

Plantilla de Documento de Arquitectura de Software (SAD)

Esta plantilla está diseñada para documentar la arquitectura de software de la Bolsa de Empleados Conjunta – Cámara de Industrias de Loja, siguiendo un enfoque estructurado inspirado en vistas arquitectónicas estándar (por ejemplo, el modelo 4+1). Proporciona una visión integral del sistema, capturando decisiones clave, vistas y consideraciones. Personalice las secciones con detalles específicos del proyecto, diagramas y contenido. Utilice UML u otras herramientas de diagramación para representaciones visuales donde sea aplicable.

Aprobador(es) del Documento:

Nombre del Aprobador	Rol
Steven Neira	Software Architect

Revisores del Documento:

Nombre del Revisor	Rol
Steven Neira	Software Architect

Resumen de Cambios:

Versión	Fecha	Creado por	Descripción Corta de los Cambios
2.1	06.01.2026	Jhandry Solorzano	Seguridad
2.1	06.01.2026	Veronica Luna	Diccionario de clases
2.1	06.01.2026	Sofia Vire	Diagrama de clases
2.1	06.01.2026	Nahomi Cabrera	Diagrama DevOps
2.1	06.01.2026	Steven Neira	Glosario Terminos

Tabla de Contenidos

Plantilla de Documento de Arquitectura de Software (SAD)	1
1. INTRODUCCIÓN.....	3
1.1. Propósito	3
1.2. Alcance.....	3
1.3. Referencias	3
1.4. Visión General del Contenido del Documento	4
2. REPRESENTACIÓN ARQUITECTÓNICA.....	6
3. OBJETIVOS Y RESTRICCIONES ARQUITECTÓNICAS.....	11
4. VISTA DE CASOS DE USO	11
4.1. Especificación de casos de uso.....	15
6. VISTA LÓGICA	20
6.1. Visión General	20
6.2. Diagrama de clases	21
6.3. Diccionario de clases.....	21
7. VISTA DE PROCESOS.....	35
7.1. Visión General	35
7.2. Procesos y Flujos Clave (Diagramas de Secuencia).....	35
8. VISTA DE IMPLEMENTACIÓN.....	41
8.1. Visión General	41
8.2. Estructura de componentes de desarrollo.....	43
8.3. Estructura detallada del repositorio	43
8.4. Diagrama de componentes de desarrollo	51
9. VISTA DE DESPLIEGUE.....	54
9.1 Visión general.....	54
9.2 Diagrama de despliegue.....	54
10. VISTA DE DATOS.....	57
10.1. Visión general	57
10.1. Modelo de Datos.....	57
10.2. Diccionarios de Datos	58
11. CALIDAD	60
11.1. Visión general	60
11.2. Atributos de calidad	60
12. CALIDAD	61
12.1. Visión general	63
12.2. Atributos de calidad	65
13. INFORMACIÓN DE CONTACTO	66

1. INTRODUCCIÓN

1.1. Propósito

La aplicación Bolsa de Empleados Conjunta – Cámara de Industrias de Loja es un sistema de intermediación laboral que conecta empresas afiliadas con potenciales empleados, centralizando la gestión de vacantes, postulaciones y validación de datos de manera segura. Su enfoque principal es ofrecer a la Cámara de Industrias de Loja (CAIL) una plataforma tecnológica que mejore la eficiencia y trazabilidad de los procesos de búsqueda, selección y registro de candidatos.

El presente Documento de Arquitectura de Software (Software Architecture Document – SAD) tiene como objetivo sustentar el proceso arquitectónico en el desarrollo de este sistema, capturando las decisiones de diseño más relevantes y las vistas arquitectónicas que permiten comprender la solución propuesta. Está dirigido principalmente a arquitectos de software y desarrolladores, sirviendo como guía de referencia durante las etapas de diseño, implementación, pruebas y evolución del sistema.

La arquitectura se encuentra alineada con la norma ISO/IEC/IEEE 42010 para la descripción de arquitecturas de sistemas y se estructura siguiendo el modelo de vistas 4+1 de Kruchten, complementado con lineamientos de usabilidad, seguridad, calidad de datos y modelado basados en estándares internacionales y regulaciones nacionales aplicables.

1.2. Alcance

La arquitectura descrita en este Documento de Arquitectura de Software (Software Architecture Document – SAD) abarca el backend del sistema, las interfaces de usuario web y móvil, así como las integraciones con sistemas externos necesarios para la validación de datos de empresas y empleados. En particular, se consideran las integraciones con los servicios del Servicio de Rentas Internas (SRI) para la verificación de RUC y vigencia de las empresas, y con el Registro Civil para la validación de la identidad de los empleados.

Quedan explícitamente fuera del alcance de esta arquitectura los procesos que impliquen interacción directa especializada con el empleado, como la aplicación y gestión de pruebas psicológicas, así como los procesos de contratación directa, dado que el sistema actúa únicamente como mediador entre las partes. Tampoco se cubren los procesos internos particulares de las empresas, ni los procesos de contratación de usuarios que no pertenecen a una empresa registrada en la plataforma.

El contexto actual del sistema es el de prototipo desarrollado en el marco de un proyecto académico, con la perspectiva de evolucionar hacia un producto listo para ser adoptado por la Cámara de Industrias de Loja (CAIL) como solución productiva de intermediación laboral.

1.3. Referencias

En esta sección se listan los estándares, lineamientos y artefactos que sirven de base para la definición de la arquitectura del sistema.

#	Documento	Resumen de Contenidos
[REF1]	ISO 9241	Ergonomics of human-system interaction: Lineamientos de experiencia de usuario y diseño de interfaces. Se aplica para asegurar que las opciones de privacidad y seguridad sean claras para el empleado, reduciendo el riesgo de error humano.
[REF2]	ISO/IEC/IEEE 42010	Systems and software engineering: Marco para la descripción de arquitecturas de software y sistemas. Es la base metodológica utilizada para la elaboración y estructura de este documento (SAD).
[REF3]	ISO/IEC 27034	Application Security: Lineamientos para la gestión de seguridad en aplicaciones. Proporciona el marco técnico para implementar los controles de OWASP y asegurar que el código del backend sea resistente a ataques.
[REF4]	ISO 8000	Data quality: Principios para la calidad de datos. Es fundamental para garantizar que el "Desempeño Histórico" y el "Puntaje Global" sean veraces, íntegros y de alta calidad, cumpliendo con la LOPDP.
[REF5]	ISO/IEC 19505	Information technology – Object Management Group Unified Modeling Language (UML):

1.3.2 Marco legal nacional

El tratamiento de la información dentro de la plataforma se rige por el marco jurídico de la República del

Ecuador, el cual impone requisitos técnicos de seguridad y privacidad sobre la arquitectura del sistema:

- **Constitución de la República del Ecuador:** Art. 66, numeral 19, que reconoce y garantiza el derecho a la protección de datos de carácter personal, incluyendo el acceso y la decisión sobre información y datos de este tipo.
- **Ley Orgánica de Protección de Datos Personales (LOPDP):** Publicada en el Registro Oficial en mayo de 2021. Esta ley es el pilar principal para el diseño de la plataforma, especialmente en lo referente a:

- **Principio de Seguridad (Art. 10):** Obliga a implementar medidas técnicas (como el cifrado AES-256 que definimos) para evitar el tratamiento no autorizado.
- **Tratamiento de Datos Sensibles:** Normativa estricta para el manejo de la edad y el desempeño laboral.
- **Derecho a la revisión humana (Art. 20):** Requisito fundamental para el algoritmo de matching, asegurando que ninguna decisión laboral sea 100% automatizada.
- **Código Orgánico de la Economía Social de los Conocimientos (Código Ingenios):** En lo relativo a la protección de bases de datos y propiedad intelectual del software desarrollado.
- **Ley de Comercio Electrónico, Firmas y Mensajes de Datos:** Que otorga validez jurídica a las transacciones y comunicaciones electrónicas realizadas entre las Empresas Emisoras, Receptoras y la plataforma, respaldando el uso de certificados digitales y TLS.

1.3.3 Artefactos del proyecto

Mapa de capacidades del sistema de Bolsa de Empleo de la CAIIL.

Especificación de casos de uso y diagramas de casos de uso.

Diagrama de clases y su especificación asociada.

Diagramas de secuencia (en proceso de revisión), que describen flujos de interacción entre actores y componentes clave del sistema.

1.3.4 Tecnologías y plataformas

Node.js y Fastify para la implementación del backend.

React para la interfaz web.

Firebase Hosting para la publicación de componentes web y/o servicios estáticos.

Android Package Kit (APK) para la distribución de la aplicación móvil.

Docker para contenerización y despliegue.

GitHub y GitHub Actions para control de versiones e integración continua / entrega continua (CI/CD).

Jira para la planificación y seguimiento de tareas.

Discord como canal de comunicación del equipo de desarrollo.

1.4. Visión General del Contenido del Documento

Tras esta introducción, el Documento de Arquitectura de Software (Software Architecture Document – SAD) presenta la representación arquitectónica general del sistema, así como los objetivos y restricciones arquitectónicas que guían las decisiones de diseño. A continuación, se desarrolla la arquitectura mediante diversas vistas alineadas al modelo 4+1 de Kruchten, incluyendo la vista de casos de uso, la vista lógica, la vista de implementación y la vista de despliegue, complementadas con una vista de datos que detalla el modelo de información central del sistema.

Posteriormente, se abordan secciones específicas relativas a seguridad, tamaño y rendimiento, y atributos de calidad como extensibilidad, fiabilidad y portabilidad, junto con aspectos de logging, cacheo y, en caso de requerirse, soporte para escenarios de multicliente o multidominio. Estas secciones permiten comprender cómo la arquitectura propuesta busca satisfacer los requisitos funcionales y no funcionales, garantizando un diseño coherente con los estándares de referencia y las necesidades de la Cámara de Industrias de Loja (CAIL).

1.5. Glosario De Términos

A

Término	Definición
API	Application Programming Interface (Interfaz de Programación de Aplicaciones). Conjunto de protocolos y herramientas que permiten la comunicación entre diferentes componentes de software.
APK	Android Package Kit. Formato de archivo utilizado para distribuir e instalar aplicaciones en dispositivos Android.
Argon2	Algoritmo de hashing de contraseñas diseñado para ser resistente a ataques de fuerza bruta y ataques mediante hardware especializado.

B

Término	Definición
Backend	Capa del sistema que gestiona la lógica de negocio, el procesamiento de datos y la comunicación con bases de datos y servicios externos.

Bcrypt	Función criptográfica de hash utilizada para proteger contraseñas mediante la adición de salt aleatorio.
---------------	--

C

Término	Definición
CAIL	Cámara de Industrias de Loja. Entidad que representa a las empresas industriales afiliadas en la provincia de Loja, Ecuador.
CI/CD	Continuous Integration/Continuous Deployment (Integración Continua/Despliegue Continuo). Prácticas de desarrollo que automatizan la integración de código y el despliegue de aplicaciones.
CRUD	Create, Read, Update, Delete (Crear, Leer, Actualizar, Eliminar). Operaciones básicas de manipulación de datos en sistemas de información.
CVE	Common Vulnerabilities and Exposures (Vulnerabilidades y Exposiciones Comunes). Sistema de identificación de vulnerabilidades de seguridad conocidas públicamente.

D

Término	Definición
DevOps	Development and Operations (Desarrollo y Operaciones). Conjunto de prácticas que combinan el desarrollo de software y las operaciones de TI para acortar el ciclo de vida del desarrollo.
Docker	Plataforma de contenedorización que permite empaquetar aplicaciones y sus dependencias en contenedores virtuales que pueden ejecutarse en cualquier sistema operativo.
DXA	Twentieth of a Point (Ventésima de punto). Unidad de medida utilizada en documentos de Microsoft Word donde 1440 DXA equivalen a 1 pulgada.

E

Término	Definición
ELK Stack	Elasticsearch, Logstash y Kibana. Conjunto de herramientas para búsqueda, análisis y visualización de logs en tiempo real.

Endpoint	Punto de entrada específico en una API REST que permite realizar operaciones sobre recursos del sistema.
-----------------	--

F

Término	Definición
Fastify	Framework web de alto rendimiento para Node.js enfocado en proporcionar la mejor experiencia de desarrollo con mínima sobrecarga.
Firebase	Plataforma de Google que proporciona servicios backend como autenticación, base de datos en tiempo real, almacenamiento y hosting para aplicaciones web y móviles.
Firestore	Base de datos NoSQL orientada a documentos de Firebase que permite sincronización en tiempo real y funciona sin conexión.
Frontend	Capa de presentación del sistema que interactúa directamente con el usuario, incluyendo interfaces gráficas web y móviles.

G-H

Término	Definición
GitHub Actions	Plataforma de automatización integrada en GitHub que permite crear flujos de trabajo para CI/CD, pruebas y despliegues.
HTTPS	HyperText Transfer Protocol Secure. Protocolo de comunicación seguro que cifra las comunicaciones entre cliente y servidor mediante TLS/SSL.

I

Término	Definición
IDOR	Insecure Direct Object Reference (Referencia Directa Insegura a Objetos). Vulnerabilidad de seguridad donde un atacante puede acceder a objetos mediante la modificación de identificadores.
ISO/IEC/IEEE 42010	Estándar internacional que proporciona un marco conceptual para la descripción de arquitecturas de sistemas y software.
ISO 27034	Estándar internacional que proporciona lineamientos para la gestión de seguridad en aplicaciones.

J

Término	Definición
Jest	Framework de pruebas para JavaScript que permite realizar testing unitario, de integración y de snapshot con configuración mínima.
Jira	Herramienta de gestión de proyectos y seguimiento de incidencias utilizada en metodologías ágiles.
JWT	JSON Web Token. Estándar abierto para transmitir información de forma segura entre partes como un objeto JSON, comúnmente usado para autenticación.

L-M

Término	Definición
LOPDP	Ley Orgánica de Protección de Datos Personales de Ecuador. Normativa que regula el tratamiento de datos personales en el país.
Matching	Proceso algorítmico de emparejamiento entre empleados y vacantes laborales basado en competencias, experiencia y requisitos específicos.
MFA	Multi-Factor Authentication (Autenticación Multifactor). Método de seguridad que requiere múltiples formas de verificación de identidad.
Middleware	Componente de software que actúa como intermediario entre diferentes aplicaciones o capas del sistema, procesando peticiones antes de que lleguen a su destino final.

N-O

Término	Definición
Node.js	Entorno de ejecución de JavaScript del lado del servidor basado en el motor V8 de Chrome, diseñado para construir aplicaciones escalables.
NoSQL	Categoría de sistemas de gestión de bases de datos que no utilizan el modelo relacional tradicional, optimizados para escalabilidad y flexibilidad.

OAuth 2.0	Protocolo de autorización que permite a aplicaciones de terceros acceder a recursos de usuario sin exponer credenciales.
ORM	Object-Relational Mapping (Mapeo Objeto-Relacional). Técnica que convierte datos entre sistemas de tipos incompatibles usando programación orientada a objetos.
OWASP	Open Web Application Security Project. Comunidad global enfocada en mejorar la seguridad del software mediante la identificación de las vulnerabilidades más críticas.

