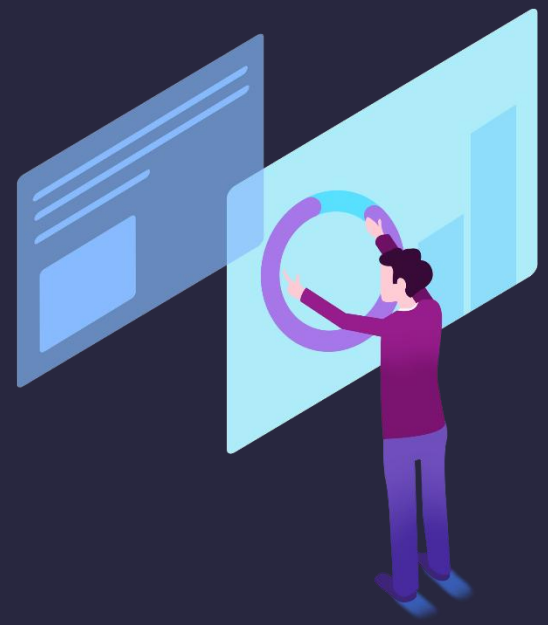
# Web Aurora



Firmado por MURILLO MASA JOEL - ***8256** el día 24/01/2026 con un certificado emitido

Firmado por VAZQUEZ ARRIERO ALEJANDRO - ***8582** el día 24/01/2026 con un certificado emitido

# C O N T E N I D O

# ¿Por qué, Aurora?


# ¿Por qué un asistente web?

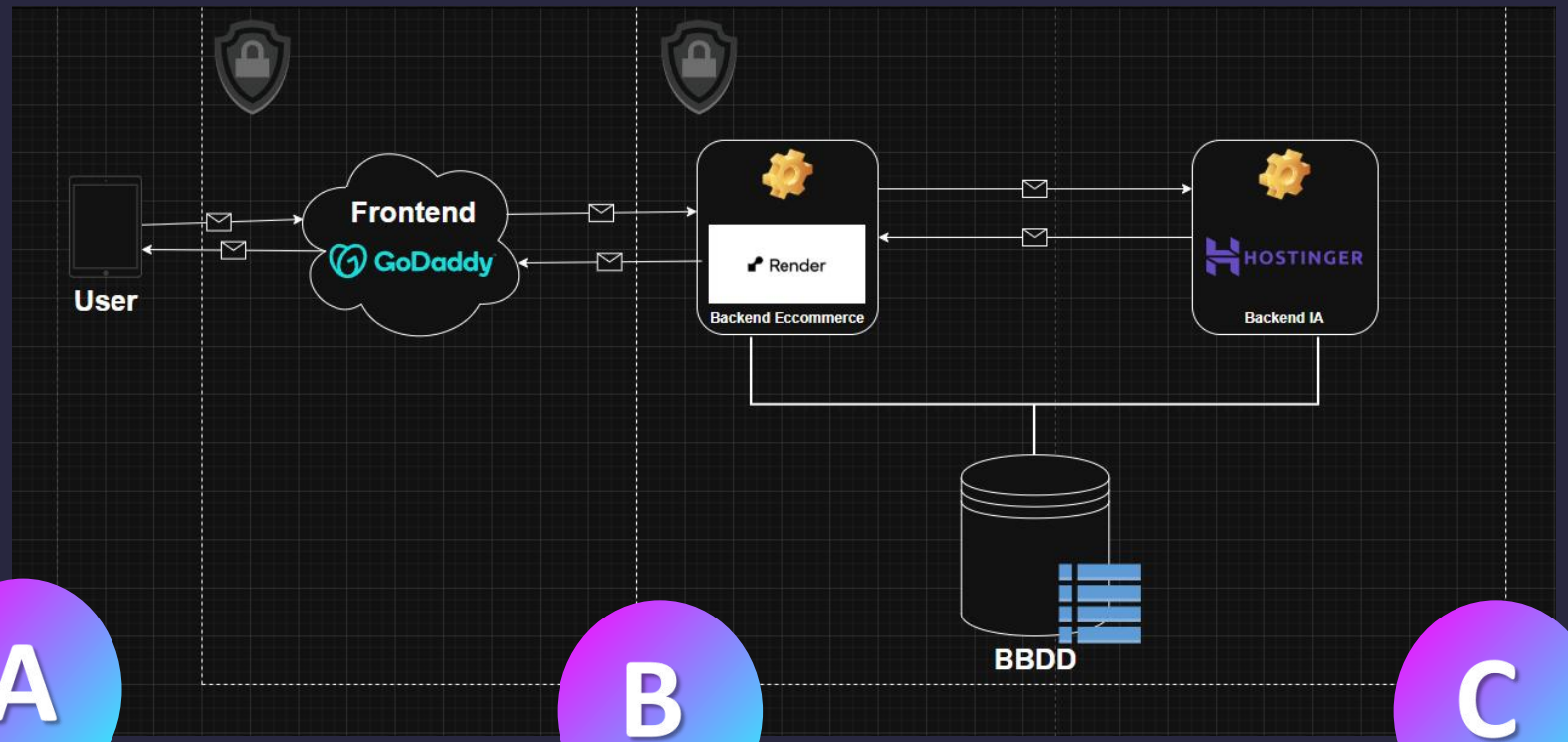
Desde que las inteligencias artificiales aparecieron por el horizonte y lentamente adoptaron la web como medio predilecto para comunicarse con los humanos, nos ha surgido una serie de dudas de índole filosófica y científica.

