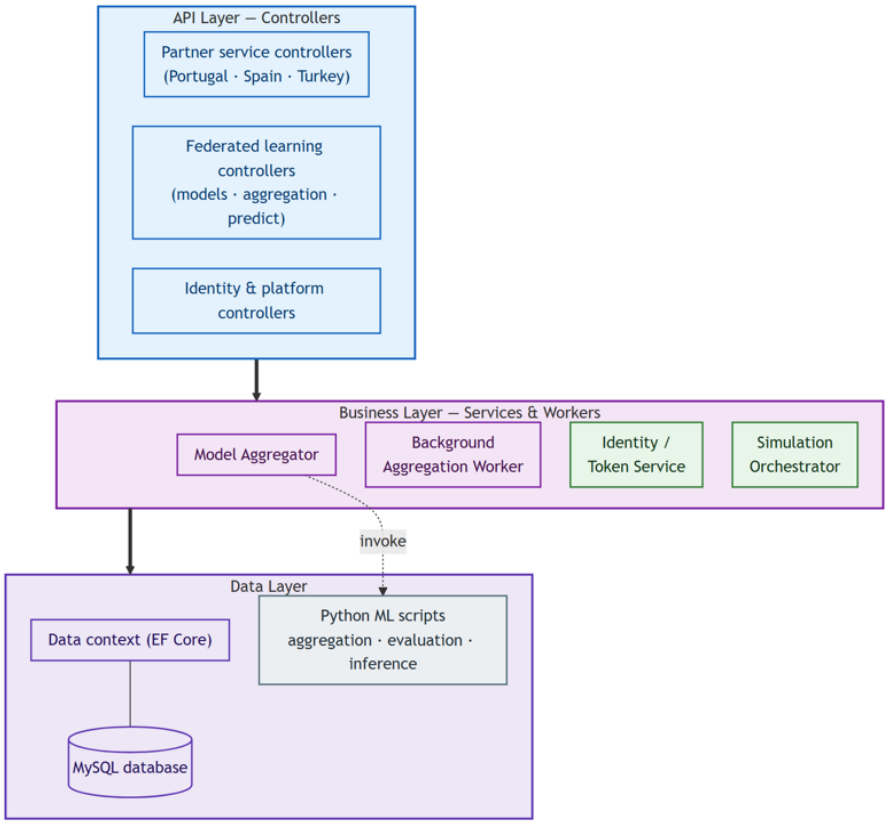

Figure 2. Backend component and layer map

3.2 The unified service surface

Each partner group is handled by its own controller, but all of them follow the same routing convention and the same request/response style. Adding a new partner service means adding it under the same pattern — nothing about the calling convention changes. This uniformity is what lets the dashboard, and any external client, treat every service the same way regardless of which country or technology is behind it.

3.3 The federated learning engine

The engine is built around a model aggregator service and a background worker. The aggregator collects the eligible locally trained models for a given platform and version, combines them using a selected strategy, evaluates the combined model, and stores it as the new global model along with an analysis of its structure. A strategy registry holds the available aggregation methods (for example FedAvg, the standard federated-averaging method, alongside more robust variants), and a policy decides which method applies. The mathematics runs in external Python scripts; the .NET service decides what to run, with which inputs, and records the outcome.

3.4 Data layer

The database is organised into a few coherent groups: identity (users and the platform each user belongs to), models (locally uploaded models, and the aggregated global models with their analyses), and aggregation configuration and logs (the policies, defaults, strategy registry, and the record of every aggregation round and recommendation). Keeping these groups distinct makes the data model easy to follow.

