Documento de Diseño de Arquitectura (ADD)

Documento de arquitectura de alto nivel se sugieren los siguientes elementos:

1. Introducción

- **Propósito del Documento:**

  El presente documento tiene como objetivo principal la definición de la arquitectura de software para la construcción del aplicativo ***"Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)"***. Este sistema está diseñado para optimizar la gestión de vacantes, postulaciones y la evaluación de candidatos, facilitando la transparencia, eficiencia y trazabilidad durante todo el proceso de selección. La arquitectura propuesta se enfoca en proporcionar una solución escalable, segura y fácil de usar para empresas de diferentes tamaños y sectores, permitiendo que los responsables de recursos humanos gestionen los procesos de selección de manera independiente pero bajo una infraestructura tecnológica común.

  El sistema utilizará tecnologías modernas como React y Next.js para el desarrollo del frontend, con un backend robusto basado en Node.js y PostgreSQL, asegurando que se cumplan los requisitos funcionales y no funcionales definidos en las fases anteriores del proyecto. Además, se integrarán servicios externos como Firebase Auth para la autenticación segura de usuarios y Socket.io para la comunicación en tiempo real entre usuarios.

  Este documento también aborda los componentes clave de la arquitectura, la elección de las tecnologías, las soluciones de almacenamiento, los servicios en la nube, y las medidas de seguridad que garantizarán el correcto funcionamiento del sistema, a la vez que se prioriza la experiencia del usuario y la adaptabilidad a futuras necesidades del sistema.

- **Alcance**:

El presente documento de arquitectura de software define el diseño técnico y estructural de la plataforma Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub), orientada a la optimización y automatización de los procesos de reclutamiento y selección de personal. El objetivo principal de este sistema es proporcionar a empresas, departamentos de recursos humanos, y candidatos una solución eficiente, transparente y escalable que facilite la gestión de vacantes, postulaciones, entrevistas y selección de manera totalmente digitalizada.

El alcance de esta solución abarca todas las funcionalidades esenciales para gestionar usuarios (registro, autenticación, asignación de roles), la creación y administración de vacantes, el seguimiento detallado de los estados de las postulaciones, la notificación en tiempo real a candidatos y reclutadores, la evaluación de candidatos y la generación de reportes estadísticos. Además, el sistema incluye herramientas para la colaboración entre los miembros del equipo de selección, como comentarios y retroalimentación interna sobre los candidatos, mejorando la comunicación y las decisiones estratégicas.

Desde el punto de vista técnico, este documento describe la arquitectura por capas del sistema, sus componentes lógicos, los flujos de información entre las diferentes funcionalidades, el diseño de la base de datos, y las vistas de despliegue. La solución está diseñada para operar en un entorno web, utilizando tecnologías modernas que permiten escalabilidad y mantenimiento ágil, basándose en una arquitectura de servicios desacoplados. Esta estructura garantiza que el sistema sea fácilmente mantenible y ampliable, permitiendo adaptaciones y mejoras futuras.

Cabe destacar que en esta fase no se contempla la integración con sistemas externos como redes sociales profesionales ni la personalización de la interfaz para dispositivos móviles. Sin embargo, estas funcionalidades podrán considerarse en futuras iteraciones del proyecto.

- **Audiencia:**

1.1 Administradores de Empresas (Clientes)

Los administradores de empresas son los principales usuarios del sistema, encargados de gestionar todo el proceso de selección de personal, desde la creación de vacantes hasta la evaluación final de los candidatos. Su principal interés es contar con una plataforma eficiente que les permita gestionar múltiples vacantes, realizar seguimientos detallados del estado de las postulaciones y obtener reportes clave sobre la efectividad del proceso de selección. Además, necesitan un sistema seguro que garantice la protección de los datos sensibles de los candidatos y les permita gestionar los permisos de acceso de los usuarios dentro de la plataforma, garantizando un control adecuado de la información.

