CÁMARA DE COMERCIO DE QUITO

Área de Tecnología

Documento Técnico

Proyecto MERA

Frontend, Backend, APIs, IA, Búsqueda, Moodle y Base de Datos

Versión	Fecha	Autor	Descripción de cambios
1.0	13/03/2026	Equipo de TI	Creación del documento técnico maestro.
1.1	17/03/2026	Equipo de TI	Revisión, actualizaciones y diseño de portada.

Fuentes primarias analizadas en el repositorio:

- `03. API .NET`
- `04. CODIGO`
- `01. BDD/SQL/Script Completo/tables_ef_utf8_clean.sql`
- `01. BDD/SQL/Mera Scripts/script_unificado_4_6_3_5_7.sql`

Índice

1. Diccionario de términos

Este diccionario estandariza el significado de los términos técnicos y siglas utilizados en el documento.

término	**Definición**
Proyecto MERA (MERA)	Nombre del proyecto/sistema documentado en este archivo.
CCQ	Cámara de Comercio de Quito.
SPA	Single Page Application: aplicación web de una sola página que navega sin recargar completamente.
Frontend	Capa cliente (navegador) responsable de UI/UX y consumo de la API.
Backend	Capa servidor que expone la API y ejecuta la lógica de negocio.
API	Interfaz de programación de aplicaciones; en este contexto, endpoints HTTP.
API REST	Estilo de API HTTP basada en recursos (GET/POST/PUT/DELETE) y JSON.
Endpoint	Ruta/operación especifica expuesta por la API (por ejemplo `/api/...`).
Ruta (route)	URL o path utilizado para acceder a una pantalla (frontend) o endpoint (backend).
Payload	Cuerpo de una solicitud/respuesta (normalmente JSON) enviado entre frontend y backend.
Query string	Parámetros en la URL (por ejemplo `?page=1&pageSize=10`).
React	Librería de JavaScript para construir interfaces de usuario.
Vite	Herramienta de desarrollo/build para proyectos frontend modernos.
Redux	Librería para manejo de estado global en frontend.
Axios	Cliente HTTP usado desde el frontend para consumir la API.
.NET Framework Web API	Framework/stack backend utilizado para exponer controladores HTTP en C#.
Controlador (Controller)	Clase backend que define endpoints HTTP y orquesta servicios.
Servicio (Service)	Componente de negocio que implementa reglas y orquestación de procesos.
Repositorio (Repository)	Capa de acceso a datos que encapsula consultas y persistencia.
Entity Framework (EF)	ORM para mapear tablas SQL a entidades y ejecutar operaciones de datos.
Dapper	Micro-ORM para ejecutar SQL directo con mapeo liviano.
SQL Server	Motor de base de datos principal del sistema.
MySQL	Motor de base de datos usado por Moodle (en modo de integración directa).
JWT	JSON Web Token: token firmado para autenticación/autorización.
Claim	Campo dentro del JWT (por ejemplo, rol, email, id).
Issuer/Audience	Parámetros de validación de JWT (emisor/destinatario).
sessionStorage/localStorage	Almacenamiento del navegador para persistir sesión/datos del usuario.
CORS	Política de intercambio de recursos entre orígenes; controla llamadas cross-domain.

Dashboard	Pantalla de indicadores, tablas y visualizaciones para administración/seguimiento.
KPI	Indicador clave de rendimiento mostrado en el dashboard.
CRUD	Operaciones básicas: Create, Read, Update, Delete.
Matching IA	Proceso de análisis y ranking que estima compatibilidad candidato-vacante.
Ranking	Lista ordenada de candidatos (o empleos) según un score.
Precomputo	Estrategia donde resultados (por ejemplo, ranking) se calculan antes y se consultan luego.
Fallback	Comportamiento alternativo cuando un componente falla o no está disponible.
Gemini	Proveedor/modelo IA usado para análisis semántico (según configuración).
OCR	Reconocimiento óptico de caracteres para extraer texto de documentos.
OCR.space	Servicio OCR opcional mencionado/consumido por el sistema.
SHA-256	Algoritmo hash usado para detectar cambios en fuentes textuales.
GUID	Identificador único global (UUID) usado en tokens/lógica.
Moodle	Plataforma LMS externa integrada para cursos/badges/certificados.
OAuth	Protocolo para autorización/autenticación delegada (flujo de vinculación con Moodle).
Web.config	Archivo de configuración del backend .NET (claves, endpoints, connection strings).
Connection string	Cadena de conexión para acceder a una base de datos desde el backend.
Auditoría	Registro de accesos/eventos (por ejemplo, logins, acciones administrativas).

2. Resumen ejecutivo

El sistema analizado corresponde al Proyecto MERA, compuesto por cuatro dominios técnicos principales:

- Un frontend SPA desarrollado con React 18 y Vite, responsable de la experiencia de usuario para candidatos, empresas, administradores y visitantes públicos.
- Un backend en .NET Framework Web API con arquitectura por capas, responsable de autenticación, negocio, integración con base de datos, IA, Moodle, auditoría y notificaciones.
- Una base de datos SQL Server que modela candidatos, empresas, empleos, postulaciones, catálogos, auditoria, configuración y objetos auxiliares para IA.
- Un subsistema de Matching inteligente que combina análisis por Gemini, OCR opcional, heurísticas de fallback y persistencia precomputada para ranking de candidatos por vacante.

El sistema no es un prototipo aislado; contiene flujos end to end operativos para:

- Registro e inicio de sesión de candidatos, empresas y administradores.
- Gestión integral de perfil del candidato: hoja de vida, resumen, experiencia, formación, idiomas, intereses, habilidades, fotografía y certificados.
- Publicación, actualización, consulta y desactivación de vacantes por parte de empresas.
- Exploración y búsqueda de empleos por filtros combinados.
- Postulación de candidatos a vacantes y seguimiento.
- Dashboard administrativo con indicadores, tablas, auditoria y configuración.
- Integración con Moodle por API REST, OAuth y acceso MySQL para recuperación de información académica y certificados.
- Motor de IA de apoyo al reclutamiento basado en preanálisis y ranking persistido.

Este documento sirve como base de sustentación técnica, memoria del sistema, manual de arquitectura y referencia para anexar capturas funcionales.