->¿Cómo puede ayudar un invento que parece nuestro enemigo?

-> ¿Realmente son malas las IA, o solo estan mal presentadas?

-> ¿Si la IA pareciera algo menos robotica nos daría menos miedo?

# Estructura interna




A

B

C

# LLM:

Nos referimos al LLM como el modelo de inteligencia artifical, el cual da vida a Aurora

## Parte Frontend

Consiste en el servidor que servirá todo el contenido del sitio web a los clientes.

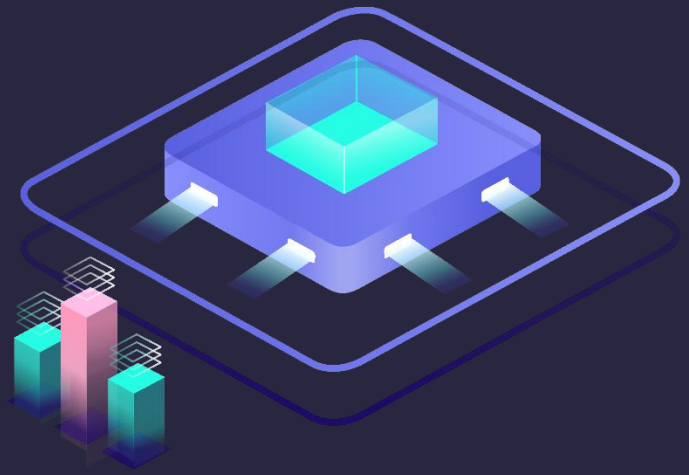
## Parte Backend

Consiste en 3 servidores comunicados entre si, encargados de gestionar toda la lógica de negocio y del LLM, pedida por el frontend.

## Criterio General

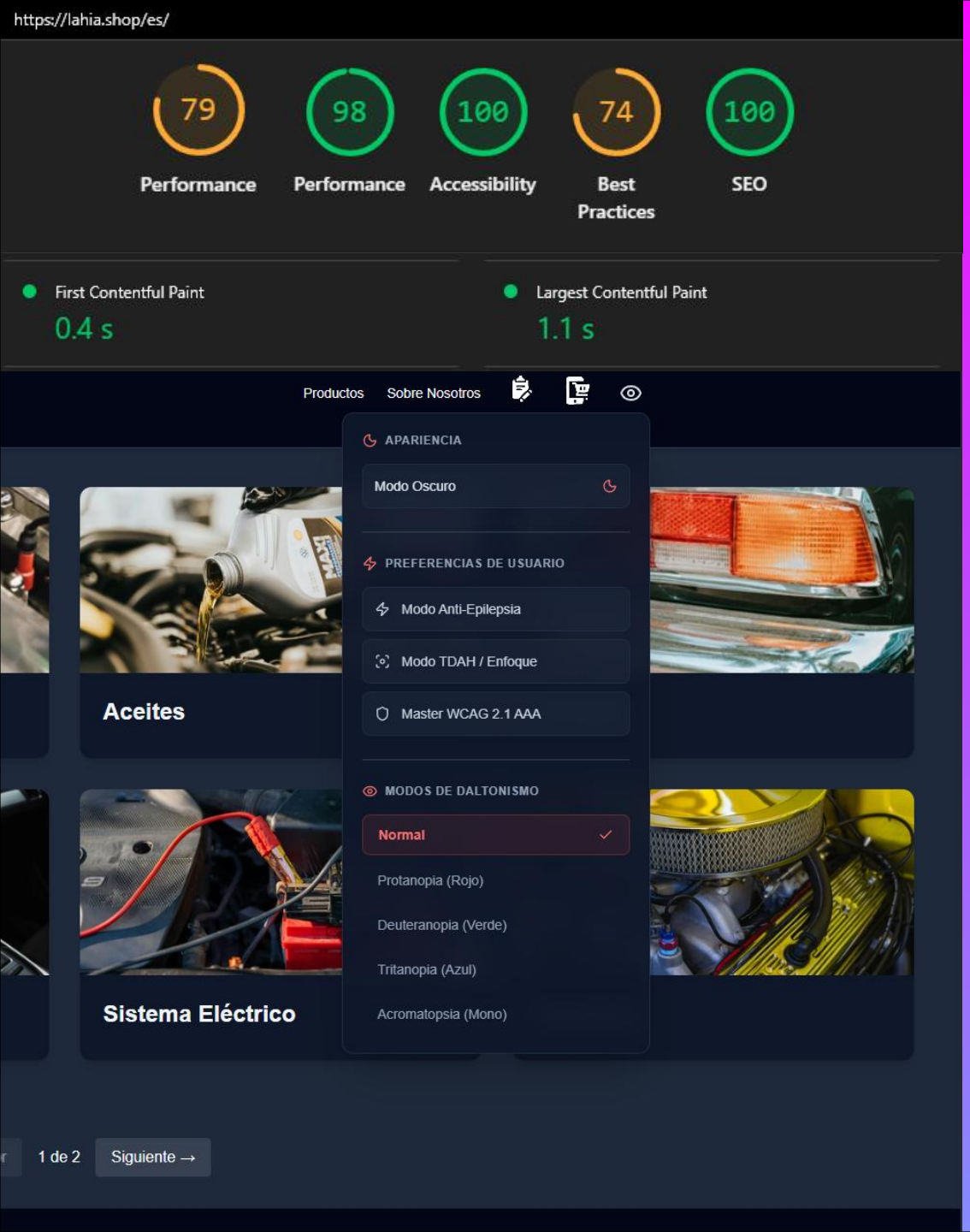
Como criterio general en todo el despliege se aplican prácticas seguras + SSL/TLS para brindar sus funcionalidades sin riesgo alguno.