R

Término	Definición
RBAC	Role-Based Access Control (Control de Acceso Basado en Roles). Método de seguridad que restringe el acceso al sistema basándose en los roles de los usuarios.
React	Biblioteca de JavaScript desarrollada por Facebook para construir interfaces de usuario mediante componentes reutilizables.
Registro Civil	Organismo público ecuatoriano responsable de registrar hechos vitales y validar la identidad de los ciudadanos mediante cédula de identidad.
REST	Representational State Transfer. Estilo arquitectónico para sistemas distribuidos que utiliza HTTP para operaciones CRUD sobre recursos.
RUC	Registro Único de Contribuyentes. Identificador fiscal de 13 dígitos asignado por el SRI a personas naturales y jurídicas en Ecuador.

S

Término	Definición
SAD	Software Architecture Document (Documento de Arquitectura de Software). Documento que captura las decisiones arquitectónicas clave de un sistema.
SRI	Servicio de Rentas Internas. Organismo técnico autónomo de Ecuador encargado de la gestión tributaria y validación fiscal de empresas.

SSRF	Server-Side Request Forgery (Falsificación de Solicitud del Lado del Servidor). Vulnerabilidad que permite a un atacante hacer que el servidor realice solicitudes no autorizadas.
-------------	--

T-U

Término	Definición
TLS	Transport Layer Security. Protocolo criptográfico que proporciona comunicaciones seguras sobre redes, sucesor de SSL.
UML	Unified Modeling Language (Lenguaje Unificado de Modelado). Lenguaje de modelado estándar para visualizar el diseño de sistemas de software.
UX	User Experience (Experiencia de Usuario). Conjunto de percepciones y respuestas de un usuario derivadas del uso de un producto o servicio.

2. REPRESENTACIÓN ARQUITECTÓNICA

Esta sección describe el enfoque utilizado para documentar la arquitectura de la aplicación Bolsa de Empleados Conjunta – Cámara de Industrias de Loja, siguiendo la norma ISO/IEC/IEEE 42010 y el modelo de vistas 4+1 de Kruchten. La arquitectura se presenta a través de múltiples vistas complementarias que permiten abordar las preocupaciones de diferentes interesados, incluyendo arquitectos de software, desarrolladores y representantes de la Cámara de Industrias de Loja (CAIL).

Las vistas que se utilizan en este Documento de Arquitectura de Software (Software Architecture Document – SAD) son: vista de casos de uso, vista lógica, vista de implementación (desarrollo), vista de despliegue y vista de datos, además de secciones específicas para seguridad, rendimiento, atributos de calidad, logging, cacheo y, de ser necesario, multicliente o multidominio. Cada vista se apoya en diagramas UML según ISO/IEC 19505 y en descripciones textuales que explican los componentes, sus responsabilidades y las decisiones de diseño relevantes para el sistema.

3. OBJETIVOS Y RESTRICCIONES ARQUITECTÓNICAS

Los objetivos arquitectónicos de la aplicación se centran en proporcionar una plataforma de intermediación laboral segura, extensible y preparada para evolucionar desde prototipo académico hacia un producto potencialmente adoptable por la Cámara de Industrias de Loja (CAIL). La arquitectura busca facilitar la integración con sistemas externos críticos (SRI y Registro

Civil), garantizar la calidad y protección de los datos personales de empleados y empresas, y ofrecer una buena experiencia de usuario en interfaces web y móviles.

3.1 Requisitos no funcionales

Requisito no funcional	Descripción
Usabilidad	La aplicación deberá proporcionar interfaces web y móviles claras, consistentes y fáciles de usar, siguiendo principios de diseño centrado en el usuario y las recomendaciones de la norma ISO 9241, de forma que empresas y empleados puedan completar sus tareas con errores mínimos y tiempos de aprendizaje reducidos.
Seguridad de la aplicación	La arquitectura deberá contemplar controles de seguridad alineados con la norma ISO 27034 y con la normativa de protección de datos personales vigente en Ecuador, incluyendo autenticación, autorización, protección de datos en tránsito y en reposo, y registros de auditoría para operaciones sensibles.
Calidad de datos	La información de empresas y empleados deberá cumplir criterios de calidad de datos (completitud, consistencia, exactitud y trazabilidad) de acuerdo con los lineamientos de ISO 8000, apoyándose en validaciones automáticas (por ejemplo, contra SRI y Registro Civil) y reglas de negocio que reduzcan la presencia de datos erróneos o duplicados.
Interoperabilidad	El sistema deberá interoperar de forma fiable con los servicios externos del SRI y del Registro Civil, respetando los formatos de datos, protocolos y políticas de seguridad que estos organismos definan, de modo que las validaciones de RUC, vigencia de empresas e identidad de empleados se integren de manera transparente en los flujos de negocio.
Escalabilidad	La arquitectura deberá poder evolucionar para soportar un aumento en el número de empresas, ofertas y postulaciones, así como mayor tráfico concurrente, mediante técnicas como despliegues escalables (por ejemplo, contenedores Docker) y separación clara de responsabilidades entre backend, frontend y servicios externos.

Rendimiento	Los tiempos de respuesta percibidos por los usuarios deberán ser adecuados para el uso interactivo, particularmente en operaciones frecuentes como búsqueda de ofertas, postulación y revisión de candidatos, evitando bloqueos innecesarios y aplicando optimizaciones donde las integraciones externas añadan latencia.
Disponibilidad	La solución deberá diseñarse para minimizar el impacto de fallos parciales (por ejemplo, caída temporal de un servicio externo), contemplando estrategias de manejo de errores y reintentos, de forma que el sistema pueda seguir brindando funcionalidades esenciales incluso ante incidencias puntuales.
Mantenibilidad	El sistema deberá estar organizado en capas y componentes claramente definidos (presentación, lógica de negocio, acceso a datos, integración con terceros) que faciliten el mantenimiento, la corrección de errores y la incorporación de nuevas funcionalidades sin afectar innecesariamente a otras partes de la aplicación.
Extensibilidad	La arquitectura deberá permitir la incorporación futura de nuevas funcionalidades, tipos de usuarios, integraciones con otros servicios o canales (por ejemplo, nuevos portales de empleo o otros organismos públicos) sin requerir cambios disruptivos en los componentes existentes.
Portabilidad	Siempre que sea posible, se priorizará el uso de tecnologías y servicios portables entre entornos (por ejemplo, contenedores Docker, frameworks multiplataforma) para facilitar el despliegue en diferentes infraestructuras (cloud, on-premise o entornos mixtos) si la CAIIL así lo requiere en el futuro.
Observabilidad y logging	El sistema deberá generar registros (logs) suficientes para monitorear el comportamiento de la aplicación, detectar fallos y auditar operaciones clave, separando eventos técnicos y de negocio, y facilitando el análisis durante soporte y evolución de la solución.

3.2 restricciones arquitectónicas

Restricciones	Descripción
Tecnologías definidas del proyecto	La solución deberá implementarse utilizando las tecnologías acordadas para el proyecto: Node.js y Fastify en el backend, React en el frontend web, distribución móvil mediante APK Android, Firebase Hosting para despliegue web estático o servicios específicos, Docker para contenerización y GitHub/GitHub Actions para control de versiones y CI/CD.
Herramientas de gestión y comunicación	La planificación y seguimiento del trabajo del equipo deberá realizarse en Jira, mientras que la coordinación diaria y comunicación se gestionará mediante Discord, lo que condiciona la forma en que se organizan los entregables y el flujo de trabajo del equipo.
Integración obligatoria con SRI y Registro Civil	La arquitectura deberá adaptarse a las APIs, protocolos de seguridad y restricciones operativas que definen el Servicio de Rentas Internas (SRI) y el Registro Civil, lo que puede imposibilitar el uso de ciertas tecnologías o patrones de integración y exige cumplir sus requisitos de autenticación, autorización y disponibilidad.
Cumplimiento legal en Ecuador	El tratamiento de datos personales de empleados y empresas deberá ajustarse a la Constitución y a la legislación nacional sobre protección de datos, lo que limita el almacenamiento, procesamiento y exposición de información sensible e impone requisitos específicos de confidencialidad, consentimiento y auditoría.
Contexto de prototipo académico	En la fase actual, el sistema se desarrolla como prototipo en el marco de un proyecto académico, lo que implica restricciones de tiempo, recursos y alcance, si bien la arquitectura debe prever su posible evolución a un producto listo para uso productivo por parte de la CAIIL.

4. VISTA DE CASOS DE USO

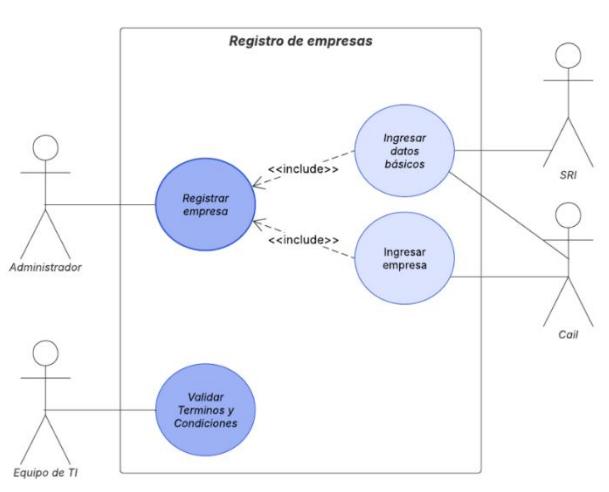
4.1. Especificación de casos de uso

A continuación, se describen los casos de uso considerados arquitectónicamente significativos para el sistema:

CU Registro de empresas (Figura 1)

Este caso de uso permite registrar una nueva empresa afiliada a la CAIIL para que pueda participar como emisora y/o receptora de empleados, integrando la validación del RUC con el Servicio de Rentas Internas (SRI) y generando las credenciales de acceso iniciales. Impacta directamente en la vista de datos (creación de entidades Empresa), en la vista de implementación (servicios de integración con SRI) y en la vista de despliegue (conectividad con servicios externos).

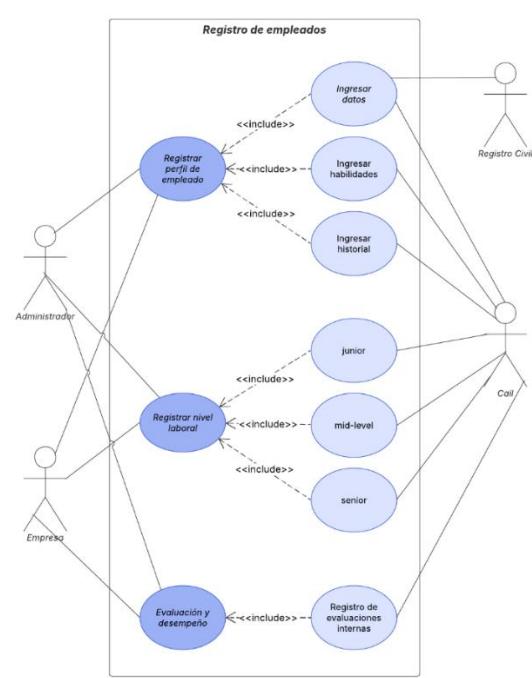
*Figura 1
CU Registro de empresas*



CU Registro de empleados (Figura 2)

Este caso de uso gestiona el registro de empleados en la bolsa de empleo, incluyendo la validación de identidad con el Registro Civil, el registro de historial laboral y competencias, y la preparación de la información necesaria para evaluación y publicación. Involucra datos personales sensibles, dependencias con servicios externos, generación de estados iniciales del empleado (por ejemplo, En Revisión) y actualización del modelo de datos y servicios de negocio asociados.

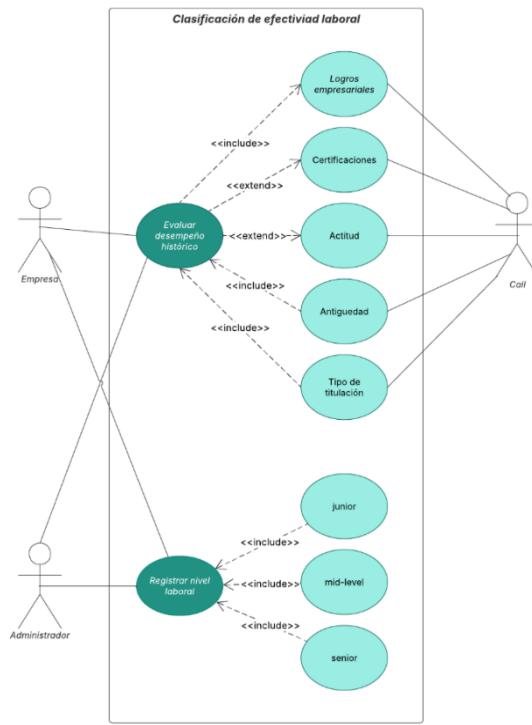
*Figura 2
CU Registro de empleados*



CU Clasificación de efectividad laboral (Figura 3)

Agrupa los casos de Evaluar desempeño histórico, Registrar nivel laboral, Consolidar resultados y Asignar puntaje global de efectividad, con el objetivo de calcular un puntaje 0–100 y un nivel laboral (Junior, Mid-Level, Senior) basado en criterios ponderados como logros, certificaciones, actitud, antigüedad y titulación. Este proceso alimenta el motor de matching, impacta en la vista lógica (módulos de evaluación y reportes), en la vista de datos (historial de evaluaciones e informes de ponderación) y en decisiones de rendimiento si los cálculos se realizan sobre volúmenes grandes de empleados.

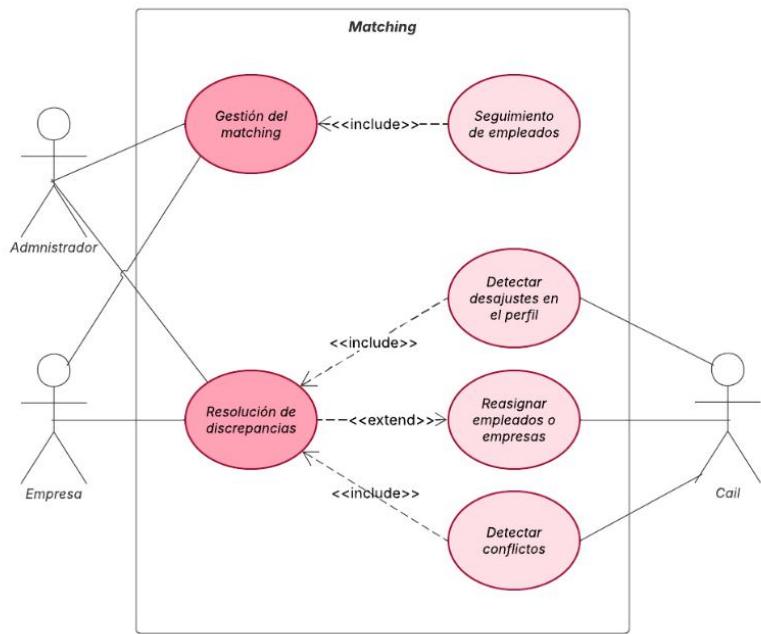
*Figura 3
CU Clasificación de efectividad laboral*



CU de Matching (Figura 4)

Corresponde a la gestión y ejecución del algoritmo de matching entre empleados disponibles y vacantes activas, incluyendo la generación automática de coincidencias, el cálculo de scores mínimos (por ejemplo, ≥ 60) y la resolución de discrepancias cuando se detectan conflictos o desajustes. Este caso de uso exige un diseño cuidadoso de los servicios de negocio y del modelo de datos para soportar búsquedas eficientes, ordenamiento por score y ejecución periódica o bajo demanda, afectando también a la vista de despliegue si se requieren estrategias de escalabilidad.