3. Objetivo del documento

Este documento tiene cuatro objetivos:

1. Describir la arquitectura real del sistema a partir del código fuente y los scripts SQL.
2. Inventariar componentes funcionales: pantallas, servicios, controladores, rutas, tablas, procedimientos y procesos.

3. Explicar el flujo técnico de negocio, especialmente en autenticación, búsqueda, postulación, IA y Moodle.
4. Proveer una estructura formal sobre la que se puedan insertar capturas de pantalla y evidencias de ejecución.

4. Alcance

El alcance de esta documentación cubre:

- Solución backend `03. API .NET`.
- Aplicación frontend `04. CODIGO`.
- Scripts SQL base y complementarios ubicados en `01. BDD/SQL`.
- Configuraciones visibles en `Web.config`, `package.json` y archivos README del repositorio.

No cubre infraestructura externa no incluida en el repositorio, como configuración real de IIS, DNS, certificados SSL, pipelines CI/CD, servidores productivos o manuales operativos institucionales.

5. Visión general del sistema

5.1. Problema que resuelve

La plataforma centraliza el proceso de intermediación laboral entre candidatos y empresas. Permite a los postulantes construir y mantener un perfil profesional digital completo, a las empresas publicar oportunidades y gestionar postulaciones, y a los administradores monitorear el comportamiento global del ecosistema.

Sobre esa base tradicional se incorporan dos capacidades diferenciadoras:

- Integración académica con Moodle para enriquecer la evidencia formativa del candidato.
- Matching inteligente precomputado para acelerar el proceso de selección por parte de la empresa.

5.2. Actores del sistema

Actor	**Responsabilidades y capacidades principales**
Visitante publico	Navega empleos, empresas, página principal, FAQ, acerca de y registro.

Candidato	Se registra, inicia sesión, completa perfil, sube CV, gestiona certificaciones, consulta postulaciones, guarda empleos, cambia contraseña, se vincula con Moodle y postula a vacantes.
Empresa	Se registra, inicia sesión, mantiene perfil corporativo, publica y edita vacantes, gestiona empleos y revisa candidatos o postulaciones.
Administrador	Inicia sesión, consultas estadísticas, analítica, auditoria, datos agregados, configura sistema y administra estados de candidatos y empresas.
Servicios externos	Gemini, OCR.space opcional, Moodle REST, proveedor MySQL de Moodle, envío de correo.

5.3. Arquitectura lógica de alto nivel

1. El usuario interactúa con una SPA React.
2. El frontend consume una API REST .NET mediante Axios.
3. El backend delega la lógica a servicios de negocio.
4. Los servicios de negocio usan repositorios para acceder a SQL Server.
5. Ciertas operaciones disparan procesos secundarios: JWT, IA, Moodle, correo y auditoria.
6. El resultado vuelve al frontend para renderizar vistas, dashboards, formularios y tablas.

6. Estructura física del repositorio

6.1. Carpetas principales

Ruta	**Descripción**
`01. BDD`	Scripts de base de datos, migraciones, objetos SQL y anexos de auditoria e IA.
`02. POSTMAN`	Colecciones o evidencias de consumo API.
`03. API .NET`	Solución backend compuesta por API, lógica de negocio, entidades, dominio y acceso a datos.
`04. CODIGO/JobBoard_Startup`	Frontend React/Vite, con páginas, servicios, store Redux, CSS y build.
`05. PLANTILLA`	Área adecuada para alojar este documento y posibles entregables.

6.2. Solución backend

La solución `api.mera.ccq.ec.sln` contiene los siguientes proyectos:

- `APIMera`: capa de presentación HTTP y configuración Web API.
- `lógica Negocio`: reglas de negocio, integraciones, servicios y orquestación.

- `AccesoDatos`: repositorios, persistencia con Entity Framework y Dapper.
- `Entidades`: modelos de datos transferidos entre capas.
- `Dominio`: constantes y utilidades auxiliares.

6.3. Aplicación frontend

El frontend se organiza sobre la carpeta `src` con la siguiente separación:

- `jsx/pages`
- `services`
- `store`
- `layout`
- `components`
- `config`
- `css`
- `context`
- `utils`
- `jsx/constant/alldata.jsx`

7. Stack tecnológico

7.1. Frontend

- React 18.3.1
- React Router DOM 7.4.0
- Redux 5.0.1 y Redux Thunk 3.1.0
- Axios 1.8.4
- Vite 6.2.0
- React Bootstrap 2.10.9
- Chart.js 4.5.1
- AG Charts 13.1.0
- SweetAlert 2.1.2
- React Quill
- html2canvas y jsPDF

7.2. Backend

- ASP.NET Web API sobre .NET Framework
- C#
- JWT
- Dapper
- RestSharp y HttpClient
- MySQL Connector/NET
- Newtonsoft.Json y System.Text.Json

7.3. Base de datos e integraciones

- SQL Server como base principal
- Moodle vía REST API o vía MySQL
- Gemini como proveedor opcional de análisis IA
- OCR.space como OCR opcional
- Proveedor de correo para notificaciones

8. Configuración del sistema

8.1. Configuración del frontend

El frontend toma la URL base de la API desde `VITE_API_URL`; en ausencia de dicha variable usa:

```
http://20.81.20.15/api.proyecto.mera
```

8.2. Configuración del backend

El archivo `APIMera/Web.config` concentra:

- Claves JWT.
- Credenciales y modelo para Gemini.
- Configuración opcional de OCR.
- Configuración Moodle REST.
- Configuración Moodle OAuth.

- Configuración Moodle MySQL.
- Connection strings.

8.3. Observación de seguridad

El repositorio contiene valores sensibles que en un entorno productivo deben rotarse y externalizarse.

9. Arquitectura backend

9.1. Capa API

La capa `APIMera` expone endpoints REST con enrutamiento por atributos. `WebApiConfig.cs`:

- Habilita CORS de forma abierta con `"*"`.
- Activa `MapHttpAttributeRoutes()`.
- Mantiene una ruta convencional de respaldo `api/{controller}/{id}`.

9.2. Capa de negocio

La carpeta `lógicaNegocio/Servicios` contiene el núcleo funcional:

- Candidatos
- Empresas
- Empleos
- Postulaciones
- Catálogos
- Datos académicos
- Experiencia laboral
- Idiomas
- Intereses
- Referencias laborales
- Favoritos
- Auditoría
- Dashboard administrativo
- Configuración del sistema

- Integración Moodle
- Matching IA

9.3. Capa de acceso a datos

La carpeta `AccesoDatos/Repositorios` materializa las operaciones de persistencia. El módulo IA utiliza Dapper y SQL directo con `MERGE`, `SELECT` y creación dinámica de esquema.