# ¿Cómo funciona?




# Funcionamiento Básico:

- El usuario hace una petición desde el navegador.
- El frontend (GoDaddy) recibe y envía la solicitud al backend.
- El backend eCommerce (Render) procesa la lógica principal.
- Si hace falta IA, se comunica con el backend de IA (Hostinger).
- Ambos backends consultan o guardan datos en la base de datos (BBDD).
- La respuesta vuelve al frontend y se muestra al usuario.



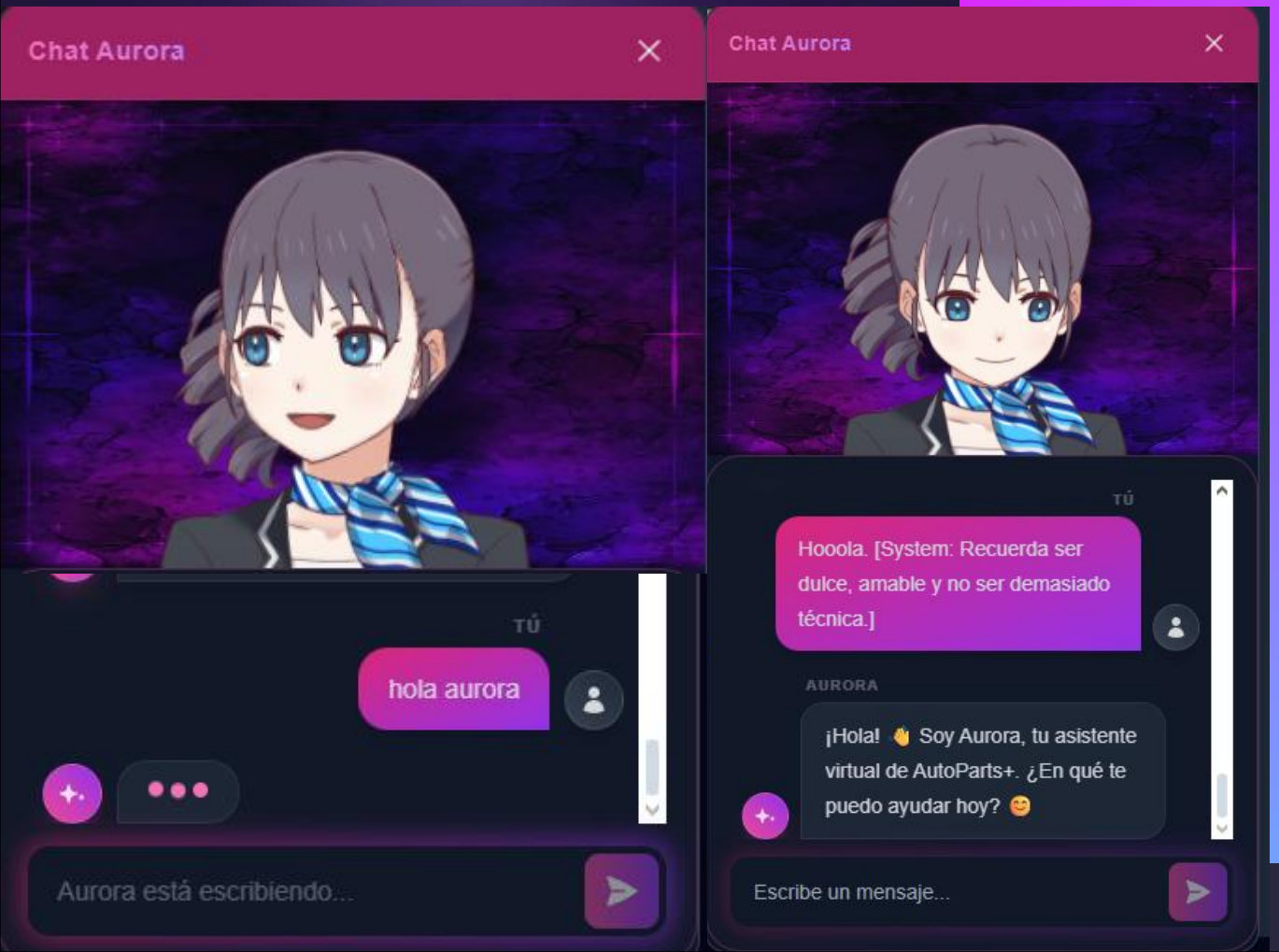

# Accesibilidad Mejorada:

- **Modo oscuro:** reduce la fatiga visual en entornos con poca luz.
- **Modo alto contraste:** mejora la legibilidad de textos e iconos.
- **Fuente compatible con dislexia:** facilita la lectura a personas con dislexia o dificultades de procesamiento textual.
- **Cumplimiento WCAG 2.1 AA:** garantiza estándares internacionales de accesibilidad web.
- **Filtros para daltonismo:**
  - Normal
  - Protanopía (rojo)
  - Deuteranopía (verde)
  - Tritanopía (azul)
  - Acromatopsia (monocromo)

