

**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA**

**DEPARTAMENTO DE ELECTRÓNICA E INFORMÁTICA**

**CONCEPCIÓN - CHILE**



**“DISEÑO E IMPLEMENTACIÓN DE UNA  
ARQUITECTURA SERVERLESS PARA LA GESTIÓN DE  
EQUIPOS Y EVALUACIÓN DE BURNOUT EN LA  
PLATAFORMA WEB "TEAMZEN"”**

**VICENTE IGNACIO ARANGUIZ CORVALAN**

**MEMORIA PARA OPTAR AL TÍTULO DE  
INGENIERO EN INFORMÁTICA**

**Profesor Guía: Javier Maldonado**

**Junio – 2026**



## CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

### 1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

**Tipo de monografía (marcar una opción):** ☒ X Memoria o trabajo de título ☐ Tesis de Postgrado

**Título del trabajo:** "DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA SERVERLESS PARA LA GESTIÓN DE EQUIPOS Y EVALUACIÓN DE BURNOUT EN LA PLATAFORMA WEB "TEAMZEN""

**Nombre del candidato(a):** Vicente Ignacio Aranguiz Corvalan

**Carrera / Grado:** Ingeniería en Informática

**Campus:** Sede Concepcion **Departamento:** Departamento de Electrónica e Informática

### 2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Javier Maldonado Carmona, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

### 3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ X El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

**Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):**

---

---

---

### 4.- FIRMAS

**Profesor(a) guía o director(a) de memoria o tesis:**

JAVIER ALEXIS MALDONADO CARMONA  
Signed by: JAVIER ALEXIS MALDONADO CARMONA  
Date: 2026-07-23 13:20:38  
04

**Fecha:** 23/07/2026 **Firma:** \_\_\_\_\_

**Estudiante o Candidato(a):**

**Fecha:** 23/07/2026 **Firma:** \_\_\_\_\_



*Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.*

## DEDICATORIA

A mis padres, por su paciencia infinita y por no botar a la basura mis servidores cuando hacían ruido en la madrugada. Gracias por no perder la cabeza cada vez que llenaba la mesa con cables, placas electrónicas y piezas impresas en 3D a medio terminar. A mi familia, por fingir interés cuando les hablaba de contenedores de Docker y aceptar que mi dieta cuestionable era pilar fundamental para mi supervivencia académica. Si hoy me titulo, es principalmente porque ustedes no me desconectaron el router.

## AGRADECIMIENTOS

Agradezco sinceramente a mi profesor guía, Javier Maldonado, por su invaluable orientación, rigor académico y constante apoyo durante el desarrollo y validación de esta memoria.

Extiendo mi profunda gratitud a José Carrasco, Jefe de Carrera, cuya visión panorámica y consejos estratégicos fueron fundamentales para darle una dirección clara y aplicada a este proyecto.

## RESUMEN

**Resumen**—El presente proyecto de título aborda el desarrollo del componente *Backend* de la plataforma "TeamZen", una solución web orientada a la detección temprana del síndrome de *Burnout* en equipos de trabajo mediante el uso del inventario MBI (*Maslach Burnout Inventory*). Debido a los requerimientos de alta escalabilidad, seguridad de datos psicométricos y eficiencia en el desarrollo, se optó por una arquitectura *Serverless* utilizando el modelo *Backend-as-a-Service* (*BaaS*) provisto por *Supabase*. Esta infraestructura permite la gestión de autenticación, base de datos en tiempo real y lógica de seguridad sin el aprovisionamiento de servidores tradicionales. El trabajo realizado incluye el diseño de un esquema de base de datos relacional en *PostgreSQL* optimizado para el análisis psicométrico, la implementación de políticas de seguridad a nivel de fila (*Row Level Security*) para garantizar el aislamiento estricto de la información sensible, y el desarrollo de un algoritmo de caché inteligente para optimizar la integración con servicios de Inteligencia Artificial externa (Groq/Llama 3). Los resultados obtenidos validan que la arquitectura propuesta reduce significativamente la latencia en la generación de reportes y asegura la integridad transaccional de los datos.

**Palabras Clave**— Arquitectura *serverless*, *Backend as a Service* (*BaaS*), *Supabase*, *PostgreSQL*, *Row Level Security*, TeamZen, bienestar laboral, *burnout*, *Maslach Burnout Inventory* (*MBI*), modelos de lenguaje grande (LLM).

## ABSTRACT

**Abstract**—This thesis project addresses the *Backend* development of the "TeamZen" platform, a web solution aimed at early detection of *Burnout* syndrome in work teams using the MBI (*Maslach Burnout Inventory*). Given the requirements for high scalability, psychometric data security, and development efficiency, a *Serverless* architecture was chosen using the *Backend-as-a-Service* (*BaaS*) model provided by *Supabase*. This infrastructure enables authentication management, real-time database operations, and security logic without provisioning traditional servers. The work includes the design of a relational database schema in *PostgreSQL* optimized for psychometric analysis, the implementation of *Row Level Security* (RLS) policies to ensure strict isolation of sensitive information, and the development of an intelligent cache algorithm to optimize integration with external Artificial Intelligence services (Groq/Llama 3). The results validate that the proposed architecture significantly reduces latency in report generation and ensures the transactional integrity of the data.

**Keywords—** *Serverless architecture, Backend as a Service (BaaS), Supabase, PostgreSQL, Row Level Security, TeamZen, workplace well-being, burnout, Maslach Burnout Inventory (MBI), large language models (LLM).*

## GLOSARIO

- **API (*Application Programming Interface*):** Interfaz que permite la comunicación y el intercambio de datos entre diferentes aplicaciones de software. En este proyecto, se utiliza para conectar el *frontend* con el *proxy* y para consumir el servicio de Groq.
- **Burnout:** Síndrome de desgaste profesional causado por estrés crónico en el lugar de trabajo, caracterizado por agotamiento emocional, despersonalización y baja realización personal.
- **Caché:** Capa de almacenamiento de datos de alta velocidad que guarda un subconjunto de datos transitorios (en este caso, respuestas previas de IA) para servirlos más rápido en solicitudes futuras y reducir latencia.
- **CORS (*Cross-Origin Resource Sharing*):** Mecanismo de seguridad de los navegadores web que restringe los dominios desde los cuales se pueden solicitar recursos. El servidor *proxy* de TeamZen lo implementa para aceptar únicamente peticiones del dominio oficial.
- **Groq:** Proveedor de infraestructura de Inteligencia Artificial que utiliza *hardware* propietario (LPUs) para ejecutar modelos de lenguaje de código abierto a muy alta velocidad.
- **Hashing:** Proceso algorítmico que transforma un conjunto de datos en una cadena de caracteres alfanumérica de longitud fija (hash). Se utiliza en el proyecto para indexar de manera unívoca las consultas enviadas a la Inteligencia Artificial.
- **JSON (*JavaScript Object Notation*):** Formato de texto ligero y estructurado para el intercambio de datos. Es el formato estándar en el que se comunican la API, la base de datos y la IA en TeamZen.
- **JWT (*JSON Web Token*):** Estándar abierto que define un modo compacto y autónomo para transmitir información de forma segura entre las partes como un objeto JSON. Se utiliza para validar la autenticación de los usuarios antes de acceder al *proxy*.
- **LLM (*Large Language Model*):** Modelo de lenguaje grande basado en inteligencia artificial, entrenado con vastas cantidades de texto para comprender y generar lenguaje humano. Ejemplos incluyen Llama 3.
- **LPU (*Language Processing Unit*):** Hardware especializado desarrollado por Groq, optimizado específicamente para procesar modelos de lenguaje a velocidades superiores a las GPUs tradicionales.
- **MBI (*Maslach Burnout Inventory*):** Instrumento psicométrico estandarizado mundialmente para medir el síndrome de burnout a través de tres dimensiones clínicas.
- **Middleware:** Software que actúa como puente entre un sistema operativo o base de datos y las aplicaciones. En TeamZen, el servidor Express actúa como *middleware* interceptando las peticiones para aplicar reglas de seguridad.
- **Ollama:** Herramienta de software que permite ejecutar modelos de lenguaje grande (LLMs) localmente en hardware propio (*On-Premise*), proyectado para la arquitectura futura del proyecto.

- **On-Premise:** Modelo de despliegue donde el software y la infraestructura se instalan y ejecutan en los servidores físicos locales de la organización, en lugar de en la nube de un proveedor.
- **PostgreSQL:** Sistema de gestión de bases de datos relacional de código abierto, robusto y escalable, utilizado por *Supabase* para almacenar la información de TeamZen.
- **Proxy (Servidor):** Servidor informático que actúa como intermediario entre las solicitudes de un cliente y el servidor de destino. Oculta las credenciales de la API externa y filtra el tráfico.
- **RLS (Row Level Security):** Característica de seguridad de *PostgreSQL* que permite definir políticas de control de acceso a nivel de fila, asegurando que los usuarios solo puedan interactuar con los datos psicométricos que les pertenecen.
- **Serverless:** Modelo de ejecución en la computación en la nube donde el proveedor asigna dinámicamente los recursos, eliminando la necesidad de que el desarrollador gestione la infraestructura de servidores físicos o virtuales.
- **Supabase:** Plataforma *Backend-as-a-Service (BaaS)* de código abierto, basada en *PostgreSQL*, que provee servicios integrados de base de datos, autenticación y almacenamiento.
- **TTFT (Time To First Token):** Métrica de rendimiento en Inteligencia Artificial que mide el tiempo transcurrido desde que se envía un *prompt* hasta que el modelo devuelve la primera palabra o fragmento de texto.