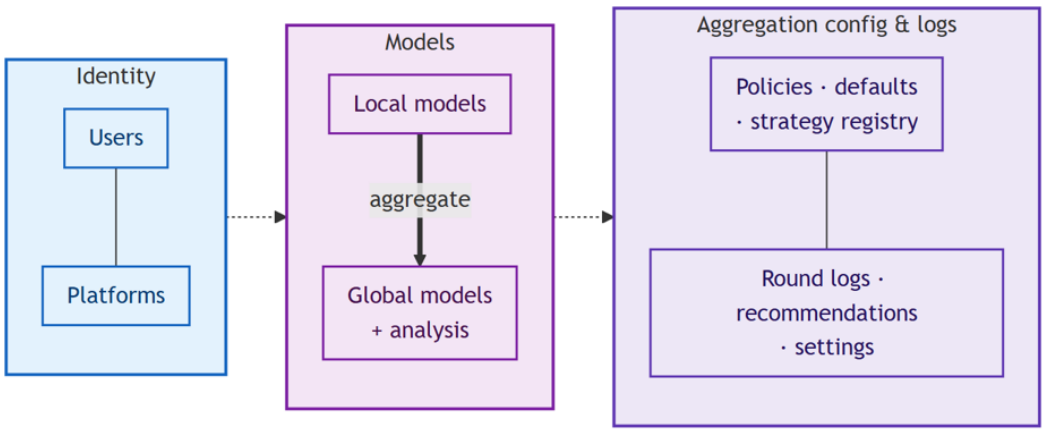

Figure 3. Data model groups

3.5 Web dashboard

The dashboard is a set of browser pages built with lightweight front-end tools. It provides a catalogue of all registered services, an integration map that shows partners by country and by flow, metrics pages, and a “try it out” page that calls a chosen endpoint directly. It is the human-facing window onto the platform.

4 Service Integration Model

4.1 Uniform routing and the service catalogue

All partner services are reached through one host and one address pattern. The platform keeps a catalogue of these services, so the full set of consortium capabilities can be seen and exercised from one place. Because the convention is the same everywhere, the effort of integrating an additional service stays small and predictable.

4.2 Partner service groups

The services are described here by what they do; the contributing partner is named in parentheses where it helps.

Portugal — product recommendation with explanations. A recommendation service family for an e-commerce catalogue (Norsafe). It offers generic recommendations for a shopper, items similar to a given product, complementary products of a source product, and it returns ranked items with scores and short natural-language explanations. It also exposes model training and display-tuning functions for administrators.

Spain — content, clinical, and agent services. Three service groups: a news/content service built on natural-language processing and a knowledge graph (SQ1) that ingests text, extracts entities and topics, and answers content and recommendation queries; a clinical-AI group (Glintt) that analyses medical images and audio — for example pneumonia detection, breast-tissue segmentation, and audio transcription; and an agent group (Dextro) that provides a conversational assistant, access-scope and prompt-safety checks, explanations, and inventory alerts.

Turkey — federated recommendation. A federated two-tower recommendation pipeline (contributed by Donanım Haber, Doğuş Teknoloji, and adesso) in which each provider trains a local model and ships it to a central point for aggregation and inference. Alongside it sits the central Federated Model Server (FMS) for uploading models, recommending an aggregation strategy, running inference, and analysing model structure. This group is the closest match to the platform's own federated-learning engine.

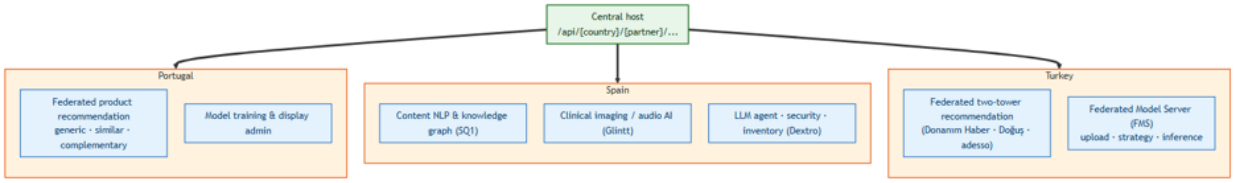

Figure 4. Unified service catalogue by country and function

4.3 Supported interface patterns

Partner services do not all behave the same way, and the surface accommodates three patterns: synchronous request/response (the call returns the result immediately — most recommendation and analysis services); asynchronous task polling (a long-running job returns a task identifier, and the platform polls until the result is ready — used for heavy training and natural-language jobs); and streaming (the service pushes a continuous flow of events — used by the conversational agent and the real-time insight feed). Presenting all three behind one consistent surface is part of what the platform provides.

4.4 From a shared surface to live integration

The architecture is deliberately staged. The uniform surface and routing are defined first, so that the shape of the integration is fixed and every partner is reached the same way. Each service behind the surface can then be connected to its live partner endpoint without changing anything the caller sees. This lets the consortium demonstrate the complete, end-to-end picture early, and bring services fully live one by one.

