CAÑAVERAL

Aurora

El chatbot virtual.
Memoria Técnica

Clase: 2 DAW

Nombre: Alejandro Morón Turiel

Fecha: 18/01/2026

Informática y Comunicaciones

Tabla de contenido:

Introducción:

En un *entorno digital* en constante *crecimiento*, las empresas y organizaciones deben enfrentar el reto de ofrecer un servicio de atención al cliente eficiente, así como experiencias nuevas para *competir* en un *mercado tecnológico* cada vez más *competitivo* y *globalizado*. Nuestro ChatBot surge como una *solución innovadora* a esta necesidad, al proporcionar una asistencia automatizada y amena para los usuarios que se traduce en un *factor diferenciador* y *de ahorro* para aquellas empresas que quieran implementarlo, siendo de gran ayuda para los sistema de posicionamiento web o SEO, al aumentar la retención de un usuario en el sitio web con el chatbot instalado.

El presente proyecto tiene como *propósito* el *desarrollo* integral de un *ChatBot* inteligente y accesible, destinado a su *implantación en páginas web de diversa índole*, cuyo *objetivo central es asistir al usuario en la resolución de dudas, consultas frecuentes y problemas comunes que, de otro modo, requerirían la intervención de un equipo de soporte técnico humano*, inexistente en algunas Pymes o autónomos sobretodo en las empresas de menos de 10 trabajadores registradas. *Fuente: (Instituto Nacional de Estadística (INE). (2024). Encuesta sobre el uso de las tecnologías de la información y las comunicaciones (TIC) y el comercio electrónico en las empresas. Año 2023 - Primer trimestre de 2024.)*

Indicadores sobre uso TIC en las empresas - 2023-2024

		Empresas con menos de 10 empleados	Empresas con más de 10 empleados
Disponen de ordenadores	1	87,90	99,58
Tiene conexión a internet	1	83,81	99,13
Tiene conexión a internet y página web	2	33,21	81,84
Utilizan medios sociales	2	34,58	64,70
Realizan ventas por comercio electrónico	1	13,06	30,67
Realizan compras por comercio electrónico	1	19,27	39,04

Tal y como se puede apreciar en los datos obtenidos del INE, el *33,21% de las empresas con menos de 10 empleados cuentan con un sitio web, normalmente limitado por su presupuesto*, siendo un proyecto interesante para agilizar o posibilitar una asistencia al usuario en empresas de tan bajo perfil.

Teniendo esto en mente y el contexto del mercado actual de soluciones digitales muestra un creciente interés en herramientas que mejoren la experiencia del usuario dentro de las plataformas web. Entre estas herramientas, los ChatBots destacan por su *capacidad para orientar a los visitantes, ayudándoles a localizar rápidamente secciones, funcionalidades o información relevante.*

Estas soluciones integran tecnologías de procesamiento de lenguaje natural (NLP) que facilitan la comprensión de las consultas de los usuarios y permiten ofrecer respuestas claras, precisas y contextualmente relevantes, lo que contribuye a una navegación más eficiente y a una mayor satisfacción del cliente.

Otro elemento a tener en cuenta es la *accesibilidad*, el proyecto se plantea con un firme *compromiso hacia la accesibilidad y la inclusión digital*, incorporando características que faciliten la interacción de personas con discapacidades físicas, visuales o auditivas.

Tres de cada cuatro personas con discapacidad grave de entre 16 y 74 años en España -concretamente, el **75,9%- utilizan Internet con regularidad**, cuando ese porcentaje entre la población sin discapacidad es del **93,5%**.

Fuente: (Discapnet. (2025). El 76% de las personas con discapacidad en España usan Internet, 20 puntos menos que el resto)

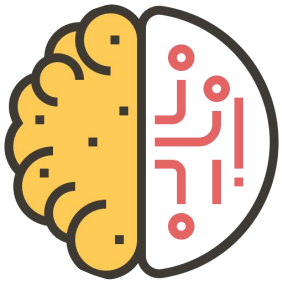
Tal y como se puede apreciar en los datos aportados en la fuente, el *75,9% de los discapacitados españoles usan internet siendo otro colectivo de usuarios a tener en cuenta en lo que al diseño de proyectos web se refiere*, haciendo énfasis más que nunca en la accesibilidad web un requisito y objetivo de nuestro proyecto.

Por eso es necesario garantizar la compatibilidad con lectores de pantalla, comandos de voz para usuarios con limitaciones motrices, así como opciones de alto contraste y textos ampliables para personas con dificultades visuales siguiendo las recomendaciones de las Directrices de Accesibilidad para el Contenido Web (WCAG 2.1), el estándar actual.

Otro factor importante para el proyecto es el aspecto visual de la inteligencia artificial. *La apariencia del ChatBot*, incluyendo su diseño gráfico, expresiones faciales y elementos humanizados, *influye directamente en la percepción de cercanía, confianza y accesibilidad por parte del usuario*. Un diseño cuidadosamente humanizado y atractivo puede mejorar la interacción, hacerla más intuitiva y generar una experiencia más agradable, favoreciendo la retención y satisfacción del usuario en la plataforma web.

Un ejemplo a tener en cuenta es el contexto de ventas en línea, la aceptación de interfaces de inteligencia artificial con apariencia humana presenta matices importantes. *Un estudio realizado en China reveló que características humanas en los chatbots, como la empatía y la calidez, incrementan la confianza del consumidor, lo que puede traducirse en una mayor intención de compra y satisfacción en entornos de comercio electrónico* *Fuente:* (*Ding, Y., & Najaf, M. (2024). Interactividad, humanidad y confianza: un enfoque psicológico para la adopción de chatbots de IA en el comercio electrónico. BMC Psychology, 12(1), 83.)*

Motivación del proyecto:



En la actualidad, Aurora se concibe como una respuesta humana, ética y tecnológica ante los desafíos crecientes de la interacción digital. A medida que los entornos web incorporan más automatización e inteligencia artificial, *aumentan también las dificultades de acceso y comprensión para amplios sectores de la población. Las personas mayores, con discapacidad o con baja alfabetización digital siguen encontrando barreras que limitan su participación en la sociedad digital*. Según la UNESCO (2025), más del *40 % de las páginas y aplicaciones públicas no cumplen los estándares* mínimos de accesibilidad establecidos por la Web Accessibility Initiative (WAI), lo que refuerza la urgencia de soluciones centradas en la inclusión y el acompañamiento tecnológico.

La motivación principal de Aurora *es poner a las personas en el centro de la experiencia tecnológica*, transformando la relación tradicional entre usuario y web en un vínculo más empático y adaptativo. No se trata únicamente de un chatbot que responde preguntas, sino de un asistente inteligente que *acompaña al usuario en su recorrido digital, ajustándose a sus capacidades y contexto en tiempo real. Este enfoque, basado en la accesibilidad emocional y cognitiva*, busca crear una experiencia fluida, segura y amable, donde cada persona pueda desenvolverse sin miedo a equivocarse ni a sentirse excluida. Tal y como señala la UNESCO (2025) en su Recomendación sobre la Ética de la Inteligencia Artificial, la tecnología debe fortalecer las capacidades humanas, respetar la diversidad cultural y promover la justicia social. *Fuente: (UNESCO. (2025). Recommendation on the Ethics of Artificial Intelligence: Implementation Updates).*

Aurora *se dirige* especialmente a *personas mayores, con discapacidad, dificultades cognitivas o baja alfabetización digital*, ofreciendo un entorno accesible y guiado. El sistema integra técnicas de simplificación de interfaces, asistencia paso a paso y comunicación clara en lenguaje natural, permitiendo adaptar la interacción a cada nivel de comprensión. Esta aproximación coincide con las recomendaciones del European Accessibility AI Action Plan (2025), que insta a diseñar *sistemas inteligentes que detecten y compensen dificultades de uso antes de que generen frustración o abandono digital*. *Fuente: (European Accessibility AI Action Plan. (2025). AI for Inclusive Europe: Designing accessibility-first systems. European Commission).*

El proyecto *se apoya en un marco ético sólido*, alineado con las nuevas directrices de la Comisión Europea (2025) sobre el *uso responsable de la inteligencia artificial*. Aurora garantiza *transparencia, privacidad y autonomía*: las decisiones se explican de forma clara, los datos se procesan con cifrado local y el usuario conserva siempre el control sobre sus preferencias. La normativa europea recalca la prohibición del uso de IA para manipular decisiones o explotar vulnerabilidades derivadas de la edad o la discapacidad, asegurando así un entorno de confianza y respeto. *Fuente: (European Commission. (2021). Ethics guidelines for trustworthy AI.)*