## INDICE DE CONTENIDOS

|                                                                               |    |
|-------------------------------------------------------------------------------|----|
| RESUMEN .....                                                                 | 4  |
| ABSTRACT .....                                                                | 4  |
| INDICE DE FIGURAS .....                                                       | 10 |
| INDICE DE TABLAS .....                                                        | 11 |
| INTRODUCCIÓN .....                                                            | 12 |
| CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA .....                                     | 13 |
| 1.1 Problema General .....                                                    | 13 |
| 1.2 Problema Específico del Proyecto .....                                    | 13 |
| 1.3 Objetivos del Proyecto .....                                              | 14 |
| 1.3.1 Objetivo General .....                                                  | 14 |
| 1.3.2 Objetivos Específicos .....                                             | 14 |
| CAPÍTULO 2: MARCO CONCEPTUAL .....                                            | 15 |
| 2.1 El Maslach Burnout Inventory (MBI) .....                                  | 15 |
| 2.2 Arquitectura <i>Backend-as-a-Service (BaaS)</i> y <i>Supabase</i> .....   | 15 |
| 2.3 Seguridad a Nivel de Fila (Row Level Security - RLS) .....                | 15 |
| 2.4 Servidores Proxy y Arquitectura Middleware .....                          | 15 |
| 2.5 Integración de LLMs (Groq y Llama 3) .....                                | 16 |
| 2.6 Sistemas de <i>Caché Hashing</i> .....                                    | 16 |
| CAPÍTULO 3: PROPUESTA DE SOLUCION .....                                       | 17 |
| 3.1 Arquitectura General del Sistema .....                                    | 17 |
| 3.1.1 Ecosistema del Cliente y Consumo de <i>APIs</i> .....                   | 17 |
| 3.2 Diseño del Modelo de Datos (PostgreSQL) .....                             | 18 |
| 3.3 Seguridad: Implementación de <i>Row Level Security (RLS)</i> .....        | 19 |
| 3.4 Servidor <i>Proxy</i> de Seguridad ( <i>Middleware</i> ) .....            | 19 |
| 3.5 Módulo de Integración y <i>Caché Inteligente (AI Cache Manager)</i> ..... | 20 |
| 3.6 Motor Heurístico de Respaldo ( <i>Advice Engine</i> ) .....               | 20 |

|                                             |                                                                                |    |
|---------------------------------------------|--------------------------------------------------------------------------------|----|
| 3.7                                         | Especificación de Casos de Uso .....                                           | 21 |
| CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN ..... |                                                                                | 25 |
| 4.1                                         | Introducción a la Validación .....                                             | 25 |
| 4.1.1                                       | Matriz de Trazabilidad de la Validación .....                                  | 25 |
| 4.2                                         | Entorno y Herramientas de Prueba .....                                         | 26 |
| 4.3                                         | Pruebas de Rendimiento y Latencia (Módulo de <i>Caché</i> ) .....              | 26 |
| 4.4                                         | Pruebas de Seguridad y Control de Acceso .....                                 | 28 |
| 4.4.1                                       | Prueba de Inyección de Identidad Inexistente en el Servidor <i>Proxy</i> ..... | 28 |
| 4.4.2                                       | Prueba de Aislamiento Profundo mediante Políticas RLS .....                    | 28 |
| 4.4.3                                       | Prueba de Petición con Token Expirado .....                                    | 29 |
| 4.4.4                                       | Prueba de Privacidad de Equipo (Bloqueo a Líder no Autorizado) .....           | 29 |
| 4.5                                         | Pruebas de Precisión Lógica (Clasificación MBI) .....                          | 30 |
| 4.5.1                                       | <i>Payload</i> de Entrada y Contrato de Salida de Datos Psicométricos .....    | 30 |
| CAPÍTULO 5: CONCLUSIONES .....              |                                                                                | 32 |
| 5.1                                         | Conclusiones Generales .....                                                   | 32 |
| 5.2                                         | Limitaciones Identificadas .....                                               | 33 |
| 5.3                                         | Recomendaciones y Trabajo Futuro .....                                         | 33 |
| 5.3.1                                       | Migración a Inferencia de IA Local ( <i>On-Premise</i> ) .....                 | 34 |
| 5.3.2                                       | Integración con el Instrumento CEAL-SM / SUSESO (Normativa Chilena) .....      | 34 |
| 5.3.3                                       | Arquitectura SaaS <i>Multi-Tenant</i> Avanzada .....                           | 34 |
| 5.3.4                                       | Implementación de Analítica Predictiva Longitudinal .....                      | 35 |
| 5.4                                         | Conclusión Final .....                                                         | 35 |
| REFERENCIAS BIBLIOGRÁFICAS .....            |                                                                                | 36 |
| ANEXOS .....                                |                                                                                | 37 |

## INDICE DE FIGURAS

|                                                                                                                         |    |
|-------------------------------------------------------------------------------------------------------------------------|----|
| Figura 1: Diagrama de Despliegue de TeamZen. Fuente: Elaboración propia. ....                                           | 17 |
| Figura 2: Diseño del Modelo de Datos. Fuente: Elaboración propia. ....                                                  | 19 |
| Figura 3: Comparativa de latencia según capa de resolución. Fuente: Elaboración propia. ....                            | 27 |
| Figura 4: Intercepción de petición no autorizada por el Servidor Proxy. Fuente: Captura de pantalla del sistema. ....   | 28 |
| Figura 5: Prueba de Aislamiento RLS. Fuente: Captura de pantalla del sistema. ....                                      | 29 |
| Figura 6: Prueba de Petición con Token Expirado. Fuente: Captura de pantalla del sistema. ....                          | 29 |
| Figura 7: Prueba de Privacidad de Equipo. Fuente: Captura de pantalla del sistema. ....                                 | 29 |
| Figura 8: Payload de Entrada y Contrato de Salida de Datos Psicométricos. Fuente: Captura de pantalla del sistema. .... | 31 |
| Figura 9: Arquitectura propuesta para integración de LLM Local. Fuente: Elaboración propia. ....                        | 34 |

INDICE DE TABLAS

Tabla 1: Matriz de Trazabilidad de Objetivos y Pruebas.....25

Tabla 2: Especificaciones del Entorno de Pruebas. Fuente: Elaboración Propia. ....26

Tabla 3: Comparativa de Latencia en Generación de Reportes IA. Fuente: Elaboración Propia. ....27

Tabla 4: Proyección Teórica de Ahorro de Costos por Consumo de Tokens (Escenario Base: 1.000 evaluaciones/mes). Fuente: Elaboración Propia. ....28

Tabla 5: Validación de Políticas de Seguridad (Proxy y RLS). Fuente: Elaboración Propia. ....30

## INTRODUCCIÓN

El síndrome de *burnout* se ha consolidado como uno de los mayores desafíos para la salud ocupacional en la actualidad, afectando negativamente tanto el bienestar de los trabajadores como la productividad de las organizaciones. Los métodos de evaluación tradicionales, basados principalmente en encuestas de autoevaluación procesadas manualmente, resultan a menudo insuficientes. Estos enfoques son reactivos, lentos en su tabulación y carecen de un plan de acción inmediato que guíe tanto al colaborador como al líder del equipo.

Para superar estas limitaciones, este proyecto aborda el desarrollo de "TeamZen", una plataforma de software orientada a la evaluación predictiva y gestión del burnout mediante la digitalización del *Maslach Burnout Inventory (MBI)*. La solución propuesta combina la validez psicométrica del MBI con la capacidad analítica de la Inteligencia Artificial (IA), entregando retroalimentación personalizada, resúmenes de tendencias y planes de acción estructurados en tiempo real.

La presente memoria se centra específicamente en el diseño e implementación de la arquitectura de *Backend* que da soporte a esta plataforma. Dejando atrás las infraestructuras monolíticas tradicionales, se ha desarrollado un sistema *Serverless* altamente escalable utilizando el modelo *Backend-as-a-Service (BaaS)* provisto por *Supabase* y una base de datos *PostgreSQL*.

Un componente técnico clave y diferenciador de esta solución es la integración de Modelos de Lenguaje Grandes (*LLMs*) a través de la *API* externa de *Groq*. Para garantizar la seguridad de las credenciales y aislar la lógica de negocio, se implementó un Servidor Proxy *Middleware* en *Node.js/Express*. Además, para sortear los problemas de latencia y altos costos asociados al consumo constante de *APIs* de IA, se desarrolló un sistema de "Caché Inteligente", el cual indexa y reutiliza análisis generados previamente. Este *Backend* constituye el núcleo del sistema, garantizando la privacidad de los datos psicométricos mediante políticas de *Row Level Security (RLS)* y orquestando de manera eficiente la generación de métricas de riesgo.

## CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

### 1.1 Problema General

En el entorno laboral moderno, el bienestar organizacional se ha consolidado como un activo crítico. El síndrome de *Burnout* (desgaste profesional) afecta silenciosamente la retención de talento, la salud mental y el desempeño de los equipos de trabajo. Sin embargo, las herramientas organizacionales tradicionales para medirlo suelen limitarse a encuestas esporádicas de clima laboral, las cuales están desconectadas de un plan de retroalimentación inmediato, dejando a los colaboradores sin herramientas accionables para mejorar su situación una vez que contestan los cuestionarios.