5 Service Interaction and Runtime Flows

5.1 A service request, end to end

When a user asks for something — say, a set of recommendations — the request reaches the central platform, which resolves the correct route, attaches the user's context, and forwards the call to the partner service. The service returns its result, and the platform passes it back in the platform's uniform response shape. The caller never deals with the partner's specific address or conventions; it always talks to the platform.

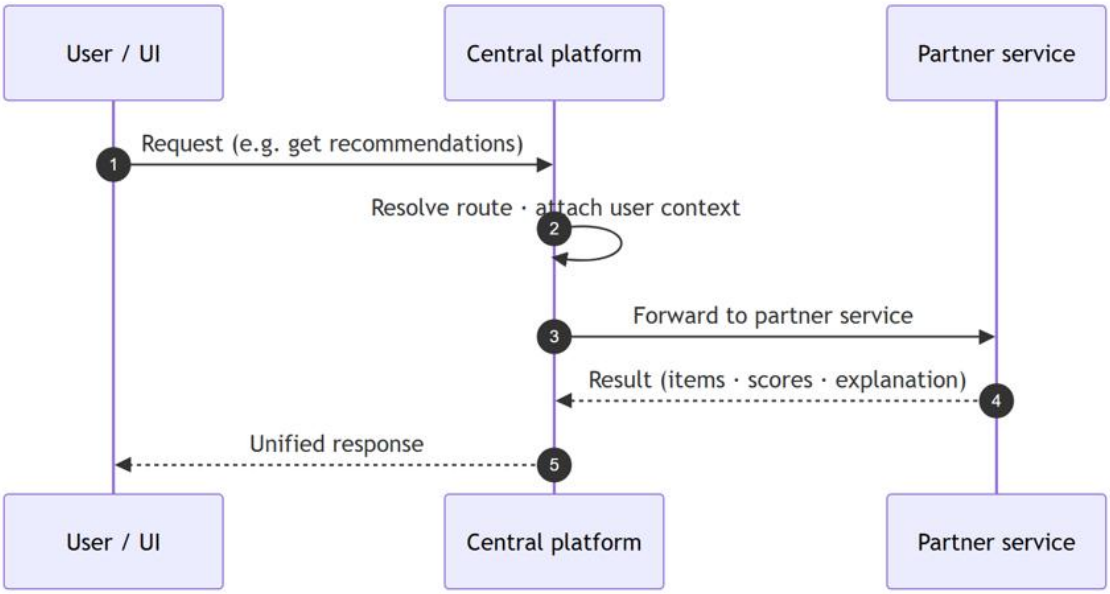

Figure 5. A service request, end to end

5.2 The federated model lifecycle

The federated-learning flow is the platform's core internal process. A partner node trains a model on its own data and uploads it. The platform gathers the eligible uploaded models, aggregates them with the chosen strategy, evaluates the combined model, and stores it as the new global model. From that global model the platform serves recommendations and inference. Each new round folds the previous global model back in, so the shared model improves over time without any raw data ever leaving its owner.

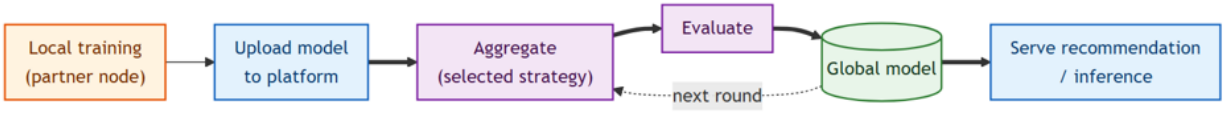

Figure 6. The federated model lifecycle

5.3 Strategy recommendation

Because the best way to combine models depends on the situation — how unevenly data is distributed, how many clients take part, how much risk there is of poor updates — the platform includes a step that recommends an aggregation strategy from a description of the case. This keeps the choice of method explicit and recorded, rather than fixed in advance.

6 Authentication and Access Unification

Partners secure their services in different ways — some with bearer tokens, some with API keys, some with signed tokens and roles. Asking a user to deal with all of these would defeat the purpose of a single entry point. The platform therefore unifies access behind one central identity and presents the same two-layer scheme to every partner service.

The first layer is the platform's own identity. The platform proves to each partner service that a request genuinely comes from the trusted central platform — for example through a shared key sent with every call. This lets a partner accept platform traffic with confidence, without opening its service to arbitrary callers.