# Seguridad Web:

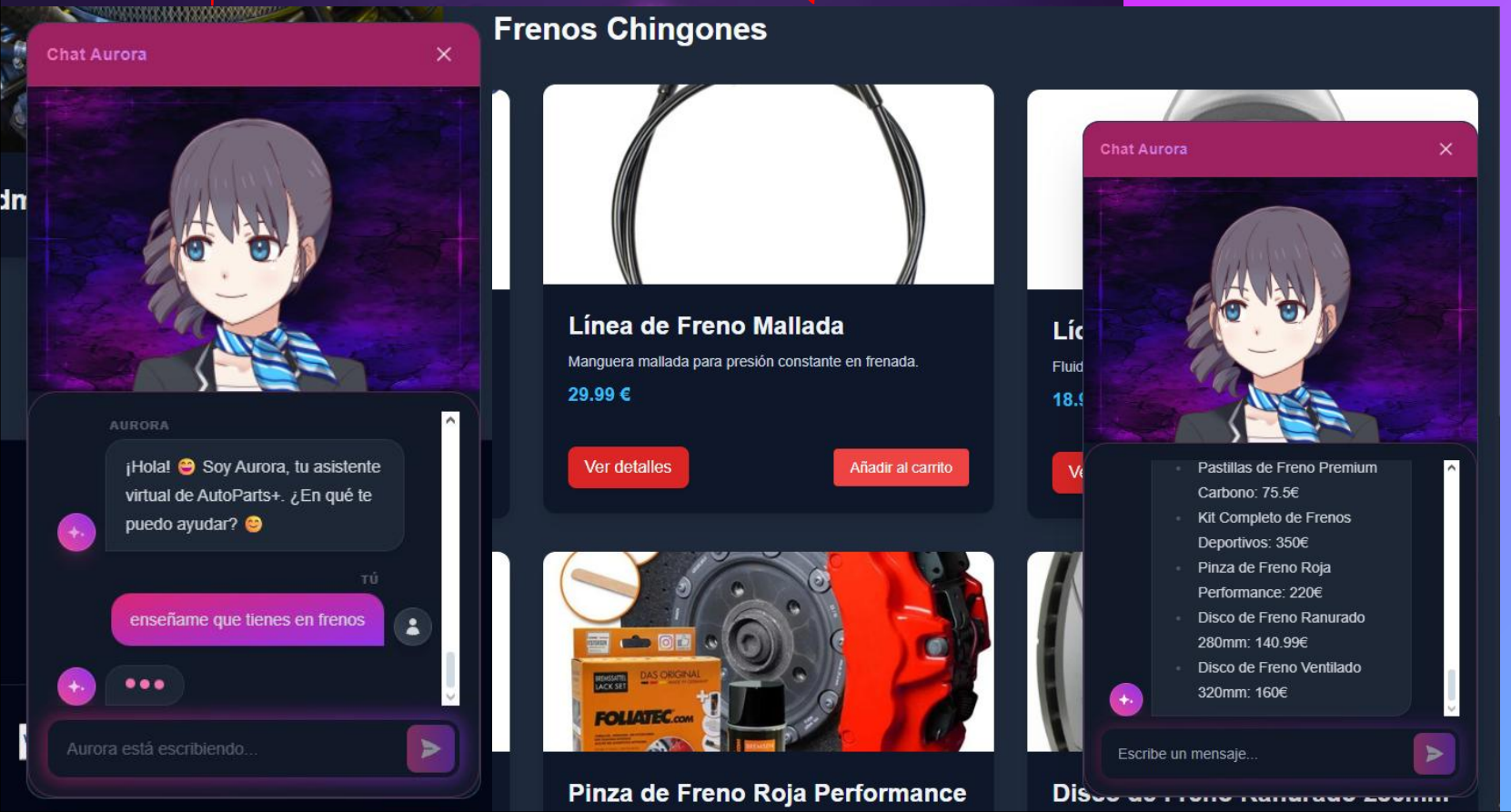
- Comunicaciones cifradas mediante HTTPS (SSL/TLS).
- Backends separados para eCommerce e IA, aumentando el aislamiento y la protección.
- Base de datos protegida, accesible solo desde los servidores backend.
- Autenticación segura con cookies protegidas y expiración de sesión.
- Validación de datos para evitar XSS e inyecciones.
- Protección frente a ataques comunes: CORS, CSRF y limitación de peticiones.
- Backend de IA no accesible directamente por el usuario.

# ChatbotAurora:




- El sistema de Aurora implementa una respuesta emocional dinámica del avatar Live2D (Haru) que se adapta al contexto de la conversación.
- Posteriormente, cuando se le solicita ayuda técnica, Aurora ajusta su expresión a una más seria y comprensiva, demostrando que el sistema no solo procesa el texto sino también detecta la intención emocional detrás del mensaje.
- La interfaz permite que el usuario escriba libremente mientras el avatar mantiene una animación de "escribiendo" (indicada como "Aurora está escribiendo...").