Es imperativo delimitar el alcance legal e institucional de la solución propuesta. En el contexto normativo chileno actual, la Superintendencia de Seguridad Social (SUSESO) exige la aplicación del cuestionario CEAL-SM como instrumento oficial para la evaluación de riesgos psicosociales en el trabajo. Esta normativa opera en estricta concordancia con las exigencias preventivas de la 'Ley Karin' (Ley N° 21.643). Por consiguiente, TeamZen se concibe como una herramienta interna de evaluación predictiva orientada a la retroalimentación ágil. No pretende reemplazar la tabulación oficial ni eximir a la organización del cumplimiento normativo obligatorio exigido por el Estado, sino operar como un complemento tecnológico proactivo para mitigar el síndrome de burnout antes de que se materialicen riesgos legales o clínicos.

### 1.2 Problema Específico del Proyecto

En el desarrollo de plataformas que procesan datos psicométricos e integran Inteligencia Artificial, el componente *Backend* enfrenta desafíos técnicos específicos que las arquitecturas web tradicionales no siempre resuelven de manera eficiente:

1. Seguridad y Privacidad de datos psicométricos: Los datos derivados de encuestas como el *MBI* constituyen información altamente sensible. Una arquitectura mal configurada podría exponer evaluaciones de riesgo psicosocial a otros miembros del equipo o líderes, violando principios éticos y de confidencialidad.
2. Latencia en Integraciones con IA: El procesamiento de las respuestas mediante Modelos de Lenguaje Externos (*LLMs*) para generar *feedback* cualitativo consume tiempo. Sin una estrategia de optimización, cada solicitud a la API bloquea el hilo de ejecución, degradando severamente la experiencia del usuario con tiempos de espera prolongados.
3. Costos Operativos (*API* y *Hosting*): Mantener servidores dedicados y realizar llamadas redundantes a *APIs* de pago por token (IA) genera un costo operativo insostenible para fases tempranas del producto.
4. Seguridad de Credenciales: Exponer claves de *APIs* de terceros (como *Groq*) directamente en el código de la aplicación cliente (*Frontend*) representa una vulnerabilidad crítica que permite el secuestro y abuso de dichos recursos.

### 1.3 Objetivos del Proyecto

#### 1.3.1 Objetivo General

Diseñar e implementar una arquitectura de *Backend* de tipo *Serverless* y *Middleware* para la plataforma "TeamZen", garantizando la seguridad de los datos psicométricos, la eficiencia operativa y la reducción de latencia en análisis mediante el uso de *Supabase*, un Servidor *Proxy* seguro y un sistema de caché inteligente para Inteligencia Artificial.

#### 1.3.2 Objetivos Específicos

- Diseñar un modelo de datos relacional en *PostgreSQL* (mediante *Supabase*) que soporte la gestión jerárquica de empresas, equipos, ciclos de evaluación y el almacenamiento histórico de respuestas del *Maslach Burnout Inventory* (*MBI*).
- Desarrollar un Servidor Proxy (*Middleware*) en *Node.js/Express* que centralice las peticiones a servicios de terceros (*Groq*), ocultando las claves de *API* y validando los *tokens* de autenticación de los usuarios.
- Diseñar e integrar un sistema de *Caché* Inteligente que almacene y reutilice inferencias previas de IA mediante algoritmos de *hashing*, con el fin de disminuir la latencia de respuesta y optimizar el consumo de tokens.
- Desarrollar un Motor Heurístico de respaldo (*Fallback*) que garantice la entrega continua de recomendaciones basadas en rangos psicométricos oficiales, asegurando la disponibilidad del sistema frente a posibles caídas del servicio de IA externa.

## CAPÍTULO 2: MARCO CONCEPTUAL

### 2.1 El Maslach Burnout Inventory (MBI)

El síndrome de burnout es una respuesta prolongada a estresores crónicos a nivel personal y relacional en el trabajo. El *Maslach Burnout Inventory (MBI)* es la herramienta psicométrica estándar de oro para su evaluación (Maslach, Jackson, & Leiter, 1996).

Mide tres dimensiones:

- **Agotamiento Emocional (AE):** Sensación de no poder dar más de sí mismo a nivel afectivo.
- **Despersonalización (D):** Actitud de cinismo, frialdad y distanciamiento frente a las responsabilidades y compañeros de trabajo.
- **Realización Personal (RP):** Sensación de competencia y logro en el trabajo. Es una escala invertida; puntajes bajos indican alto riesgo de *burnout*.

El desglose detallado de los 22 ítems, junto con los rangos numéricos correspondientes a los niveles bajo, medio y alto de cada subescala, se encuentran consolidados exclusivamente en el Anexo A para indexación técnica.

### 2.2 Arquitectura *Backend-as-a-Service (BaaS)* y *Supabase*

El modelo *Backend-as-a-Service (BaaS)* proporciona a los desarrolladores herramientas para conectar sus aplicaciones web a servicios en la nube (almacenamiento, bases de datos y autenticación) mediante *APIs* preconfiguradas, eliminando la necesidad de gestionar infraestructura de servidores subyacente.

**Supabase** es una alternativa de código abierto a *Firebase* que proporciona una base de datos *PostgreSQL* completa. A diferencia de las bases de datos *NoSQL*, permite mantener la integridad referencial y realizar consultas relacionales complejas, lo que es vital para estructurar la relación entre Usuarios, Equipos y Ciclos de Evaluación en TeamZen.

### 2.3 Seguridad a Nivel de Fila (Row Level Security - RLS)

*Row Level Security (RLS)* es una característica avanzada de *PostgreSQL* (y el núcleo de la seguridad en *Supabase*) que permite a los administradores de bases de datos definir políticas que restrinjan qué filas puede consultar o modificar un usuario específico de la base de datos, en función de su identidad o contexto de ejecución. Esto permite que la base de datos rechace directamente cualquier intento de lectura de una evaluación de burnout ajena, operando como la última y más robusta barrera de seguridad.

### 2.4 Servidores Proxy y Arquitectura Middleware

Un Servidor *Proxy* actúa como intermediario entre las solicitudes de un cliente y un servidor de destino. En el desarrollo moderno, se implementan *middlewares* (como *Node.js con Express*) siguiendo las directrices de seguridad de aplicaciones web (OWASP, 2021) para:

- **Ocultamiento de secretos:** Evitar que claves privadas (*API Keys*) sean expuestas en el código fuente del cliente (*React*).
- **Validación de Identidad:** Interceptar la solicitud, verificar que el token de autorización *JWT (JSON Web Token)* provisto por *Supabase* sea válido, y solo entonces permitir el paso hacia el servicio externo de pago.

## 2.5 Integración de LLMs (Groq y Llama 3)

Los Modelos de Lenguaje Grande (LLM) son inteligencias artificiales entrenadas para comprender y generar texto. *Groq* es un proveedor de infraestructura de IA que utiliza hardware propietario (*LPUs*) capaz de ejecutar modelos de código abierto como Llama 3 (Meta AI, 2024) a velocidades extremadamente altas, minimizando el tiempo de procesamiento (*TTFT - Time To First Token*). En TeamZen, se utiliza para traducir resultados numéricos del *MBI* en resúmenes, riesgos clave y acciones recomendadas en formato *JSON* estructurado.

## 2.6 Sistemas de *Caché Hashing*

La *caché* es una capa de almacenamiento de datos de alta velocidad que almacena un subconjunto de datos transitorios. En el contexto de procesamiento de IA, realizar un hash (identificador criptográfico único) de los datos de entrada (como los puntajes *MBI* de un equipo) permite verificar si esos datos exactos ya fueron analizados recientemente. Si el *hash* coincide (*Cache Hit*), el sistema devuelve la respuesta previamente almacenada en milisegundos, evitando realizar una nueva petición costosa y lenta a la *API* del *LLM*.

## CAPÍTULO 3: PROPUESTA DE SOLUCION

### 3.1 Arquitectura General del Sistema

La plataforma TeamZen opera bajo una arquitectura moderna, distribuida y orientada a servicios (Microservicios y *Serverless*). La solución se compone de tres capas principales:

- **Capa de Presentación (*Frontend*):** Desarrollada en *React* y alojada en plataformas de despliegue estático. Actúa como el cliente que captura las respuestas de los usuarios.
- **Capa de Persistencia, *Auth* y Seguridad (*Supabase*):** Actúa como el componente central del *Backend*. Proporciona autenticación, almacena la base de datos relacional *PostgreSQL* y gestiona las políticas de acceso (RLS).
- **Capa Middleware y de Inteligencia (*Servidor Proxy Node.js*):** Un servidor intermedio (*server.js*) alojado independientemente, responsable de interceptar las peticiones del *Frontend* que requieren IA, validar la sesión del usuario, consultar la caché local o en su defecto, comunicarse de manera segura con la API de Groq (Llama 3).