Además, Aurora se construye como una plataforma viva y en mejora continua, *capaz de aprender de la interacción con sus usuarios para ofrecer una asistencia cada vez más precisa y humana*. Gracias al aprendizaje adaptativo y a las tecnologías de accesibilidad integradas, puede anticiparse a los problemas más comunes y proponer soluciones proactivas, contribuyendo a una experiencia digital más equitativa. Según Forbes (2025), las tendencias en gobernanza de IA destacan la necesidad de sistemas auditables, explicables y evolutivos, que mantengan la confianza del usuario como eje central. *Fuente: (Forbes. (2025, enero 9). AI Governance in 2025: Expert predictions on ethics, tech, and law).*

Finalmente, Aurora *aspira a convertirse en un referente de accesibilidad ética y diseño humano en la web del futuro*. No se limita a resolver un problema puntual, sino que propone un nuevo paradigma donde la empatía, la comprensión y la inclusión sean principios estructurales del desarrollo digital. Tal como expone Frontiers in Digital Health (2025), el diseño ético de la IA debe centrarse en la equidad y en el fortalecimiento de las capacidades humanas, no en la sustitución de la interacción social. En este sentido, Aurora no solo acompaña al usuario, sino que inspira un cambio de mentalidad: hacia una web más consciente, respetuosa y emocionalmente inteligente. *Fuente: (Frontiers in Digital Health. (2025). Biases in AI: Acknowledging and addressing the inevitable.)*

Beneficios esperados:

La implementación y despliegue exitoso de Aurora promete generar un *impacto profundo y multidimensional*. No se trata únicamente de una mejora tecnológica, sino de un cambio de paradigma en cómo las empresas digitales interactúan con sus usuarios, generando *valor en cuatro ejes fundamentales: social, económico, técnico y psicológico*.

Impacto Social y Democratización Digital (Beneficios para la Sociedad)

- *Reducción Significativa de la Brecha Digital:* Aurora actúa como un "puente cognitivo" para usuarios no nativos digitales. Al sustituir interfaces de navegación complejas (menús anidados, filtros paramétricos) por una conversación natural ("Quiero unos zapatos rojos baratos"), se empodera a colectivos tradicionalmente excluidos tecnológica y socialmente.

- *Accesibilidad Universal Real (Tecnología Asistiva):* Gracias a la integración del módulo LUCIA (Layer for Universal Cognitive & Inclusive Access), el proyecto trasciende el cumplimiento burocrático de la normativa. Ofrece beneficios tangibles para:

  - *Neurodivergencia (TDAH/Autismo):* Modos de "Enfoque" que eliminan distracciones visuales.

  - *Discapacidad Visual:* Soporte nativo para lectores de pantalla y modos de alto contraste validados (WCAG AAA).

  - *Fotosensibilidad:* Protección activa contra estímulos epilépticos.

- Inclusión Generacional: Facilita que la tercera edad realice gestiones online (*compras, consultas*) con la misma autonomía que un usuario joven, fomentando su independencia y autoestima.

Retorno de Inversión y Eficiencia Operativa (Beneficios para la Organización)

- *Optimización de Recursos Humanos:* La automatización de la atención al cliente de primer nivel (Preguntas Frecuentes, Estado de Pedidos, Ayuda de Navegación) libera al personal humano de tareas repetitivas, permitiéndoles centrarse en incidencias complejas que requieren juicio crítico y creatividad. Esto reduce el coste operativo por ticket y mejora la satisfacción del empleado.

- *Mejora del SEO y Dwell Time:* La presencia de un avatar interactivo retiene al usuario en la página por más tiempo (aumentando el "Time on Site"). Los motores de búsqueda como Google interpretan esta retención como una señal de calidad, mejorando el posicionamiento orgánico (SEO) de la plataforma.

- *Incremento en la Tasa de Conversión:* Un asistente proactivo que resuelve dudas en el momento de la decisión de compra ("¿Tenéis talla 42?", "¿Llega para el viernes?") reduce drásticamente el abandono del carrito, actuando como un vendedor experto 24/7.

- *Diferenciación Competitiva:* En un mercado saturado de e-commerces idénticos (plantillas estándar), Aurora ofrece una identidad de marca única, memorable e innovadora ("La tienda que te habla").

Excelencia Técnica y Escalabilidad (Beneficios Tecnológicos)

- *Arquitectura de Rendimiento Extremo (Web Vitals):* El uso de *Islands Architecture (Astro)* garantiza que la inclusión de funcionalidades avanzadas de IA no penalice la velocidad de carga. Esto demuestra que es posible crear webs ricas e interactivas que funcionen fluidamente incluso en dispositivos móviles de *gama baja y redes 4G precarias*.

- *Modularidad y Reusabilidad:* El desarrollo basado en componentes aislados (Módulo AURORA, Módulo CART) *permite que el chatbot pueda ser "exportado" e integrado en otros sistemas* o páginas web con cambios mínimos, funcionando como un micro-frontend independiente.

- *Seguridad y Privacidad por Diseño:* Al procesar la lógica crítica en el cliente o mediante APIs seguras, y al delegar los pagos a pasarelas certificadas (Stripe), se minimiza la superficie de ataque y se protege la integridad de los datos sensibles del usuario.

Bienestar Psicológico y Experiencia de Usuario (UX Emocional)

- *Reducción de la Ansiedad Tecnológica:* Para muchos usuarios, enfrentarse a formularios de error o procesos de pago complejos genera estrés. La interfaz conversacional, con su tono amable y guía paso a paso, mitiga esta frustración, transformando una experiencia potencialmente traumática en una interacción fluida.

- *Conexión Emocional (HCI - Human Computer Interaction):* El uso de un rostro antropomórfico (*Avatar*) activa mecanismos neuronales de empatía y confianza en el usuario (*Pareidolia positiva*). Esto genera una lealtad hacia la plataforma mucho más fuerte que la que se consigue con una interfaz puramente utilitaria.

- *Autonomía Percibida:* El usuario siente que tiene el control de la interacción (*"Yo pregunto, ella responde"*), en lugar de sentirse perdido en un laberinto de menús diseñados por terceros.

Objetivos:

General:

El objetivo principal de este Trabajo de Fin de Grado es diseñar, desarrollar e implementar una plataforma de comercio electrónico de nueva generación, integral y escalable, potenciada por un *Asistente Virtual Inteligente y Empático*.

Esta solución tecnológica busca demostrar empíricamente cómo la convergencia de tecnologías web de alto rendimiento (*Arquitectura de Islas*) con *Inteligencia Artificial Generativa y principios de Accesibilidad Universal*, puede resolver de raíz las barreras de fricción, exclusión y deshumanización que aquejan al sector del retail digital actual.

El sistema resultante, *Aurora Chat & E-commerce*, debe trascender la funcionalidad básica de una tienda on-line para convertirse en un Sistema de Acompañamiento Digital Adaptativo, capaz de:

1. Comprender el lenguaje natural y el contexto emocional del usuario.

2. Adaptar su interfaz visual a las capacidades cognitivas y sensoriales del visitante.

3. Ofrecer una experiencia de compra fluida, segura y técnicamente robusta, validada mediante estándares industriales.

Específicos:

Para materializar el objetivo general, se ha desglosado el proyecto en una estructura jerárquica de metas técnico-operativas, diseñadas para cubrir todo el *ciclo de vida del software (SDLC)* y las múltiples dimensiones del problema.

Investigación, Análisis y Diseño (Fase de Fundamentación)

Esta fase busca establecer los cimientos teóricos y arquitectónicos antes de escribir código.

- Fundamentación en HCI (*Interacción Humano-Ordenador*):
  - Investigar el Efecto Proteus y la teoría del Valle Inquietante para diseñar un avatar (*"Haru"*) que genere confianza sin causar rechazo.
  - Analizar patrones de diseño conversacional híbridos (*Texto + Voz + Gesto*) para reducir la carga cognitiva del usuario.

- Arquitectura de Software y Patrones:
  - Diseñar una arquitectura modular basada en *Domain-Driven Design (DDD)*, separando claramente los dominios de *AURORA (IA) y SHOP (E-commerce)*.
  - Definir una estrategia de *Micro-Frontends lógicos mediante Astro Islands*, permitiendo que el chatbot y la tienda evolucionen a velocidades distintas.
- Estrategia de Accesibilidad Normativa:
  - Establecer una matriz de cumplimiento cruzada con la norma *UNE-EN 301 549:2022* y las pautas *WCAG 2.1 Nivel AA*.
  - Definir criterios de aceptación específicos para discapacidades visuales (*ceguera, baja visión, daltonismo*), auditivas y cognitivas.