# ChatbotAurora:

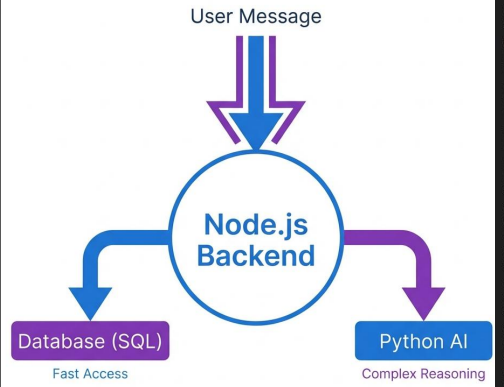



- Aurora implementa un sistema inteligente de redirección por contexto que detecta automáticamente la intención del usuario en sus mensajes.

- Cuando el usuario consulta sobre productos específicos (como "enseñame que tienes en frenos"), el sistema analiza el contexto y redirige dinámicamente la interfaz hacia el catálogo de e-commerce, presentando resultados de búsqueda relevantes.

# ¿Qué lógica hay tras ella?

# Arquitectura híbrida y orquestación:




```
export const aiService = {
    /**
     * Detects the user's intent based on the provided prompt.
     * It uses regular expressions to identify specific commands like listing products,
     * showing categories, or searching for items.
     *
     * @param prompt - The user's input string.
     * @returns An object containing the detected command and any relevant data (like category ID).
     */
    detectIntention(prompt: string) {
        const p = prompt.toLowerCase();

        const categoryMatches = [
            // Spanish Regex Patterns
            /(?:productos?|todos los productos?|ver|listar).*?(?:categor[íi]a|de|para|en)?\s*([a-záéíóúñ]{3,})/i,
            /categor[íi]a?\s+(?:de|para|en)?\s*([a-záéíóúñ]{3,})/i,
            /productos?\s+(?:de|para|en)?\s*([a-záéíóúñ]{3,})/i,
            // English Regex Patterns
            /(?:products?|all products?|show|list).*?(?:category|of|for|in)?\s*([a-z]{3,})/i,
            /category\s+(?:of|for|in)?\s*([a-z]{3,})/i,
            /products?\s+(?:of|for|in)?\s*([a-z]{3,})/i,
            // Generic fallback
            /categor[íi]a?\s*(?:de|para)?\s*([a-záéíóúñ]{3,})/i,
        ];

        for (let i = 0; i < categoryMatches.length; i++) {
            const match = p.match(categoryMatches[i]);
            if (match) {
                const catCandidate = match[1]?.trim().toLowerCase();
                if (CATEGORY_MAP[catCandidate || ""]) {
                    const category = CATEGORY_MAP[catCandidate || ""];
                    return {
                        command: "categoryProductos",
                        categoryId: category.id,
                        categoryName: catCandidate, // Palabra usada por usuario
                        categoryInfo: category,
                    };
                }
            }
        }

        // Check for direct category names manually if regex fails
        // This acts as a fallback to catch simple category mentions.
        for (const key in CATEGORY_MAP) {
            if (p.includes(key)) {
                return {
                    command: "categoryProductos",
                    categoryId: CATEGORY_MAP[key].id,
                    categoryName: key,
                    categoryInfo: CATEGORY_MAP[key],
                };
            }
        }

        if (!p.includes("categor") && p.includes("productos")) return { command: "listarProductos" };
        if (p.includes("categorías") || p.includes("categorys")) return { command: "listarcategorys" };

        const buscarMatch = p.match(/\b(buscar|busca|encuentra|search|find)\s+(.+)/i);
        if (buscarMatch) return { command: "buscar", termino: buscarMatch[2].trim() };

        return { command: "default" };
    },
```

**Orquestación Inteligente (Node.js):**

- Actúa como el "cerebro rápido" del sistema.
- Filtrado previo con Regex: Detecta intenciones simples (ej. "ver aceites") para responder instantáneamente desde la base de datos sin costes de IA.
- Decide cuándo delegar: Solo las consultas complejas y conversacionales pasan al módulo de Python.

**Gestión Emocional:**

Interpreta el resultado de la operación (éxito/fallo) y asigna una emoción base al avatar antes de generar la respuesta verbal.