9.4. Entidades

Las entidades cumplen roles de modelo transaccional, transferencia entre capas e integración externa.

10. Autenticación, autorización y sesión

10.1. Modelo de autenticación

El sistema utiliza JWT:

1. Usuario envía credenciales.
2. Backend valida correo y contraseña.
3. Si la autenticación es correcta, genera token con email, rol e id.
4. El frontend guarda el payload en `sessionStorage` o `localStorage`.
5. Axios adjunta el token como `Authorization: Bearer`.
6. Los controladores protegidos aplican `JwtAuthentication`.

10.2. Claims del token

El generador de tokens incorpora:

- `sub`: email
- `jti`: GUID
- `ClaimTypes.Email`: email
- `ClaimTypes.Role`: rol
- claim `id`: identificador del usuario

10.3. Roles utilizados

- `Candidato`
- `Empresa`
- `Administrador`

10.4. Control de bloqueo

`JwtAuthenticationAttribute` valida firma, audiencia, issuer y expiración, y también consulta el estado del usuario:

- Si el candidato está bloqueado, responde `403 USER_BLOCKED`.
- Si la empresa esta inactiva, responde `403 USER_BLOCKED`.

10.5. Persistencia de sesión en frontend

El archivo `src/services/AuthService.jsx` administra guardado de usuario, lectura de sesión, logout, rehidratación automática y temporizador de expiración.

11. Catálogo de APIs backend

11.1. Resumen

La API se divide en dominios:

- Candidato
- Empresa
- Empleos
- Postulaciones
- Favoritos
- Catálogos
- Datos complementarios del candidato
- Administración
- Configuración
- Notificaciones
- IA Matching
- Moodle

- Migración de contraseñas

11.2. APIs de candidato

Método	Ruta	Descripción funcional
POST	`/api/candidato/AgregarCandidato`	Registro de candidato.
POST	`/api/candidato/login`	Inicio de sesión y entrega de JWT.
POST	`/api/candidato/ActualizarFotoCandidato/{IdCandidato}`	Carga y actualización de foto.
GET	`/api/candidato/ObtenerPorId/{id}`	Consulta integral del perfil.
POST	`/api/candidato/ActualizarCandidato`	Actualiza datos del candidato.
GET	`/api/candidato/ObtenerHabilidades/{id}`	Obtiene habilidades.
POST	`/api/candidato/VerificarCorreoExistente`	Valida correo repetido.
POST	`/api/candidato/ActualizarHabilidades`	Actualiza habilidades.
POST	`/api/candidato/ActualizarCV/{IdCandidato}`	Carga CV principal.
GET	`/api/candidato/ObtenerCertificados/{IdCandidato}`	Lista certificados.
POST	`/api/candidato/AgregarCertificado/{IdCandidato}`	Carga certificado.
POST	`/api/candidato/EliminarCertificado/{IdCandidato}/{IdCertificado}`	Elimina certificado.
POST	`/api/candidato/ActualizarContrasenia`	Cambio de contraseña.
GET	`/api/candidato/ProxyImagen`	Proxy de imágenes remotas.

11.3. APIs de empresa

Método	Ruta	Descripción funcional
GET	`/api/empresa/ObtenerEmpresasActivas`	Lista publica de empresas activas.
GET	`/api/empresa/ObtenerPorId/{id}`	Consulta de empresa por identificador.
POST	`/api/empresa/AgregarEmpresa`	Registro de empresa.
POST	`/api/empresa/login`	Login de empresa y JWT.
POST	`/api/empresa/ActualizarEmpresa`	Actualiza perfil empresarial.
POST	`/api/empresa/CambiarEstadoEmpresa/{id}/{estado}`	Cambia estado de empresa.
GET	`/api/empresa/ObtenerDetallesEmpresa/{id}`	Vista detallada de empresa.
POST	`/api/empresa/VerificarCorreoExistente`	Valida correo.
POST	`/api/empresa/ActualizarContrasenia`	Cambio de contraseña empresarial.

11.4. APIs de empleos

Método	Ruta	Descripción funcional
POST	`/api/empleos/AgregarEmpleo`	Publica vacante y encola IA.
GET	`/api/empleos/ObtenerEmpleosActivosPorEmpresa/{IdEmpresa}`	Empleos activos por empresa.
GET	`/api/empleos/ObtenerTodosPorEmpresa/{IdEmpresa}`	Todos los empleos de una empresa.
GET	`/api/empleos/ObtenerPorId/{id}`	Consulta base de vacante.
POST	`/api/empleos/ActualizarEmpleo`	Edita vacante y reindexa IA.
POST	`/api/empleos/DesactivarEmpleo/{IdEmpleo}/{IdMotivo}`	Baja lógica de vacante.
GET	`/api/empleos/ObtenerEmpleosActivos`	Lista publica de vacantes activas.
GET	`/api/empleos/BuscarEmpleos`	Motor de búsqueda con filtros.
GET	`/api/empleos/ObtenerDetallesEmpleo/{id}`	Vista enriquecida de empleo.

11.5. APIs de postulaciones

Método	Ruta	Descripción funcional
POST	`/api/postulaciones/AgregarPostulacion`	Registra una postulación y dispara procesamiento IA best effort.
POST	`/api/postulaciones/VerificarPostulacionExistente`	Valida duplicidad de postulación.
GET	`/api/postulaciones/VerPostulacionesPorCandidato/{id}`	Historial del candidato.
GET	`/api/postulaciones/VerPostulacionesPorEmpleo/{idEmpleo}/{idEmpresa}`	Postulaciones por vacante.
POST	`/api/postulaciones/ActualizarEstadoPostulacion`	Cambio de estado.
GET	`/api/postulaciones/NumPostulacionsYPorEmpresa/{idEmpresa}`	Indicadores para empresa.
GET	`/api/postulaciones/NumPostulacionsYPorCandidato/{idCandidato}`	Indicadores para candidato.
GET	`/api/postulaciones/VerUltimasPostulacionesPorEmpresa/{idEmpresa}`	Ultimas postulaciones para panel empresa.
GET	`/api/postulaciones/VerUltimasPostulacionesPorCandidato/{idCandidato}`	Ultimas postulaciones para panel candidato.