The second layer is the end-user's identity. A user signs in once to the platform and receives a single identity token that carries who they are and which platform they belong to. When a request is routed onward, the platform passes this user context to the partner service in whichever form suits that partner: as plain request headers it can read directly (the simplest option), as the forwarded token for it to decode, or by exchanging the token for the partner's own. The partner service still applies its own access rules; the platform only supplies the user context in a consistent shape.

The design is kept intentionally lightweight. A single shared identity, plus a small amount of user context travelling with each routed call, is enough to let many differently secured partner services work together — without forcing any partner to change how it protects its own service.

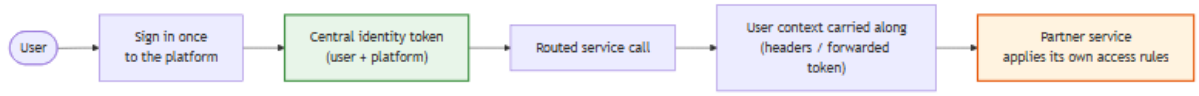

Figure 7. Unified authentication and user-context flow

7 Deployment and Environment

The platform runs as one application on a central server, together with its MySQL database and the Python runtime used for model computations. Partner services run wherever their owners host them and are reached over the network; the only requirement is that the central server can call them. To keep its picture of the system current, the platform also checks each partner's health endpoint on a regular interval, so a service that is temporarily unavailable is shown as such rather than failing silently.

Configuration is held outside the code: the service catalogue, the aggregation settings, and the environment-specific values — such as the database connection and identity keys — are all set through configuration, so the same software can run in different environments without change.

Moving from a demonstration setup to a production one is mainly a matter of connecting each service to its live partner endpoint and tightening the operational settings: identity, secrets, and monitoring. Because the surface and the flows stay the same, this transition does not change the shape of the system the user sees.

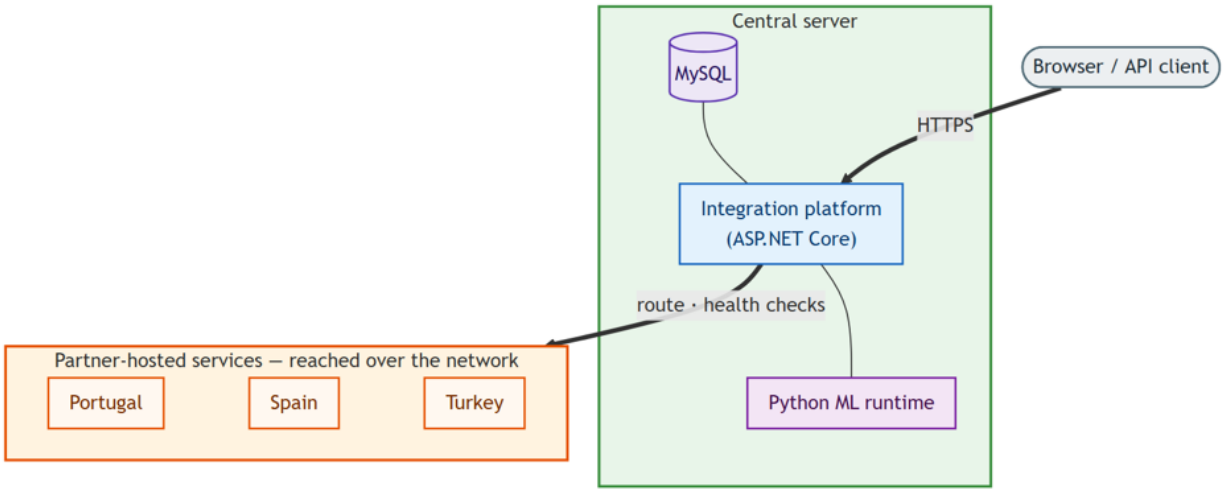

Figure 8. Deployment topology

8 Conclusion

The EARS central integration platform brings the consortium's services together behind one consistent surface, adds a working federated-learning engine that turns many local models into one shared model, and unifies user access behind a single identity. Its software is organised in clear layers, follows one uniform convention for every partner service, and is designed so that services can be brought live one at a time without disturbing the whole. The result is a practical foundation for operating the partners' capabilities — across Turkey, Portugal, and Spain — as a single system.