Ingeniería Frontend y Optimización (Fase de Núcleo)

Objetivos centrados en la excelencia técnica y el rendimiento, críticos para la retención de usuarios.

- Implementación de Arquitectura de Islas (Island Architecture):
  - Utilizar Astro como meta-framework para renderizar HTML estático (*Zero JS*) en el 90% de la superficie de la web (*Header, Footer, Listados*).
  - Configurar directivas de hidratación (*client:visible, client:idle*) para diferir la ejecución de scripts pesados hasta que sean estrictamente necesarios.
- Optimización de Core Web Vitals:
  - Alcanzar un LCP (*Largest Contentful Paint*) inferior a 2.5 segundos en redes 4G simuladas.

- Minimizar el CLS (*Cumulative Layout Shift*) reservando espacio estructural para imágenes y esqueletos de carga.

- Reducir el TBT (*Total Blocking Time*) delegando el procesamiento de IA y lógica compleja a Web Workers o servicios backend.

➢ Gestión de Estado Reactiva y Persistente:

- Implementar un store atómico con Jotai que elimine los re-renderizados innecesarios (*problema común en Context API*).

- Desarrollar un middleware de persistencia que sincronice automáticamente el carrito de compras con *SessionStorage*, resistente a recargas accidentales.

Inteligencia Artificial y Experiencia Multimodal (Fase de Valor)

El corazón diferenciador del proyecto, donde la tecnología se encuentra con la emoción.

➢ Desarrollo del Motor ANA (*Aurora Neural Analyzer*):

- Implementar un proxy de API que enriquezca los prompts del usuario con contexto de navegación (*ej: "El usuario está viendo la página de Zapatos"*).

- Diseñar el sistema de clasificación de intenciones: Navegación, Soporte Técnico, Consulta de Producto, Charla Casual.

➢ Sistema de Renderizado de Avatar Live2D:

- Integrar el *SDK Cubism 4.0* en un contexto *WebGL* optimizado con *PixiJS*.

- Programar una máquina de estados finitos (FSM) para gestionar las transiciones suaves entre animaciones (*Idle -> Talking -> Happy*).

- Sincronización Labial y Voz (*TTS*):

  - Crear un algoritmo *heurístico de Lip-Sync* en el cliente que analice la densidad de vocales del texto para modular la apertura de la boca del avatar en tiempo real, evitando latencias de audio externas.

  - Integrar la *Web Speech API* para síntesis de voz nativa, reduciendo la dependencia de APIs de terceros costosas.

Accesibilidad Cognitiva y Sensorial (Fase de Inclusión - Módulo LUCIA)

Objetivos destinados a derribar barreras de acceso.

- Simulación y Corrección de Daltonismo:

  - Implementar filtros SVG (*<feColorMatrix>*) a nivel de raíz (*<html>*) que reasignen el espectro de colores para usuarios con *Protanopia, Deuteranopia y Tritanopia*.

- Protección Neurológica (*Modo Epilepsia*):

  - Desarrollar un sistema de eventos global que, al activarse, deshabilite todas las animaciones CSS, GIFs y movimientos del avatar (*Kill-switch*).

- Soporte de Tecnologías Asistivas:

  - Implementar una estructura semántica HTML5 rigurosa (*<main>, <nav>, <aside>*).

  - Configurar regiones *ARIA-Live* para que los lectores de pantalla anuncien los mensajes del chat automáticamente sin perder el foco de navegación.

Infraestructura, Seguridad y Calidad (Fase de Operaciones)

Garantizar que el sistema sea *robusto, seguro y mantenible* a largo plazo.

- Seguridad Transaccional (*PCI-DSS*):
  - Integrar Stripe Elements para encapsular la entrada de datos de tarjetas en iFrames seguros, garantizando que el servidor propio nunca procese información financiera sensible (*SAQ A*).
- Estrategia SEO para SPAs/Islas:
  - Generar metadatos dinámicos (*Open Graph, Twitter Cards*) para cada ficha de producto en tiempo de construcción (*SSG*) o solicitud (*SSR*).
  - Implementar generación automática de *sitemap.xml* y *robots.txt* para maximizar la indexación.
- Automation Testing y CI/CD:
  - Configurar un pipeline en GitHub Actions que bloquee integraciones si no pasan los tests.
  - Alcanzar una cobertura de código (*Code Coverage*) superior al 80% en los módulos críticos (*Carrito, ANA*) mediante Jest y React Testing Library.

Contexto actual:

Estado del arte:

En la actualidad, los asistentes virtuales han evolucionado significativamente, *integrando inteligencia artificial (IA) para ofrecer interacciones más naturales y personalizadas. Plataformas como ChatGPT* de OpenAI y Character.AI han destacado por su capacidad para *generar respuestas coherentes y contextualmente precisas* Sin embargo, estas soluciones suelen depender de servicios backend complejos y no siempre permiten una personalización directa de la interfaz visual en el frontend.

Fuente: (IMB. What is ChatGPT and a LLM Model?)

Por otro lado, aplicaciones como Replika han incorporado *avatares virtuales para mejorar la interacción con los usuarios*, buscando una experiencia más cercana y humanizada. Sin embargo, la mayoría de estas implementaciones requieren recursos gráficos pesados o entornos 3D complejos, lo que puede dificultar su integración en aplicaciones web ligeras.

En este contexto, Grok, desarrollado por xAI, ha emergido como una alternativa innovadora. Lanzado en noviembre de 2023, *Grok es un chatbot de IA generativa que se distingue por su capacidad para acceder a información en tiempo real a través de la plataforma X* (anteriormente conocida como Twitter) y por su tono de respuesta único, *que combina humor y rebeldía, algo muy humano*. Grok ha sido diseñado para ofrecer respuestas rápidas y precisas, integrándose de manera fluida con el ecosistema digital actual *Fuente: (Uniathena. What is Grok?).*

Grok utiliza una arquitectura avanzada basada en transformadores, similar a otros modelos de lenguaje de gran escala, pero con mejoras específicas que optimizan su rendimiento y capacidad de respuesta. Además, se ha integrado con diversas plataformas, incluyendo aplicaciones móviles y vehículos Tesla, ampliando su alcance y funcionalidad

Fuente: (Everyday AI. (2025). Grok: Architecture and integrations.)

Otro elementos que ha germinado en los últimos años son los VTubers, que han revolucionado la interacción digital *al combinar avatares virtuales animados con comunicación en tiempo real*. Plataformas como YouTube y Twitch han permitido a creadores transmitir *utilizando avatares 2D o 3D* que reaccionan a sus movimientos y expresiones faciales mediante técnicas de captura de movimiento o software de seguimiento facial. Este enfoque demuestra cómo *los avatares virtuales pueden generar una conexión más cercana con los usuarios, aumentando la participación y la sensación de interacción directa*. En proyectos de ChatBots, esta misma lógica puede aplicarse para que un avatar virtual 2D interactúe con la IA alojada en el backend, mostrando respuestas visuales y expresiones que refuercen la comunicación textual generada por el sistema.

Fuente: (Hyte. What is a VTuber?)

Un ejemplo destacado de esta integración es *Neuro-sama*, un *VTuber controlado completamente por IA*. Desarrollada por el programador Vedal, Neuro-sama utiliza un modelo de lenguaje grande para generar respuestas en tiempo real, mientras controla un avatar 2D animado *que interactúa con los usuarios a través de* plataformas como *Twitch y YouTube*. Esta implementación demuestra cómo la combinación de IA avanzada y avatares virtuales puede ofrecer experiencias conversacionales totalmente autónomas, personalizadas y visualmente atractivas, sirviendo como referente para la implementación de ChatBots con avatares 2D interactivos en el frontend.

Fuente: (Wikipedia. Neuro-sama.)

El presente proyecto se inspira en estas tendencias, *proponiendo un asistente virtual que combina IA en el backend con un avatar virtual 2D en el frontend*. Esta combinación busca ofrecer una experiencia de usuario inmersiva y eficiente, aprovechando las capacidades de la IA para la comprensión y generación de lenguaje, mientras se mantiene una interfaz visual ligera y accesible.

Conceptos clave:

Para *comprender el desarrollo y alcance del proyecto*, es fundamental establecer los conceptos clave que servirán de base teórica y técnica. Estos términos permiten definir con claridad los elementos que intervienen en la creación del ChatBot, la integración de la inteligencia artificial en el backend y la implementación del avatar virtual 2D en el frontend:

Palabra:	Definición:
Inteligencia Artificial (IA)	Conjunto de técnicas y algoritmos que permiten a un sistema imitar capacidades cognitivas humanas, como razonamiento, aprendizaje y generación de lenguaje.
Procesamiento de Lenguaje Natural (PLN / NLP)	Rama de la IA que permite interpretar, comprender y generar lenguaje humano de manera coherente.