11.6. APIs de apoyo al perfil candidato

Dominio	Rutas base	Operación
Datos académicos	`/api/datosacadÃ¯micos/*`	Agregar, actualizar, consultar y eliminar.
Experiencia laboral	`/api/experiencialaboral/*`	Agregar, actualizar, consultar y eliminar.
Referencia laboral	`/api/referencialaboral/*`	Agregar, actualizar, consultar y eliminar.
Idiomas	`/api/idiomaspostulantes/*`	Gestión de idiomas.
Intereses	`/api/interesespostulantes/*`	Gestión de intereses y sugerencias para agregar.
Favoritos	`/api/favoritos/*`	Guardado y eliminación de favoritas.

11.7. APIs de catálogos

El dominio `/api/catálogos` entrega catálogos maestros para formularios y filtros:

- categorías de empleo,
- nivel laboral,
- modalidad de trabajo,
- jornada,
- provincias,
- ciudades por provincia,
- motivos de eliminación,
- tipo de identificación,
- género,
- estado civil,
- nacionalidad,
- nivel de estudio,
- estados académicos,
- relaciones laborales,
- idiomas,
- niveles de idioma,
- FAQ,
- sectores de Quito,
- sectores de empresas.

11.8. APIs administrativas

Método	Ruta	Descripción funcional
POST	`/api/admin/login`	Login administrador.
POST	`/api/admin/crear`	Creación de administrador.
GET	`/api/admin/ObtenerPorId/{id}`	Consulta administrador.
GET	`/api/admin/estadÃŋsticas`	KPIs globales.
GET	`/api/admin/candidatos`	Vista administrativa de candidatos.
GET	`/api/admin/empresas`	Vista administrativa de empresas.
GET	`/api/admin/empleos`	Vista administrativa de empleos.
GET	`/api/admin/postulaciones`	Vista administrativa de postulaciones.
GET	`/api/admin/auditorias`	Registro de accesos y auditoria.
GET	`/api/admin/analÃŋtica`	Series y agregados para dashboard.
POST	`/api/admin/candidato/estado`	Bloqueo o cambio de estado de candidato.
POST	`/api/admin/empresa/estado`	Activación o desactivación de empresa.
POST	`/api/admin/candidato/password`	Forzado de password candidato.
POST	`/api/admin/empresa/password`	Forzado de password empresa.

11.9. APIs de configuración y notificaciones

El dominio `/api/admin/configuración` expone lectura integral y actualización dinámica. El controlador `API_EMAIL_Controller` expone notificaciones para postulación, recuperación de contraseña y contacto.

11.10. APIs de IA

Método	Ruta	Descripción funcional
POST	`/api/ai-matching/analizar-empleo/{idEmpleo}`	Genera o regenera índice IA del empleo.
POST	`/api/ai-matching/analizar-candidato/{idCandidato}`	Genera o regenera índice IA del candidato.
POST	`/api/ai-matching/procesar-postulacion/{idEmpleo}/{idCandidato}`	Calcula match y persiste score.
GET	`/api/ai-matching/ranking-empleo/{idEmpleo}`	Obtiene ranking precomputado.
GET	`/api/ai-matching/mÃľtricas-empleo/{idEmpleo}`	KPIs del matching por empleo.
GET	`/api/ai-matching/cola`	Estado de cola interna.
GET	`/api/ai-matching/diagnostico?live=false|true`	Diagnostico de Gemini, OCR y cola.
GET	`/api/ai-matching/documentos-candidato/{idCandidato}`	Extracciones documentales del candidato.

11.11. APIs de Moodle

Método	Ruta	Descripción funcional
GET	`/api/moodle/diagnostico`	Verifica configuración disponible.
POST	`/api/moodle/oauth-url`	Construye URL de autenticación OAuth.
POST	`/api/moodle/oauth-exchange`	Intercambia el code OAuth por token y userinfo.
POST	`/api/moodle/vincular`	Busca y valida usuario Moodle por identificador.
POST	`/api/moodle/sincronizar`	Sincroniza cursos, badges y certificados.
GET	`/api/moodle/perfil-acadÃ¯mico/{idCandidato}`	Consulta perfil académico.
POST	`/api/moodle/importar-certificados`	Importa certificados al perfil del candidato.

12. Arquitectura del frontend

12.1. Patrón general

La aplicación es una SPA. El componente raíz define rutas públicas, rutas de dashboard empresa, redirecciones heredadas y pantallas de autenticación.

12.2. Servicios frontend

El frontend encapsula casi toda la comunicación HTTP en `src/services`.

Servicio	Responsabilidad principal
`apiClient.js`	Cliente Axios, timeout, encabezados, JWT, formateo de errores y logout por bloqueo.
`AuthService.jsx`	Login, registro, recuperación de contraseña, persistencia de sesión.
`CandidatoService.js`	CRUD del candidato, CV, foto, habilidades y certificados.
`EmpresasService.js`	Operaciones empresariales.
`EmpleosService.js`	Listado, detalle, búsqueda y gestión de vacantes.
`PostulacionesService.js`	Alta y consulta de postulaciones.
`CatalogosService.js`	Catálogos maestros.
`AcademicosService.js`	Formación del candidato.
`ExperienciaService.js`	Experiencias del candidato.
`LaboralService.js`	Referencias del candidato.
`IdiomasService.js`	Idiomas del candidato.
`PostulanteService.js`	Intereses del candidato.
`FavoritosService.js`	Empleos favoritos.
`AdminService.js`	Dashboard y gestión administrativa.
`AiMatchingService.js`	Consumo del módulo IA.

`MoodleService.js`	Diagnostico, OAuth, vinculación, sincronización e importación Moodle.