1.2 Recursos Humanos / Reclutadores

El departamento de recursos humanos y los reclutadores son usuarios esenciales del sistema, pues son los encargados de interactuar directamente con los candidatos durante todo el proceso de selección. Su interés principal es contar con herramientas que les permitan recibir, organizar y evaluar las postulaciones de forma ágil, así como asignar y gestionar las entrevistas. Buscan una plataforma que les permita filtrar candidatos de manera eficiente, colaborar dentro del equipo de selección y tener acceso a métricas y reportes que les ayuden a tomar decisiones informadas y mejorar continuamente el proceso de selección.

1.3 Candidatos (Usuarios Finales)

Los candidatos son los usuarios finales de la plataforma, y su experiencia es clave para el éxito del sistema. Su principal interés es tener un proceso de postulación claro, accesible y transparente, que les permita postularse a vacantes de manera sencilla, gestionar su perfil profesional y consultar el estado de sus postulaciones en tiempo real. Los candidatos también valoran la seguridad de sus datos personales y el seguimiento claro de su proceso de selección. La plataforma debe garantizarles una experiencia fluida y sin fricciones, desde la postulación hasta la notificación final.

1.4 Desarrolladores y Equipo Técnico

El equipo de desarrollo y los responsables técnicos del proyecto tienen como interés diseñar y construir una plataforma robusta, escalable y fácil de mantener. Se encargan de implementar la arquitectura técnica, las funcionalidades del sistema y de garantizar que la plataforma cumpla con los

requisitos funcionales y no funcionales establecidos. Este equipo también es responsable de la integración de servicios y tecnologías, como bases de datos, autenticación, almacenamiento de archivos, y notificaciones en tiempo real. Su objetivo es proporcionar un sistema eficiente, seguro y optimizado que sea fácilmente extensible para futuras mejoras y adaptaciones.

1.5 Stakeholders Académicos (Profesor y Supervisores del Proyecto)

Los stakeholders académicos, como el profesor y los supervisores del proyecto, tienen un rol crucial en la supervisión y evaluación del desarrollo del sistema. Su principal interés es asegurar que el proyecto esté alineado con los objetivos académicos, garantizando que se sigan las mejores prácticas de desarrollo de software, que se cumplan los plazos establecidos y que el sistema cumpla con los requisitos definidos en el marco académico. También proporcionan retroalimentación sobre la viabilidad y calidad del proyecto, asegurando que el mismo se mantenga dentro de los lineamientos definidos.

1.6 Usuarios Administrativos del Sistema (Superadministradores)

Los superadministradores son responsables de la gestión general de la plataforma, incluyendo la creación de cuentas de usuario, la asignación de roles y permisos, y la configuración del sistema a nivel global. Su interés principal es garantizar la correcta administración de las empresas dentro de la plataforma, supervisando los diferentes usuarios y asegurando que cada uno tenga acceso a las funcionalidades correspondientes según su rol. También deben gestionar aspectos como la seguridad de la información y el control sobre los datos almacenados.

1.7 Usuarios Externos y Partners (Posibles Integraciones Futuras)

Los usuarios externos y partners son aquellas entidades o sistemas de terceros con los cuales se podría integrar la plataforma en futuras fases del proyecto. Estos interesados tienen como principal interés ampliar las funcionalidades del sistema mediante la integración con plataformas de evaluación externa, redes sociales profesionales (como LinkedIn) y herramientas de videoconferencia para entrevistas. Aunque las integraciones externas no se consideran en esta fase, el diseño modular del sistema permite adaptaciones y ampliaciones futuras.

1.8 Usuarios de Soporte y Mantenimiento

Los usuarios de soporte y mantenimiento son los encargados de garantizar que el sistema se mantenga operativo y funcional después de su implementación. Su interés radica en gestionar incidencias técnicas, realizar mantenimiento preventivo y actualizaciones periódicas del sistema, asegurando que este se mantenga eficiente y seguro a lo largo del tiempo. Además, se encargan de gestionar las solicitudes de soporte de los usuarios finales (candidatos, reclutadores, administradores) y de solucionar cualquier problema que pueda surgir en el uso del sistema.