Palabra:	Definición:
Modelo de lenguaje (LLM, Large Language Model)	Algoritmo entrenado con grandes volúmenes de texto para generar respuestas contextualmente precisas.
UI (User Interface)	Diseño de los elementos visuales con los que interactúa el usuario.
UX (User Experience)	Experiencia del usuario; se centra en la eficiencia, satisfacción y facilidad de uso del sistema.
Avatar virtual 2D	Personaje animado que representa al ChatBot y que expresa emociones y gestos dentro del frontend.
Interfaz multimodal	Sistema que permite interacción a través de distintos medios, como texto, voz y gráficos animados.

Captura de movimiento (Motion Capture)	Técnica que registra los movimientos de una persona para animar un avatar virtual.
Controlador de eventos	Sistema que detecta y gestiona acciones del usuario, disparando respuestas de IA y animaciones del avatar.
Microinteracciones	Pequeñas animaciones o efectos que refuerzan la respuesta visual del avatar frente a las acciones del usuario.
Backend	Parte del sistema que procesa la lógica de la aplicación, ejecuta algoritmos de IA, gestiona datos y envía respuestas al frontend.
Frontend	Parte de la aplicación que interactúa directamente con el usuario, gestionando la presentación visual y la experiencia de uso.

Palabra:	Definición:
API (Application Programming Interface)	Conjunto de reglas que permite la comunicación entre sistemas o entre frontend y backend.
Framework	Estructura de desarrollo que facilita la organización del código y la integración de funcionalidades en aplicaciones web.
Islas de interactividad (Islands Astro) Architecture)	Técnica de Astro que permite cargar únicamente componentes interactivos con JavaScript, manteniendo el resto del sitio estático.
SSR (Server-Side Rendering)	Renderizado de páginas en el servidor antes de enviarlas al cliente, mejorando tiempo de carga y SEO.
Hydration	Proceso de activar un componente estático para hacerlo interactivo en el navegador.

Lazy loading	Técnica que carga imágenes o recursos solo cuando son visibles en pantalla, mejorando rendimiento y SEO.
SEO (Search Engine Optimization)	Conjunto de prácticas que mejoran la visibilidad de un sitio web en motores de búsqueda como Google.
Core Web Vitals	Métricas de rendimiento que Google usa para evaluar la experiencia de usuario, incluyendo LCP, FID y CLS.
CLS (Cumulative Layout Shift)	Mide la inestabilidad visual de una página, es decir, cómo y cuánto se mueven los elementos mientras se carga.
LCP (Largest Contentful Paint)	Mide el tiempo que tarda en cargarse el elemento más grande visible en la ventana del usuario.

Análisis de requisitos

Diagrama de casos de uso:

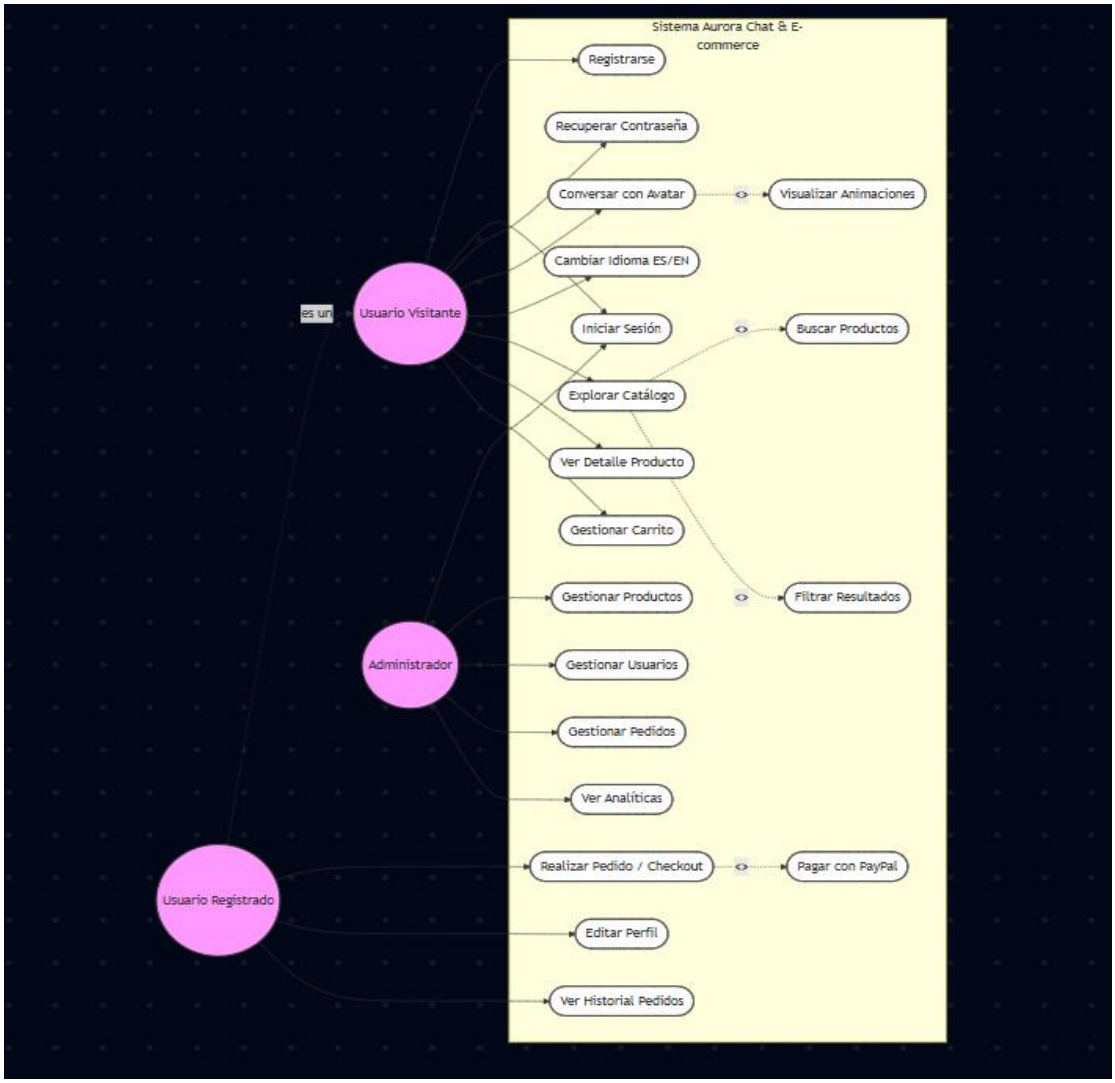
El proceso de análisis de requisitos constituye una fase crítica en el ciclo de *vida de desarrollo del software (SDLC)* para el proyecto *Aurora Chat & E-commerce*. En esta etapa, el objetivo primordial ha sido transformar las necesidades abstractas y los problemas detectados en especificaciones *técnicas concretas, inequívocas y verificables*. Este capítulo documenta exhaustivamente los modelos conceptuales construidos, sirviendo como contrato entre las partes interesadas y el equipo de desarrollo, y garantizando que el producto final no solo cumpla con las expectativas funcionales, sino que también satisfaga los estándares de calidad esperados.

A continuación, se desglosan los requisitos del sistema, comenzando por el modelado visual de las interacciones (*Casos de uso*), seguido de la especificación detallada de funciones (*Requisitos funcionales*) y culminando con las restricciones y atributos de calidad (Requisitos no funcionales).

Diagrama de Casos de Uso:

El diagrama de casos de uso es una representación gráfica de alto nivel que ilustra el comportamiento del sistema desde el punto de vista del usuario final. Este diagrama define el alcance del proyecto (*System Boundary*) y muestra cómo los diferentes actores externos interactúan con las funcionalidades principales del sistema para alcanzar objetivos específicos.

Para el sistema *Aurora*, se ha diseñado el siguiente diagrama UML que encapsula la complejidad de los tres subsistemas principales: la Inteligencia Artificial (*Avatar*), la Plataforma de Comercio Electrónico y el Panel de Administración.



Explicación del Diagrama y Relaciones:

El diagrama anterior no solo lista las funciones, sino que establece la jerarquía y dependencia entre ellas. A continuación se interpreta la semántica de las relaciones representadas:

1. Herencia de Actores (Generalización):

- ✧ Se observa una relación donde el *Usuario Registrado (Cliente)* es un *Usuario Visitante*. Esto implica que el cliente autenticado conserva todas las capacidades del visitante *(navegar, chatear, usar el carrito)* y añade nuevas capacidades exclusivas *(checkout, perfil)*. Esto simplifica el modelo al evitar duplicar líneas de relación.