12.3. Ruteo principal

Las rutas principales declaradas en `src/jsx/root/root.jsx` son:

Ruta	**Pantalla**	**Observaciones**
`/`	Homepage1	Landing principal.
`/jobs-profile`	Jobprofile	Perfil rápido del candidato.
`/jobs-my-resume`	Jobmyresume	Pantalla central de gestión curricular.
`/jobs-applied-job`	Jobsappliedjob	Historial de postulaciones.
`/jobs-saved-jobs`	Jobsavedjobs	Vacantes favoritas.
`/jobs-cv-manager`	Jobcvmanager	Gestión de CV.
`/jobs-change-password`	Changepasswordpage	Cambio de contraseña candidato.
`/candidate/choose-profile`	CandidateChooseProfile	Selección de tipo de perfil.
`/moodle/callback`	MoodleOAuthCallback	Retorno del flujo OAuth.
`/company-profile/\{id\}`	CompanyProfileView	Vista publica de empresa.
`/browse-candidates`	Browsecandidates	Vista de candidatos para empresa.
`/create-account`	create-account	Selector de cuenta tipo candidato.
`/account-fresher`	CandidateAccountSetup	Alta de perfil sin experiencia.
`/account-professional`	CandidateAccountSetup	Alta de perfil profesional.
`/about-us`	Aboutus	página informativa.
`/job-detail/\{id\}`	Jobdetail	Detalle de vacante.
`/companies`	Companies	Listado público de empresas.
`/browse-job-list`	Browsejoblist	Búsqueda y listado de vacantes.
`/faq`	Faq	Preguntas frecuentes.
`/login`	Login	Login general.
`/register`	Register	Registro.
`/admin/login`	Login	Login administrador con parámetros.
`/admin/dashboard`	AdminDashboard	Panel administrativo integral.
`/company/dashboard`	CompanyDashboardHome	Panel empresa.
`/company/profile`	Companyprofile	Perfil empresa.
`/company/post-job`	CompanyPostJobs	Publicación de empleo.
`/company/edit-job/\{id\}`	CompanyPostJobs	Edición usando el mismo formulario.
`/company/manage-jobs`	CompanyManageJobs	Gestion de vacantes.
`/company/candidate-profile/\{candidateId\}`	CandidateProfileView	Perfil detallado del candidato para empresa.
`/company/change-password`	Companyresume	Pantalla de seguridad empresa.

13. Inventario de pantallas

13.1. Pantallas publicas

13.1.1. Homepage

Objetivo: presentar la propuesta de valor, accesos rápidos y navegación inicial.

Fuentes de datos: combina contenido real y contenido heredado del template. Importa `Desirecategorydata` desde `alldata.jsx`.

Captura sugerida: banner principal, buscador inicial, CTA y pie de página.

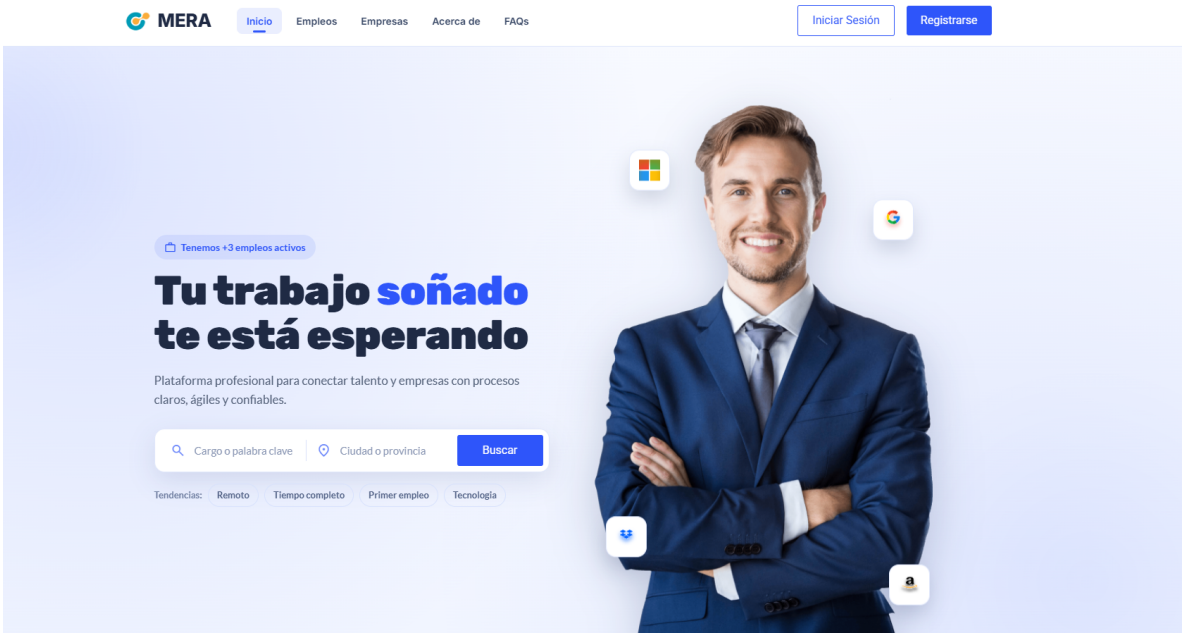

Figura 1: Homepage del sistema

13.1.2. Listado de empleos

Ruta: `/browse-job-list`

Responsabilidad: ejecutar búsqueda paginada con filtros por texto, ubicación, provincia, categoría, nivel, jornada y fecha relativa.

Servicios consumidos:

- `buscarEmpleos`
- `getCategoriasEmpleo`
- `getProvincias`
- `getNivelLaboral`
- `getJornada`

- `agregarPostulacion`
- `getPostulacionesPorCandidato`

Comportamiento importante: si el backend desplegado no tiene `/BuscarEmpleos` y devuelve 404, el frontend aplica **fallback local** filtrando los empleos activos en memoria.

13.1.3. Detalle de empleo

Ruta: `/job-detail/{id}`

Responsabilidad: mostrar información completa de la vacante y permitir acceso a postulación.

13.1.4. Listado de empresas y perfil publico

Rutas: `/companies` y `/company-profile/{id}`

Responsabilidad: exponer empresas activas y su detalle público.

13.1.5. FAQ y Acerca de

Son páginas institucionales o de soporte. `Aboutus.jsx` consume datos de `alldata.jsx`.

13.2. Pantallas de autenticación

13.2.1. Login

Rutas: `/login` y `/admin/login`

Detalles técnicos relevantes:

- El frontend envía payload exacto `{Çorreo": "...", "Password": "..."}`.
- Hay funciones separadas para login de candidato, empresa y admin.
- El token se almacena en navegador y luego se reinyecta automáticamente en Axios.