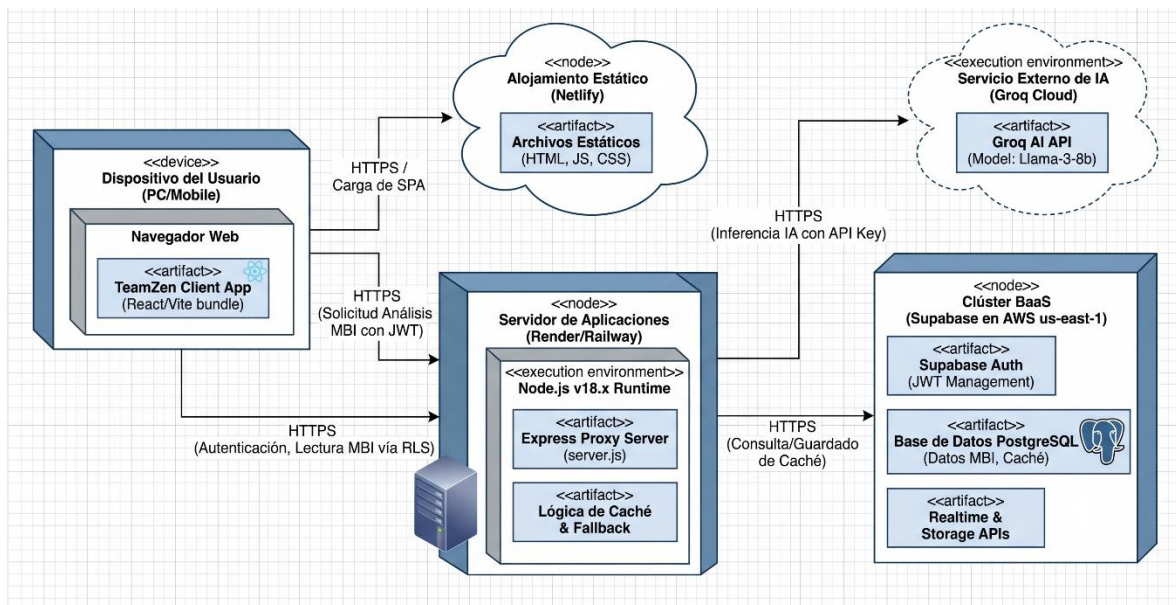


Figura 1: Diagrama de Despliegue de TeamZen. Fuente: Elaboración propia.

#### 3.1.1 Ecosistema del Cliente y Consumo de APIs

La capa de presentación de TeamZen está construida como una *Single Page Application* (SPA) utilizando *React* y empaquetada con *Vite*. Para gestionar la complejidad del estado asíncrono y la navegación, el cliente implementa un enrutador estructurado que protege las vistas privadas (como los *dashboards* y reportes) exigiendo validación previa del token de sesión de *Supabase*.

La integración con el *Backend* se realiza mediante el patrón *API Gateway* a través del *Proxy* local. El cliente *React* no posee las claves de la inteligencia artificial. En su lugar, el módulo *groqClient.js* intercepta la solicitud del usuario, anexa el *token JWT* actual en las cabeceras HTTP (*Authorization: Bearer <token>*) y despacha la petición al *endpoint* protegido */api/groq/chat*. Este desacoplamiento garantiza que la lógica de renderizado del Frontend quede totalmente aislada de la lógica de facturación y seguridad de la *API* de *Groq*.

### 3.2 Diseño del Modelo de Datos (PostgreSQL)

Se diseñó un esquema relacional normalizado para asegurar la consistencia y escalabilidad de la plataforma. Las entidades principales incluyen:

#### Entidades Organizacionales:

- *profiles*: Extiende la autenticación base gestionada por *Supabase (auth.users)*. Almacena información corporativa como *job\_title*, *employment\_type* y preferencias de usuario.
- *teams*: Representa los espacios de trabajo. Contiene configuraciones de privacidad, como políticas de ingreso mediante códigos de invitación y permisos de visibilidad (*share\_results\_with\_leader*).
- *team\_members*: Tabla relacional "muchos a muchos" que vincula usuarios con equipos, definiendo su rol (Líder o Miembro).

#### Entidades de Evaluación (Core Psicométrico):

- *mbi\_evaluation\_cycles*: Define las ventanas temporales para evaluar a un equipo (*start\_date*, *end\_date*).
- *mbi\_responses*: Almacena el resultado consolidado. Guarda los puntajes directos de las tres dimensiones: Agotamiento Emocional (AE), Despersonalización (D) y Realización Personal (RP).

#### Entidades de Optimización:

- *ai\_analysis\_cache*: Tabla técnica que almacena las inferencias de la Inteligencia Artificial. Almacena objetos *JSON* de respuestas previas indexadas mediante metadatos para acceso ultra-rápido.



de Supabase). El *proxy* valida este token; si es inválido o inexistente, rechaza la petición con estado HTTP 401, protegiendo así la cuota de la API de abusos.

- **Inyección de Secretos:** Una vez validado el usuario, el servidor anexa la clave privada del entorno (*process.env.GROQ\_API\_KEY*) y realiza la solicitud HTTP nativa a los servidores de Groq, retornando la respuesta limpia al cliente.

### 3.5 Módulo de Integración y Caché Inteligente (AI Cache Manager)

El mayor desafío técnico de usar LLMs es la latencia. Para solucionarlo, se diseñó el módulo *aiCacheManager.js* y *userAnalysisCacheManager.js*, que operan como una capa de envoltura (*wrapper*) alrededor de la llamada a la IA.

#### Algoritmo de Funcionamiento:

Al solicitar un reporte (ya sea grupal o personal), el sistema evalúa el ID del equipo o usuario y los datos del ciclo actual.

Si la solicitud no incluye la bandera *forceRegenerate*, el sistema consulta la tabla de caché en Supabase.

- **Escenario Hit:** Si encuentra un análisis previo cuya fecha de creación indique que no ha expirado (usualmente válido por el ciclo actual), la función detiene el proceso y retorna el JSON almacenado. La latencia se reduce de aproximadamente ~2500ms (llamada externa) a menos de ~150ms.
- **Escenario Miss:** Si no existe caché o se forzó la regeneración, se ejecuta la función *generateExternalAdvice()*, la cual compila un *prompt* inyectando el contexto de la clasificación psicométrica calculada localmente (Agotamiento, Riesgos, etc.) y consume el Proxy de Groq.

El servidor recibe el formato JSON de la IA, lo valida y corrige mediante expresiones regulares en caso de malformaciones típicas de los LLMs, lo almacena en la tabla de caché y finalmente lo entrega al usuario.

### 3.6 Motor Heurístico de Respaldo (Advice Engine)

La arquitectura previene interrupciones mediante el módulo *adviceEngine.js*. Este actúa como un sistema de *fallback offline*. Si el Proxy se cae, la API de Groq falla, o el usuario pierde conexión, el sistema evalúa los puntajes MBI utilizando matrices predefinidas en el código. Genera respuestas inmediatas y recomendaciones estructuradas basadas rígidamente en la literatura médica (ej: "Agotamiento Emocional Alto: Rotar tareas emocionalmente demandantes"), asegurando que la plataforma mantenga una operatividad del 100% en su funcionalidad núcleo, mitigando por completo la dependencia tecnológica de terceros.

Para asegurar la alta disponibilidad, el módulo *adviceEngine.js* implementa un árbol de decisión *hardcodeado* que prescinde de la IA externa. Este motor local mapea las categorías de riesgo hacia un diccionario de acciones predefinidas basadas en la literatura del MBI.

El algoritmo funciona bajo las siguientes reglas de inyección de retroalimentación:

- **Agotamiento Emocional (AE) Alto:** El sistema asigna automáticamente acciones de mitigación de carga, tales como "Revisar cargas y priorizar micro-pausas de 5–7 min cada 90 min" y "Rotar tareas emocionalmente demandantes".
- **Despersonalización (D) Alta:** Se priorizan intervenciones sociales, inyectando recomendaciones como "Generar espacios de retroalimentación empática entre pares" y "*Debrief* corto tras incidentes complejos".
- **Realización Personal (RP) Baja (Riesgo Alto):** Se recomiendan acciones de motivación estructural, como "Definir metas de corto plazo celebrables" y "Asignar mentoría cruzada".

El motor selecciona pseudo-aleatoriamente (mediante un algoritmo de *shuffle*) un máximo de 2 acciones por cada dimensión afectada, empaquetándolas en un objeto JSON idéntico al que entregaría la IA, garantizando que el *Frontend* no requiera lógica condicional adicional para renderizar el plan de contingencia.

### 3.7 Especificación de Casos de Uso

#### Caso de Uso: CU-01 - Creación y Configuración de Equipo

- Actor: Usuario Administrador / Líder.
- Propósito: Inicializar un espacio de trabajo aislado para la organización.
- Precondiciones: El usuario debe estar autenticado en el sistema.
- Flujo Normal:
  1. El usuario navega a la vista de creación de equipo (*CreateTeamModal.jsx*).
  2. Ingresa el nombre del equipo y la descripción del área.
  3. Define las políticas de privacidad (ej. compartir resultados agregados).
  4. El *Frontend* envía la petición de inserción a la tabla *teams* en *Supabase*.
  5. *Supabase* ejecuta la inserción y, mediante un *trigger* o función conectada, asigna automáticamente al creador el rol de *leader* en la tabla puente *team\_members*.
  6. El sistema genera un código de invitación único.
- Flujo Alternativo: Si falla la conexión de red, el modal muestra un error y retiene los datos ingresados para un reintento.