- **Glosario:**

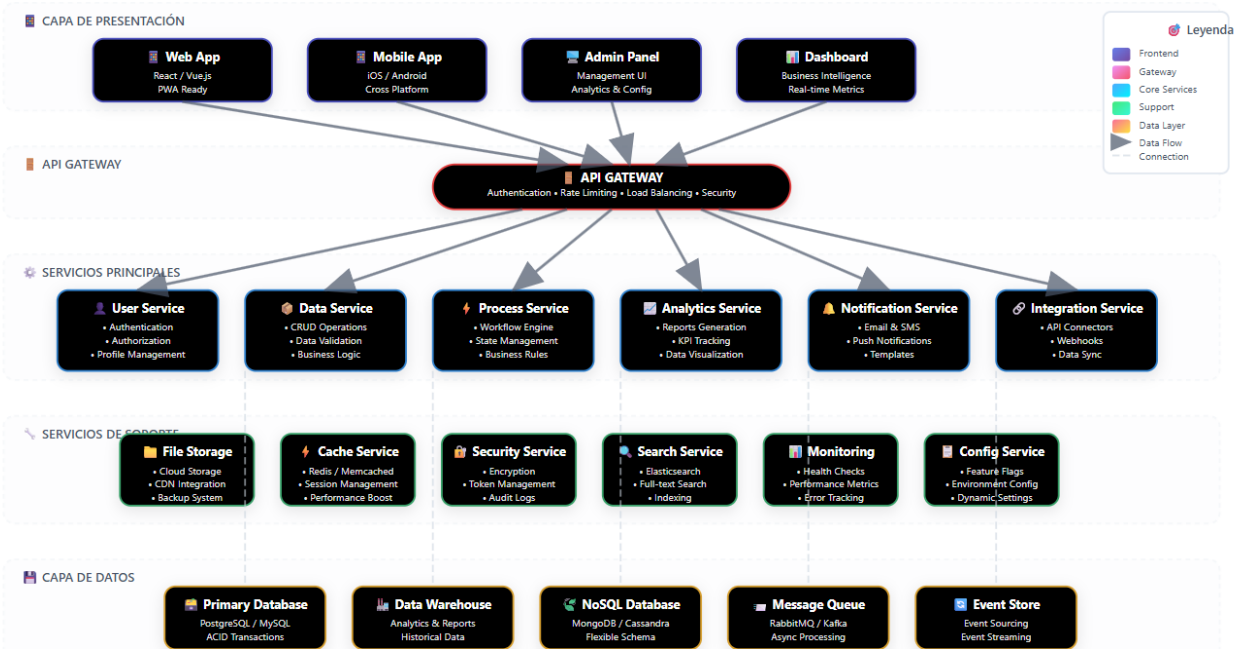
Abreviatura / Término	Definición
Frontend	Parte visual o cliente del sistema. En este proyecto se desarrolla con React, Next.js y Tailwind CSS.
React	Librería de JavaScript para construir interfaces de usuario dinámicas y reactivas.
Next.js	Framework basado en React que permite renderizado del lado del servidor (SSR) y generación estática de páginas (SSG).
Tailwind CSS	Framework de diseño CSS basado en utilidades para diseño rápido y responsivo.
Backend	Parte lógica y funcional del sistema que gestiona la lógica de negocio y el acceso a datos.
Node.js	Entorno de ejecución de JavaScript en el servidor, utilizado en el backend del sistema.

Express.js	Framework web para Node.js utilizado para gestionar rutas, peticiones HTTP y middleware en el backend.
SupaBase	Plataforma de backend como servicio basada en PostgreSQL que proporciona base de datos, autenticación, almacenamiento y APIs listas para usar. En este proyecto se utiliza como motor de base de datos y entorno backend.
PostgreSQL	Sistema de gestión de bases de datos relacional utilizado como motor de base de datos en este proyecto.
Firebase Auth	Servicio de autenticación de Google utilizado para el registro e inicio de sesión de usuarios mediante tokens seguros.
Cloud Storage	Servicio de almacenamiento en la nube para guardar archivos, como PDFs, imágenes y recursos educativos.
Socket.io	Biblioteca para la comunicación en tiempo real entre cliente y servidor, utilizada para funcionalidades de chat y notificaciones.
PDFKit	Librería para la generación dinámica de documentos PDF, utilizada para emitir certificados automáticos.
API REST	Interfaz de programación de aplicaciones basada en HTTP para la comunicación entre frontend y backend. En este proyecto, las APIs son gestionadas mediante Express.js.
JWT (JSON Web Tokens)	Mecanismo de autenticación basado en tokens, utilizado para mantener la sesión del usuario.
Nginx	Servidor web utilizado como reverse proxy para dirigir las peticiones hacia el backend y mejorar el rendimiento del sistema.