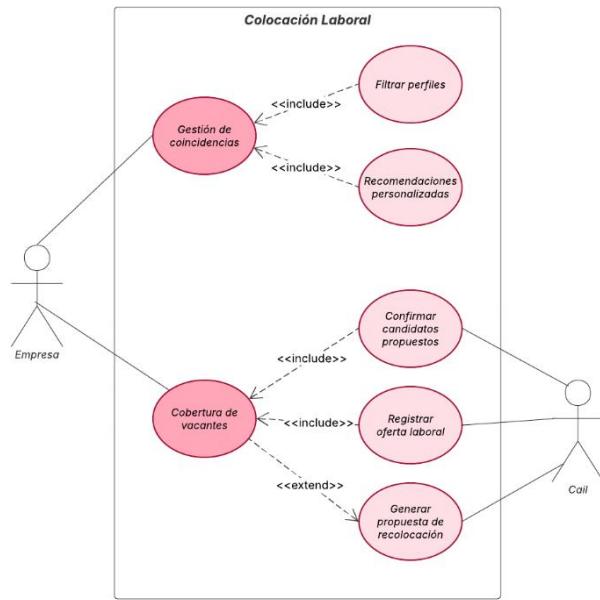
*Figura 4
CU Matching*



CU de Colocación laboral

Incluye la gestión de coincidencias y la cobertura de vacantes, permitiendo que la empresa receptora explore las recomendaciones del motor de matching, seleccione candidatos de interés y avance hacia la fase de negociación. Impacta en la vista lógica (módulos de búsqueda y filtrado), en la vista de interfaz de usuario (interacciones para revisar perfiles y estados de vacantes) y en la vista de datos (registro de preferencias, estados de vacantes y de empleados).

*Figura 5
CU Matching*

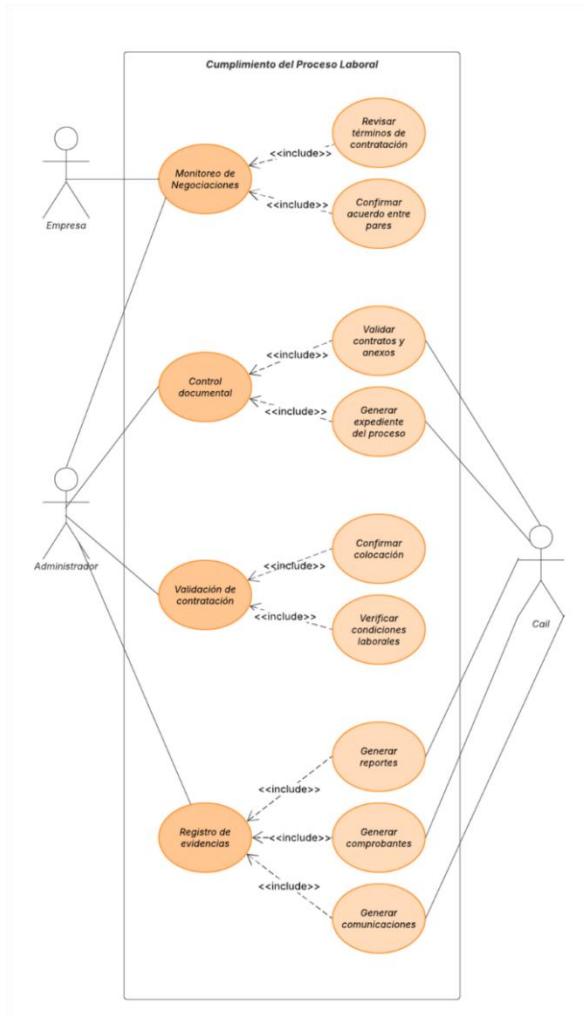


CU de Cumplimiento del proceso laboral (Figura 6)

Cubre la fase final del proceso, desde la validación de la negociación hasta la confirmación de la contratación, la generación del expediente del proceso laboral y la activación de procesos posteriores como monetización y analítica de resultados. Este caso de uso requiere un modelo de datos capaz de registrar de forma inmutable el expediente, reglas de negocio para validar documentación y tiempos, así como servicios capaces de coordinar estados finales y disparar cálculos de comisiones y reportes analíticos, afectando tanto a la vista de datos como a la de implementación y calidad.

Con esta selección, la vista de casos de uso cubre el ciclo completo de valor de la bolsa de empleo: incorporación de actores (empresas y empleados), evaluación de efectividad, recomendación mediante matching, selección y negociación, y cierre formal del proceso de colocación laboral.

*Figura 6
Cumplimiento del Proceso Laboral*



6. VISTA LÓGICA

6.1. Visión General

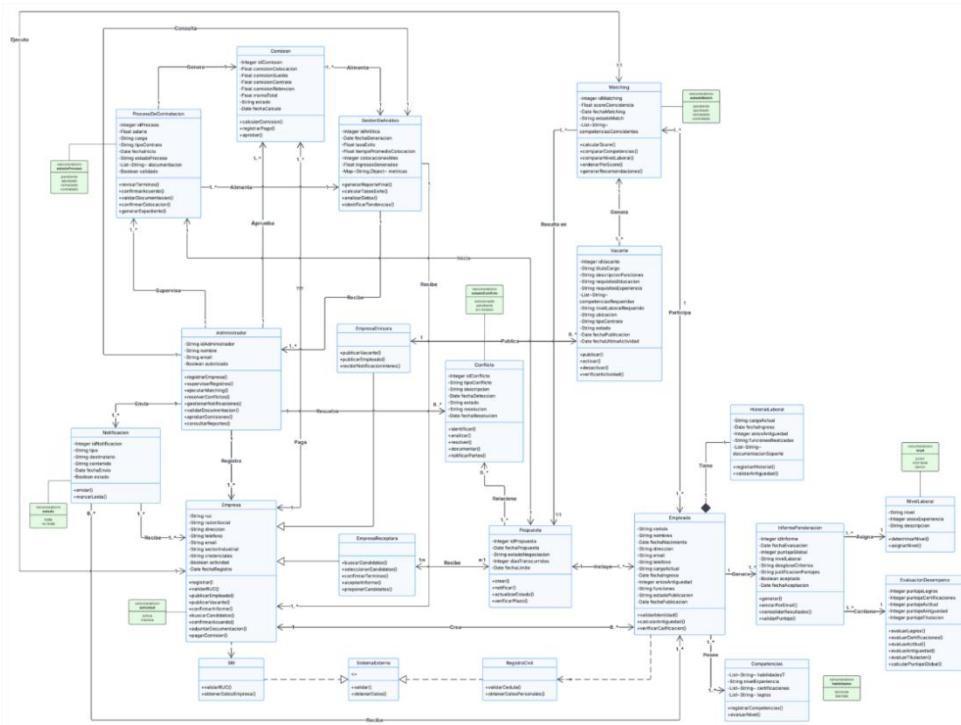
La vista lógica describe los principales componentes y clases de dominio de la Bolsa de Empleados Conjunta – CAIL, así como sus relaciones, para mostrar cómo se implementan los casos de uso arquitectónicamente significativos definidos en la sección 5. El modelo se organiza en paquetes que agrupan entidades de negocio (core.entities), procesos de contratación (core.processes), lógica de matching (core.matching), evaluación y ponderación (core.evaluation), análisis y monetización (core.analytics), gestión de notificaciones y conflictos (core.notifications, core.management) e integración con sistemas externos (external.services).

En el núcleo del modelo se encuentran las entidades Empresa, Empleado, Vacante, Matching, Propuesta, ProcesoDeContratacion e InformePonderacion, complementadas

por clases de apoyo como HistorialLaboral, Competencias, NivelLaboral y EvaluacionDesempeno, que enriquecen el perfil del empleado y respaldan los algoritmos de clasificación y emparejamiento. La clase Administrador y los servicios externos SRI y RegistroCivil completan la estructura lógica, al encargarse de la supervisión de procesos, la resolución de conflictos y la validación de datos legales e identitarios.

6.2. Diagrama de clases

*Figura 7
Diagrama de clases*



6.3. Diccionario de clases

Clasificación de Clases

Clases Principales: Empresa, Empleado, Administrador, Vacante y Matching.

Clases de Proceso: ProcesoDeContratacion, Comision y Propuesta.

Clases de Información: HistorialLaboral, Competencias, NivelLaboral, EvaluacionDesempeno y InformePonderacion.

Clases de Gestión: Notificacion, Conflicto y GestionDeAnalisis.

Clases Externas: SistemaExterno (Interfaz), RegistroCivil y SRI.

Enumeraciones: EstadoConflict, Nivel, Habilidades, Estado, Actividad, EstadoMatch y EstadoProceso.

Clase: Empresa

Aspecto	Detalle
Nombre	Empresa
Tipo	Clase abstracta (tiene subclases: EmpresaEmisora, EmpresaReceptora)
Paquete/Módulo	core.entities
Descripción	Representa una empresa afiliada a CAIL que puede actuar como emisora (ofrece empleados) o receptora (busca empleados)
Atributos	
ruc(String)	Registro Único de Contribuyentes, Único, 13 dígitos, requerido
razonSocial(String)	Nombre legal de la empresa, Requerido, max 200 caracteres
direccion(String)	Dirección física de la empresa, Requerido, max 200 caracteres
telefono(String)	Teléfono de contacto , Formato: 10 dígitos
email(String)	Correo electrónico corporativo , Formato válido, único
sectorIndustrial(String)	Sector económico al que pertenece, Catálogo predefinido
credenciales(String)	Credenciales de acceso al sistema, Hasheadas, encriptadas
actividad(String)	Número de contrataciones de la empresa, cuenta para la parte de las recomendaciones , Formato: 10 dígitos
fechaRegistro(String)	Fecha de registro en el sistema, Auto-generada
Métodos	
registrar()	Registra una nueva empresa en el sistema. Valida RUC y datos básicos.
validarRUC()	Consulta al SRI para validar que el RUC sea válido y esté activo.
publicarEmpleado()	Publica un empleado disponible para recolocación (solo EmpresaEmisora).
publicarVacante()	Publica una vacante disponible (solo EmpresaReceptora).
confirmarInforme()	Confirma y acepta un informe de ponderación de empleado.
buscarCandidatos()	Busca empleados que coincidan con criterios específicos.
confirmarAcuerdo()	Confirma un acuerdo de recolocación entre empresas, de acuerdo al mejor matching elegido por la empresa.
adjuntarDocumentacion()	Adjunta documentación legal requerida para validación.
pagarComision()	Realiza el pago de una comisión generada.
Relaciones	
Empleado(0..*)	Una empresa crea múltiples empleados
Notificacion(1..*)	Una empresa puede recibir múltiples notificaciones
Matching(1)	Una empresa ejecuta un matching
Comision(1..*)	Una empresa paga múltiples comisiones
Administrador(1)	Una empresa es registrada por un Administrador
GestionDeAnalisis(1)	Una empresa recibe un Análisis

SubClase: Empresa Emisora

Información General

Aspecto	Detalle
Nombre	EmpresaEmisora
Hereda de	Empresa
Descripción	Empresa que ofrece empleados senior para recolocación
Métodos Adicionales	
publicarVacante()	Publica una vacante disponible
publicarEmpleado()	Publica un empleado disponible para su recolocación.
recibirNotificacionInteres()	Recibe notificación cuando una empresa receptora muestra interés en un empleado.
Relaciones	
Vacante(0..*)	Una empresa emisora puede o no publicar múltiples vacantes

SubClase: Empresa Receptora

Información General

Aspecto	Detalle
Nombre	EmpresaReceptora
Hereda de	Empresa
Descripción	Empresa que busca empleados senior con experiencia
Métodos Adicionales	
buscarCandidatos()	Busca empleados según criterios específicos.
seleccionarCandidatos()	Selecciona candidatos de interés para proceso de contratación.
confirmarTerminos()	Confirma los términos de una propuesta de contratación.
aceptarInforme()	Acepta el informe de ponderación de un empleado.
proponerCandidatos()	Propone candidatos para una vacante específica.
Relaciones	
Propuesta(1..*)	Una empresa receptora recibe múltiples propuestas

Clase: Empleado

Información General

Aspecto	Detalle
Nombre	Empleado
Tipo	Clase concreta
Paquete/Módulo	core.entities
Descripción	Representa un empleado senior disponible para recolocación laboral

Atributos	
cedula(String)	Cédula de identidad ecuatoriana, Único, 10 dígitos, requerido
nombres(String)	Nombres completos del empleado, Requerido
fechaNacimiento(Date)	Fecha de nacimiento, Requerido, mayor de 18 años
direccion(String)	Dirección de residencia, Requerido
email(String)	Correo electrónico personal, Formato válido
telefono(String)	Teléfono de contacto, 10 dígitos
estadoPublicacion(String)	Estado de publicación del perfil, Enum: disponible, en_proceso, colocado
fechaPublicacion(Date)	Fecha en que se publicó el perfil, Auto-generada
Métodos	
validarIdentidad()	Valida la identidad del empleado consultando el Registro Civil.
calcularAntiguedad()	Calcula los años de antigüedad desde fecha de ingreso hasta hoy.
verificarCalificacion()	Verifica si el empleado cumple con los requisitos mínimos (antigüedad, evaluación).
Relaciones	
Competencias(1..*)	Un empleado posee múltiples competencias
HistorialLaboral(1..*)	Un empleado tiene múltiples historiales laborales
InformePonderacion(1..*)	Un empleado genera múltiples informes de ponderación.
Matching(1..*)	Un empleado participa en múltiples matchings
Notificacion(0..*)	Un empleado puede recibir múltiples notificaciones
Propuesta(1)	Un empleado es incluido a una propuesta
Empresa (1)	Un empleado es creado por una empresa

Clase: Administrador

Información General

Aspecto	Detalle
Nombre	Administrador
Tipo	Clase concreta
Paquete/Módulo	core.users
Descripción	Usuario administrador del sistema CAIL con permisos especiales
Atributos	
idAdministrador(String)	Identificador único del administrador, Único
nombre(String)	Nombre completo del administrador, Requerido
email(String)	Correo electrónico institucional, Único, formato válido
autorizado(Boolean)	Indica si el administrador está autorizado, Default: true
Métodos	
registrarEmpresa()	Registra una nueva empresa en el sistema.
supervisarRegistros()	Supervisa y revisa registros pendientes de validación.
ejecutarMatching()	Ejecuta el algoritmo de matching para generar coincidencias.

resolverConflictos()	Resuelve conflictos entre empresas o en procesos.
gestionarNotificaciones()	Gestiona el envío y seguimiento de notificaciones.
validarDocumentacion()	Valida la documentación subida por las empresas.
aprobarComisiones()	Aprueba o rechaza el pago de comisiones.
consultarReportes()	Consulta reportes y análisis del sistema.
Relaciones	
Empresa(1..*)	Un administrador registra múltiples empresas
Conflicto(0..*)	Un administrador resuelve múltiples conflictos
Notificacion(1..*)	Un administrador envía múltiples notificaciones
Comision(1..*)	Un administrador aprueba múltiples comisiones
GestionDeAnalisis(1..*)	Un administrador recibe múltiples análisis del sistema
ProcesoDeContratacion(1..*)	Un administrador supervisa múltiples procesos de contratación de empleados