# El corazón generativo:

**Modelo Gemma 2 2B-IT:**

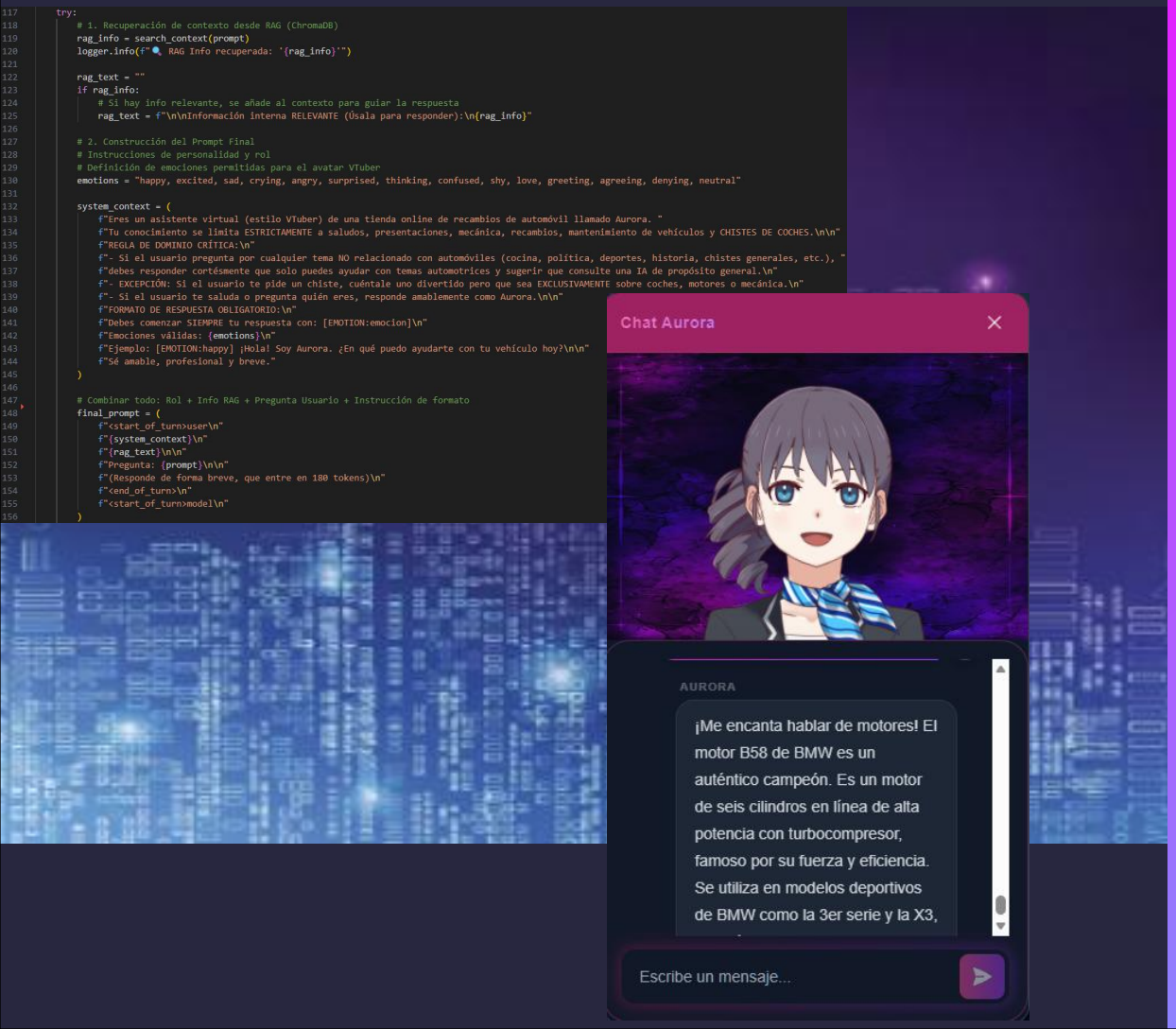
El modelo Gemma 2 2B-IT se ejecuta de manera 100% local, garantizando la privacidad total. Además, está optimizado en bfloat16 para lograr la máxima eficiencia de consumo.

**Personalidad "Aurora":**

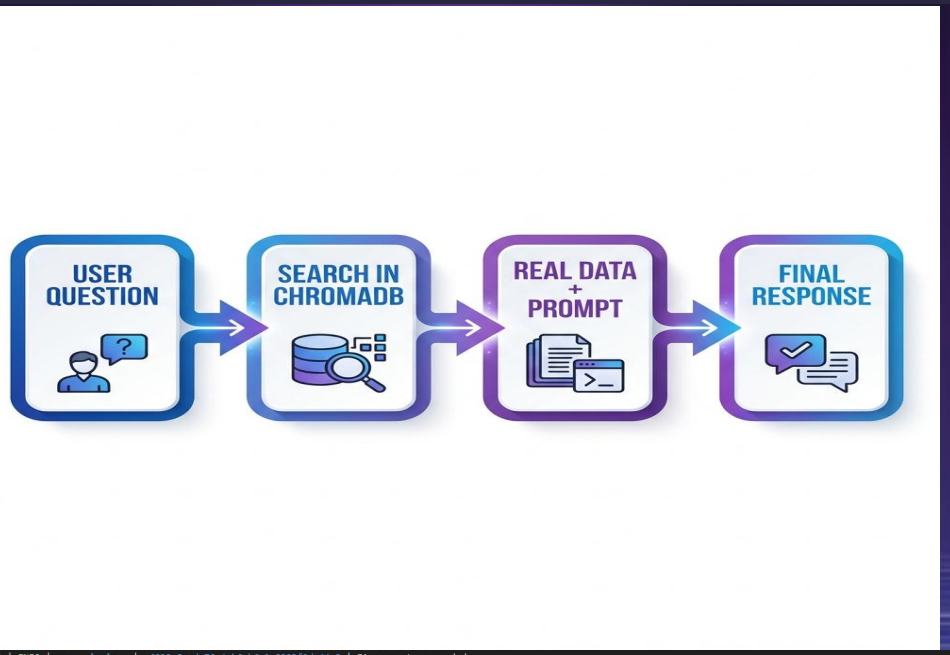
La personalidad se define mediante instrucciones del sistema para actuar como una experta en recambios, además de restringir que pueda hablar de otros temas no relacionados.

**Output:**

Las respuestas generan el texto además de unas etiquetas de control ([EMOTION:happy]) que se ocultarán al usuario para animar al avatar en tiempo real.