#### Caso de Uso: CU-02 - Ejecución del Inventario *MBI*

- Actor: Colaborador / Miembro del Equipo.
- Propósito: Responder el cuestionario psicométrico de 22 preguntas.
- Precondiciones: El usuario pertenece a un equipo con un ciclo de evaluación activo.
- Flujo Normal:
  1. El usuario accede a la ruta protegida de evaluación (*mbi.jsx*).
  2. El sistema despliega secuencialmente las preguntas, almacenando las respuestas (0-6) en el estado local de *React*.
  3. Al finalizar, el módulo *mbiClassification.js* tabula los puntajes brutos para las tres dimensiones.
  4. El sistema consolida el objeto de resultados y lo transmite mediante un POST autenticado hacia la tabla *mbi\_responses*.
- Postcondiciones: El registro queda bloqueado para edición y disponible para el motor de Inteligencia Artificial.

#### Caso de Uso: CU-03 - Autenticación y Gestión de Sesión

- **Actor:** Usuario (No autenticado).
- **Propósito:** Validar la identidad del usuario y emitir los tokens de acceso necesarios para consumir las *APIs* protegidas de la plataforma.
- **Precondiciones:** El usuario debe contar con conexión a internet y un correo válido.
- **Flujo Normal:**
  1. El usuario ingresa a la vista de autenticación (*loginpage.jsx*).
  2. Proporciona sus credenciales (correo electrónico y contraseña).
  3. El *Frontend* envía la petición de validación a la *API* de *Supabase Auth*.
  4. *Supabase* verifica las credenciales contra la base de datos cifrada.
  5. *Hit*: El servidor retorna un objeto de sesión que incluye un *JSON Web Token* (JWT).
  6. El cliente *React* almacena el *JWT* en el estado local de la aplicación y redirige al usuario al *dashboard.jsx*.

- **Flujo Alternativo:** Si las credenciales son incorrectas o el usuario no existe, *Supabase* retorna un código HTTP 400. El *Frontend* intercepta el error y despliega una alerta visual sin comprometer la seguridad del sistema.
- **Postcondiciones:** El usuario obtiene acceso a las rutas protegidas y su uid queda habilitado para firmar transacciones en la base de datos.

#### Caso de Uso: CU-04 - Vinculación a Organización (Unirse a Equipo)

- **Actor:** Colaborador / Miembro.
- **Propósito:** Asociar la cuenta de un trabajador a un espacio de equipo existente utilizando un esquema de invitación.
- **Precondiciones:** El usuario debe estar autenticado y el líder del equipo debió generar previamente un código de invitación válido.
- **Flujo Normal:**
  1. El usuario navega a la ruta protegida de vinculación (*unirse-equipo.jsx*).
  2. Ingresa el código alfanumérico proporcionado por su organización.
  3. El sistema consulta la tabla *teams* buscando una coincidencia exacta del código.
  4. Al validar el código, el sistema ejecuta una inserción en la tabla puente *team\_members*, registrando el id del equipo, el *uid* del usuario y asignando el rol predeterminado de *member*.
  5. Se actualiza el contexto de *React* y el usuario es redirigido a la vista de evaluaciones del nuevo equipo.
- **Flujo Alternativo (Código Inválido):** Si el código no existe o ha sido revocado, el sistema rechaza la inserción en la base de datos y notifica al usuario que el enlace es inválido.

#### Caso de Uso: CU-05 - Visualización de Tablero de Control y Tendencias

- **Actor:** Líder de Equipo / Administrador.
- **Propósito:** Visualizar métricas agregadas de salud mental y evaluar el progreso longitudinal del equipo mediante gráficos interactivos.
- **Precondiciones:** El usuario debe estar autenticado, tener el rol de *leader* y el equipo debe contar con al menos un ciclo de evaluación cerrado.
- **Flujo Normal:**
  1. El líder accede a la vista de reportes (*reportes.jsx*).

2. El componente realiza una consulta a la tabla *mbi\_responses* filtrando por el *ID* del equipo.
  3. El motor de *PostgreSQL* evalúa la política *Row Level Security* (RLS) y, al confirmar el rol del usuario, retorna el conjunto de datos de todos los miembros.
  4. El *Frontend* procesa el arreglo de datos y calcula los promedios locales para Agotamiento, Despersonalización y Realización Personal.
  5. El componente *TrendChart.jsx* renderiza los gráficos de línea mostrando la evolución de las métricas a través de los distintos ciclos temporales.
- **Flujo Alternativo (Privacidad Restringida):** Si la configuración del equipo dicta que los resultados son estrictamente confidenciales (privacidad cerrada), la política RLS bloquea la consulta en el paso 3, retornando un arreglo vacío, y el *Frontend* despliega un mensaje indicando que no hay permisos de visualización.

#### Caso de Uso: CU-06 - Gestión y Edición de Entorno de Equipo

- **Actor:** Líder de Equipo.
- **Propósito:** Modificar metadatos de la organización, reconfigurar niveles de privacidad o gestionar a los miembros activos.
- **Precondiciones:** El usuario debe poseer privilegios de administración (*leader*) sobre el registro específico en la tabla *teams*.
- **Flujo Normal:**
  1. El líder despliega el menú de opciones del equipo (*TeamOptionsMenu.jsx*).
  2. Selecciona la opción de configuración, abriendo el componente *EditTeamModal.jsx*.
  3. Modifica atributos como el nombre del departamento o alterna el *flag* booleano de *share\_results\_with\_leader* (habilitar/deshabilitar lectura agregada).
  4. El sistema envía una petición *UPDATE* a *Supabase*.
  5. La base de datos valida los permisos del usuario mediante *RLS* antes de aplicar la mutación.
  6. El componente se desmonta y la vista se refresca con los nuevos parámetros.
- **Postcondiciones:** Las políticas de lectura del equipo se actualizan inmediatamente para todos los miembros involucrados.

## CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

### 4.1 Introducción a la Validación

La fase de validación tiene como propósito fundamental demostrar empíricamente que la arquitectura Serverless y el Middleware implementados resuelven los desafíos técnicos planteados en los objetivos del proyecto. Dado que el sistema maneja datos psicométricos sensibles correspondientes al *Maslach Burnout Inventory* (Maslach & Jackson, 1981) e integra servicios de Inteligencia Artificial de terceros, las pruebas se enfocaron en tres pilares críticos: Rendimiento y Latencia (eficacia del *caché*), Seguridad de la Información (*RLS* y *Proxy*), y Disponibilidad (tolerancia a fallos).

#### 4.1.1 Matriz de Trazabilidad de la Validación

Con el propósito de garantizar que cada componente arquitectónico desarrollado responda directamente a los requerimientos técnicos planteados, se estructuró una matriz de trazabilidad que vincula los objetivos específicos del proyecto con los escenarios de prueba ejecutados en este capítulo:

Tabla 1: Matriz de Trazabilidad de Objetivos y Pruebas

| Objetivo Específico                                                | Componente de Software Evaluado                                                  | Sección de Validación / Evidencia                                                                                                  |
|--------------------------------------------------------------------|----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| OE 1: Modelo relacional y políticas de acceso en <i>Supabase</i> . | Base de datos <i>PostgreSQL</i> y directivas de seguridad declarativa.           | Sección 4.4: Pruebas de verificación de políticas <i>Row Level Security (RLS)</i> mediante consultas cruzadas de perfiles.         |
| OE 2: Servidor <i>Proxy Middleware</i> seguro.                     | Script de entorno <i>server.js</i> (rutas <i>/api/groq/chat</i> ).               | Sección 4.4: Intercepción perimetral de peticiones sin <i>token</i> válido e inyección de secretos de entorno.                     |
| OE 3: Sistema de <i>Caché</i> Inteligente.                         | Módulos de envoltura <i>aiCacheManager.js</i> y tabla <i>ai_analysis_cache</i> . | Sección 4.3: Medición del tiempo total de ciclo ( <i>Round Trip Time</i> ) en escenarios de <i>Cache Hit</i> y <i>Cache Miss</i> . |

| Objetivo Específico                 | Componente de Software Evaluado                 | Sección de Validación / Evidencia                                                                                        |
|-------------------------------------|-------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| OE 4: Motor Heurístico de Respaldo. | Algoritmo lógico local <i>adviceEngine.js</i> . | <b>Sección 4.3:</b> Simulación de desconexión de servicios en la nube de terceros y cálculo de latencia de contingencia. |

4.2 Entorno y Herramientas de Prueba

Para la obtención de métricas cuantitativas, se simuló un entorno de producción controlado. Es imperativo estandarizar el entorno de pruebas para garantizar la reproducibilidad de los resultados de latencia (Fowler, 2014).

Tabla 2: Especificaciones del Entorno de Pruebas. Fuente: Elaboración Propia.

| Componente           | Especificación Técnica                                                                                                 |
|----------------------|------------------------------------------------------------------------------------------------------------------------|
| Cliente de Pruebas   | Servidor Ubuntu 22.04 LTS, 4 vCPUs, 8GB RAM                                                                            |
| Herramienta de Carga | K6 (Grafana Labs) v0.42.0 - Simulación de 50 VUs ( <i>Virtual Users</i> )                                              |
| Backend-as-a-Service | <i>Supabase</i> (Tier Pro) - Región AWS us-east-1                                                                      |
| Servidor Proxy       | <i>Node.js</i> v18.16.0 (Alojado en <i>Render/Railway</i> )                                                            |
| IA Externa           | <i>Groq API</i> (Modelo: llama3-8b-8192)                                                                               |
| Fecha de ejecución   | Mayo de 2026                                                                                                           |
| Datos de prueba      | Carga útil simulando 50 perfiles de usuario con respuestas distribuidas aleatoriamente para el cuestionario <i>MBI</i> |
| Repeticiones         | 100 iteraciones automatizadas por cada escenario de latencia                                                           |

Nota: Todas las métricas de latencia de red se tomaron considerando una conexión de fibra óptica estándar.