Clase: Vacante

Información General

Aspecto	Detalle
Nombre	Vacante
Tipo	Clase concreta
Paquete/Módulo	core.entities
Descripción	Representa una vacante laboral publicada por una empresa receptora
Atributos	
idVacante(Integer)	Identificador único de la vacante, Auto-incremental, único
tituloCargo(String)	Título del cargo ofrecido, 200 caracteres
descripcionFunciones(String)	Descripción detallada de funciones, Max 2000 caracteres
requisitosEducacion(String)	Requisitos educativos necesarios , Requerido
añosExperiencia(String)	Años de experiencia requeridos, Mínimo 3 competencias
competenciasRequeridas(String)	Lista de competencias necesarias, Mínimo 3 competencias
nivelLaboralRequerido(String)	Nivel laboral requerido, Enum: junior, mid-level, senior
ubicacion(String)	Ubicación física del puesto, ciudad, Requerido
tipoContrato(String)	Tipo de contrato ofrecido, Enum predefinido, preestablecido
estado(String)	Estado actual de la vacante, Enum: activa, pausada, cerrada, cubierta
fechaPublicacion(Date)	Fecha de publicación de la vacante, Enum: activa, pausada, cerrada, cubierta. Auto-generada
fechaUltimaActividad(Date)	Última fecha de actividad en la vacante, Auto-actualizada
Métodos	

publicar()	Publica la vacante y la hace visible en el sistema.
activar()	Activa una vacante pausada.
desactivar()	Pausa temporalmente una vacante activada.
verificarActividad()	Verifica si la vacante ha tenido actividad reciente.
Relaciones	
EmpresaEmisora(1)	Una vacante es publicada por una empresa emisora
Matching(1)	Una vacante genera un matching

Clase: Matching

Información General

Aspecto	Detalle
Nombre	Matching
Tipo	Clase concreta
Paquete/Módulo	core.matching
Descripción	Representa una coincidencia entre un empleado y una vacante
Atributos	
idMatching(Integer)	Identificador único del matching, Auto-incremental, único
scoreCoincidencia(Float)	Porcentaje de coincidencia (0-100)
fechaMatching(String)	Fecha en que se generó el matching, Auto-generada
estadoMatch(String)	Estado actual del matching, Enum: pendiente, aprobado, rechazado, contratado, precargada
competenciasCoincidentes(String)	Lista de competencias que coinciden, Auto-calculada
Métodos	
calcularScore()	Calcula el porcentaje de coincidencia entre empleado y vacante.
compararCompetencias()	Compara las competencias del empleado con las requeridas.
compararNivelLaboral()	Verifica si el nivel laboral coincide.
ordenarPorScore()	Ordena una lista de matchings por score descendente.
generarRecomendaciones()	Genera recomendaciones sobre el matching para las partes.
Relaciones	
Empleado(1..*)	En un matching participan múltiples empleados
Vacante(1..*)	Un matching relaciona múltiples vacantes
Empresa(1)	Un matching involucra una empresa
Propuesta(1)	Un matching resulta en una propuesta

Especificación de Clases de Proceso

Clase: ProcesoDeContratacion

Información General

Aspecto	Detalle
Nombre	ProcesoDeContratacion
Tipo	Clase concreta
Paquete/Módulo	core.processes
Descripción	Gestiona el proceso completo de contratación desde matching hasta expediente
Atributos	
idProceso(Integer)	Identificador único del proceso, Auto-incremental, único
salario(Float)	Salario acordado para el empleado, Mayor a 0, en USD
cargo(String)	Cargo ofrecido al empleado Requerido, 200 caracteres
tipoContrato(String)	Tipo de contrato, Enum, predefinido
fechalinicio(Date)	Fecha de inicio del contrato, Requerida
estadoProceso(String)	Estado actual del proceso, Enum: pendiente, aprobado, rechazado, contratado
documentacion(String)	Lista de documentos del proceso, Mínimo 3 documentos
validado(Boolean)	Indica si el proceso está validado, Default: false
Métodos	
revisarTerminos()	Revisa y valida los términos del contrato.
confirmarAcuerdo()	Confirma el acuerdo entre empresa receptora y empleado.
validarDocumentacion()	Valida que toda la documentación esté completa y vigente.
confirmarColocacion()	Confirma la colocación exitosa del empleado.
generarExpediente()	Genera el expediente digital del proceso completo.
Relaciones	
Propuesta(1)	Un proceso de contratación se inicia con una propuesta
Comision(1)	Un proceso de contratación genera una comisión
GestionDeAnalisis(1)	Un proceso de contratación alimenta un mismo análisis del sistema
Administrador(1)	Un proceso de contratación es supervisado y validado por un administrador

Clase: Comision

Información General

Aspecto	Detalle
Nombre	Comision
Tipo	Clase concreta
Paquete/Módulo	core.financial
Descripción	Gestiona el cálculo y pago de comisiones por colocación exitosa
Atributos	

idComision	Identificador único de la comisión
comisionColocacion	Comisión fija por colocación
comisionSueldo	Porcentaje del salario anual
comisionContrato	Comisión según tipo de contrato
comisionRetencion	Bono por retención del empleado
montoTotal	Monto total de la comisión
estadoComision	Estado de la comisión
fechaCalculo	Fecha de cálculo de la comisión
Métodos	
calcularComision()	Calcula el monto total de la comisión según los parámetros del proceso.
registrarPago()	Registra el pago efectuado de la comisión.
aprobar()	Aprueba la comisión para pago (requiere administrador).
Relaciones	
ProcesoDeContratacion(1..*)	Una comisión se generan múltiples procesos de contratación
Empresa(1)	Una comisión es pagada por un administrador
Administrador(1)	Una comisión es aprobada por un administrador
GestionDeAnalisis(1)	Una comisión alimenta un mismo análisis

Clase: Propuesta

Información General

Aspecto	Detalle
Nombre	Propuesta
Tipo	Clase concreta
Paquete/Módulo	core.matching
Descripción	Representa una propuesta de contratación generada por un matching
Atributos	
idPropuesta(Integer)	Identificador único de la propuesta
fechaPropuesta(Date)	Fecha de creación de la propuesta
estadoNegociacion(String)	Estado actual de negociación
diasTranscurridos(Integer)	Días desde la creación
fechaLimite(Date)	Fecha límite para respuesta
Métodos	
fechaPropuesta	Fecha de creación de la propuesta
estadoNegociacion	Estado actual de negociación
diasTranscurridos	Días desde la creación
fechaLimite	Fecha límite para respuesta
Relaciones	
Matching(1..*)	Una propuesta es resultado de un matching.
EmpresaReceptora(1..*)	Una propuesta recibe una empresa receptora.

Conflicto(0..*)	Una propuesta puede relacionarse o no con múltiples conflictos.
Empleado(1..*)	Una propuesta puede incluir múltiples empleados

Especificación de Clases de Información

Clase: HistorialLaboral

Información General

Aspecto	Detalle
Nombre	HistorialLaboral
Tipo	Clase concreta
Paquete/Módulo	core.employee
Descripción	Registro del historial laboral de un empleado
Atributos	
idHistoriaLaboral(Integer)	Identificador unico de historia laboral
cargoActual(String)	Cargo actual del empleado
fechalIngreso(Date)	Fecha de ingreso a la empresa
aniosAntiguedad(Integer)	Años de antigüedad
funcionesRealizadas(String)	Descripción de funciones
documentacionSoporte(List<String>)	Documentos que respaldan el historial
Métodos	
registrarHistorial()	Registra un nuevo historial laboral en el sistema.
validarAntiguedad()	Valida que la antigüedad sea mayor o igual a 15 años.
Relaciones	
Empleado(1)	Un historial pertenece a un empleado

Clase: Competencias

Información General

Aspecto	Detalle
Nombre	Competencias
Tipo	Clase concreta
Paquete/Módulo	core.employee
Descripción	Conjunto de habilidades, certificaciones y logros de un empleado
Atributos	
IdCompetencias(Integer)	Nivel de experiencia general
HabilidadesT(List<String>)	Lista de certificaciones obtenidas
HabilidadesB(List<String>)	Lista de logros profesionales destacados
nivelExperiencia(String)	Nivel de experiencia general

certificaciones(List<String>)	Lista de certificaciones obtenidas
logros(List<String>)	Lista de logros profesionales destacados
Métodos	
registrarCompetencias()	Registra nuevas competencias del empleado.
evaluarNivel()	Evalúa y determina el nivel de experiencia del empleado.
Relaciones	
Empleado(1...*)	Las competencias pueden pertenecer a múltiples empleados

Clase: NivelLaboral

Información General

Aspecto	Detalle
Nombre	NivelLaboral
Tipo	Clase concreta
Paquete/Módulo	core.employee
Descripción	Define el nivel laboral de un empleado (junior, mid-level, senior)
Atributos	
IdNivelLaboral(Integer)	Identificador unico
nivel(String)	Nivel laboral del empleado
aniosExperiencia(Integer)	Años de experiencia requeridos para el nivel
descripcion(String)	Descripción del nivel laboral
Métodos	
determinarNivel()	Determina el nivel laboral basado en experiencia y competencias.
asignarNivel()	Asigna un nivel laboral específico al empleado.
Relaciones	
Relaciones(1...*)	Un mismo nivel laboral se puede asignar a múltiples informades de ponderación

Clase: EvaluacionDesempeno

Información General

Aspecto	Detalle
Nombre	EvaluacionDesempeno
Tipo	Clase concreta
Paquete/Módulo	core.evaluation
Descripción	Evaluación detallada del desempeño de un empleado
Atributos	
idEvaluacionDesempeño(Integer)	Identificador unico
puntajeLogros(Integer)	Puntaje por logros profesionales

puntajeCertificaciones(Integer)	Puntaje por certificaciones
puntajeActitud(Integer)	Puntaje por actitud y comportamiento
puntajeAntiguedad(Integer)	Puntaje por años de antigüedad
puntajeTitulacion(Integer)	Puntaje por nivel educativo
Métodos	
evaluarLogros()	Evalúa y calcula puntaje de logros profesionales.
evaluarCertificaciones()	Evalúa y calcula puntaje de certificaciones de acuerdo a la vigencia y numero de certificados.
evaluarActitud()	Evalúa actitud basada en referencias del empleador. (1-5)
evaluarAntiguedad()	Calcula puntaje basado en años de antigüedad.
evaluarTitulacion()	Evalúa el nivel educativo del empleado de acuerdo al nivel de estudios del empleado.
calcularPuntajeGlobal()	Calcula el puntaje global promedio ponderado, uniendo las evaluaciones.
Relaciones	
InformePonderacion(1..*)	Una evaluación de desempeño forma parte múltiples informes de ponderación

Clase: InformePonderacion

Información General

Aspecto	Detalle
Nombre	InformePonderacion
Tipo	Clase concreta
Paquete/Módulo	core.evaluation
Descripción	Informe consolidado de ponderación de un empleado
Atributos	
idInforme(Integer)	Identificador único del informe
fechaEvaluacion(Date)	Fecha de generación del informe
puntajeGlobal(Integer)	Puntaje global del empleado
nivelLaboral(String)	Nivel laboral determinado
desgloseCriterios(String)	Desglose detallado de criterios evaluados
justificacionPuntajes(String)	Justificación de los puntajes asignados
aceptado(Boolean)	Indica si el informe fue aceptado
fechaAceptacion(Date)	Fecha en que se aceptó el informe
Métodos	
generar()	Genera el informe completo basado en la evaluación.
enviarPorEmail()	Envía el informe por correo electrónico.
consolidarResultados()	Consolida todos los resultados en un formato estructurado.
validarPuntaje()	Valida que los puntajes estén dentro de rangos permitidos.
Relaciones	

Empleado(1)	Un informe de ponderación es generado para un empleado
EvaluacionDesempeno(1)	Un informe de ponderación contiene de una evaluación de desempeño
NivelLaboral(1)	Un informe de ponderación asigna un nivel laboral

Especificación de Clases de Gestión

Clase: Notificacion

Información General

Aspecto	Detalle
Nombre	Notificacion
Tipo	Clase concreta
Paquete/Módulo	core.notifications
Descripción	Sistema de notificaciones para comunicar eventos importantes
Atributos	
idNotificacion(Integer)	Identificador único de la notificación
tipoNotificacion(String)	Tipo de notificación
Destinatario(String)	Email o ID del destinatario
Contenido(String)	Contenido del mensaje
fechaEnvio(Date)	Fecha de envío de la notificación
Estado(Boolean)	Estado de lectura
Métodos	
enviar()	Envía la notificación al destinatario.
marcarLeida()	Marca la notificación como leída.
Relaciones	
Empresa(1..*)	Una notificación es recibida por múltiples empresas
Empleado(1..*)	Una notificación es recibida por múltiples empleados
Administrador(1)	Una notificación es enviada por un solo administrador

Clase: Conflicto

Información General

Aspecto	Detalle
Nombre	Conflicto
Tipo	Clase concreta
Paquete/Módulo	core.management
Descripción	Gestiona conflictos entre entidades del sistema
Atributos	

idConflicto(Integer)	Identificador único del conflicto
tipoConflicto(String)	Tipo de conflicto
descripcion(String)	Descripción detallada del conflicto
fechaDeteccion(Date)	Fecha en que se detectó el conflicto
estadoConflicto(String)	Estado actual del conflicto
resolucion(String)	Descripción de la resolución
fechaResolucion(String)	Fecha en que se resolvió
Métodos	
identificar()	Identifica y registra un nuevo conflicto.
analizar()	Analiza el conflicto y genera recomendaciones.
resolver()	Registra la resolución del conflicto.
documentar()	Genera documentación del conflicto y su resolución.
notificarPartes()	Notifica a las partes involucradas sobre el estado.
Relaciones	
Administrador	Cero o múltiples conflictos son resueltos por un solo administrador
Propuesta	Un conflicto puede relacionarse con múltiples propuestas

Clase: GestionDeAnalisis

Información General

Aspecto	Detalle
Nombre	GestionDeAnalisis
Tipo	Clase concreta
Paquete/Módulo	core.analytics
Descripción	Gestiona análisis y métricas del sistema CAIL
Atributos	
idAnalitica(Integer)	Identificador único del análisis
fechaGeneracion(Date)	Fecha de generación del reporte
tasaExito(Float)	Porcentaje de colocaciones exitosas
tiempoPromedioColocacion(Float)	Días promedio desde matching hasta colocación
colocacionesMes(Integer)	Número de colocaciones en el mes
ingresosGenerados(Float)	Ingresos totales por comisiones
métricas(Map<String, Object>)	Métricas adicionales del sistema
Métodos	
generarReporteFinal()	Genera reporte consolidado del periodo especificado.
calcularTasasExito()	Calcula la tasa de éxito de colocaciones.
analizarDatos()	Analiza todos los datos disponibles y genera insights.
identificarTendencias()	Identifica tendencias en el mercado laboral.
Relaciones	
Administrador(1)	Una gestión de análisis es recibida por un solo administrador

ProcesoDeContratacion(1..*)	Una gestión de análisis es alimentada por múltiples procesos de contratación
Empresa(1)	Una gestión de análisis es recibida por un empleado
Comisión(1..*)	Una gestión de análisis es alimentada por múltiples comisiones

Especificación de Clases Externas

Interfaz: SistemaExterno

Información General

Aspecto	Detalle
Nombre	SistemaExterno
Tipo	Interfaz
Paquete/Módulo	external.services
Descripción	Interfaz para integración con sistemas externos (SRI, Registro Civil)
Métodos Abstractos	
validar()	Valida información en el sistema externo.
obtenerDatos()	Obtiene datos completos del sistema externo.