2. Relaciones de Inclusión (<<include>>):

- ✧ *Conversar con Avatar* `<<include>>` *Visualizar Animaciones*: Esta relación denota obligatoriedad. No es posible mantener una conversación con el sistema sin que el módulo de visualización renderice las respuestas emocionales del avatar. El caso de uso base (Conversar) invoca necesariamente al incluido.

- ✧ *Realizar Pedido* `<<include>>` *Pagar con PayPal*: Para completar exitosamente un pedido, es condición sine qua non ejecutar el proceso de pago. El sistema no permite finalizar una orden sin la transacción económica.

3. Relaciones de Extensión (<<extend>>):

- ✧ *Buscar Productos* y *Filtrar Resultados* `<<extend>>` *Explorar Catálogo*: Estas son funcionalidades opcionales que enriquecen la experiencia base. Un usuario puede navegar por el catálogo página a página (caso base) o, opcionalmente, activar los mecanismos de búsqueda o filtrado para refinar su vista.

Requisitos Funcionales Principales:

Los *Requisitos Funcionales (RF)* describen los comportamientos específicos del software: qué debe hacer el sistema en respuesta a las entradas y en diferentes estados. La extracción de estos requisitos proviene directamente del análisis de los casos de uso presentados anteriormente.

Para una *mayor claridad y rigor técnico*, se han agrupado por los subsistemas lógicos que componen la arquitectura de Aurora.

2.1 Subsistema de Inteligencia Artificial y Avatar (Módulo AURORA)

Este subsistema representa el valor diferencial del proyecto, combinando capacidades cognitivas con presentación visual avanzada.

RF-01: Renderizado de Avatar Live2D

- *Descripción*: El sistema debe cargar y renderizar en el navegador del cliente un modelo Live2D (formato *`.model3.json`* y asociados). El modelo debe mantener un estado *"Idle"* (reposo) con movimiento de respiración automático cuando no hay interacción.

- *Entrada*: Archivos de recursos del modelo (texturas, físicas, mocap).

- *Salida*: Lienzo WebGL interactivo con el personaje "Haru" visualizado.

RF-02: Procesamiento de Lenguaje Natural (Conversación):

- *Descripción*: El sistema debe proporcionar una interfaz de chat donde el usuario pueda introducir texto libre. Este texto debe ser enviado al backend para su procesamiento por *el modelo de lenguaje (LLM)*, retornando una respuesta coherente en el contexto de un asistente de tienda virtual.

- *Restricción*: La longitud máxima del mensaje de entrada será de 500 caracteres para asegurar tiempos de respuesta óptimos.

RF-03: Análisis Emocional y Expresividad (Módulo ANA)

- *Descripción*: Cada respuesta generada por el sistema debe venir acompañada de una etiqueta emocional (metadata). El frontend debe interpretar esta etiqueta (*ej. `happy`, `sad`, `surprised`*) y ejecutar la animación facial y corporal correspondiente en el avatar.

- *Mapeo*:

  - Alegría -> Animación *`motion_happy`* + Expresión *`f01`*.

  - Tristeza -> Animación *`motion_sad`* + Expresión *`f02`*.

  - Ira -> Animación *`motion_angry`* + Expresión *`f03`*.

RF-04: Sincronización Labial (Lip-Sync) y TTS

- *Descripción:* El sistema debe convertir la respuesta textual en audio utilizando la API de síntesis de voz del navegador (Web Speech API). Simultáneamente, debe modular la apertura de la boca del avatar en tiempo real basándose en el volumen o los fonemas del audio generado, creando una ilusión de habla realista.

2.2 Subsistema de Comercio Electrónico (E-commerce)

Este bloque abarca la lógica de negocio tradicional de una tienda en línea.

RF-05: Navegación y Filtrado de Catálogo:

- *Descripción:* La plataforma debe listar los productos disponibles recuperados de la base de datos.

✧ *Funcionalidades Detalladas:*

- ➢ Paginación asíncrona de resultados (carga progresiva o por páginas).
- ➢ Barra de búsqueda predictiva que filtre resultados por coincidencia en el nombre o descripción.

RF-06: Gestión del Carrito de Compras (Persistencia)

✧ *Descripción:* El sistema debe permitir a los usuarios (tanto visitantes como registrados) añadir productos a una "cesta" virtual.

✧ *Comportamiento Crítico:* El estado del carrito debe persistir localmente (SessionStorage) si el usuario cierra la pestaña o recarga el navegador, garantizando que no pierda su selección.

✧ *Cálculo:* El subtotal y total deben recalcularse automáticamente al modificar cantidades.

RF-07: Pasarela de Pago (Checkout)

✧ *Descripción:* El sistema debe guiar al usuario a través de un flujo de compra seguro. Debe recolectar datos de envío y facturación mediante formularios validados.

✧ *Integración:* La transacción final se debe delegar a la *API de Stripe.* El sistema debe manejar los callbacks de éxito (`onApprove`) y error (`onError`), registrando el pedido en la base de datos interna solo tras la confirmación del pago.

2.3 Subsistema de Administración (Backoffice)

Herramientas exclusivas para el mantenimiento del negocio, invisibles para el usuario final.

RF-08: Gestión de Inventario (CRUD de Productos)

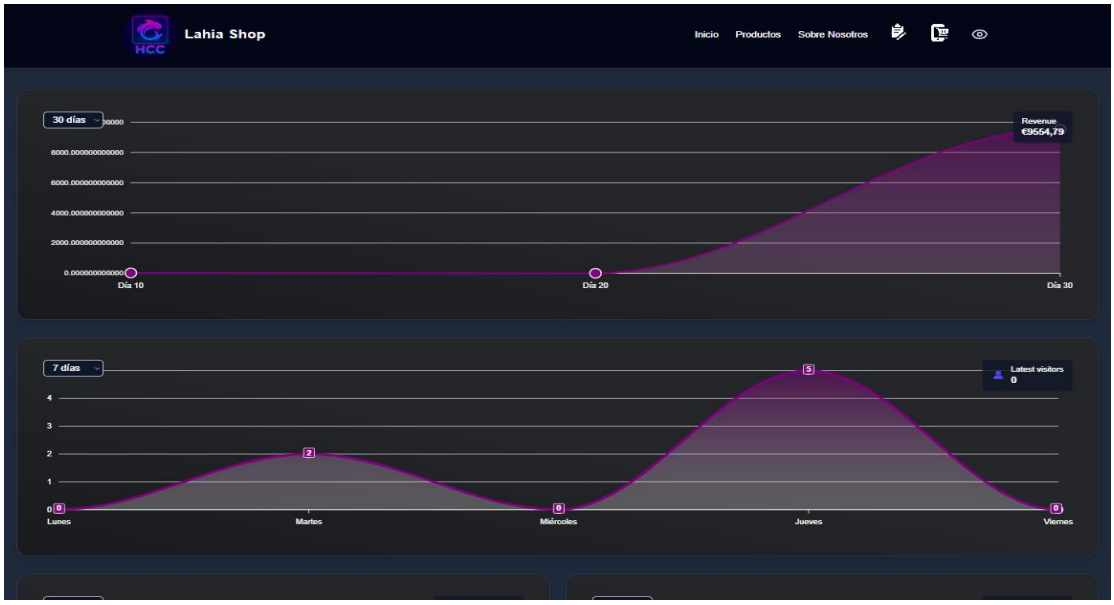
- *Descripción:* El administrador debe disponer de interfaces para Crear, Leer, Actualizar y Borrar productos del catálogo.

- *Validaciones:* El sistema debe impedir la creación de productos con precio negativo o sin nombre. Debe permitir la carga de imágenes, gestionando su almacenamiento y vinculación.

RF-09: Panel de Control Analítico (Dashboard)

- *Descripción:* El sistema debe agregar los datos transaccionales para presentar visualizaciones estratégicas.

- *Métricas Requeridas:* Gráfico de tendencia de ventas (últimos 30 días), distribución de pedidos por estado, y listado de productos con bajo stock.



Requisitos No Funcionales

Los *Requisitos No Funcionales (RNF)* definen los *"atributos de calidad"* del sistema. A diferencia de los funcionales que dicen *qué* hace el sistema, estos dictan *cómo* debe comportarse. Son cruciales para la satisfacción del usuario y la viabilidad técnica. Se clasifican a continuación según atributos estándar de calidad de software (*ISO/IEC 25010*).