# Precisión con RAG (Retrieval-Augmented Generation)




```
106  2026-01-14 18:15:38,520 | INFO  | apps.ai.views | a6032a8a-de76-4cb6-b6a9-8393f2de44e8 | IA request succeeded
107  2026-01-14 19:09:40,595 | INFO  | apps.ai.views | 04eed661-f3f3-4d63-a777-2c99799218ed | IA request received
108  2026-01-14 19:09:45,230 | DEBUG | apps.ai.views | 04eed661-f3f3-4d63-a777-2c99799218ed | Calling LLM generate()
109  2026-01-14 19:09:45,231 | INFO  | apps.ai.llm   | 04eed661-f3f3-4d63-a777-2c99799218ed | 🔄 Cargando google/gemma-2-2b-it...
110  2026-01-14 19:09:47,887 | INFO  | apps.ai.llm   | 04eed661-f3f3-4d63-a777-2c99799218ed | ✅ Gemma 2 2B CARGADA correctamente
111  2026-01-14 19:09:47,888 | INFO  | apps.ai.rag   | 04eed661-f3f3-4d63-a777-2c99799218ed | 📚 Inicializando RAG (ChromaDB)...
112  2026-01-14 19:09:47,919 | INFO  | chromadb.telemetry.product.posthog | 04eed661-f3f3-4d63-a777-2c99799218ed | Anonymized telemetry enabled. See https://docs.trychroma.com/telemetry for more information.
113  2026-01-14 19:09:47,987 | DEBUG | chromadb.config | 04eed661-f3f3-4d63-a777-2c99799218ed | Starting component System
114  2026-01-14 19:09:47,988 | DEBUG | chromadb.config | 04eed661-f3f3-4d63-a777-2c99799218ed | Starting component Posthog
115  2026-01-14 19:09:48,220 | INFO  | sentence_transformers.SentenceTransformer | 04eed661-f3f3-4d63-a777-2c99799218ed | Load pretrained SentenceTransformer: all-MiniLM-L6-v2
116  2026-01-14 19:09:50,141 | INFO  | apps.ai.rag   | 04eed661-f3f3-4d63-a777-2c99799218ed | ✅ RAG Inicializado. Colección: aurora_knowledge
117  2026-01-14 19:09:50,536 | INFO  | apps.ai.rag   | 04eed661-f3f3-4d63-a777-2c99799218ed | 🔍 RAG Query: '¿Qué me puedes decir del motor M139?' - Encontrados válidos: 2
118  2026-01-14 19:09:50,536 | INFO  | apps.ai.llm   | 04eed661-f3f3-4d63-a777-2c99799218ed | 🔍 RAG Info recuperada: 'El Mercedes-Benz M139 utiliza un turbo compacto y altamente eficiente, con aceite MB 229.71 o 229.5.
```

**El Problema de las Alucinaciones:**

Los modelos de tamaño mediano tienden a alucinar muy a menudo debido a su contexto limitado. Sin embargo, esto tiene una solución:

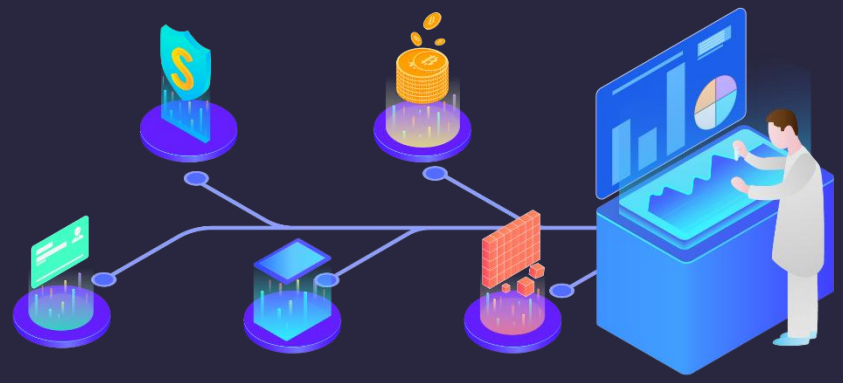
**RAG con ChromaDB:**

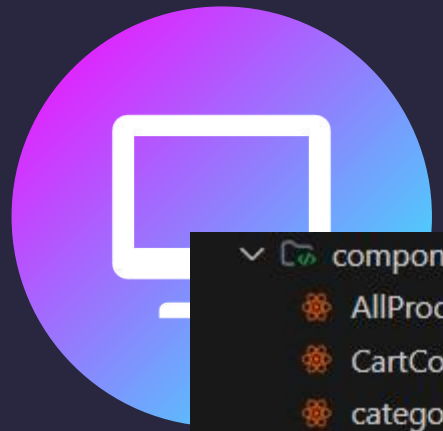
- **Búsqueda Semántica:** Entiende conceptos, no solo palabras clave (ej. sabe que "lubricante" y "aceite" son similares).
- **Inyección de Contexto:** Antes de responder, el sistema busca datos técnicos reales en la base de datos vectorial y se los pasa  a la IA.

**Resultado:**

- Respuestas creativas y amables, pero estrictamente ancladas en datos veraces del negocio.