Clase: RegistroCivil

Información General

Aspecto	Detalle
Nombre	RegistroCivil
Tipo	Clase concreta
Implementa	SistemaExterno
Paquete/Módulo	external.services
Descripción	Servicio de integración con el Registro Civil Ecuatoriano
Métodos	
validarCedula()	Valida que una cédula sea válida y exista en el Registro Civil.
obtenerDatosPersonales()	Obtiene datos personales completos de una persona.
validar()	Implementación de método de interfaz.
obtenerDatos()	Implementación de método de interfaz.

Clase: SRI

Información General

Aspecto	Detalle
Nombre	SRI

Tipo	Clase concreta
Implementa	SistemaExterno
Paquete/Módulo	external.services
Descripción	Servicio de integración con el Servicio de Rentas Internas (SRI)
Métodos	
validarRUC()	validarRUC()
obtenerDatosEmpresa()	obtenerDatosEmpresa()

7. VISTA DE PROCESOS

7.1. Visión General

Los diagramas de secuencia son una herramienta de modelado en la Ingeniería de Software que forma parte del Lenguaje Unificado de Modelado (UML). Se utilizan para describir cómo los objetos de un sistema interactúan entre sí a través del tiempo, mostrando el flujo de mensajes o llamadas entre ellos. Estos diagramas son especialmente útiles para representar procesos y casos de uso en sistemas orientados a objetos.

Estructura de Diagrama de Secuencia

1. Actores y Objetos: Representan las entidades que participan en la interacción. Los actores se representan como figuras humanas, y los objetos como rectángulos.
2. Líneas de vida: Indican la existencia de un actor u objeto durante el proceso. Son líneas verticales que comienzan debajo de cada actor u objeto.
3. Mensajes: Representan la comunicación entre actores y objetos mediante flechas horizontales.
4. Bloques de ejecución: Rectángulos sobre las líneas de vida que representan la ejecución de un proceso o método.
5. Tiempo: Se representa de forma implícita en la dirección descendente del diagrama.

7.2. Procesos y Flujos Clave (Diagramas de Secuencia)

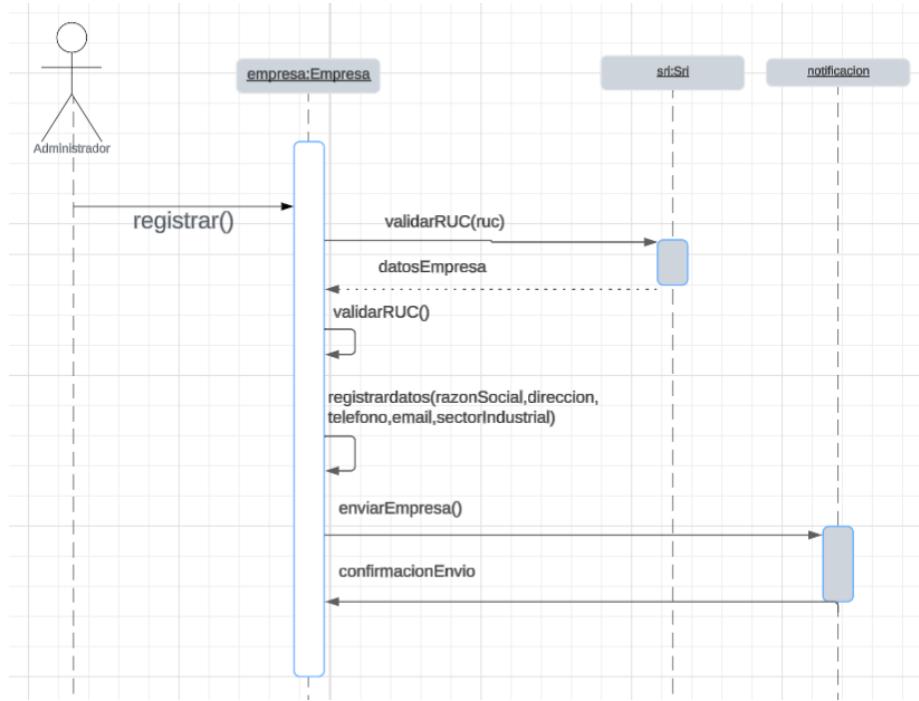
Registro de empresa

Este diagrama describe cómo se agrega nuevas empresas en el sistema y ademas valida que los datos agregados sean correctos, es decir si cumplen con las restricciones y si estan agregados en el sistema del SRI. Los pasos clave incluyen:

- Verificar datos ingresados.
- Vallidar RUC.

- Enviar datos registrados.

*Figura 8
Diagrama de secuencia de registro de empresa*

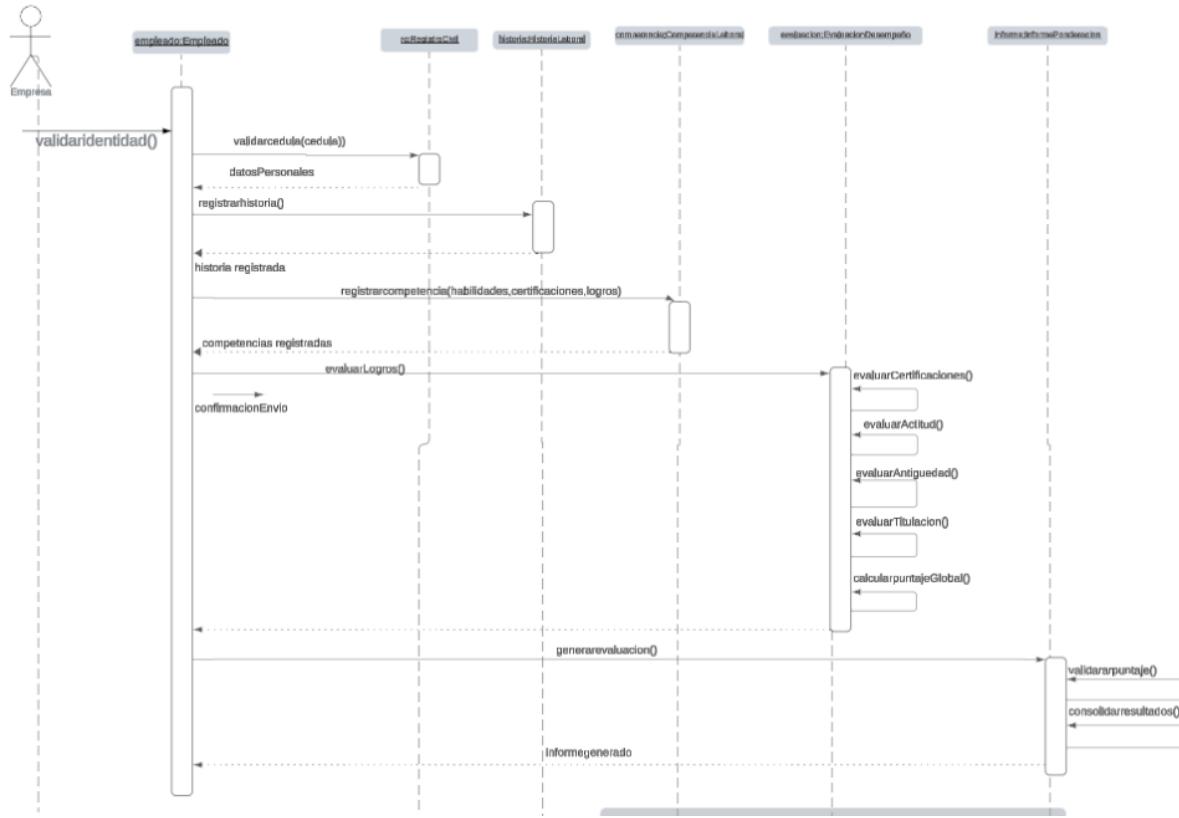


Registro de Empleados

Este diagrama describe proceso de registro y evaluación de empleados senior en el sistema CAIL. La Empresa Emisora proporciona el historial laboral, competencias, certificaciones y logros del empleado. El sistema valida su identidad mediante el Registro Civil y calcula un puntaje global de efectividad laboral (0-100) evaluando cinco criterios ponderados: logros (20%), certificaciones (15%), actitud (25%), antigüedad (25%) y titulación (15%). Con base en este análisis, se genera un Informe de Ponderación que determina el nivel laboral del empleado (Junior/Mid-Level/Senior). El flujo incluye:

- Registrar el empleado.
- Generar informe evaluacion del empleado.
- Clasifica el estado del empleado.

*Figura 9
Diagrama de secuencia de registro de empleados*



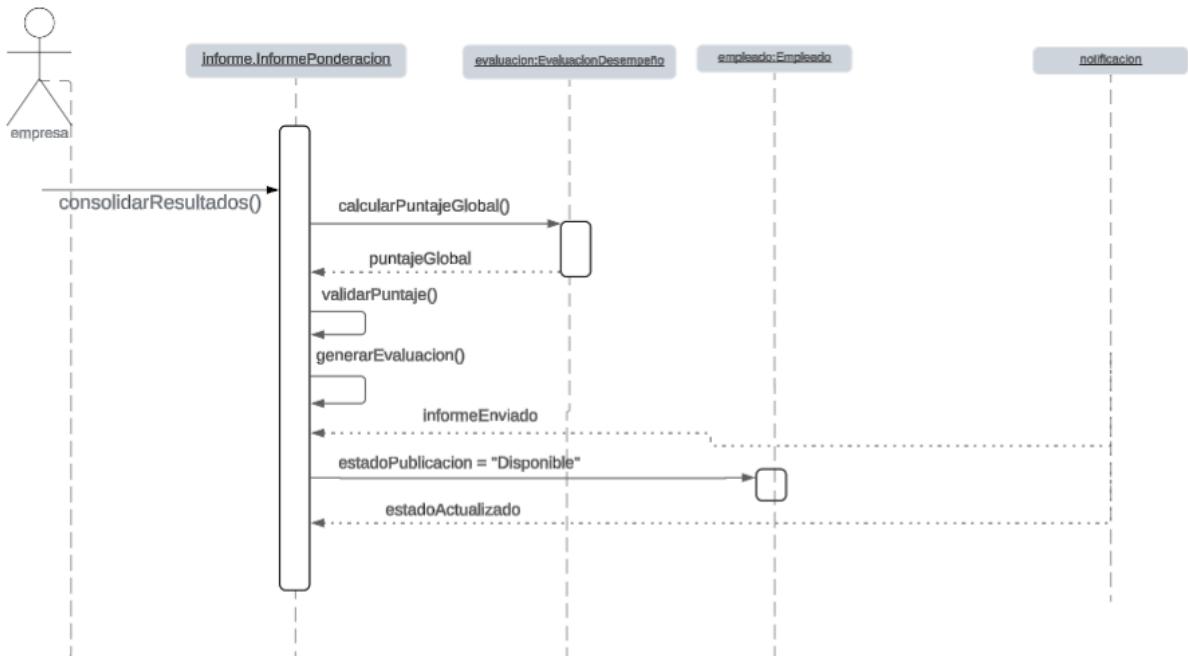
Consolidar Resultados, Asignar Puntaje y Confirmar

Este diagrama muestra el proceso de generación, validación y aceptación del Informe de Ponderación. El flujo incluye:

- El sistema valida que el puntaje esté dentro del rango permitido y que cumpla con el mínimo requerido para su publicación.
- El informe se envía automáticamente por email tanto al empleado como a la Empresa Emisora.
- La empresa confirma su aceptación del informe.

Figura 10

Diagrama de secuencia de generación, validación y aceptación del Informe de Ponderación

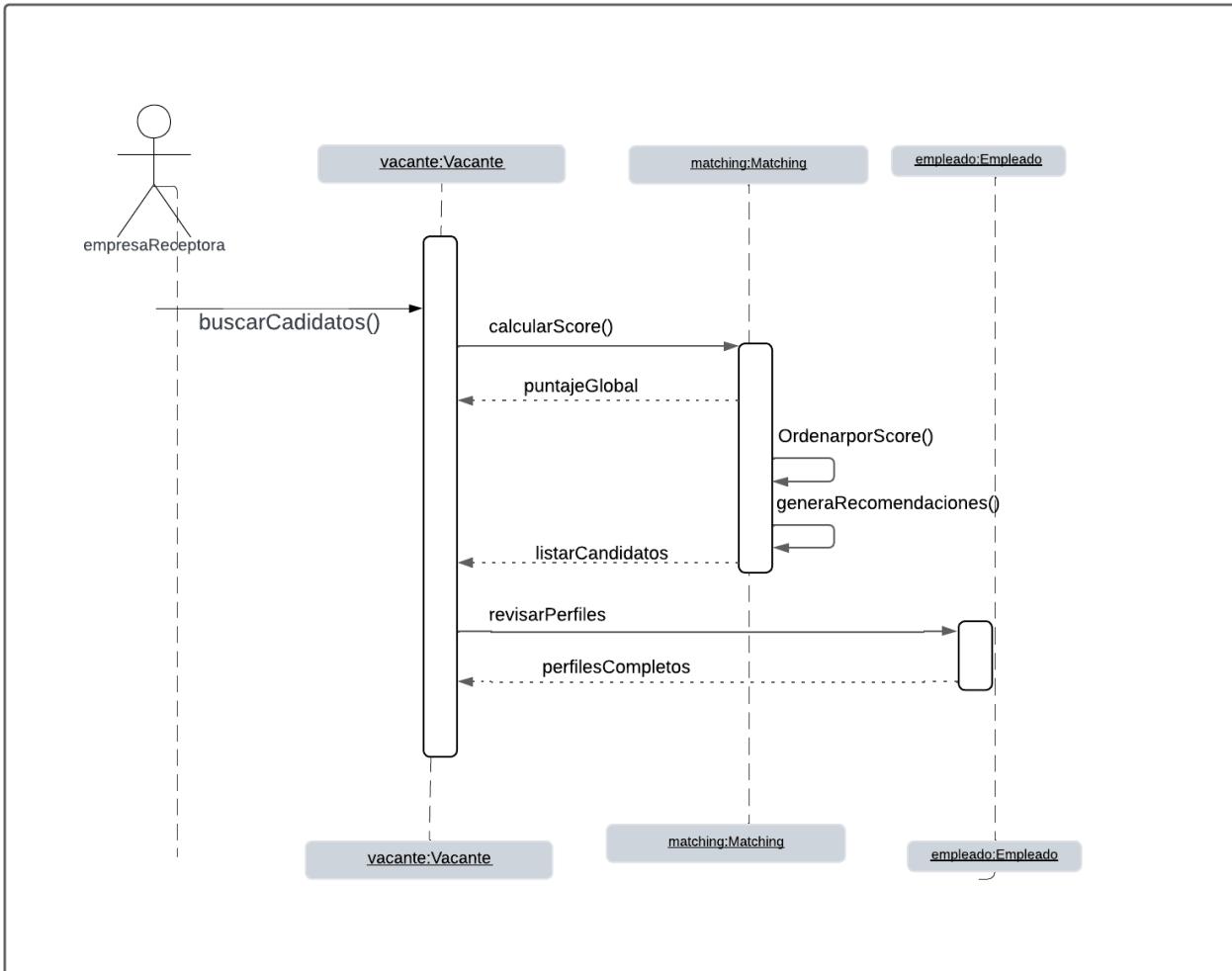


Gestión de Coincidencias

Este diagrama representa el proceso de búsqueda y filtrado de candidatos por parte de Empresas Receptoras. El flujo incluye:

- La empresa genera una vacante específica.
- El Matching compara las competencias con los empleados disponibles.
- El sistema ordena los resultados presentando únicamente empleados con coincidencia mínima.