Rendimiento y Eficiencia:

El rendimiento es vital en un e-commerce, donde cada segundo de retraso impacta directamente en la tasa de conversión.

RNF-01 (Tiempo de Respuesta Interactivo):

- El sistema debe ofrecer tiempos de respuesta inferiores a *200 milisegundos* para interacciones de interfaz de usuario (clics en botones, apertura de menús) para garantizar una sensación de fluidez (percepción de inmediatez).

RNF-02 (Core Web Vitals - LCP):

- El indicador *Largest Contentful Paint* (tiempo de carga del elemento más grande) debe mantenerse por debajo de los *2.5 segundos* en condiciones de red 4G, asegurando una buena indexación SEO y experiencia de usuario.

RNF-03 (Tasa de Fotogramas del Avatar):

- La renderización del modelo Live2D debe mantener una tasa estable de *60 FPS* (*fotogramas por segundo*) en dispositivos de gama media y superior, y degradarse elegantemente (*mínimo 30 FPS*) en dispositivos móviles de gama baja.

Seguridad (Confidencialidad, Integridad y Disponibilidad)

Dada la naturaleza transaccional del sistema, la seguridad es un pilar innegociable.

RNF-04 (Autenticación y Sesiones):

- ✧ El acceso a áreas restringidas (Perfil, Dashboard) debe protegerse mediante fichas *JWT (JSON Web Tokens)* firmadas criptográficamente. El sistema no debe permitir el acceso a rutas *`/admin/`* sin un token válido con el rol *`admin`*.

RNF-05 (Integridad de Datos):

- ✧ Todas las comunicaciones cliente-servidor deben realizarse exclusivamente sobre *HTTPS (TLS 1.2 o superior)* para prevenir ataques de *"Man-in-the-Middle"*.

RNF-06 (Delegación de Datos Sensibles):

- ✧ El sistema *NO* debe almacenar, procesar ni transmitir datos de tarjetas de crédito o débito en sus propios servidores. Toda información de pago debe ser introducida directamente en los marcos seguros (*iFrame/Pop-up*) proporcionados por la pasarela de Stripe (*Cumplimiento PCI-DSS simplificado*).

Usabilidad y Accesibilidad

El sistema debe ser inclusivo y fácil de usar por la mayor audiencia posible.

RNF-07 (Diseño Responsivo Universal):

- ✧ La interfaz gráfica debe adaptarse dinámicamente (*`Fluid Layout`*) a cualquier resolución de pantalla, desde dispositivos móviles (*320px ancho*) hasta monitores de escritorio 4K. El diseño no debe presentar barras de desplazamiento horizontal no intencionadas.

RNF-08 (Feedback del Sistema):

- El sistema debe proporcionar retroalimentación visual clara ante cualquier acción (*éxito, error, carga*). Se debe utilizar un sistema de notificaciones tipo *"Toast"* (Módulo ALBA) para informar de errores de red o validación sin bloquear la navegación del usuario.

RNF-09 (Internacionalización - i18n):

- El sistema debe estar preparado arquitectónicamente para soportar múltiples idiomas. Inicialmente, debe soportar *Español* e *Inglés*, permitiendo el cambio en tiempo de ejecución sin recargar la aplicación completa.

Descripción de los Usuarios y sus Necesidades

Basándonos en el análisis de los casos de uso, se han identificado tres roles fundamentales que interactúan con el sistema. A continuación se desarrollan en profundidad mediante la técnica de *"Personas"* para contextualizar sus necesidades.

1. El Visitante Casual (Rol: Guest)

Este es el punto de entrada más común al sistema. Representa a usuarios no identificados que llegan a la web, posiblemente desde buscadores o redes sociales.

- *Perfil Demográfico/Técnico*: Variado. Desde usuarios con baja competencia tecnológica hasta expertos. Utilizan principalmente dispositivos móviles.

- *Motivaciones Principales*:

  - Curiosidad por la tecnología de "IA Emocional" promocionada.

  - Interés en consultar precios o especificaciones de un producto específico.

✧ *Necesidades y Expectativas*:

- *Inmediatez:* Necesita que la página cargue rápido. Si tarda más de 3 segundos, abandonará el sitio.
- *Baja Fricción:* Requiere poder interactuar con el Avatar y ver productos *sin* necesidad de registrarse. El registro forzoso es una barrera de entrada que debe evitarse.
- *Claridad:* Necesita entender intuitivamente qué es Aurora (¿es un juego? ¿es una tienda?) en los primeros instantes de navegación.

2. El Cliente Recurrente (Rol: User)

Es un visitante que ha decidido establecer una relación de confianza con la plataforma creando una cuenta.

✧ *Perfil Demográfico/Técnico:* Usuario habituado a compras online. Valora la eficiencia y la seguridad.

✧ *Motivaciones Principales:*

- Completar compras de productos seleccionados.
- Hacer seguimiento de sus envíos.
- Recibir atención personalizada del Avatar (que recuerde su nombre o preferencias en futuras versiones).

✧ *Necesidades y Expectativas:*

- *Seguridad Percibida:* Necesita garantías visuales de que su pago es seguro.
- *Comodidad:* Espera que el sistema recuerde sus datos de envío para no tener que rellenar formularios extensos en cada compra.

- *Autonomía:* Desea poder consultar el estado de sus pedidos (Enviado/Entregado) sin tener que contactar soporte humano.

3. El Administrador del Negocio (Rol: Admin)

Es el usuario interno, responsable de la operativa diaria de la tienda y la gestión del contenido.

✧ *Perfil Demográfico/Técnico:* Personal de la empresa propietaria de Aurora. Conocimientos intermedios de gestión web.

✧ *Motivaciones Principales:*

- Mantener el catálogo actualizado, atractivo y sin errores.
- Maximizar el beneficio operativo gestionando eficientemente los pedidos.
- Entender el comportamiento de los clientes mediante datos.

✧ *Necesidades y Expectativas:*

- *Eficiencia CRUD:* Necesita herramientas rápidas para subir múltiples productos o cambiar precios masivamente sin lidiar con código o bases de datos directas.
- *Control Total:* Requiere capacidad para intervenir en cualquier pedido (cancelarlo, reembolsarlo) o moderar usuarios si fuera necesario.
- *Visibilidad:* Necesita un "Dashboard" que le diga de un vistazo cómo va el negocio hoy (ventas, alertas de stock bajo), para tomar decisiones rápidas.

Diseño de la Aplicación

El diseño de la aplicación *Aurora Chat & E-commerce* representa la culminación del análisis de requisitos, transformando las necesidades funcionales en una solución técnica tangible y estructurada. Este capítulo detalla la arquitectura de software, las decisiones de diseño de interfaz y el modelado de objetos que sustentan el sistema. Se ha puesto especial énfasis en la modularidad, la experiencia de usuario *(UX) inmersiva* y la aplicación de patrones de diseño de software robustos.

Prototipos de Interfaz de Usuario (Mockups) y Experiencia de Usuario (UX)

El diseño visual de Aurora trasciende la estética convencional de un e-commerce para adentrarse en el terreno de las "Interfaces Conversacionales" y el "Diseño Emocional". La interfaz no es solo un medio para realizar transacciones, sino un entorno construido para facilitar la interacción natural con una Inteligencia Artificial personificada (Vease Anexo1).

1. Filosofía de Diseño: Glassmorphism y Profundidad

Se ha adoptado una estética basada en el *Glassmorphism* (Vidrismo), caracterizada por el uso de capas semitransparentes con desenfoque de fondo (`backdrop-filter: blur`).

- *Justificación Visual:* Esta técnica permite establecer una clara jerarquía visual en el eje Z (profundidad). El Avatar 3D reside en el "plano base", mientras que los paneles de chat y las tarjetas de productos flotan en un "plano superior". Esto asegura que la interfaz de usuario (UI) sea legible sin ocultar completamente al personaje, manteniendo la sensación de presencia del asistente virtual.

- *Paleta Cromática:* Se ha diseñado un sistema de colores semántico y atmosférico.

  - *Primario (Violeta Eléctrico #7c3aed):* Utilizado para llamadas a la acción (CTAs) y elementos activos. Evoca modernidad, misterio y tecnología cognitiva.

  - *Secundario (Cyan Neón):* Usado para retroalimentación de "pensamiento" de la IA y estados de carga, complementando al violeta en una tríada análoga.

- *Neutros:* Blancos y grises muy claros con alta luminosidad, evitando el negro puro para los textos a fin de reducir la fatiga visual en pantallas de alta densidad (Retina/OLED).