Docker	Plataforma de contenedores utilizada para empaquetar y desplegar el backend y frontend en un entorno controlado y replicable.

2. Visión General de la Arquitectura

- Diagrama de la Arquitectura:



- **Descripción de las Capas**:

1. Capa de Presentación

- **Responsabilidad**: Esta capa es la encargada de la interacción del usuario con el sistema. Incluye todas las interfaces visuales (web, móvil, panel de administración, y dashboard).

- **Componentes**:
    - **Web App**: Es una aplicación de interfaz web desarrollada con tecnologías como React o Vue.js, que está lista para ser utilizada como una Progressive Web App (PWA).
    - **Mobile App**: Una aplicación móvil desarrollada para plataformas iOS y Android, que ofrece funcionalidad en dispositivos móviles de manera cross-platform.
    - **Admin Panel**: Un panel para la gestión del sistema, donde los administradores pueden configurar y gestionar usuarios, datos y más.
    - **Dashboard**: Una interfaz para análisis y visualización de métricas e informes en tiempo real, orientada a proporcionar información a los usuarios.

2. Capa de Aplicación

- **Responsabilidad**: Esta capa gestiona la lógica del negocio, autenticación, seguridad y servicios que permiten que la aplicación funcione correctamente.
- **Componentes**:
    - **API Gateway**: Gestiona la autenticación, la limitación de tasas, el balanceo de carga y la seguridad, redirigiendo las peticiones de los usuarios a los servicios correspondientes.
    - **User Service**: Maneja la autenticación, autorización y gestión de perfiles de usuario.
    - **Data Service**: Se encarga de las operaciones CRUD, validación de datos y lógica de negocio.
    - **Process Service**: Gestiona el motor de flujos de trabajo, administración de estados y reglas de negocio.

- **Analytics Service**: Genera informes, realiza el seguimiento de KPIs y proporciona visualizaciones de datos.
- **Notification Service**: Permite enviar notificaciones por correo electrónico, SMS y otros canales de comunicación.
- **Integration Service**: Gestiona la integración con otros sistemas mediante conectores API y sincronización de datos.

3. Capa de Datos

- **Responsabilidad**: Esta capa se ocupa del almacenamiento de datos y de la gestión de bases de datos, tanto relacionales como no relacionales.
- **Componentes**:
  - **Primary Database**: Base de datos principal para almacenar información estructurada (por ejemplo, PostgreSQL o MySQL) con transacciones ACID.
  - **Data Warehouse**: Almacén de datos que contiene informes y análisis históricos.
  - **NoSQL Database**: Base de datos no relacional como MongoDB o Cassandra, utilizada para datos flexibles y escalables.
  - **Message Queue**: Cola de mensajes como RabbitMQ o Kafka, para procesar datos de manera asincrónica.
  - **Event Store**: Almacena eventos de manera persistente para el procesamiento de eventos.
- **Tecnologías Clave**:

1. Capa de Presentación

- **Web App:**
    - React.js / Vue.js: Frameworks JavaScript para la construcción de interfaces de usuario interactivas en aplicaciones web.
    - HTML5 / CSS3: Lenguajes estándar para la estructuración y el diseño de las páginas web.
    - JavaScript (ES6+): Lenguaje de programación para añadir interactividad y lógica en el frontend.
    - PWA (Progressive Web App): Tecnología para que la web funcione como una app nativa en dispositivos móviles, con capacidades de almacenamiento offline y notificaciones push.
- **Mobile App:**
    - React Native / Flutter: Frameworks para el desarrollo de aplicaciones móviles multiplataforma (iOS/Android).
    - Swift (para iOS) y Kotlin (para Android): Lenguajes nativos para aplicaciones móviles (si se opta por desarrollo nativo).
- **Admin Panel:**
    - React.js / Vue.js: Frameworks JavaScript para crear una interfaz dinámica y fácil de usar para los administradores.
    - HTML5 / CSS3 / JavaScript: Para la creación de interfaces de usuario (UI) atractivas y funcionales.
- **Dashboard:**
    - React.js / Vue.js: Frameworks de JavaScript para crear dashboards interactivos.
    - D3.js / Chart.js: Bibliotecas para la visualización de datos e informes en tiempo real.