13.2.2. Registro

Ruta: `/register`

Observación: el frontend arma payloads mínimos y completa ciertos campos con defaults para satisfacer al backend.

13.3. Pantallas de candidato

13.3.1. CandidateChooseProfile y CandidateAccountSetup

Permiten seleccionar si el usuario es *fresher* o *professional*. Son parte del onboarding.

13.3.2. Jobmyresume

Ruta: `/jobs-my-resume`

Importancia: es una de las pantallas más complejas del sistema. Funciona como consola integral de perfil profesional.

Módulos gestionados:

- Datos básicos
- Resumen del perfil con editor rico
- Habilidades
- Educación
- Experiencia laboral
- Referencias
- Idiomas
- Intereses
- CV principal
- Certificados
- Foto de perfil
- Integración Moodle
- Generación de CV en PDF

Aspectos técnicos destacados:

- Usa React Quill para resumen profesional.
- Genera PDF mediante `generateCvPdf`.
- Soporta carga de foto y recarga de sesión para refrescar el header.
- Administra certificados con restricciones de tipo MIME, tamaño y cantidad.
- Integra flujo OAuth de Moodle y restauración de sesión post callback.
- Puede importar certificados y sincronizar perfil académico.

13.3.3. Otras pantallas de candidato

- `Jobprofile`: resumen del perfil.
- `Jobsappliedjob`: historial de postulaciones.
- `Jobsavedjobs`: vacantes favoritas.
- `Jobcvmanager`: gestión de CV.
- `Changepasswordpage`: cambio de contraseña.
- `MoodleOAuthCallback`: retorno del flujo OAuth.

13.4. Pantallas de empresa

- `CompanyDashboardHome`: panel principal de empresa.
- `Companyprofile`: mantenimiento del perfil empresarial.
- `CompanyPostJobs`: alta y edición de vacantes.
- `CompanyManageJobs`: administración de vacantes.
- `CandidateProfileView`: revisión de perfil candidato para empresa.
- `Companyresume`: seguridad o cambio de contraseña.

13.5. Pantallas administrativas

13.5.1. AdminDashboard

Ruta: `/admin/dashboard`

Tabs implementados:

- Panel general
- Candidatos
- Empresas
- Empleos
- Postulaciones
- Auditoría
- Gestion candidatos
- Gestion empresas
- Configuración

Fuentes de datos:

- `getEstadisticas`
- `getCandidatos`
- `getEmpresas`
- `getEmpleos`
- `getPostulaciones`
- `getAuditorias`
- `getAnalitica`

Visualizaciones: KPIs, line charts, área charts, pie y donut charts, tablas filtrables y paginadas.

14. Motor de búsqueda de empleos

14.1. Arquitectura funcional

El motor de búsqueda se construye sobre dos niveles:

1. Búsqueda primaria en backend mediante `GET /api/empleos/BuscarEmpleos`.
2. Fallback local en frontend si el endpoint no existe.

14.2. Parámetros de búsqueda

Los filtros implementados son:

- `texto`
- `ubicación`
- `idProvincia`
- `categoría`
- `nivel`
- `jornada`
- `diasRecientes`
- `page`
- `pageSize`

14.3. Proceso backend

El controlador recibe parámetros por query string, normaliza página y tamaño, delega a `EF_EMPLEOS_Service.BuscarEmpleosActivos` y devuelve total, página, tamaño e items.

14.4. Proceso fallback en frontend

Si el backend responde 404:

1. se descargan todos los empleos activos,
2. se filtra en memoria,
3. se ordena por `FechaInicio` descendente,
4. se página localmente.

14.5. Ventajas y limitaciones

Ventajas: continuidad de servicio y desacople parcial entre despliegues.

Limitaciones: descarga completa de vacantes, escalabilidad limitada y posibles diferencias entre entornos.

15. Motor de matching con IA

15.1. Enfoque arquitectónico

El sistema implementa un **matching precomputado orientado a lectura rápida**. La IA no se ejecuta en cada consulta del empleador, sino en eventos gatillo.

15.2. Eventos que disparan procesamiento

- actualización de perfil del candidato,
- actualización de habilidades,
- carga o actualización de CV,
- alta o baja de certificados,
- actualización de empleo,
- creación de postulación.

15.3. Tablas auxiliares del módulo IA

Tabla	Propósito
AI_JOB_ANALYSIS	Perfil semántico extraído de una vacante.
AI_CANDIDATE_ANALYSIS	Perfil semántico extraído de un candidato.
MATCH_EMPLEO_CANDIDATO	Score final y desagregado por empleo-candidato.
CANDIDATE_EXTRACT	Cache de texto extraído desde CV y certificados.

15.4. Flujo completo del matching

1. Se construye una fuente textual del empleo.
2. Se construye una fuente textual del candidato.
3. Cada fuente se hashifica con SHA-256.
4. Si el hash no cambio, se reutiliza el análisis previo.
5. Si cambio, se llama a Gemini o se usa fallback heurístico.
6. Se guarda el índice estructurado.
7. Para una postulación, se combinan ambos índices y se calcula un score compuesto.
8. El resultado se persiste y luego se consulta como ranking precomputado.

15.5. Uso de Gemini y fallback

El servicio `AI_GEMINI_MVP_Service` construye prompts JSON estrictos, invoca `generateContent`, exige JSON valido y parsea campos normalizados. Si no existe API key o la llamada falla, activa fallback heurístico sin interrumpir el sistema.

15.6. Scoring y OCR

El score total, en versión `score-v3`, combina skills, experiencia, certificaciones, idiomas, ubicación, educación y keywords. También aplica reglas críticas, reglas bloqueantes, relajación para internship y factor de confianza.

Si existe `OCR_SPACE_API_KEY`, el sistema intenta extraer texto del CV y certificados y lo cachea en `AI_CANDIDATE_DOCUMENT_EXTRACT`. Si falla OCR, el proceso continuo.

15.7. Ventaja operacional

La empresa consulta un ranking ya materializado. Esto reduce latencia en lectura comparado con un matching calculado en tiempo real.

16. Integración con Moodle

16.1. Objetivo

Enriquecer el perfil del candidato con evidencia académica externa: usuario Moodle, cursos, badges y certificados.

16.2. Modos soportados

Modo	Descripción
`api`	Consume funciones REST de Moodle con token y nombres de funciones web service.
`mysql`	Consulta la base Moodle directamente por medio de `DbProviderFactories` y proveedor MySQL.