2. Pantalla Principal: La Experiencia "Hero"

La *Landing Page* rompe con la estructura tradicional de *Header-Slider-Grid*.

- *Composición Asimétrica*: El diseño divide la pantalla aurea. El Avatar (Haru) se sitúa estratégicamente para respetar la "Regla de los Tercios", dirigiendo su mirada hacia el área de chat o contenido, lo que psicológicamente guía la atención del usuario hacia esos elementos (fenómeno de *Gaze Cueing*).

- *Chat como Ciudadano de Primera Clase*: El widget de chat no está relegado a una esquina oculta. Es un componente central, flotante y persistente. Su diseño incluye micro-interacciones sutiles: las burbujas de mensaje aparecen con una animación de entrada (`fade-in-up`) y escalado elástico, otorgando una sensación de "peso" y realidad física a la conversación digital.

3. Catálogo de Productos Inmersivo

El desafío del catálogo era integrar la densidad de información de una tienda (precios, fotos, nombres) sin romper la inmersión del chat.

- *Sistema de Tarjetas (Card UI):* Cada producto es una entidad visual autocontenida.

  - *Contención:* Las tarjetas utilizan sombras suaves difusas (`box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1)`) que aumentan al hacer *hover*, elevando perceptualmente el elemento para indicar interactividad (Affordance).

  - *Tipografía:* Se emplea una tipografía Sans-Serif geométrica (Inter o Roboto) con pesos variables. El precio se destaca visualmente mediante tamaño y color, jerarquizando la información para facilitar el escaneo rápido (Skimming) característico de los usuarios digitales.

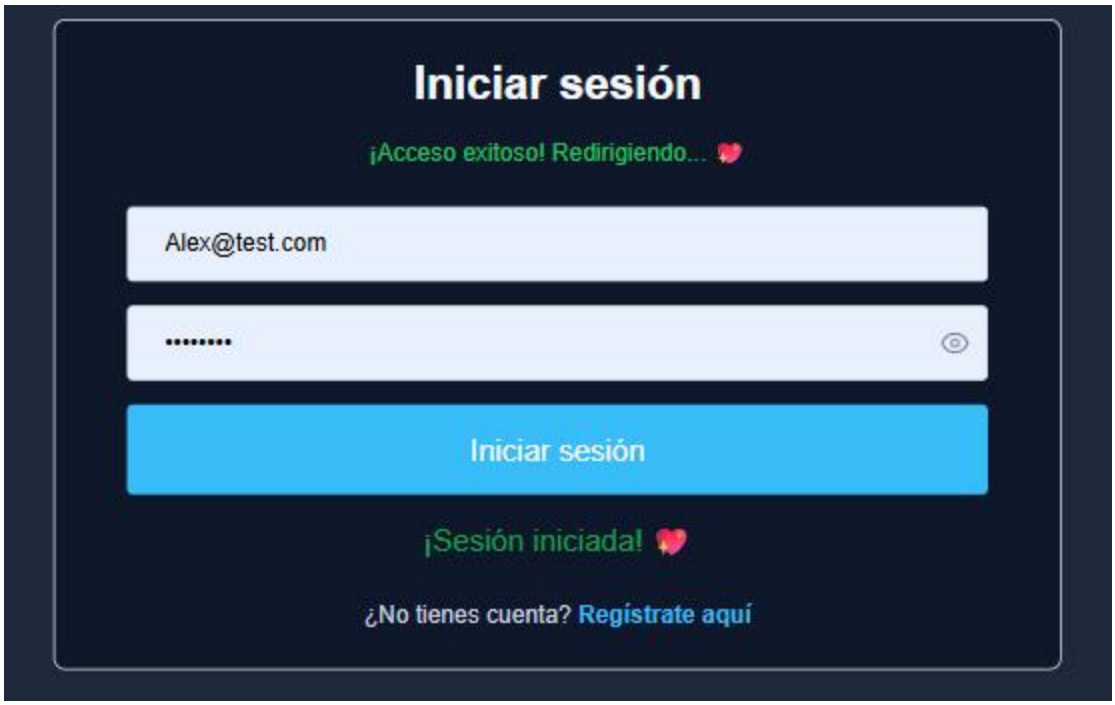
- *Navegación Facetada Adaptativa:* Los filtros de búsqueda no bloquean la vista. En escritorio se presentan lateralmente, mientras que en dispositivos móviles se despliegan en un panel inferior (*Bottom Sheet*) deslizable, optimizando el área táctil para el pulgar (*Thumb Zone)*.

4. Dashboard: Ergonomía y Eficiencia

Para el panel de administración, el paradigma de diseño cambia de "inmersión" a "eficiencia cognitiva".

- *Diseño de Alta Densidad:* Se reducen los espacios en blanco innecesarios (*padding*) para mostrar más filas de datos en tablas. Se utiliza tipografía monoespaciada para identificadores y cifras, facilitando la comparación vertical de datos numéricos.

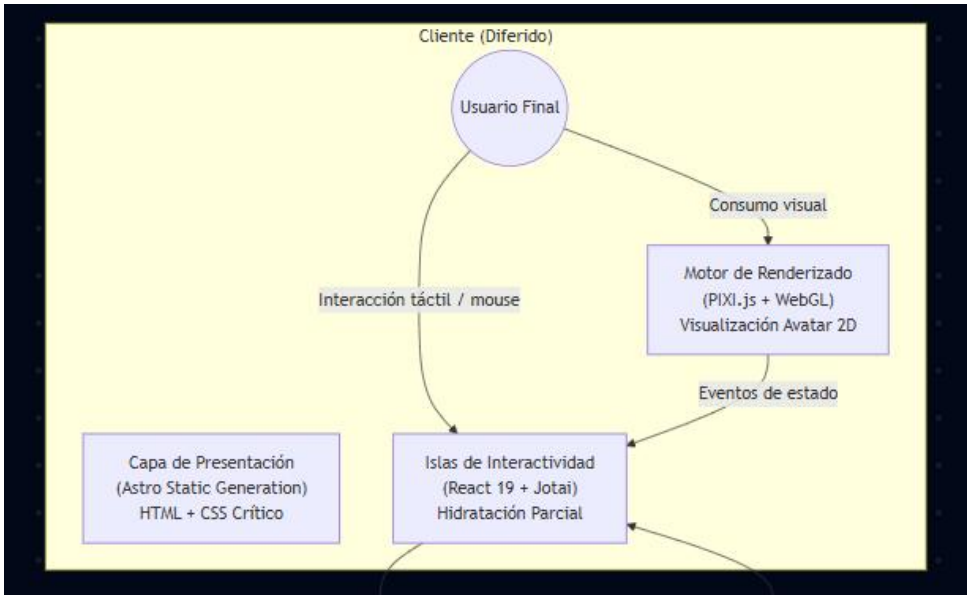
- *Feedback de Estado:* El sistema de estados de pedidos utiliza "Píldoras" (Badges) con codificación de color semántica universal (Verde=Completado, Amarillo=Pendiente, Rojo=Cancelado), permitiendo al administrador evaluar la salud del negocio de un solo vistazo.



Arquitectura del Sistema

La arquitectura de software de Aurora ha sido concebida bajo el paradigma de *Sistemas Distribuidos* y *Componentización*, priorizando la escalabilidad, la mantenibilidad y el rendimiento. Se ha optado por un enfoque desacoplado donde el Frontend y el Backend evolucionan de forma independiente, comunicándose a través de contratos de API estrictos.

Diagrama de Arquitectura (Nivel de Contenedores C4)



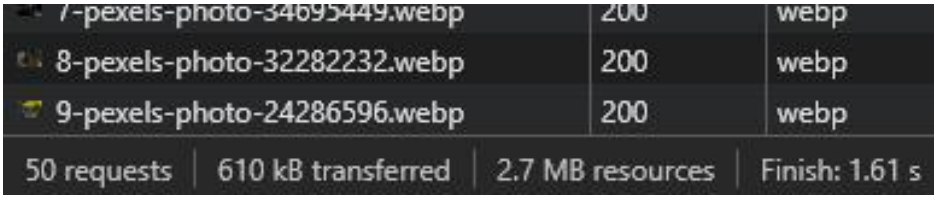
Arquitectura de Islas (Islands Architecture)

Una de las decisiones técnicas más relevantes ha sido la implementación de la *Arquitectura de Islas* mediante el framework *Astro*.

- *El Problema de las SPAs:* Las Single Page Applications tradicionales (React, Vue) envían grandes cantidades de JavaScript al navegador, bloqueando el hilo principal durante la hidratación y perjudicando métricas web vitales como el *First Contentful Paint (FCP)* y el *Time to Interactive (TTI)*.