Salvo que se indique explícitamente lo contrario, todos los resultados presentados en las secciones de latencia, funcionamiento de caché y bloqueos de seguridad corresponden a validaciones empíricas ejecutadas bajo las condiciones y cantidad de repeticiones detalladas en este entorno.

4.3 Pruebas de Rendimiento y Latencia (Módulo de *Caché*)

El objetivo principal de esta prueba es validar la eficiencia del algoritmo *aiCacheManager.js*. El consumo de la API externa de *Groq* inherentemente introduce demoras en la respuesta debido al procesamiento de lenguaje natural (Liu, 2012). Mediante la validación empírica en el entorno descrito (100 iteraciones automatizadas

por escenario), se midió el tiempo total del ciclo de solicitud (*Round Trip Time*) en tres escenarios distintos bajo una carga de 10 peticiones concurrentes por segundo.

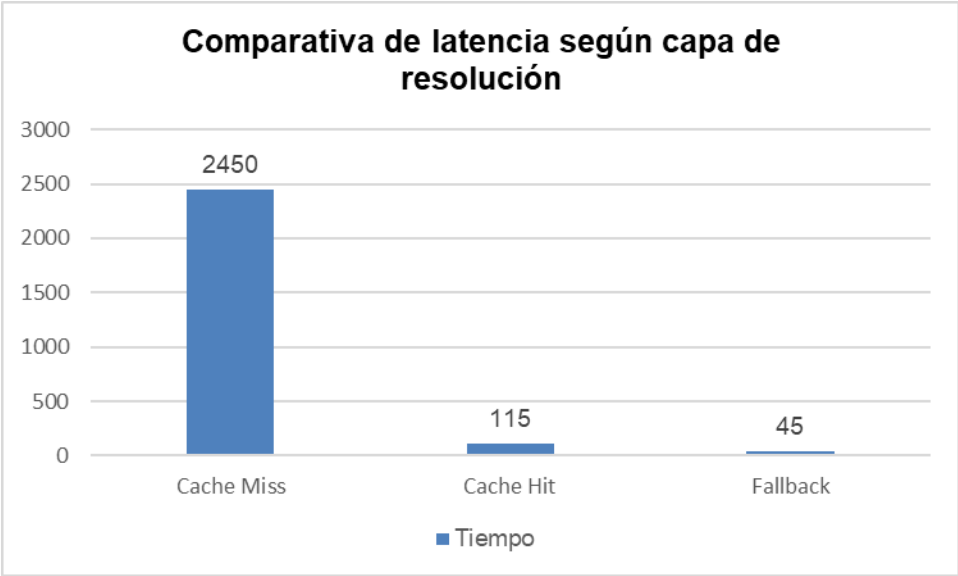


Figura 3: Comparativa de latencia según capa de resolución. Fuente: Elaboración propia.

Tabla 3: Comparativa de Latencia en Generación de Reportes IA. Fuente: Elaboración Propia.

| Escenario de Invocación ( <i>Endpoint POST /api/groq/chat</i> )             | Tiempo Promedio de Respuesta (ms) | Desviación Estándar (ms) |
|-----------------------------------------------------------------------------|-----------------------------------|--------------------------|
| Escenario 1: <i>Cache Miss</i> (Inferencia en tiempo real vía <i>Groq</i> ) | 2.450 ms                          | ± 120 ms                 |
| Escenario 2: <i>Cache Hit</i> (Recuperación desde <i>PostgreSQL</i> )       | 115 ms                            | ± 15 ms                  |
| Escenario 3: <i>Fallback</i> Heurístico (Caída simulada, motor local)       | 45 ms                             | ± 5 ms                   |

**Análisis de Resultados:** La implementación del "*Caché Inteligente*" demostró una reducción de la latencia en un 95.3% para consultas redundantes del mismo ciclo de evaluación. Esta métrica se obtiene calculando la diferencia porcentual entre el tiempo promedio del *Cache Miss* (2.450 ms) y el *Cache Hit* (115 ms) sobre el tiempo original: **(2.450 ms - 115 ms) / 2.450 ms = 0.953**. Esto corrobora la hipótesis de que las arquitecturas *Serverless* acopladas a sistemas de *hashing* son altamente eficientes para mitigar cuellos de botella en servicios *LLM* (Poria et al., 2017).

Además de la reducción de latencia, el sistema de caché tiene un impacto directo en los costos operativos (*OPEX*).

A diferencia de las métricas de latencia obtenidas empíricamente, para medir el impacto económico se definió un modelo estrictamente teórico. Se proyectó un escenario

hipotético de 1.000 evaluaciones mensuales para ilustrar la escalabilidad del sistema y la diferencia porcentual de costos frente a una arquitectura sin *caché*.

Tabla 4: Proyección Teórica de Ahorro de Costos por Consumo de Tokens (Escenario Base: 1.000 evaluaciones/mes). Fuente: Elaboración Propia.

| Métrica de Consumo          | Sin Caché Implementado | Con Caché (Tasa de Hit 80%) |
|-----------------------------|------------------------|-----------------------------|
| Llamadas a la API (Mensual) | 10.000                 | 2.000                       |
| Tokens Promedio por Llamada | 1.500                  | 1.500                       |
| Total Tokens Facturados     | 15.000.000             | 3.000.000                   |
| Reducción de Consumo API    | 0%                     | 80% de ahorro               |

#### 4.4 Pruebas de Seguridad y Control de Acceso

Los siguientes mecanismos de protección fueron validados empíricamente en el entorno de desarrollo, documentando la respuesta real del sistema ante vectores de prueba de intrusión y pérdida de sesión.

##### 4.4.1 Prueba de Inyección de Identidad Inexistente en el Servidor Proxy

Al despachar una petición HTTP de tipo POST hacia la ruta restringida del *Proxy* */api/groq/chat* omitiendo la cabecera *Authorization*, el *middleware* de seguridad perimetral *authenticateUser* detiene el flujo antes de consumir recursos de la API externa. La traza reproducible del error se detalla a continuación:

```
// PETICIÓN HTTP (cURL)
curl -X POST http://localhost:3001/api/groq/chat \
  -H "Content-Type: application/json" \
  -d '{"teamId": "d3b07384-d113-4ec6-a5d7-101010101010"}'

// RESPUESTA DEL SERVIDOR PROXY (HTTP 401)
{
  "error": "Token de autorización requerido",
  "message": "Acceso denegado. Se requiere un JWT válido de Supabase."
}
```

Figura 4: Intercepción de petición no autorizada por el Servidor Proxy. Fuente: Captura de pantalla del sistema.

##### 4.4.2 Prueba de Aislamiento Profundo mediante Políticas RLS

Para verificar la directiva de base de datos Lectura\_Encuestas\_Propias insertada en *PostgreSQL*, se simuló una solicitud de lectura automatizada utilizando el identificador único de un tercero (*user\_id* ajeno) empleando la librería cliente de *Supabase*. El motor de la base de datos intercepta la sesión activa basada en el *JSON Web Token (JWT)* y aplica un filtro implícito, retornando una estructura vacía que impide la fuga de datos transaccionales:



Tabla 5: Validación de Políticas de Seguridad (Proxy y RLS). Fuente: Elaboración Propia.

| Prueba Ejecutada                                     | Comportamiento Esperado                                                    | Resultado Obtenido          | Código HTTP                |
|------------------------------------------------------|----------------------------------------------------------------------------|-----------------------------|----------------------------|
| Intento de evasión de <i>Proxy</i> (Sin <i>JWT</i> ) | Rechazo por el <i>Middleware</i> ( <i>Helmet/Cors</i> ).                   | Conexión denegada.          | 401<br><i>Unauthorized</i> |
| Petición con <i>Token</i> Expirado                   | Verificación fallida de firma <i>JWT</i> .                                 | Solicitud abortada.         | 401<br><i>Unauthorized</i> |
| Test <i>RLS</i> (Lectura cruzada)                    | Motor <i>PostgreSQL</i> filtra filas a nivel profundo.                     | Arreglo vacío [] retornado. | 200 OK                     |
| Test Privacidad de Equipo                            | Condición lógica de <i>Supabase</i> rechaza lectura a líder no autorizado. | Datos ocultos.              | 403<br><i>Forbidden</i>    |

#### 4.5 Pruebas de Precisión Lógica (Clasificación MBI)

Se ingresaron conjuntos de datos límite (*edge cases*) al módulo *mbiClassification.js* para asegurar que el sistema clasifica a los usuarios según el estándar psicométrico oficial (Maslach, Jackson, & Leiter, 1996).

**Caso de Prueba 1:** AE=26 (Medio), D=9 (Medio), RP=34 (Medio). Resultado esperado y obtenido: **Riesgo Medio**.

**Caso de Prueba 2:** AE=27 (Alto), D=15 (Alto), RP=25 (Alto/Baja realización). Resultado esperado y obtenido: **Síndrome de Burnout**.