16.3. Flujo OAuth

1. El frontend solicita `/api/moodle/oauth-url`.
2. El backend construye `state` con id de candidato y returnUrl.
3. El usuario es redirigido al autorizador de Moodle.
4. Moodle devuelve `code` y `state` a `/moodle/callback`.
5. El frontend llama a `/api/moodle/oauth-exchange`.
6. El backend intercambia el code por token y consulta `userinfo` si existe.
7. El frontend restaura la sesión Moodle y llama a `/api/moodle/sincronizar`.

16.4. Sincronización académica

En modo API se consumen funciones como:

- `core_user_get_users_by_field`
- `core_enrol_get_users_courses`
- `core_completion_get_course_completion_status`
- `core_badges_get_user_badges`
- función configurada en `MOODLE_CUSTOMCERT_FUNCTION`

En modo MySQL se consultan tablas `mdl_user`, `mdl_user_enrolments`, `mdl_enrol`, `mdl_course`, `mdl_course_completions`, `mdl_badge_issued`, `mdl_badge`, `mdl_customcert_issues` y `mdl_customcert`

16.5. Integración en la UI

La integración se expone principalmente dentro de `Jobmyresume`. Esto es correcto porque Moodle se usa como enriquecimiento del expediente del candidato.

17. Base de datos

17.1. Visión general

La base de datos principal se implementa en SQL Server. Los scripts analizados muestran una evolución incremental sobre una base inicial del sistema y posteriores extensiones de seguridad, administración, auditoria, sectores, certificados e IA.

17.2. Tablas principales detectadas

Tabla	**Dominio**
EF_CANDIDATO	Maestro de candidatos.
EF_EMPRESAS	Maestro de empresas.
EF_EMPLEOS	Vacantes publicadas.
EF_POSTULACION	Relación candidato-empleo.
EF_DATOS_ACADEMICOS	Formación académica.
EF_EXPERIENCIA_LABORAL	Historial laboral.
EF_REFERENCIAS_LABORA	Referencias profesionales.
EF_IDIOMAS_POSTULANTE	Idiomas del candidato.
EF_FAVORITOS	Vacantes favoritas.
EF_INTERESES	Catálogo de intereses.
EF_INTERESES_CANDIDATOS	Relación candidato-interés.
EF_CANDIDATO_CERTIFICADO	Certificados del candidato.
EF_CORREO_CONTACTANOS	Bandeja de contacto.
EF_AUDITORIA_ACCESO	Registro de accesos.
EF_ADMINISTRADORES	Usuarios administradores.
EF_SISTEMA_CONFIGURACION	Parámetros editables.
AI_JOB_ANALYSIS_INDEX	Índice IA de empleos.
AI_CANDIDATE_ANALYSIS	Índice IA de candidatos.
MATCH_EMPLEO_CANDIDATO	Scores IA por par empleo-candidato.
AI_CANDIDATE	Cache OCR/documental.

17.3. Catálogos principales

- EF_CAT_CATEGORIA_EMPLEOS
- EF_CAT_CIUDAD

- EF_CAT_ESTADO_CIVIL
- EF_CAT_ESTADO_EMPLEO
- EF_CAT_ESTADO_POSTULACION
- EF_CAT_GENERO
- EF_CAT_IDIOMAS
- EF_CAT_JORNADA
- EF_CAT_MODALIDAD_TRABAJO
- EF_CAT_MOTIVO_ELIMINACION
- EF_CAT_NACIONALIDAD
- EF_CAT_NIVEL_ACADEMICO
- EF_CAT_NIVEL_ESTUDIO
- EF_CAT_NIVEL_IDIOMA
- EF_CAT_NIVEL_LABORAL
- EF_CAT_PROVINCIAS
- EF_CAT_TIPO_IDENTIFICACION
- EF_CAT_SECTORES_QUITO
- EF_CAT_SECTORES_EMPRESA

17.4. Procedimientos y trigger detectados

- ef_sp_GuardarContraseniaTemporalCandidato
- ef_sp_GuardarContraseniaTemporalEmpresa
- ef_sp_ActualizarContraseniaCandidato
- ef_sp_ActualizarContraseniaEmpresa
- ef_sp_ObtenerCertificadosPorCandidato
- ef_sp_AgregarCertificadoCandidato
- ef_sp_EliminarCertificadoCandidato
- EF_CORREO_CONTACTANOS_AIU

17.5. Campos de seguridad y estado agregados

Los scripts complementarios agregan, entre otros:

- `DebeCambiarContrasenia` en candidato y empresa.
- `IsBloqueado` en candidato.
- `IdSectorQuito` en candidato.

18. Procesos de negocio principales

18.1. Registro de candidato

1. El usuario envía formulario desde frontend.
2. El frontend transforma a payload mínimo aceptado.
3. El backend crea el registro en `EF_CANDIDATO`.
4. El usuario puede completar después el expediente en `Jobmyresume`.

18.2. Registro de empresa

1. Se envía payload con datos base de empresa.
2. Se crea la empresa activa por defecto.
3. Posteriormente puede ampliar descripción, datos de contacto y publicar empleos.

18.3. Publicación de empleo

1. La empresa autenticada envía una vacante.
2. El backend la marca activa.
3. Se guarda en `EF_EMPLEOS`.
4. Se encola reindexación IA del empleo.

18.4. Postulación a empleo

1. El candidato navega el listado de empleos.
2. El frontend valida sesión y rol.
3. Se confirma postulación en modal.
4. Se envía `AgregarPostulacion`.
5. El backend registra la postulación.
6. Best effort: se calcula o reevalúa el match IA.

18.5. Actualización del perfil del candidato

1. El usuario edita datos personales, educación, experiencia o habilidades.
2. El frontend consume servicios especializados por modulo.
3. El backend persiste los cambios.
4. Si el cambio afecta el perfil profesional, se encola reanálisis IA.

18.6. Sincronización con Moodle

1. El candidato vincula su cuenta Moodle por identificador o OAuth.
2. El backend localiza al usuario Moodle.
3. Se consultan cursos, badges y certificados.
4. El frontend presenta el perfil académico.
5. Opcionalmente se importan certificados al expediente del candidato.