*Figura 10
Diagrama de secuencia de coincidencias*



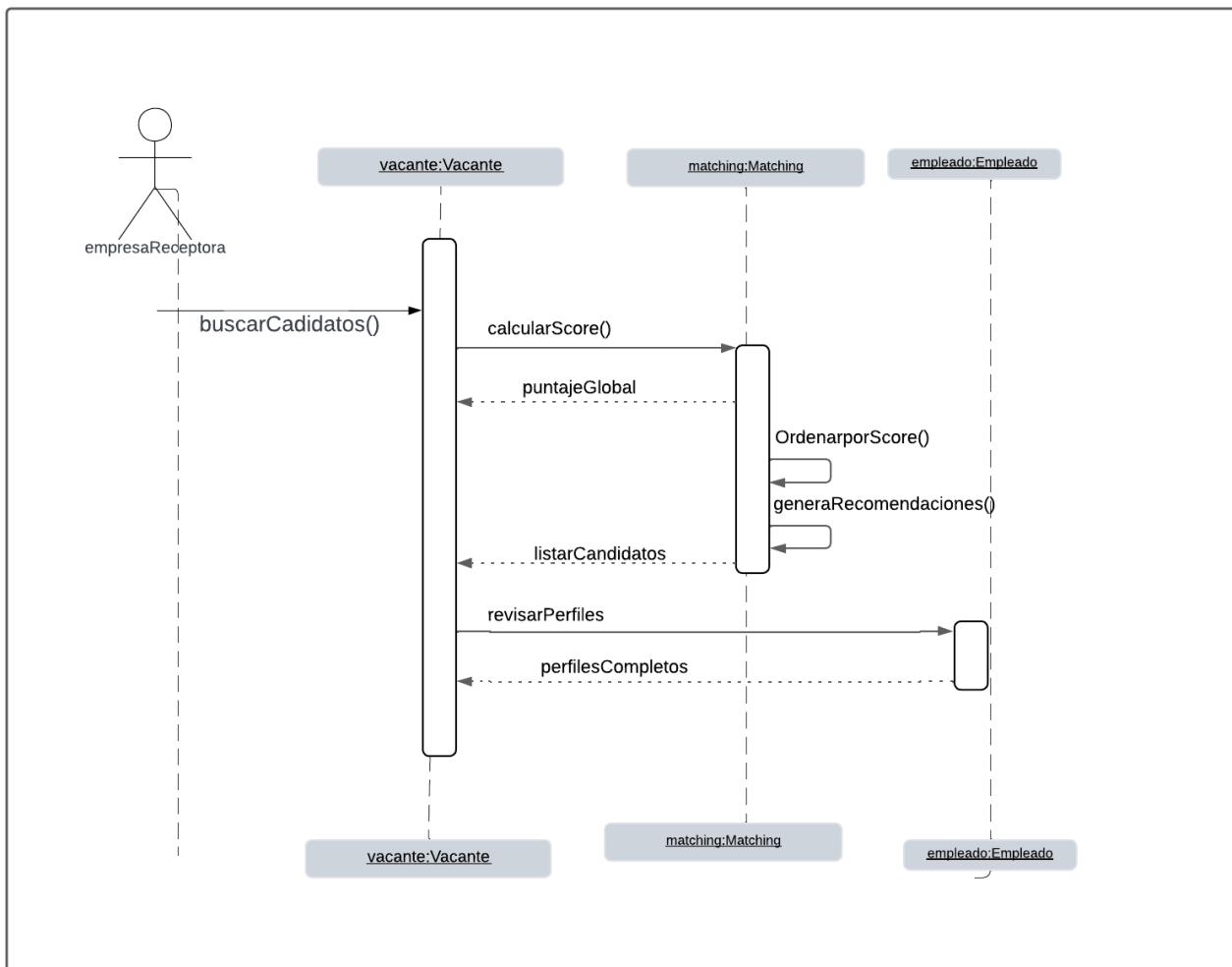
Cobertura de Vacantes

Este diagrama ilustra el proceso de selección y proposición formal de candidatos para vacantes abiertas. El flujo incluye:

- La Empresa Receptora revisa los candidatos y selecciona los empleados interés.
- Al confirmar su selección, el sistema crea una Propuesta formal que establece un plazo de negociación de 15 días hábiles.
- Los empleados seleccionados pasan automáticamente a estado "En Negociación", bloqueándolos temporalmente para otras ofertas.
- El sistema notifica a la Empresa Emisora sobre el interés mostrado en sus empleados y alerta al Administrador para seguimiento del proceso.

Figura 10

Diagrama de secuencia cobertura de vacantes



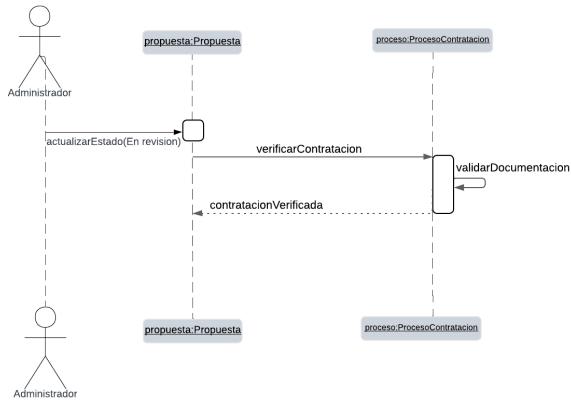
Cumplimiento del Proceso Laboral

Explicación: Este diagrama detalla el proceso de validación y formalización de contrataciones exitosas. El flujo incluye:

- El Administrador supervisa propuestas "En Revisión", verificando que los términos acordados sean confirmados formalmente tanto por la Empresa Emisora como por la Empresa Receptora.
- Ambas empresas deben adjuntar documentación legal completa (contrato firmado, anexos laborales, certificaciones).
- El Administrador valida exhaustivamente toda la documentación, rechazándola si no cumple requisitos mínimos.

- Una vez aprobada, el sistema confirma la colocación exitosa y genera un expediente digital.
- El empleado pasa a estado "Contratado", se calculan las comisiones y las partes reciben notificación formal de confirmación.

*Figura 13
Diagrama de secuencia proceso laboral*

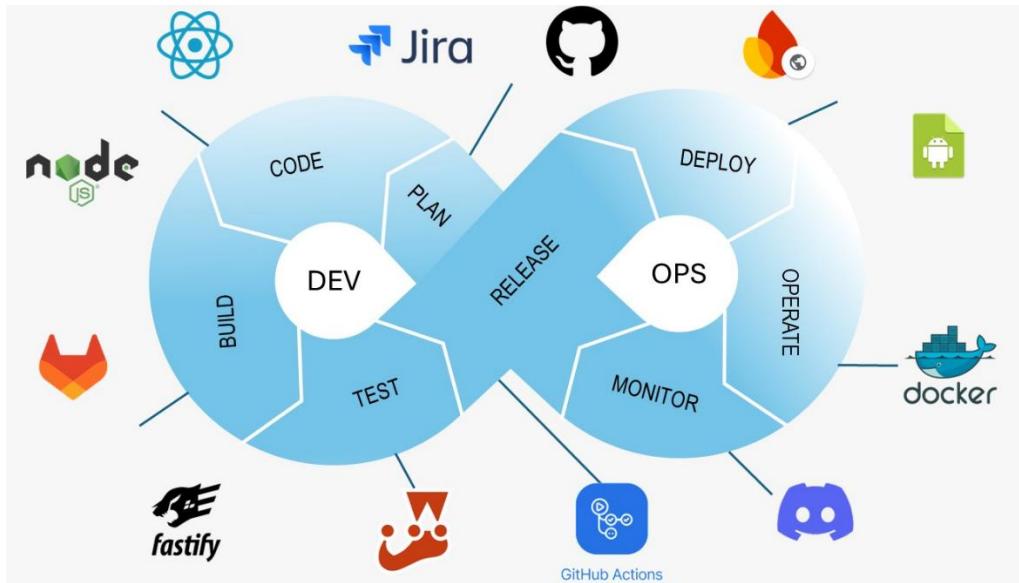


8. VISTA DE IMPLEMENTACIÓN

8.1. Visión General

El diagrama de componentes de desarrollo muestra la organización lógica del sistema Bolsa de Empleados Conjunta – CAIL, destacando los principales componentes de software y sus dependencias. La arquitectura se estructura en capas claramente diferenciadas: presentación, gestión de APIs, lógica de negocio y persistencia de datos.

*Figura 15
Flujo DevOps*



Fase DevOps	Plataformas	Descripción
Plan	GitHub	Se utilizará para gestionar el desarrollo colaborativo, organizar tareas mediante branches.
	Jira	Se empleará para el seguimiento de tareas y actualizaciones del estado de los incidentes detectados.
Code	React	Será empleado para desarrollar la interfaz móvil, garantizando una experiencia de usuario eficiente.
	Node.js	Se utiliza Node.js para escribir el código del backend, aprovechando su capacidad para manejar múltiples conexiones simultáneas y su ecosistema de módulos para integrar funcionalidades necesarias en el proyecto.
Build	GitHub Actions	Se utilizará para la automatización del testeo de cada parte del código.
	Fastify	Su arquitectura modular facilita la integración de diferentes componentes del sistema.
Test	Jest	Realiza pruebas unitarias e integradas para validar que las funciones, componentes y módulos se comporten correctamente antes de integrar cambios al repositorio.
Release	Fastify	Permite preparar la aplicación para su lanzamiento mediante la configuración de rutas y middleware, asegurando que todos los endpoints estén listos para ser utilizados en producción.
Deploy	Firebase Hosting	Permite implementar el backend sin preocuparse por la infraestructura subyacente, asegurando una alta disponibilidad.

		disponibilidad y escalabilidad del sistema de gestión de inventario.
	Android Package Kit	Se genera automáticamente dentro del pipeline de CI/CD, contiene el código, recursos, librerías y configuraciones necesarias para ejecutar la aplicación.
Operate	Docker	Se encarga de ejecutar la aplicación en contenedores estables y aislados dentro del entorno de producción, además asegura que la aplicación backend se mantenga activa, segura y funcionando de manera uniforme durante su operación diaria.
Monitor	Discord	Se utilizará para la comunicación en tiempo real y notificaciones de eventos relevantes.

8.2. Estructura de componentes de desarrollo

La estructura de componentes de desarrollo define los módulos lógicos que conforman el sistema y los artefactos generados durante el proceso de desarrollo.

Componentes principales

Frontend (Web y Móvil – React)

- UI Components: Componentes reutilizables de interfaz
- Pages / Views: Pantallas principales del sistema
- State Management: Manejo del estado de la aplicación
- Services: Consumo de APIs REST
- Routing: Navegación entre vistas
- Assets: Recursos estáticos

Backend (API – Node.js + Fastify)

- Routes: Definición de endpoints REST
- Controllers: Gestión de solicitudes y respuestas
- Services: Lógica de negocio
- Models: Definición de entidades y esquemas de datos
- Middlewares: Autenticación, autorización y validaciones

- Config: Variables de entorno y configuración del sistema

Infraestructura y DevOps

- CI/CD Pipelines: Automatización con GitHub Actions
- Dockerfiles: Contenedores de ejecución
- Testing: Pruebas unitarias e integradas con Jest
- Build Artifacts: APK, builds de frontend y backend

8.3. Estructura detallada del repositorio

A continuación, se desarrolla la estructura del repositorio del proyecto Bolsa de Empleados Conjunta – CAIL

bolsa-empleados/

```
|  
|   └── frontend/  
|       |   └── web/  
|       |       └── public/  
|       |       └── src/  
|       |           └── assets/  
|       |           └── components/  
|       |               └── common/  
|       |               └── forms/  
|       |               └── layout/  
|       |               └── pages/  
|       |               └── auth/  
|       |               └── dashboard/  
|       |               └── empresas/
```

```
| | | |   └── empleados/
| | | |   └── vacantes/
| | | |   └── postulaciones/
| | | |   └── reportes/
| | |   └── services/
| | |   └── api.ts
| | |   └── auth.service.ts
| | |   └── empleados.service.ts
| | |   └── vacantes.service.ts
| | |   └── matching.service.ts
| |   └── hooks/
| |   └── context/
| |   └── routes/
| |   └── utils/
| |   └── App.tsx
|   └── package.json
|
|   └── mobile/
|       ├── android/
|       ├── ios/
|       └── src/
|           └── components/
|           └── screens/
|               └── auth/
```

```
|   |   |   |-- home/  
|   |   |   |-- empleados/  
|   |   |   |-- vacantes/  
|   |   |   \-- postulaciones/  
|   |   |-- navigation/  
|   |   |-- services/  
|   |   |-- context/  
|   |   |-- hooks/  
|   |   |-- hooks/
```

```
|   |   \-- App.tsx  
|   \-- package.json  
|  
|   \-- frontend/
```

```
|       |-- .expo/  
|       |   |-- types/  
|       |   \-- devices.json  
|       |-- app/  
|           \-- app.tsx  
|           \-- index.tsx  
|  
|   \-- src/  
|       |-- assets/  
|       |   |-- fonts/
```

```
|   |   |-- logo/
|   |   |-- text/
|   |-- components/
|   |-- config/
|   |   |-- components/
|   |   |-- screens/
|   |   |   |-- auth/
|   |   |   |-- home/
|   |   |   |-- empleados/
|   |   |   |-- vacantes/
|   |   |   \-- postulaciones/
|   |   |-- constant/
|   |   |-- context/
|   |   |-- hooks/
|   |   |-- navigation/
|   |   |-- screens/
|   |   |-- hooks/
```

```
|   |   \-- App.tsx
|   \-- package.json
|
```

```
└── backend/
    └── api-gateway/
```

```
| |   |-- routes/
| |   |-- middlewares/
| |   |-- auth/
| |       └── gateway.config.js
| |
|   |-- services/
|   |   |-- autenticacion/
|   |       |   |-- verificarRUC.js
|   |       |   |-- verificarCEDULA.js
|   |       └── auth.controller.js
|   |
|   |   |-- empresas/
|   |       |   |-- registroEmpresas.js
|   |       |   |-- empresas.controller.js
|   |       └── empresas.routes.js
|   |
|   |   |-- empleados/
|   |       |   |-- registroEmpleados.js
|   |       |   |-- empleados.controller.js
|   |       └── empleados.routes.js
|   |
|   |   |-- cumplimiento-legal/
|   |       └── cumplimientoLegal.js
|   |
```

```
| | | └─ evaluacion-desempeno/  
| | |   └─ evaluarDesempenoHistorico.js  
| | |   └─ evaluarNivelLaboral.js  
| | |   └─ evaluacion.controller.js  
| | |  
| | |  
| | | └─ vacantes/  
| | |   └─ coberturaVacantes.js  
| | |   └─ vacantes.controller.js  
| | |   └─ vacantes.routes.js  
| | |  
| | |  
| | | └─ matching/  
| | |   └─ matching.js  
| | |   └─ gestionCoincidencias.js  
| | |   └─ matching.controller.js  
| | |  
| | |  
| | | └─ colocacion-laboral/  
| | |   └─ colocacionLaboral.js  
| | |  
| | |  
| | | └─ proceso-laboral/  
| | |   └─ procesoLaboral.js  
| | |  
| | |  
| | | └─ monetizacion/  
| | |   └─ calcularComisiones.js  
| | |   └─ monetizacion.controller.js
```

```
| | |
| | └─ reportes/
| |   ├─ generarInformes.js
| |   └─ reportes.controller.js
| |
| └─ shared/
|   ├─ utils/
|   ├─ constants/
|   ├─ errors/
|   └─ logger/
| |
| └─ server.js
|
└─ package.json
|
|
└─ database/
  ├─ firestore/
  | | └─ empresas.schema.json
  | | └─ empleados.schema.json
  | | └─ documentos.schema.json
  | | └─ evaluacionDesempeno.schema.json
  | | └─ informesPonderacion.schema.json
  | | └─ coincidencias.schema.json
  | | └─ matchings.schema.json
  | | └─ procesosLaboral.schema.json
```

```
| | └ comisiones.schema.json  
|  
| └ integrations/  
|   |   └ firebase-auth/  
|   |   └ api-registro-civil/  
|   └ api-sri/  
|  
| └ devops/  
|   |   └ docker/  
|   |     |   └ Dockerfile.backend  
|   |     |   └ Dockerfile.web  
|   |     └ Dockerfile.mobile  
|   └ kubernetes/  
|     └ ci-cd/  
|     └ nginx/  
|  
| └ docs/  
|   |   └ arquitectura/  
|   |   └ diagramas/  
|   |   └ api/  
|   └ decisiones-arquitectura/  
|  
└ README.md
```

8.4. Diagrama de componentes de desarrollo

BackEnd

1. Requerimientos y diseño

- Análisis de requisitos: Identificar las funcionalidades necesarias del sistema.
- Diseño de la arquitectura: Seleccionar la estructura del backend (microservicios, monolito).
- Definición de modelos de datos: Diseñar las bases de datos y las relaciones.
- Definición de la API: Crear un contrato API (endpoints, métodos, formatos).