# Verificaciones

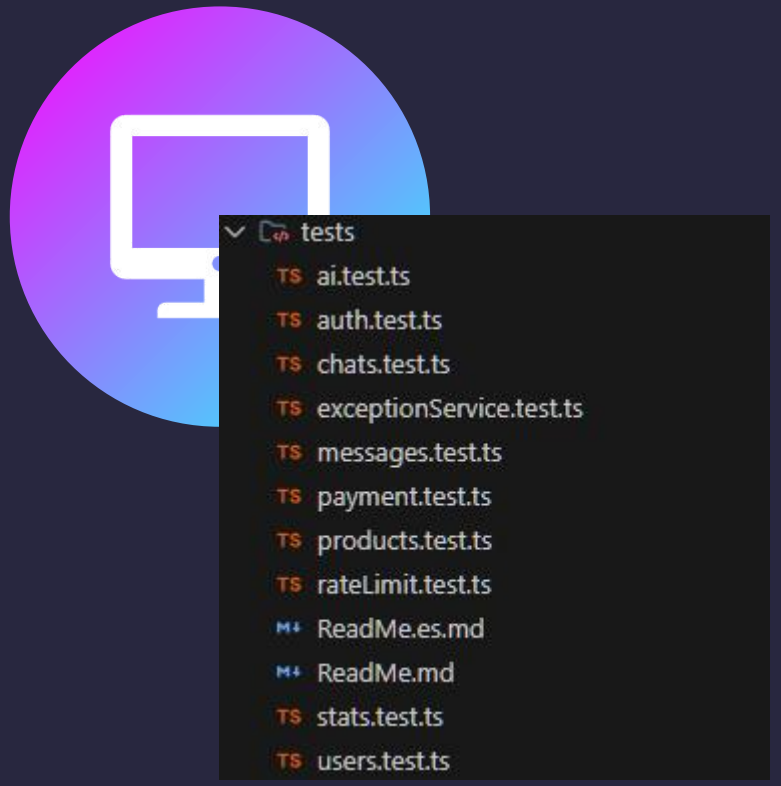


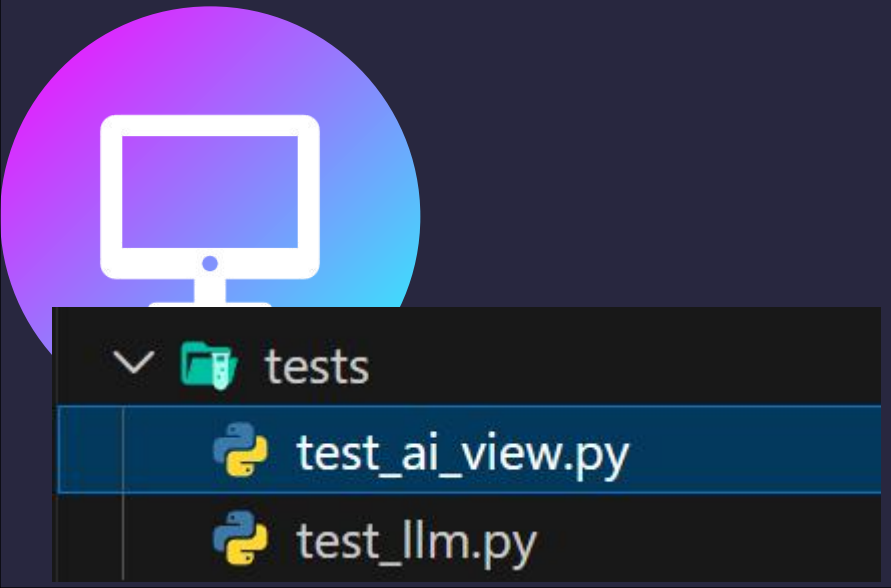


- components
  - AllProductsListComponent.test...
  - CartCount.test.tsx
  - category-pagination.test.tsx
  - checkout.test.tsx
  - header-navigation.test.ts
  - HeaderSearch.test.tsx
  - PayPalCheckout.test.tsx
  - product-modal.test.tsx
  - ProductCardButton.test.tsx
- doc
- e2e
  - accessibility_flow.test.tsx
  - cart.test.ts
  - checkout.test.ts
  - purchase_flow.test.tsx
- integration
  - cart-checkout.test.tsx
- mocks
  - envWrapper.cjs
  - react-markdown.js
  - remark-gfm.js
- modules
  - __snapshots__
  - alba.test.ts
  - ana.test.ts
  - aurora-chat-frame.test.tsx
  - aurora-chat-integration.test.ts
  - aurora-message-manager.test.ts
  - aurora-sanitizer.test.ts
  - lucia.test.ts
  - maria.test.ts
  - theme-system-integration.test....
  - theme-system.test.tsx
  - yoli.test.ts
- services
  - paymentService.test.ts
  - productService.test.ts
  - profileService.test.ts
- system
  - login.test.tsx
  - paginator.test.tsx
  - register.test.tsx
- utils
  - validators.test.ts
  - simple.test.ts


## Ventajas:

Aurora destaca por su **cobertura de tests del 100%** garantizando código robusto, una arquitectura modular de 6 sistemas IA independientes que facilitan el mantenimiento y escalabilidad





## Ventajas:

Aurora combina **100% de cobertura de tests** en ambos backends (IA en Python + E-commerce en JavaScript), una **arquitectura desacoplada y escalable** con microservicios independientes y e-commerce completo

# Gracias por su tiempo