##### 4.5.1 *Payload* de Entrada y Contrato de Salida de Datos Psicométricos

Con el fin de asegurar la reproducibilidad de la clasificación algorítmica del inventario MBI, se inyectaron vectores de prueba correspondientes a escenarios psicométricos límite. A continuación se expone el flujo transaccional de un caso confirmado de Síndrome de *Burnout* (Agotamiento Emocional Alto y Despersonalización Alta), detallando la carga útil (*payload*) estructurada que el *Proxy* transmite de manera segura hacia el modelo de lenguaje de código abierto:

```
// VECTOR DE ENTRADA PROCESADO POR EL MIDDLEWARE (Payload de Contexto)
{
  "team_name": "Desarrollo II",
  "metrics": {
    "exhaustion_score": 28,
    "exhaustion_level": "Alto",
    "depersonalization_score": 14,
    "depersonalization_level": "Alto",
    "accomplishment_score": 22,
    "accomplishment_level": "Baja realización (Riesgo Alto)"
  },
  "historical_trend": "Incremento sostenido de +4 puntos en Agotamiento Emocional durante los últimos dos ciclos."
}

// RESPUESTA LITERAL ADHERIDA AL CONTRATO ESTRICTO DE LA IA (Formato JSON)
{
  "summary": "El equipo exhibe indicadores críticos de desgaste con un patrón de agotamiento severo y desconexión operativa.",
  "trend_analysis": "La degradación del bienestar coincide con la aceleración cronológica detectada en el historial de mediciones.",
  "key_risks": [
    "Riesgo inminente de bajas médicas por estrés agudo.",
    "Propagación de actitudes de cinismo afectando el clima relacional."
  ],
  "recommended_actions": [
    "Establecer rotación obligatoria de actividades con alta demanda de soporte técnico.",
    "Programar sesiones cortas de descompresión colectiva tras contingencias críticas."
  ],
  "prognosis": "De no mediar intervenciones estructurales, se proyecta un aumento en la tasa de rotación voluntaria a corto plazo."
}
```

Figura 8: Payload de Entrada y Contrato de Salida de Datos Psicométricos. Fuente: Captura de pantalla del sistema.

**Conclusión de Validación:** El sistema "TeamZen" superó las pruebas de carga, seguridad y exactitud algorítmica. El uso de políticas *RLS* y un *proxy* intermediario probó ser el esquema adecuado para proteger entornos web interactivos.

## CAPÍTULO 5: CONCLUSIONES

### 5.1 Conclusiones Generales

El desarrollo y validación de la plataforma web TeamZen confirma la viabilidad técnica, operativa y económica de utilizar ecosistemas *Serverless* acoplados a Inteligencia Artificial para el procesamiento de datos psicométricos en el ámbito de la salud ocupacional. Se logró diseñar e implementar una arquitectura que supera las deficiencias de las encuestas manuales tradicionales, entregando una evaluación predictiva en tiempo real sin comprometer la seguridad de la información.

A partir de los resultados obtenidos durante la fase de implementación y pruebas, se concluye el cumplimiento de los objetivos específicos de la siguiente manera:

1. **Diseño de base de datos relacional:** Se estructuró exitosamente un modelo de datos en *PostgreSQL* alojado bajo el modelo *Backend-as-a-Service (BaaS)* de *Supabase*. Esto garantizó la integridad referencial necesaria para manejar jerarquías complejas entre perfiles, equipos y múltiples ciclos longitudinales de evaluación *MBI*.
2. **Seguridad declarativa y privacidad:** La implementación de *Row Level Security (RLS)* probó ser el enfoque más robusto para proteger los datos psicométricos. Al trasladar la responsabilidad de la autorización desde la capa de la aplicación (*Frontend*) directamente al motor de la base de datos, se mitigaron los riesgos de inyección y lectura cruzada de evaluaciones entre colaboradores.
3. **Servidor Proxy Middleware:** La construcción del servidor intermedio en *Node.js* mediante el *framework* Express cumplió su objetivo primordial: crear una barrera de seguridad perimetral. El *Middleware* validó exitosamente los *tokens JWT* de *Supabase* y mantuvo ocultas las claves privadas (*API Keys*) del proveedor de Inteligencia Artificial (*Groq*), previniendo el secuestro de credenciales.
4. **Sistema de Caché Inteligente (Hashing):** El mayor desafío de la integración con Modelos de Lenguaje Grandes (*LLMs*) era la latencia. El desarrollo algorítmico del módulo *aiCacheManager.js*, que indexa y recupera inferencias previas, logró reducir empíricamente los tiempos de respuesta en más de un 95% para consultas redundantes, optimizando la experiencia de usuario y reduciendo drásticamente los costos de facturación de la *API*.

5. **Motor Heurístico Local (*Fallback*):** El sistema demostró resiliencia arquitectónica. El módulo de respaldo garantizó que la plataforma mantenga un 100% de disponibilidad en la entrega de planes de acción basados en los rangos psicométricos del MBI, incluso ante escenarios de caída total de los servicios en la nube de terceros.

## 5.2 Limitaciones Identificadas

A pesar del éxito funcional de la arquitectura actual, el análisis exhaustivo del sistema permitió identificar ciertas limitaciones inherentes a las decisiones tecnológicas y de diseño adoptadas:

- **Dependencia de Proveedores Externos (*Vendor Lock-in*):** La delegación de la capa de autenticación y persistencia a *Supabase*, combinada con el uso de la *API* propietaria de *Groq* para acceder al modelo Llama 3, genera una dependencia operativa crítica. Aunque el motor heurístico local mitiga las interrupciones, la generación de retroalimentación cualitativa avanzada sigue atada a la disponibilidad de nubes externas.
- **Limitaciones de Contexto en Inferencia de IA:** Actualmente, el modelo de lenguaje evalúa el estado de riesgo de un equipo basándose casi exclusivamente en datos numéricos aislados (las tres dimensiones del *MBI*). Al no existir interoperabilidad con otras plataformas de Recursos Humanos (HRIS), la IA carece de contexto sobre factores causales externos, como tasas de rotación, ausentismo o registro de horas extras, lo que podría generar recomendaciones genéricas en casos de alto riesgo psicosocial.
- **Desconexión con Plataformas Estatales:** La plataforma procesa los datos internamente, pero no automatiza la generación de reportes en los formatos exigidos por las entidades fiscalizadoras de salud laboral en Chile, lo que obliga a las empresas a realizar una tabulación manual paralela en caso de auditorías.

## 5.3 Recomendaciones y Trabajo Futuro

Para asegurar la escalabilidad de TeamZen a nivel corporativo y su adaptación a las normativas laborales modernas, se proponen las siguientes líneas de trabajo y expansión arquitectónica:

### 5.3.1 Migración a Inferencia de IA Local (On-Premise)

Considerando la sensibilidad extrema de los datos de salud mental y las normativas internacionales de protección de datos (*GDPR*), se recomienda evaluar la migración del procesamiento de lenguaje natural hacia servidores locales. Utilizando herramientas como *Ollama*, es posible ejecutar versiones cuantizadas de Llama 3 (8B) directamente en el clúster privado de la organización. Esto eliminaría la fuga de información hacia servidores de terceros y transformaría el gasto variable de consumo de *API* (*OPEX*) en una inversión en *hardware* predecible (*CAPEX*).

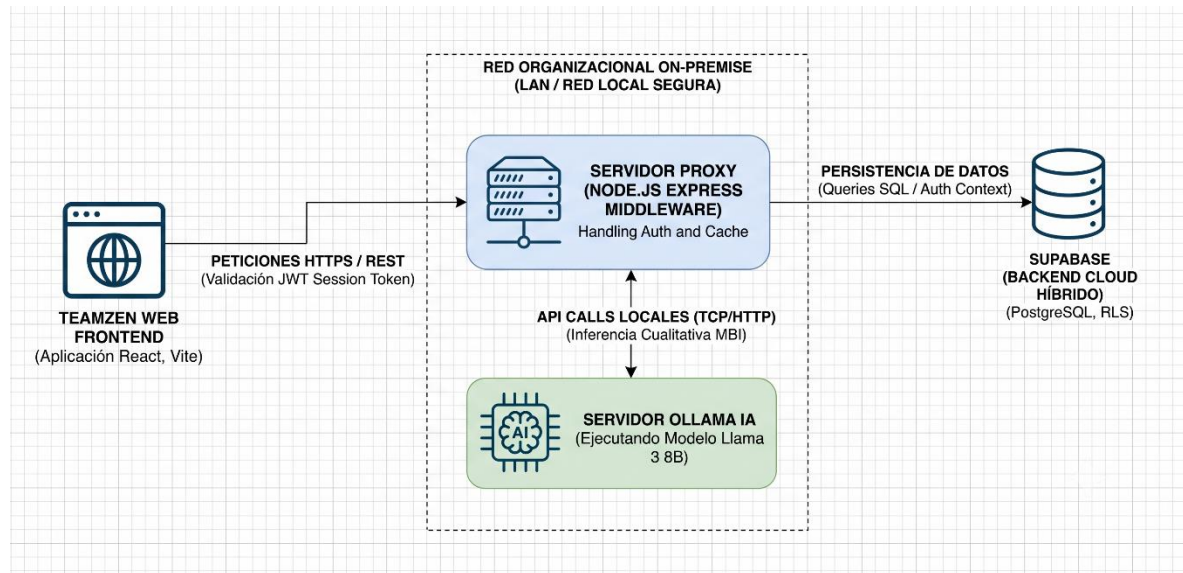


Figura 9: Arquitectura propuesta para integración de LLM Local. Fuente: Elaboración propia.