2. Configuración inicial

- Configuración del entorno de desarrollo (IDE, dependencias, frameworks).
- Creación de repositorios (Git, CI/CD pipelines).
- Configuración del entorno de Cloud Functions.
- Configuración de la base de datos (Firebase).

3. Desarrollo

- Implementación de modelos y esquemas de base de datos.
- Desarrollo de endpoints:
 - Autenticación y autorización (registro, inicio de sesión).
 - Operaciones CRUD (Create, Read, Update, Delete).
 - Integraciones externas (APIs de terceros, servicios de pago).
- Implementación de lógica de negocio.
- Manejo de errores y validación de datos.

4. Pruebas y validación

- Creación de pruebas unitarias para servicios.
- Pruebas de integración con la base de datos.
- Pruebas de rendimiento y escalabilidad.
- Validación de seguridad.

5. Implementación y despliegue

- Preparación de la base de datos.
- Despliegue en producción.
- Configuración de monitoreo y logs.

Frontend

1. Requerimientos y diseño

- Análisis de requisitos: Definir la experiencia del usuario (UX).
- Wireframes y prototipos: Crear bocetos visuales y prototipos interactivos.
- Definición de arquitectura: Estructura de componentes, rutas y estados.

2. Configuración inicial

- Configuración del entorno de desarrollo.
- Instalación de dependencias y configuración de herramientas.
- Configuración del diseño base.

3. Desarrollo

- Creación de componentes base.
- Implementación de rutas y navegación.

- Integración con APIs del backend.
- Desarrollo de lógica de estado.
- Estilizado y responsividad.

4. Pruebas y validación

- Pruebas unitarias de componentes.
- Pruebas funcionales y de interacción.
- Validación de diseño responsivo y accesibilidad.
- Pruebas de rendimiento en el navegador.

5. Implementación y despliegue

- Construcción del proyecto para producción.
- Despliegue en Google Play.
- Validación en entornos reales.
- Configuración de monitoreo.

9. VISTA DE DESPLIEGUE

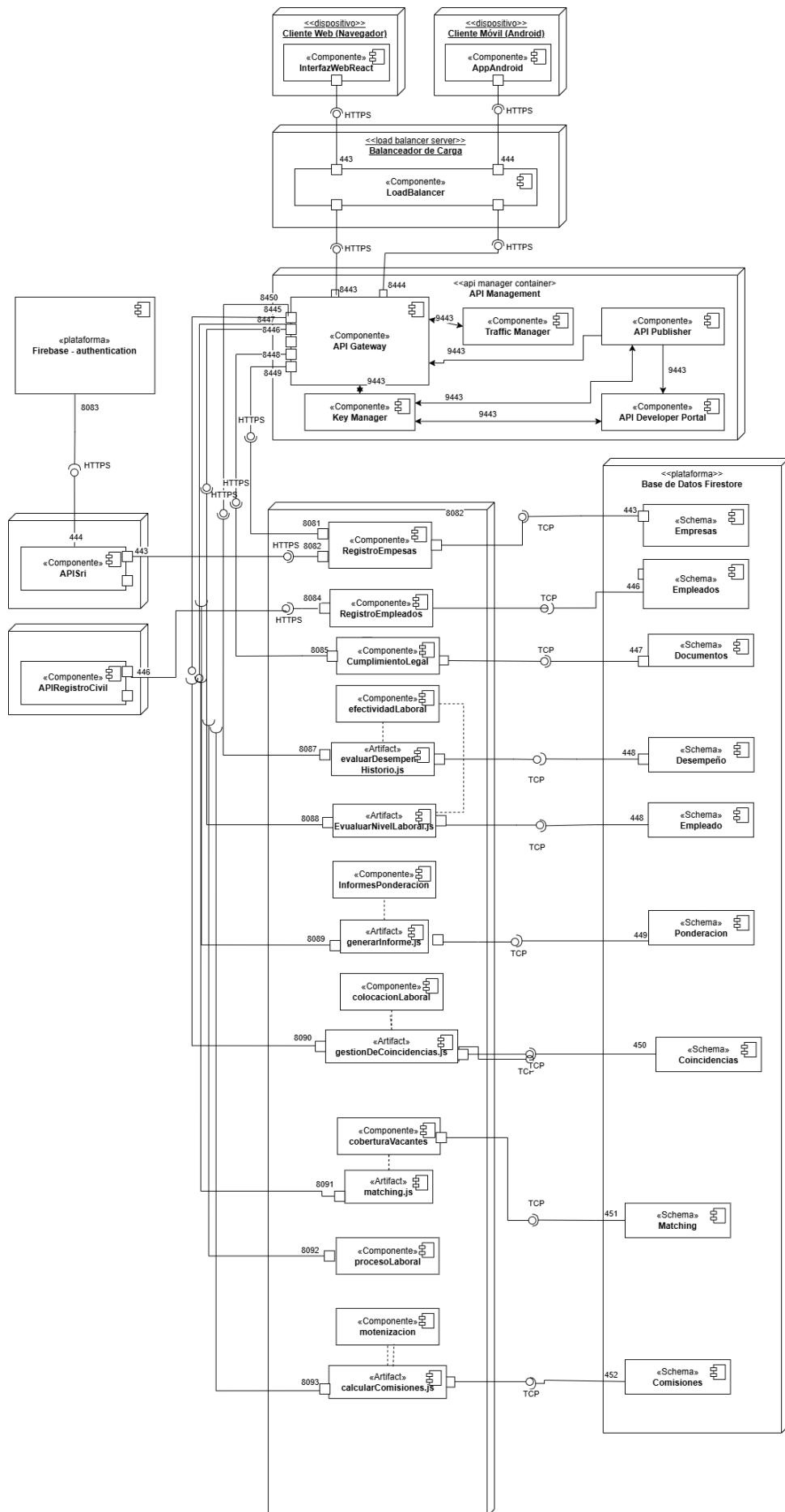
9.1 Visión general

9.2 Diagrama de despliegue

En la vista de despliegue se detallan las tecnologías empleadas en la Bolsa de Empleados Conjunta – CAIL, describiendo la infraestructura sobre la cual se ejecuta la aplicación web y móvil. Esta vista especifica los componentes del sistema, los entornos de ejecución, los servidores involucrados y las herramientas utilizadas para su funcionamiento.

Asimismo, se identifican los puertos, protocolos y tipos de conexión que permiten la comunicación entre los distintos componentes de la arquitectura, así como los mecanismos mediante los cuales se reciben, procesan y responden las solicitudes de los usuarios y de los sistemas externos, garantizando la correcta interoperabilidad, seguridad y disponibilidad de la plataforma.

Figura 16
Diagrama de despliegue



10. VISTA DE DATOS

10.1. Visión general

El Sistema de Bolsa de Empleo CAIL es una plataforma digital especializada en la recolocación de empleados con alta antigüedad laboral (mínimo 8 años de experiencia). El sistema actúa como intermediario inteligente entre empresas emisoras (que buscan reubicar empleados de larga data para evitar costos de liquidación) y empresas receptoras (que buscan talento experimentado con capacidad inmediata de contribución).

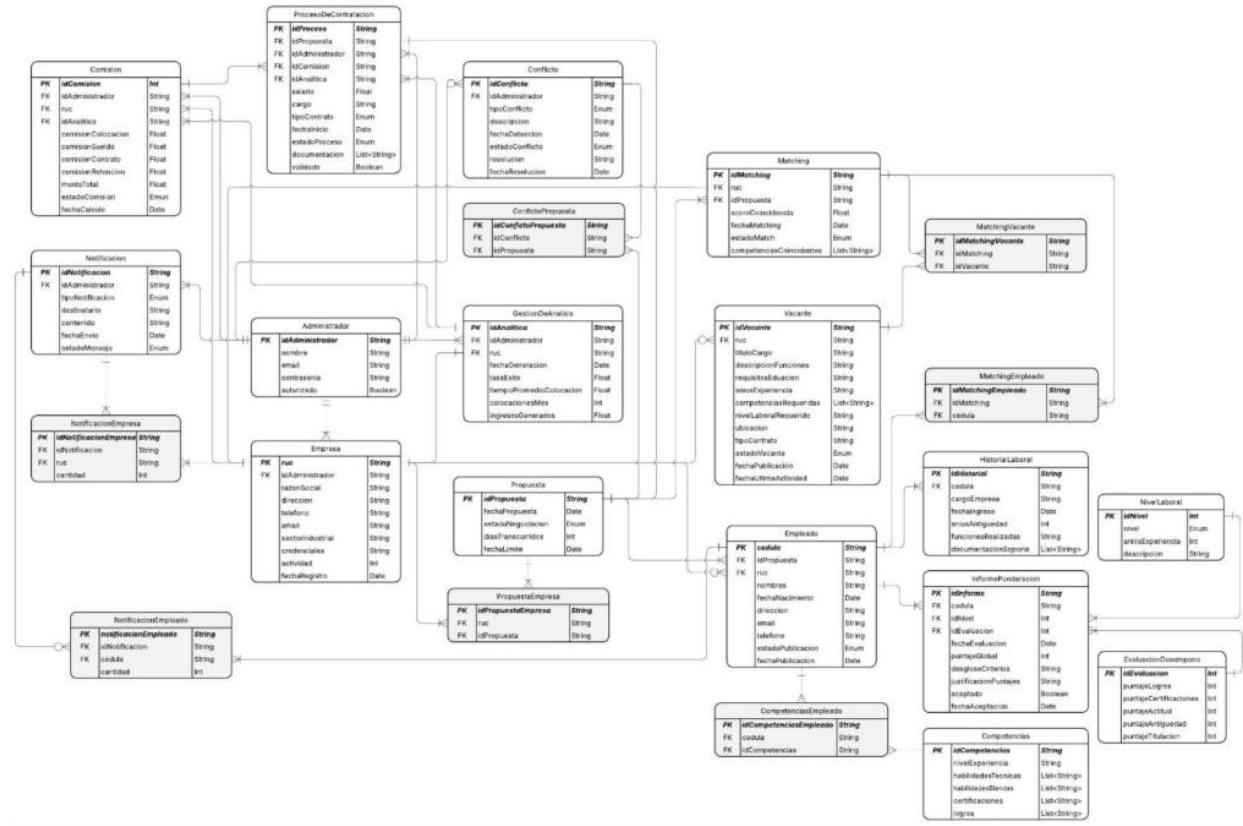
Este diccionario presenta el diseño de la base de datos mediante un modelo relacional normalizado, es decir que, posee tablas, claves primarias, claves foráneas y relaciones explícitas con la finalidad de facilitar la comprensión conceptual y comunicación efectiva con todos los stakeholders del proyecto.

El modelo relacional es universalmente reconocido en la academia y la industria, permitiendo visualizar con claridad las entidades del negocio, sus atributos y las relaciones que las conectan. Sin embargo, la implementación técnica real utilizará Firebase Firestore la cual es una base de datos NoSQL orientada a documentos, esta bd ofrece ventajas significativas para el sistema como: escalabilidad automática sin intervención manual, sincronización de datos en tiempo real para notificaciones instantáneas, modelo de datos flexible que reduce la complejidad de consultas, integración nativa con servicios de autenticación y almacenamiento en la nube y un modelo de costos optimizado para aplicaciones en crecimiento.

La transición del modelo relacional documentado a Firebase es directa, las tablas se implementan como colecciones, las filas como documentos JSON, las claves foráneas como referencias o identificadores embebidos y las relaciones muchos a muchos mediante colecciones intermedias. Esta estrategia dual combina claridad conceptual para el diseño y agilidad técnica para la ejecución.

10.1. Modelo de Datos

*Figura 17
Modelo de datos*



10.2. Diccionarios de Datos

1. Tabla: Empresa

La tabla Empresa representa a todas las organizaciones que participan en el ecosistema CAIL, ya sea como entidades que publican vacantes (empresa emisora) o como entidades que reciben propuestas (empresa receptora). Constituye una de las entidades raíz del sistema.

Función dentro del sistema

Registrar información básica y funcional sobre cada empresa participante, permitiendo la creación de vacantes, propuestas, comisiones, notificaciones y actividades de análisis.

Diccionario de Datos – EMPRESA			
Key	Nombre columna	Tipo dato	de Descripción

PK	ruc	STRING	Identificador y/o registro Único de Contribuyentes. Identificador fiscal oficial compuesto por 13 números.
FK	idAdministrador	STRING	Identificador único del administrador de CAIL que registra y valida la empresa en el sistema.
	razonSocial	STRING	Nombre comercial o razón social de la empresa registrada ante organismos oficiales.
	direccion	STRING	Dirección física de la sede principal de la empresa
	telefono	STRING	Número telefónico de contacto de la empresa.
	email	STRING	Correo electrónico corporativo principal para comunicaciones oficiales y acceso al sistema.
	sectorIndustrial	STRING	Sector económico o industria a la que pertenece la empresa
	credenciales	STRING	Contraseña de acceso a la plataforma, almacenada de forma encriptada.
	actividad	INT	Años de actividad o vigencia de la empresa en el mercado o sector industrial.
	fechaRegistro	DATE	Fecha de creación y activación del perfil de la empresa en el sistema (formato: YYYY-MM-DD).

2. Tabla: Administrador

Los Administradores son usuarios internos del sistema CAIL con permisos especiales. Son responsables de gestionar todo el ecosistema: registran empresas, supervisan procesos de contratación, aprueban comisiones, gestionan análisis del sistema y resuelven conflictos operativos.

Función dentro del sistema

Controlar, monitorear y validar el flujo operativo completo del sistema CAIL, asegurando la integridad de datos, el cumplimiento legal y la calidad de los procesos de recolocación.

Diccionario de Datos – ADMINISTRADOR			
Key	Nombre columna	Tipo de dato	Descripción
PK	idAdministrador	STRING	Identificador único del administrador.
	nombre	STRING	Nombres completos del administrador.
	email	STRING	Correo electrónico institucional único para acceso y notificaciones del sistema.
	contrasenia	STRING	Contraseña de acceso cifrada.
	autorizado	BOOLEAN	Indicador de autorización activa del administrador. (“True”, puede operar en el sistema y “False”, acceso revocado).