- HTML5 / CSS3: Lenguajes para la estructura y el estilo del dashboard.

2. Capa de Aplicación

- **API Gateway:**
  - Node.js / Express: Framework para la creación del servidor API que maneje la autenticación, seguridad y el enrutamiento de las solicitudes.
  - Nginx / HAProxy: Servidores de proxy reverso para equilibrar la carga entre los servicios.
- **User Service:**
  - Node.js / Express: Para la creación de servicios RESTful que manejen la autenticación y autorización de usuarios.
  - JWT (JSON Web Tokens): Para manejar la autenticación y autorización.
  - OAuth2: Protocolo para autorización segura en aplicaciones web.
- **Data Service:**
  - Node.js / Express: Para gestionar las operaciones CRUD y la validación de datos.
  - GraphQL: Si se usa un enfoque flexible para la consulta de datos.
- **Process Service:**
  - Node.js / Express: Para gestionar la lógica de flujos de trabajo y procesos.
  - Apache Kafka / RabbitMQ: Para manejar eventos y comunicación asincrónica entre servicios.
- **Analytics Service:**

- Python / R: Lenguajes para procesamiento y análisis de datos.
- D3.js / Chart.js: Para la visualización de métricas e informes.

- **Notification Service:**
  - Node.js / Express: Para gestionar las notificaciones y enviar correos electrónicos o SMS.
  - Twilio / Firebase Cloud Messaging (FCM): Para el envío de mensajes SMS y notificaciones push.

- **Integration Service:**
  - Node.js / Express: Para manejar la integración con sistemas externos.
  - REST APIs: Para la interacción con sistemas de terceros.
  - SOAP / GraphQL: Para la integración más detallada o para servicios específicos.

3. Capa de Datos

- **Primary Database:**
  - PostgreSQL / MySQL: Bases de datos relacionales, utilizadas para almacenar datos estructurados y transacciones ACID.

- **Data Warehouse:**
  - Google BigQuery / Amazon Redshift: Para el análisis y almacenamiento de grandes volúmenes de datos históricos.
  - ETL (Extract, Transform, Load) Tools: Para la ingesta y transformación de datos.

- **NoSQL Database:**
    - MongoDB / Cassandra: Bases de datos no relacionales utilizadas para manejar datos semi-estructurados o no estructurados.
- **Message Queue:**
    - RabbitMQ / Apache Kafka: Para manejar la mensajería asincrónica entre microservicios.
- **Event Store:**
    - Apache Kafka: Para almacenar eventos y permitir la persistencia de eventos en una arquitectura de Event Sourcing.
- **Cache Service:**
    - Redis / Memcached: Para mejorar el rendimiento mediante almacenamiento en caché de datos temporales.
- **Search Service:**
    - Elasticsearch: Para realizar búsquedas rápidas y eficientes en grandes volúmenes de datos.
- **Monitoring:**
    - Prometheus / Grafana: Herramientas para el monitoreo y análisis de métricas en tiempo real.
- **Config Service:**
    - Consul / Spring Cloud Config: Para manejar la configuración dinámica y las banderas de características.

- **Consideraciones de Calidad**

REQUERIMIENTOS NO FUNCIONALES

Los requerimientos no funcionales definen atributos de calidad del sistema, como seguridad, usabilidad, rendimiento y disponibilidad. Aunque no describen funciones específicas, son esenciales para evaluar el comportamiento y la experiencia general del usuario.

Cada requerimiento, identificado con el prefijo RNF, representa una necesidad técnica o de servicio expresada por el usuario. Estos se especifican de forma que puedan ser evaluados con métricas concretas durante las pruebas, y su cumplimiento asegura que el sistema mantenga un estándar técnico aceptable.