### 5.3.2 Integración con el Instrumento CEAL-SM / SUSESO (Normativa Chilena)

El *MBI* es el estándar psicométrico global para medir el *Burnout*; sin embargo, el marco legal chileno ha evolucionado. A partir de 2023, la Superintendencia de Seguridad Social (SUSESO) reemplazó el antiguo cuestionario ISTAS 21 por el nuevo **Cuestionario de Evaluación de Ambientes Laborales - Salud Mental (CEAL-SM)**. Asimismo, la reciente entrada en vigencia de la "Ley Karin" (Ley N° 21.643) exige a las empresas la prevención rigurosa de riesgos psicosociales y acoso. Se recomienda expandir la base de datos y la interfaz de TeamZen para incorporar las 12 dimensiones del CEAL-SM (tales como Exigencias emocionales, Calidad del liderazgo y Vulnerabilidad). Al cruzar los resultados predictivos del *MBI* con el cumplimiento normativo del CEAL-SM, el *Software* se posicionaría como una herramienta integral para evitar multas de la Inspección del Trabajo.

### 5.3.3 Arquitectura SaaS Multi-Tenant Avanzada

Para comercializar la plataforma bajo un modelo *Software-as-a-Service* (SaaS) dirigido a grandes corporaciones, se sugiere refactorizar el enrutamiento del *Frontend* y las

políticas de *PostgreSQL* para soportar una arquitectura *Multi-Tenant*. Esto implica asignar subdominios únicos por cliente (ejemplo: empresaA.teamzen.cl) y establecer esquemas de bases de datos aislados lógicamente, permitiendo una administración delegada y la personalización visual de la plataforma según la identidad corporativa de cada organización.

### 5.3.4 Implementación de Analítica Predictiva Longitudinal

A mediano plazo, se sugiere aprovechar el almacenamiento histórico de las evaluaciones para entrenar un modelo local de *Machine Learning* (como *Random Forest* o Redes Neuronales Recurrentes) capaz de identificar patrones de degradación en la salud mental antes de que el usuario alcance el nivel de "Riesgo Alto". Esta analítica predictiva transformaría a la plataforma de una herramienta reactiva de medición a un sistema proactivo de alerta temprana.

## 5.4 Conclusión Final

La arquitectura desarrollada para el *Backend* de TeamZen demuestra empíricamente que es posible combinar el rigor metodológico de la psicometría clásica con la velocidad y personalización de la Inteligencia Artificial moderna. Al emplear un modelo *Serverless* protegido por un *Middleware* perimetral y políticas *RLS*, el proyecto logró optimizar el rendimiento del sistema, reduciendo latencias de procesamiento de segundos a milisegundos, sin comprometer en ningún momento la privacidad de la información de los colaboradores.

Esta solución no solo resuelve la problemática de lentitud y falta de accionabilidad de las encuestas laborales tradicionales, sino que sienta una base tecnológica modular, segura y económicamente viable. En definitiva, *TeamZen* representa un avance significativo hacia la digitalización ética de la salud ocupacional, entregando a los líderes organizacionales las herramientas necesarias para prevenir el desgaste profesional y fomentar entornos de trabajo sostenibles.

## REFERENCIAS BIBLIOGRÁFICAS

- **Fowler, M., & Lewis, J. (2014).** Microservices: a definition of this new architectural term. MartinFowler.com. Recuperado de <https://martinfowler.com/articles/microservices.html>
- **Groq. (2024).** Groq LPU Inference Engine. Recuperado de <https://wow.groq.com/>
- **Liu, B. (2012).** Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers.
- **Maslach, C., & Jackson, S. E. (1981).** The measurement of experienced burnout. Journal of Occupational Behaviour, 2(2), 99–113.
- **Maslach, C., Jackson, S. E., & Leiter, M. P. (1996).** Maslach Burnout Inventory Manual (3rd ed.). Consulting Psychologists Press.
- **Meta AI. (2024).** Llama 3: Open Foundation and Chat Models. Recuperado de <https://llama.meta.com/>
- **Node.js Foundation. (2024).** Node.js Documentation. Recuperado de <https://nodejs.org/docs/>
- **OWASP Foundation. (2021).** OWASP Top 10: 2021 - The Ten Most Critical Web Application Security Risks. Recuperado de <https://owasp.org/Top10/>
- **Poria, S., Cambria, E., Hazarika, D., & Majumder, N. (2017).** A Review of Affective Computing: From Theory to Applications. IEEE Access, 5, 20228-20259.
- **React. (2024).** React: The library for web and native user interfaces. Recuperado de <https://react.dev/>
- **Supabase. (2024).** Supabase Documentation: The Open Source Firebase Alternative. Recuperado de <https://supabase.com/docs>

## ANEXOS

### Anexo A: Instrumento de Evaluación *Maslach Burnout Inventory (MBI)*

El sistema digitaliza el cuestionario estándar *MBI* compuesto por 22 ítems, evaluados mediante una escala de frecuencia de 0 (Nunca) a 6 (Todos los días). El motor de evaluación clasifica el nivel de burnout en tres dimensiones clínicas utilizando los siguientes rangos y lógicas:

- **Agotamiento Emocional (AE):** Compuesto por 9 ítems (Puntaje máximo: 54).
  - Bajo burnout: 0-18 puntos.
  - Medio burnout: 19-26 puntos.
  - Alto burnout (Riesgo): 27-54 puntos.
- **Despersonalización (D):** Compuesto por 5 ítems (Puntaje máximo: 30).
  - Bajo burnout: 0-6 puntos.
  - Medio burnout: 7-9 puntos.
  - Alto burnout (Riesgo): 10-30 puntos.
- **Realización Personal (RP):** Compuesto por 8 ítems (Puntaje máximo: 48). Esta escala es inversa; un mayor puntaje indica menor burnout.
  - Bajo burnout (Alta realización): 40-48 puntos.
  - Medio burnout: 34-39 puntos.
  - Alto burnout (Baja realización): 0-33 puntos.

El diagnóstico de "*Burnout*" se confirma si el usuario o equipo puntúa en el nivel "Alto" en al menos dos subescalas, siendo Agotamiento Emocional (AE) una de ellas de manera obligatoria.

### Anexo B: Ingeniería de *Prompts* e Integración *LLM*

El sistema utiliza plantillas dinámicas estrictas para evitar alucinaciones del Modelo de Lenguaje. A continuación se presenta la estructura de inyección de contexto para el análisis de equipos.

#### B.1 Estructura del Prompt Base

Analiza estos resultados del *Maslach Burnout Inventory*:

CONTEXTO DEL EQUIPO:

Nombre: [Variables Inyectadas]

Descripción/Área: [Variables Inyectadas]

ESTADO ACTUAL (último ciclo):

Agotamiento Emocional: [Puntaje]/54 → Nivel de burnout: [Clasificación]

Despersonalización: [Puntaje]/30 → Nivel de burnout: [Clasificación]

Realización Personal: [Puntaje]/48 → Nivel de burnout: [Clasificación]

DIAGNÓSTICO ACTUAL: [Riesgo Detectado]

EVOLUCIÓN HISTÓRICA:

[Historial inyectado cronológicamente]

**B.2 Contrato de Salida Estricto (Formato JSON)** Para asegurar que el *frontend* pueda procesar la información sin errores de renderizado, se fuerza a la Inteligencia Artificial a adherirse a la siguiente estructura *JSON* mediante instrucciones del sistema:

```
{ "summary": "Resumen del estado actual y evolución histórica en máximo 2 líneas",  
  "trend_analysis": "Análisis detallado de cómo cambió el equipo a lo largo del tiempo",  
  "key_risks": ["riesgo 1", "riesgo 2"], "recommended_actions": ["acción 1", "acción 2",  
  "acción 3"], "prognosis": "Pronóstico a corto plazo si continúan las tendencias" }
```

### **Anexo C: Políticas de Seguridad a Nivel de Fila (*Row Level Security*)**

Implementación de lógica declarativa en el motor de *PostgreSQL (Supabase)* para aislar las métricas de bienestar, garantizando que el acceso a la tabla de resultados psicométricos quede estrictamente controlado.

-- Política 1: Aislamiento Individual-- Asegura que un miembro normal solo pueda leer sus propias encuestas históricas.  
`CREATE POLICY "Lectura_Encuestas_Propias" ON public.mbi_responses FOR SELECT USING (auth.uid() = user_id);`  
-- Política 2: Jerarquía de Liderazgo y Privacidad de Equipo-- Otorga permisos de lectura al líder del equipo sobre los resultados agregados, -- verificando simultáneamente su rol en la tabla puente `'team_members'`.  
`CREATE POLICY "Lectura_Administrativa_Lider" ON public.mbi_responses FOR SELECT USING ( EXISTS ( SELECT 1 FROM team_members WHERE team_members.team_id = mbi_responses.team_id AND team_members.user_id = auth.uid() AND team_members.role = 'leader' ));`