11. CALIDAD

11.1. Visión general

La sección de calidad describe como la arquitectura de la Bolsa de Empleados Conjunta – CAIL cumple con los principales atributos de calidad del sistema, tales como extensibilidad, fiabilidad y portabilidad, considerando tanto la aplicación web como móvil y el enfoque DevOps adoptado.

11.2. Atributos de calidad

Extensibilidad

El uso de React en el frontend permite la construcción de componentes reutilizables, facilitando la incorporación de nuevas vistas, funcionalidades o flujos de usuario sin afectar el resto de la aplicación.

En el backend, el uso de Node.js junto con Fastify permite definir módulos independientes para la lógica de negocio, la gestión de usuarios, las vacantes y los procesos de matching. Esta estructura modular facilita la extensión del sistema mediante la adición de nuevos endpoints, servicios o integraciones externas sin necesidad de realizar cambios significativos en los módulos existentes.

Además, la integración con GitHub y GitHub Actions favorece un desarrollo colaborativo y controlado, permitiendo introducir nuevas funcionalidades de forma incremental mediante branches y pipelines automatizados, reduciendo el impacto sobre el sistema en producción.

Fiabilidad

La fiabilidad del sistema se garantiza mediante la automatización de pruebas, el control de errores y la infraestructura en la nube. El uso de Jest permite ejecutar pruebas unitarias e integradas que validan el correcto funcionamiento de los componentes antes de que los cambios sean integrados al repositorio principal, reduciendo la probabilidad de fallos en producción.

El backend desplegado en Firebase Hosting y Cloud Functions proporciona alta disponibilidad y escalabilidad automática, permitiendo que el sistema continúe operando incluso ante picos de carga o fallos temporales de infraestructura. Asimismo, Docker asegura que la aplicación se ejecute en contenedores aislados y consistentes, evitando errores derivados de diferencias entre entornos de desarrollo, prueba y producción.

Finalmente, el monitoreo y las notificaciones mediante Discord permiten detectar incidentes de forma temprana y responder rápidamente ante fallos, mejorando la continuidad operativa del sistema.

Portabilidad

La solución presenta un alto grado de portabilidad gracias al uso de tecnologías multiplataforma y contenedores. El backend desarrollado en Node.js puede ejecutarse en distintos entornos que soporten JavaScript, mientras que el uso de Docker permite desplegar la aplicación en cualquier infraestructura compatible con contenedores, ya sea en la nube o en servidores locales.

La aplicación frontend desarrollada en React puede desplegarse fácilmente en distintos entornos web, mientras que la aplicación móvil se distribuye mediante un Android Package

Kit (APK) generado automáticamente dentro del pipeline de CI/CD, lo que permite su instalación en diversos dispositivos Android sin modificaciones adicionales.

Además, el uso de servicios gestionados como Firebase reduce la dependencia de una infraestructura específica, facilitando una posible migración futura hacia otros proveedores de servicios en la nube con cambios mínimos en la configuración.

Seguridad

La seguridad del sistema se aborda desde múltiples niveles de la arquitectura. El backend implementa mecanismos de autenticación y autorización, asegurando que solo los usuarios autorizados puedan acceder a las funcionalidades del sistema. La comunicación entre el frontend y el backend se realiza mediante protocolos seguros (HTTPS), garantizando la confidencialidad e integridad de la información transmitida.

El uso de Firebase proporciona mecanismos integrados de seguridad, como la gestión segura de credenciales y el control de acceso a los datos. Adicionalmente, la ejecución del sistema en contenedores Docker contribuye a la seguridad al aislar los procesos de la aplicación y reducir la superficie de ataque.

Las pruebas automatizadas y la validación de errores ayudan a prevenir vulnerabilidades comunes, mientras que el monitoreo continuo permite detectar comportamientos anómalos o intentos de acceso no autorizados.

Escalabilidad

La arquitectura del sistema está diseñada para soportar el crecimiento progresivo del número de usuarios y solicitudes.

El uso de Firebase Hosting y Cloud Functions permite una escalabilidad automática, ajustando los recursos del sistema de forma dinámica según la demanda, sin intervención manual. Node.js y Fastify permiten manejar múltiples conexiones simultáneas de manera eficiente, lo que contribuye a un mejor rendimiento del backend bajo alta carga.

Asimismo, la separación entre frontend y backend permite escalar cada componente de forma independiente, optimizando el uso de recursos. Este enfoque garantiza que la plataforma pueda adaptarse al crecimiento de empresas y empleados registrados sin degradar la experiencia del usuario.

Mantenibilidad

La mantenibilidad del sistema se garantiza mediante buenas prácticas de desarrollo, automatización y estandarización. La arquitectura modular facilita la comprensión del sistema y permite realizar cambios o correcciones en componentes específicos sin afectar al resto de la aplicación.

El uso de GitHub Actions automatiza procesos de pruebas, construcción y despliegue, reduciendo errores humanos y facilitando la detección temprana de problemas. Las pruebas automatizadas con Jest aseguran que las modificaciones futuras no introduzcan regresiones en el sistema.

Además, el uso de Docker y la estandarización de los entornos de ejecución simplifican las tareas de mantenimiento, actualización y despliegue, permitiendo una operación más eficiente y sostenible a lo largo del tiempo.

12. SEGURIDAD

12.1. Visión general

Esta sección describe cómo la arquitectura del Sistema de Matching y Colocación Laboral garantiza la integridad, confidencialidad y disponibilidad de la información en todos sus puntos de integración. La estrategia de seguridad se basa en el estándar OWASP (Open Web Application Security Project) para mitigar vulnerabilidades web críticas y asegura el cumplimiento riguroso de la Ley Orgánica de Protección de Datos Personales (LOPD) de Ecuador, protegiendo datos sensibles como el Desempeño Histórico y el Puntaje Global de los Titulares.

12.2. Comunicación

El sistema aplica controles de seguridad en todos los canales de comunicación, integrando:

- **Estándar de Seguridad:** Alineación con ISO/IEC 27001 para la gestión de la seguridad de la información.
- **Dominios de Seguridad:** Control de acceso basado en roles (RBAC), diferenciando claramente los permisos de Empleados, Empresas Emisoras, Empresas Receptoras y Administradores.
- **Mitigación de Riesgos:** Aplicación de controles específicos frente a amenazas del OWASP Top 10, tales como Inyección, Control de Acceso Deficiente y Fallos Criptográficos.

12.3 Seguridad en los canales de comunicación

La protección de los canales se segmenta de la siguiente forma:

- **Segmento Externo (Front-to-Back):**

Comunicación cifrada mediante TLS 1.2 o superior (HTTPS) en el puerto 443, garantizando la confidencialidad e integridad de los datos en tránsito.

- **Segmento Interno (Back-to-DB):**

Comunicación restringida mediante firewalls y reglas de red que permiten tráfico únicamente desde direcciones IP autorizadas.

- **Intercambio de Mensajería y APIs:**

Uso de mecanismos de autenticación, firma de mensajes y validación de tokens para prevenir manipulación o suplantación de identidad.

12.4. Autenticación y Autorización

Las interfaces de administración se protegen bajo capas adicionales de seguridad:

- Autenticación: Se soportan mecanismos de autenticación robustos, incluyendo la integración con LDAP/Active Directory para usuarios corporativos y autenticación multifactor (MFA) obligatoria.
- Almacenamiento de Credenciales: Las contraseñas nunca se almacenan en texto plano; se utiliza el algoritmo de hashing Argon2 o Bcrypt con "salts" aleatorios para prevenir ataques de diccionario.
- Autorización: El acceso administrativo se rige por el principio de Mínimo Privilegio, asegurando que cada administrador solo pueda acceder a las funciones necesarias para su cargo (Auditoría, Gestión de Usuarios, o Monitoreo Técnico).

12.4. Seguridad de aplicaciones web OWASP

La arquitectura del sistema ha sido diseñada para neutralizar los 10 riesgos más críticos definidos por el Open Web Application Security Project (OWASP). A continuación, se detalla la estrategia de resolución técnica para cada uno:

ID	Riesgo OWASP	Implementación y Resolución Técnica en el Sistema
A01	Broken Access Control	Se implementa un Middleware de Autorización en el Backend que valida el ID del usuario contra el ID del recurso solicitado en cada transacción. Se prohíben las Referencias Directas a Objetos Inseguros (IDOR).
A02	Cryptographic Failures	Uso de TLS 1.2+ para transporte. Los datos sensibles en la base de datos (Puntaje Global y Desempeño) se cifran mediante AES-256. Las contraseñas utilizan Argon2 con sal única por usuario.
A03	Injection	Se utiliza un ORM (Object-Relational Mapping) que parametriza automáticamente todas las consultas SQL. Se aplica validación de "lista blanca" en todas las entradas del servidor para evitar inyecciones de comandos.
A04	Insecure Design	La arquitectura incluye flujos de "revisión humana" obligatoria para evitar decisiones laborales basadas puramente en algoritmos, cumpliendo con el Art. 20 de la LOPDP.
A05	Security Misconfiguration	Endurecimiento (<i>Hardening</i>) del servidor: se han eliminado servicios innecesarios, se ocultan banners de versión y se aplican cabeceras de seguridad (HSTS, X-Content-Type-Options).
A06	Vulnerable and Outdated Components	Se integra OWASP Dependency-Check en el pipeline de CI/CD para escanear y bloquear el uso de librerías de terceros con vulnerabilidades conocidas (CVEs).
A07	Identification and Authentication Failures	Implementación de MFA (Multi-Factor Authentication) para roles administrativos y políticas de bloqueo de cuenta tras 5 intentos fallidos para prevenir fuerza bruta.

A08	Software and Data Integrity Failures	Verificación de integridad mediante <i>hashes</i> para los documentos cargados (Certificados/CVs). Los despliegues de código requieren firmas digitales para asegurar que el código no fue alterado.
A09	Security Logging and Monitoring	Centralización de logs de eventos críticos. Cualquier fallo de autenticación o acceso denegado genera una alerta en tiempo real al equipo de respuesta a incidentes (IRT).
A10	Server-Side Request Forgery (SSRF)	El backend no procesa URLs suministradas por el usuario hacia recursos internos de la red. Se aplica una lista blanca estricta para cualquier integración con APIs externas.

12.5. Seguridad de aplicaciones móviles OWASP

La arquitectura del sistema ha sido diseñada para neutralizar los 10 riesgos más críticos definidos por el Open Web Application Security Project (OWASP). A continuación, se detalla la estrategia de resolución técnica para cada uno:

ID	Riesgo OWASP	Implementación y Resolución Técnica en el Sistema
A01	Broken Access Control	Se implementa un Middleware de Autorización en el Backend que valida el ID del usuario contra el ID del recurso solicitado en cada transacción. Se prohíben las Referencias Directas a Objetos Inseguros (IDOR).
A02	Cryptographic Failures	Uso de TLS 1.2+ para transporte. Los datos sensibles en la base de datos (Puntaje Global y Desempeño) se cifran mediante AES-256. Las contraseñas utilizan Argon2 con sal única por usuario.
A03	Injection	Se utiliza un ORM (Object-Relational Mapping) que parametriza automáticamente todas las consultas SQL. Se aplica validación de "lista blanca" en todas las entradas del servidor para evitar inyecciones de comandos.
A04	Insecure Design	La arquitectura incluye flujos de "revisión humana" obligatoria para evitar decisiones laborales basadas puramente en algoritmos, cumpliendo con el Art. 20 de la LOPDP.

A05	Security Misconfiguration	Endurecimiento (<i>Hardening</i>) del servidor: se han eliminado servicios innecesarios, se ocultan banners de versión y se aplican cabeceras de seguridad (HSTS, X-Content-Type-Options).
A06	Vulnerable and Outdated Components	Se integra OWASP Dependency-Check en el pipeline de CI/CD para escanear y bloquear el uso de librerías de terceros con vulnerabilidades conocidas (CVEs).
A07	Identification and Authentication Failures	Implementación de MFA (Multi-Factor Authentication) para roles administrativos y políticas de bloqueo de cuenta tras 5 intentos fallidos para prevenir fuerza bruta.
A08	Software and Data Integrity Failures	Verificación de integridad mediante <i>hashes</i> para los documentos cargados (Certificados/CVs). Los despliegues de código requieren firmas digitales para asegurar que el código no fue alterado.
A09	Security Logging and Monitoring	Centralización de logs de eventos críticos. Cualquier fallo de autenticación o acceso denegado genera una alerta en tiempo real al equipo de respuesta a incidentes (IRT).
A10	Server-Side Request Forgery (SSRF)	El backend no procesa URLs suministradas por el usuario hacia recursos internos de la red. Se aplica una lista blanca estricta para cualquier integración con APIs externas.

12.6. Análisis de Amenazas y Riesgos

Se incorpora un análisis de riesgos detallado basado en OWASP Top 10 y OWASP Mobile Top 10, mediante una matriz de amenazas que clasifica cada riesgo según:

- Nivel: Alto / Medio / Bajo
- Impacto: Confidencialidad, Integridad y Disponibilidad
- Mitigación: Controles técnicos específicos

ID	Amenaza (OWASP)	Descripción del riesgo	Nivel de riesgo	Impacto	Controles y mitigación
R1	Inyección (A03)	Inserción de código malicioso mediante entradas no validadas que puede	Alto	Integridad y confidencialidad	Uso de consultas parametrizadas, ORM, validación de entradas por lista

		alterar consultas o lógica del sistema.			blanca y sanitización de datos.
R2	Broken Access Control (A01)	Acceso no autorizado a recursos o funciones restringidas por fallos en la validación de permisos.	Alto	Confidencialidad	Implementación de RBAC, validación de identidad por middleware y prevención de IDOR.
R3	Fallos criptográficos (A02)	Exposición de datos sensibles por uso incorrecto o ausencia de cifrado.	Medio	Confidencialidad	Cifrado AES-256 para datos sensibles, TLS 1.2+ para datos en tránsito y gestión segura de claves.
R4	Configuración insegura (A05)	Uso de configuraciones por defecto o servicios innecesarios que incrementan la superficie de ataque.	Medio	Disponibilidad	Hardening de servidores, uso de cabeceras de seguridad y revisión periódica de configuraciones.
R5	Componentes vulnerables u obsoletos (A06)	Uso de librerías con vulnerabilidades conocidas que pueden ser explotadas.	Medio	Integridad	Escaneo automático con OWASP Dependency-Check y actualización periódica de dependencias.
R6	Fallos de autenticación (A07)	Compromiso de cuentas por contraseñas débiles o ataques de fuerza bruta.	Alto	Confidencialidad	Autenticación multifactor (MFA), políticas de bloqueo y hashing seguro con Argon2/Bcrypt.
R7	Registro y monitoreo insuficiente (A09)	Falta de detección oportuna de incidentes de seguridad.	Medio	Disponibilidad	Centralización de logs, alertas en tiempo real y monitoreo continuo.
R8	SSRF (A10)	Uso de URLs manipuladas para acceder a recursos internos.	Bajo	Integridad	Listas blancas de URLs y restricción de llamadas a servicios internos.

13. INFORMACIÓN DE CONTACTO

Steven Neira, Software Architect: 098 689 7878

Veronica Luna, Database Analyst: 099 131 7177

Jhandry Solorzano, Security Analyst: 096 928 0380

Nahomy Cabrera, Ux/UI: 096 846 7693

Camilo Lopez, Frontend Developer: 098 920 4851

Sofia Vire, Backend Developer: 0981867742