Seguridad

Garantiza que el sistema proteja la información y los recursos frente a accesos no autorizados, mediante autenticación, autorización, cifrado y control de sesiones

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF—4
Nombre del Requerimiento	Seguridad
Descripción del Requerimiento	El sistema debe garantizar la protección de datos sensibles de los candidatos y usuarios, implementando cifrado AES-256 en tránsito y en reposo. Además, se deben seguir las mejores prácticas de autenticación como 2FA (autenticación de dos factores) para acceso de usuarios con privilegios elevados (administradores, recursos humanos).

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)

ID	RNF—5
Nombre del Requerimiento	Autenticación y Control de Acceso
Descripción del Requerimiento	El sistema debe permitir el registro, inicio de sesión, recuperación de contraseña y autenticación mediante tokens seguros (JWT), con control de acceso basado en roles (administrador, recursos humanos, usuario) para gestionar permisos.

Usabilidad

Se refiere a la facilidad con la que los usuarios pueden aprender a utilizar el sistema y operar sus funcionalidades de manera eficiente y satisfactoria.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-1
Nombre del Requerimiento	Interfaz intuitiva y fácil de navegar
Descripción del Requerimiento	El sistema debe proporcionar una interfaz intuitiva y de fácil navegación, permitiendo que los usuarios sin conocimientos técnicos puedan realizar las tareas principales sin dificultad.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-3
Nombre del Requerimiento	Mensajes de error claros y específicos
Descripción del Requerimiento	El sistema debe proporcionar mensajes de error claros y específicos, indicando la causa del problema y sugiriendo posibles soluciones, en lugar de mostrar mensajes genéricos.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-13
Nombre del Requerimiento	Usabilidad
Descripción del Requerimiento	El sistema debe ofrecer una interfaz de usuario intuitiva, con un diseño claro y fácil de navegar, permitiendo que los usuarios sin conocimientos técnicos puedan realizar las tareas principales sin dificultad.

Rendimiento

Evalúa la capacidad del sistema para responder rápidamente ante solicitudes, manejar múltiples usuarios concurrentes y ejecutar tareas en tiempos aceptables.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-2
Nombre del Requerimiento	Rendimiento
Descripción del Requerimiento	El sistema debe ser capaz de gestionar al menos 500 postulantes por vacante y 50 vacantes activas simultáneamente sin afectar la velocidad de carga de la interfaz ni la respuesta en tiempo real. Las consultas y búsquedas deben responder en un tiempo máximo de 2 segundos.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-6
Nombre del Requerimiento	Integridad de Datos
Descripción del Requerimiento	El sistema debe garantizar que los datos de los candidatos, vacantes y procesos de selección sean consistentes y no se pierdan debido a errores del sistema. Se implementará ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) en la base de datos para mantener la integridad.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-7
Nombre del Requerimiento	Compatibilidad con Navegadores
Descripción del Requerimiento	El sistema debe ser compatible con los navegadores más utilizados, como Google Chrome, Mozilla Firefox, Microsoft Edge y Safari. Debe ser completamente funcional en versiones actuales y anteriores de estos navegadores.

Disponibilidad

Indica qué tan accesible y operativa está la plataforma en un periodo determinado. Un sistema con alta disponibilidad está diseñado para minimizar el tiempo de inactividad.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-9
Nombre del Requerimiento	Respaldo y Recuperación
Descripción del Requerimiento	El sistema debe realizar respaldos automáticos diarios de toda la base de datos, garantizando que los datos puedan ser recuperados en caso de fallos graves del

	sistema. Los respaldos deben almacenarse en un servidor seguro y ser accesibles solo por los administradores.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-10
Nombre del Requerimiento	Disponibilidad
Descripción del Requerimiento	El sistema debe tener un tiempo de disponibilidad del 99.9% durante el horario laboral (9:00 AM a 6:00 PM), con tolerancia a fallos y mecanismos de backup diario.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-12
Nombre del Requerimiento	Interoperabilidad
Descripción del Requerimiento	El sistema debe ser capaz de integrarse fácilmente con otros sistemas en el futuro, como plataformas de evaluación externa o sistemas de recursos humanos mediante API RESTful, garantizando que las integraciones no interrumpan el funcionamiento principal de la plataforma.