19. Dashboard administrativo, auditoria y seguridad

19.1. KPIs principales

El dashboard administra indicadores como total de candidatos, empresas, empleos, postulaciones, empleos vencidos, tendencias temporales, top empleos, top empresas, distribución por categoría y logins exitosos/fallidos.

19.2. Auditoría de acceso

La tabla `EF_AUDITORIA_ACCESO` almacena fecha, IP, navegador, detalle y estado exitoso o fallido.

19.3. Fortalezas

- JWT con validación de issuer, Audience, firma y expiracion.
- Validación de roles en endpoints protegidos.
- Bloqueo dinámico de candidatos y empresas.
- Separación razonable por capas.
- Fallback seguro ante indisponibilidad de IA.

19.4. Debilidades detectadas

- CORS totalmente abierto.
- Credenciales sensibles visibles en archivos de configuración.
- Dependencia de contenido mock en ciertas vistas.
- Fallback local de búsqueda con escalabilidad limitada.
- No se evidencian pruebas automatizadas en el recorte analizado.

20. Guía para inserción de capturas

Para convertir este documento en memoria final, se recomienda insertar capturas en:

1. Landing page
2. Login y registro
3. Listado de empleos con filtros
4. Detalle de empleo
5. Perfil candidato
6. CV y certificados
7. Integración Moodle
8. Dashboard empresa
9. Publicación de empleo
10. Gestion de empleos
11. Vista de candidato para empresa
12. Dashboard admin general
13. Dashboard admin candidatos
14. Dashboard admin empresas
15. Dashboard admin empleos
16. Dashboard admin postulaciones
17. Dashboard admin auditoria

```
\begin{figure}[H]
\centering
\includegraphics[width=\textwidth]{capturas/browse-job-list.png}
\caption{Pantalla de listado de empleos con filtros}
\end{figure}
```

21. Conclusiones

El Proyecto MERA posee una arquitectura funcional completa y con madurez intermedia-avanzada en los dominios centrales. Destacan la separación por capas del backend, la cobertura funcional del expediente del candidato, la gestión de vacantes y postulaciones, la analítica administrativa, la integración Moodle y el módulo de matching IA con estrategia de precomputo y fallback.

A. Anexo A: Inventario resumido de pantallas

Pantalla	**Tipo de usuario**	**Propósito**
Homepage1	Público	Landing principal.
Browsejoblist	Público/Candidato	Búsqueda de empleos.
Jobdetail	Público/Candidato	Detalle de vacante.
Companies	Público	Listado de empresas.
CompanyProfileView	Público	Perfil público de empresa.
Faq	Público	FAQ institucional.
Aboutus	Público	Información institucional.
Login	Todos	Autenticación.
Register	Público	Registro.
CandidateChooseProfile	Candidato	Selección de tipo de perfil.
CandidateAccountSetup	Candidato	Onboarding de perfil.
Jobprofile	Candidato	Resumen del perfil.
Jobmyresume	Candidato	Gestion integral del expediente profesional.
Jobsappliedjob	Candidato	Historial de postulaciones.
Jobsavedjobs	Candidato	Favoritos.
Jobcvmanager	Candidato	Gestion de CV.
Changepasswordpage	Candidato	Cambio de contraseña.
MoodleOAuthCallback	Candidato	Retorno del flujo OAuth.
CompanyDashboardHome	Empresa	Panel principal.
Companyprofile	Empresa	Perfil empresarial.
CompanyPostJobs	Empresa	Alta/edición de vacantes.
CompanyManageJobs	Empresa	Gestion de vacantes.
CandidateProfileView	Empresa	Revisión de perfil candidato.
Companyresume	Empresa	Seguridad/cambio de contraseña.
AdminDashboard	Administrador	Analítica y gestión integral.
AdminManageCandidates	Administrador	Gestion especifica de candidatos.
AdminManageCompanies	Administrador	Gestion especifica de empresas.
AdminSettings	Administrador	Parámetros del sistema.

B. Anexo B: Objetos SQL relevantes

Objeto	Clasificación
EF_CANDIDATO	Tabla transaccional
EF_EMPRESAS	Tabla transaccional
EF_EMPLEOS	Tabla transaccional
EF_POSTULACION	Tabla transaccional
EF_DATOS_ACADEMICOS	Tabla transaccional
EF_EXPERIENCIA_LABORAL	Tabla transaccional
EF_REFERENCIAS_LABORA	Tabla transaccional
EF_IDIOMAS_POSTULANTE	Tabla transaccional
EF_INTERESES	Catálogo
EF_INTERESES_CANDIDATOS	Tabla relacional
EF_FAVORITOS	Tabla transaccional
EF_CANDIDATO_CERTIFICADO	Tabla transaccional
EF_CORREO_CONTACTANOS	Tabla operativa
EF_AUDITORIA_ACCESO	Tabla de auditoria
EF_USUARIOS_ADMINISTRADORES	Seguridad/administración
EF_SISTEMA_CONFIGURACION	Parametrización
AI_JOB_ANALYSIS_INDEX	IA
AI_CANDIDATE_ANALYSIS_INDEX	IA
AI_MATCH_EMPLEO_CANDIDATO	IA
AI_CANDIDATE_DOCUMENT_EXTRACT	IA/OCR

C. Anexo C: Rutas principales del frontend

Ruta	Componente
/	Homepage1
/browse-job-list	Browsejoblist
/job-detail/\{id\}	Jobdetail
/companies	Companies
/company-profile/\{id\}	CompanyProfileView
/faq	Faq
/about-us	Aboutus
/login	Login
/register	Register
/jobs-profile	Jobprofile
/jobs-my-resume	Jobmyresume
/jobs-applied-job	Jobsappliedjob
/jobs-saved-jobs	Jobsavedjobs

/jobs-cv-manager	Jobcvmanager
/jobs-change-password	Changepasswordpage
/candidate/choose-profile	CandidateChooseProfile
/moodle/callback	MoodleOAuthCallback
/company/dashboard	CompanyDashboardHome
/company/profile	Companyprofile
/company/post-job	CompanyPostJobs
/company/edit-job/\{id\}	CompanyPostJobs
/company/manage-jobs	CompanyManageJobs
/company/candidate-profile/ \{candidateId\}	CandidateProfileView
/company/change-password	Companyresume
/admin/login	Login
/admin/dashboard	AdminDashboard