Mantenibilidad

Se refiere a qué tan fácil es realizar modificaciones, correcciones o mejoras en el sistema, sin afectar negativamente su funcionamiento.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-14
Nombre del Requerimiento	Documentación técnica y de usuario actualizada
Descripción del Requerimiento	El sistema debe contar con documentación técnica y de usuario actualizada, que facilite la administración, solución de problemas y futuras mejoras.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-15
Nombre del Requerimiento	Código flexible y modular para corrección rápida de errores
Descripción del Requerimiento	El sistema debe tener un código flexible y modular que permita corregir errores en un máximo de 48 horas desde su detección.

Módulo	Sistema de Gestión de Candidatos y Procesos de Selección (Talento-Hub)
ID	RNF-16
Nombre del Requerimiento	Integración de nuevas funcionalidades sin afectar existentes
Descripción del Requerimiento	El sistema debe permitir la integración de nuevas funcionalidades sin afectar las existentes,garantizando la estabilidad del sistema.

- **Riesgos Técnicos Iniciales (opcional)**

Riesgo Técnico	Descripción	Impacto	Mitigación Propuesta
Desconocimiento de tecnologías específicas	El equipo puede no estar familiarizado con algunas tecnologías clave como GraphQL, Kafka o Redis.	Alto	Capacitación previa, investigación técnica, uso de documentación oficial, asignar roles según experiencia y conocimiento.
Dependencia de terceros (APIs externas)	La integración con APIs externas (p.ej., Twilio, Google OAuth) podría sufrir interrupciones o cambios inesperados.	Medio	Monitoreo continuo de las dependencias, uso de pruebas de integración y manejo adecuado de errores.
Escalabilidad de la base de datos	Si no se diseña correctamente, la base de datos (relacional o no relacional) podría tener dificultades al escalar en un futuro.	Alto	Realizar pruebas de carga y dimensionamiento adecuado. Utilizar bases de datos distribuidas o particionadas si es necesario.

Sincronización de datos entre servicios	El sistema de mensajería (Kafka/RabbitMQ) puede tener problemas de latencia o caídas, afectando la sincronización entre microservicios.	Alto	Implementación de estrategias de retry, uso de mecanismos de fallos, y monitoreo en tiempo real para detectar fallas rápidamente.
Falta de pruebas automatizadas	La ausencia de pruebas automatizadas podría llevar a fallos inesperados en la producción.	Alto	Implementar pruebas unitarias, de integración y de carga desde el inicio, priorizando la cobertura de los servicios críticos.
Seguridad en la autenticación	Las vulnerabilidades en la autenticación (JWT, OAuth) podrían ser explotadas por atacantes.	Alto	Implementar seguridad en capas (encriptación, autenticación multifactor), revisar prácticas de seguridad y auditorías regulares.
Complejidad en la gestión de microservicios	El uso de microservicios aumenta la complejidad de la gestión de infraestructuras y comunicación entre servicios.	Medio	Adoptar un enfoque de documentación clara, uso de contenedores (Docker) y orquestación (Kubernetes) para facilitar la gestión.
Falta de monitoreo y alertas adecuadas	La falta de monitoreo adecuado puede llevar a no detectar fallos en producción o problemas de rendimiento.	Medio	Implementar herramientas de monitoreo como Prometheus/Grafana, configurar alertas y asegurarse de tener un plan de respuesta.

Dependencia del servicio de caché (Redis)	Si el servicio de caché (Redis) se cae, podría generar problemas de rendimiento o disponibilidad del sistema.	Medio	Implementar estrategias de "fallback" para acceder a la base de datos en caso de fallos de caché.
Complejidad en la integración de diferentes bases de datos	Integrar bases de datos relacionales y no relacionales puede ser complejo y propenso a inconsistencias si no se maneja adecuadamente.	Alto	Usar una estrategia de integración clara (ETL, sincronización periódica) y definir claramente qué tipo de datos se almacenarán en cada base.

3. **Descripción de las Capas en Detalle**

Capa de Presentación

Navegador Web

- **Rol en la interacción con el usuario**: La capa de presentación se encarga de interactuar directamente con el usuario. El navegador web es el cliente que permite al usuario acceder a la aplicación a través de una interfaz gráfica. El navegador maneja las solicitudes y respuestas entre el cliente y el servidor.

- **Uso de HTTP**: El navegador envía peticiones HTTP (Hypertext Transfer Protocol) al servidor web cada vez que el usuario solicita información o realiza una acción en la aplicación. Estas solicitudes incluyen métodos como GET (para obtener datos) y POST (para enviar datos). El servidor, a su vez, responde con datos en formato HTML, CSS, y JavaScript, que el navegador utiliza para renderizar la interfaz del usuario.

Applet Java

- Función para la interacción con la lógica de negocio a través de RMI: Los applets Java son pequeños programas escritos en Java que se ejecutan dentro del navegador y que

interactúan con la lógica de negocio en el servidor. A través de RMI (Remote Method Invocation), un applet puede invocar métodos de objetos ubicados en el servidor, lo que permite a los usuarios interactuar con la lógica del servidor sin necesidad de cargar toda la lógica en su dispositivo. Esto es útil para aplicaciones que requieren comunicación con el servidor sin recargar la página completa.

Capa de Aplicación

Servidor Web

- Gestión de peticiones HTTP: El servidor web recibe las solicitudes HTTP enviadas por el navegador y las procesa. Su principal responsabilidad es gestionar la comunicación entre el cliente (navegador) y la aplicación del servidor. El servidor web recibe las peticiones de los usuarios, las envía a los servicios adecuados y retorna las respuestas generadas. Esto incluye el procesamiento de HTML, la ejecución de código del lado del servidor, y la validación de datos. Ejemplos comunes de servidores web incluyen Apache y Nginx.

Servidor Java

- Núcleo de la lógica de negocio: El servidor Java es el corazón de la lógica de negocio de la aplicación. Este servidor se encarga de recibir peticiones del servidor web, procesarlas, y realizar las operaciones necesarias, como consultas a bases de datos, validaciones, cálculos, etc. En una arquitectura de microservicios, el servidor Java podría estar implementado utilizando Spring Boot o Java EE.

- Manejo de peticiones RMI y JDBC: A través de RMI, el servidor Java puede ofrecer métodos remotos que pueden ser invocados desde otras aplicaciones distribuidas, como un applet Java o un cliente Java en otro dispositivo. Además, el servidor utiliza JDBC (Java Database Connectivity) para interactuar con las bases de datos, permitiendo el acceso y la manipulación de los datos almacenados de manera eficiente.

Métodos Nativos (JNI) si aplica al proyecto

- Interacción directa con el Dispositivo Físico: JNI (Java Native Interface) permite que el código Java interactúe directamente con componentes de hardware o dispositivos a

través de métodos nativos escritos en otros lenguajes como C o C++. Este enfoque es necesario cuando el sistema debe interactuar con hardware específico o cuando se requiere un rendimiento muy alto que no puede lograrse completamente con Java. Por ejemplo, si el sistema necesita acceso directo a sensores o dispositivos específicos, JNI sería útil para llamar funciones nativas que interactúan con esos dispositivos.

Capa de Datos

Bases de Datos

- Rol de las bases de datos para el almacenamiento de información: Las bases de datos juegan un papel crucial en el almacenamiento persistente de la información que maneja la aplicación. Pueden almacenar datos estructurados (en bases de datos relacionales como PostgreSQL o MySQL) o no estructurados (en bases de datos no relacionales como MongoDB). Estas bases de datos mantienen la integridad de los datos y permiten su recuperación rápida y eficiente. El acceso a las bases de datos desde la aplicación es manejado a través de consultas SQL o mecanismos equivalentes.

Protocolo DBMS

- Comunicación a través de un protocolo estándar de bases de datos: La comunicación entre el servidor Java y las bases de datos se realiza utilizando un protocolo DBMS (Database Management System) estándar, como JDBC para bases de datos relacionales. El protocolo garantiza que las consultas y comandos enviados desde la aplicación se entiendan correctamente por el sistema de gestión de bases de datos (DBMS), permitiendo una ejecución de operaciones seguras y eficientes. Además, este protocolo maneja la conexión, la ejecución de consultas, y el manejo de transacciones, asegurando la consistencia de los datos.