- *La Solución de Islas:* Aurora divide la interfaz en componentes estáticos (Header, Footer, Layouts, Textos Legales) y "Islas" dinámicas (Widget de Chat, Carrito, Galería de Productos).

    - *Resultado:* El servidor renderiza HTML puro para las zonas estáticas (cero JavaScript). Solo se carga y ejecuta el JavaScript necesario para las islas interactivas. Esto reduce el peso del *bundle* inicial hasta en un 70%, acelerando drásticamente la carga en dispositivos móviles y redes lentas.

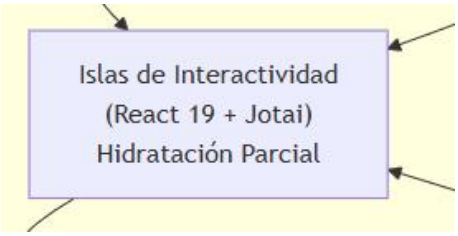


Gestión de Estado Atómico (Atomic State Management)

Para manejar la complejidad del estado en el cliente (qué productos hay en el carrito, qué emoción tiene el avatar, historial de chat), se ha descartado el patrón Flux convencional (Redux) en favor de un modelo atómico con *Jotai*.

- *Granularidad:* El estado se divide en átomos pequeños e independientes (`cartAtom`, `userAtom`, `avatarEmotionAtom`).

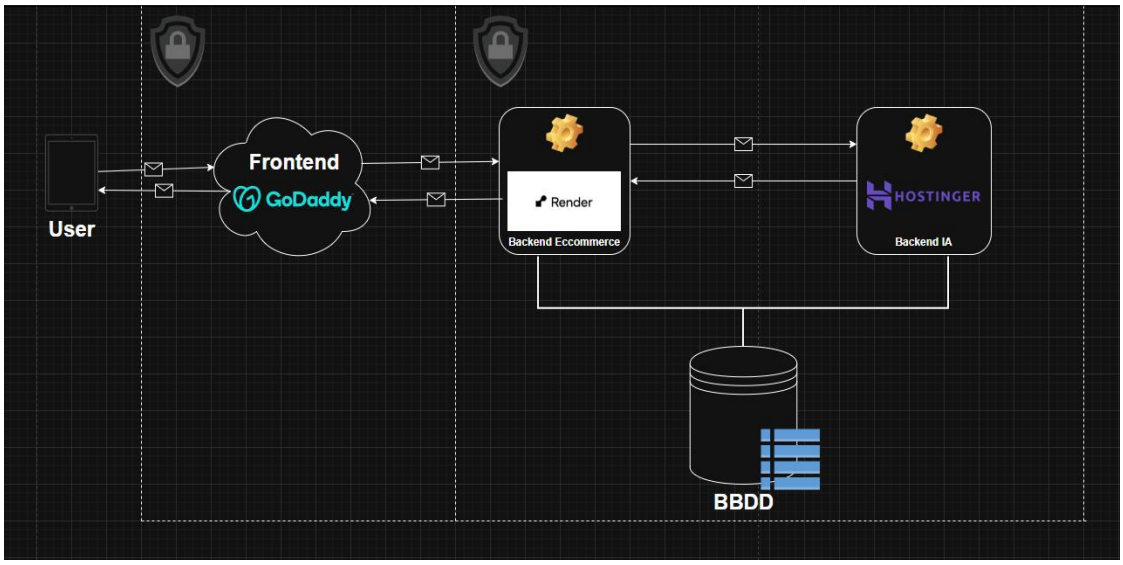
- *Beneficio de Rendimiento:* Cuando un átomo cambia, solo se renderizan los componentes que están suscritos explícitamente a ese átomo. Esto evita los costosos re-renders de árbol completo que ocurren en arquitecturas basadas en Context API mal optimizadas, asegurando que la interfaz se mantenga fluida a 60 FPS incluso con operaciones complejas.



Arquitectura de Despliegue e Infraestructura (Cloud Architecture)

El sistema Aurora utiliza una Estrategia de *Hosting Distribuido* para optimizar costes y latencias, delegando cada responsabilidad a proveedores especializados según sus capacidades de cómputo y red.

Diagrama de Infraestructura en la Nube:



Descripción de la infraestructura:

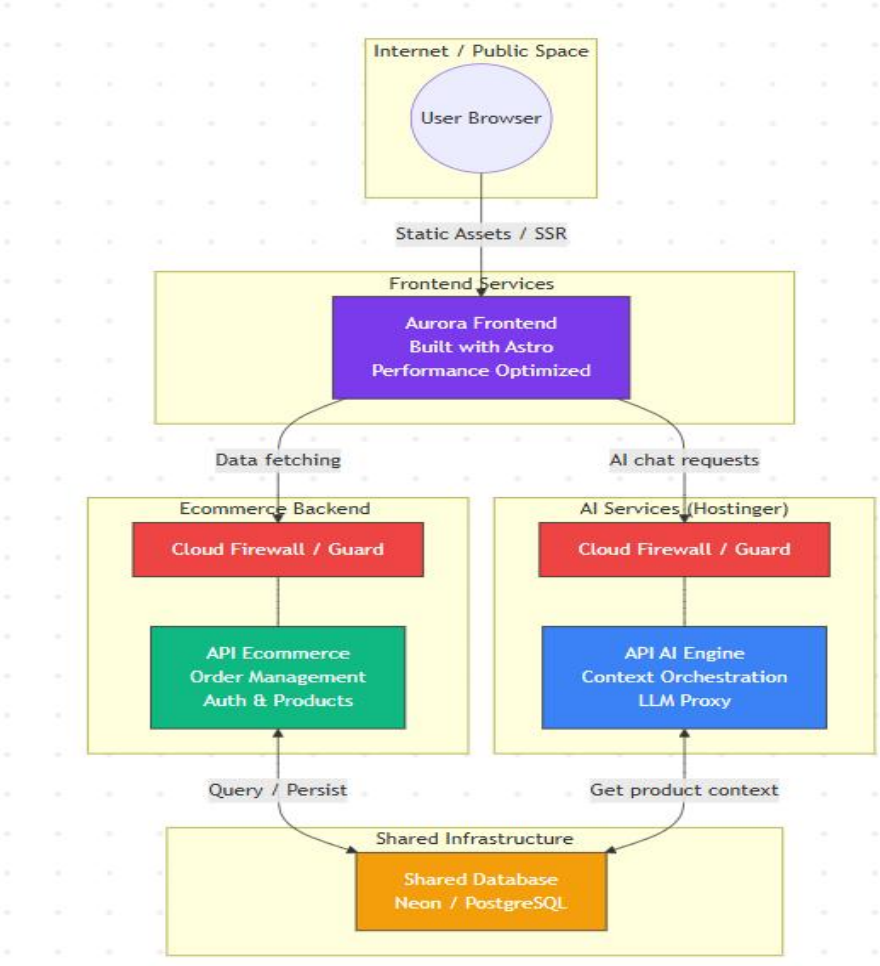
1. *Capa de Presentación (GoDaddy):* El frontend, construido con Astro, se despliega en una infraestructura optimizada para la entrega de contenido estático y renderizado en servidor (SSR). *GoDaddy actúa como la puerta de entrada principal, gestionando el dominio y la entrega de recursos mediante CDN*.

2. *Motor de E-commerce (Render):* El backend encargado de la lógica de negocio tradicional (pedidos, usuarios, productos) reside en Render. Este servicio ofrece escalado automático y una integración fluida con despliegues de Node.js, protegidos por una capa de firewall que solo permite peticiones desde el origen del frontend.

3. *Cerebro de IA (Hostinger):* Dado que el procesamiento de IA y la orquestación de prompts requieren tiempos de ejecución específicos y configuraciones de red adaptadas, este módulo se aisla en Hostinger. Actúa como un proxy de seguridad que filtra y enriquece las peticiones antes de enviarlas a los LLMs externos.

4. *Persistencia Unificada (Shared Database):* Ambos backends comparten una base de datos *PostgreSQL* alojada de forma externa. Esto asegura que la IA tenga acceso en tiempo real a la disponibilidad de stock y precios, permitiendo que Aurora responda con precisión sobre el catálogo.

5. *Capa de Seguridad (Cloud Guards):* Todas las entradas a los backends están custodiadas por *"vallas de protección"* (*Cloud Guards/Firewalls*) que monitorizan el tráfico para prevenir ataques DoS, inyecciones y garantizar que solo las peticiones autenticadas lleguen a la lógica central.



Diagramas de Clases y Diseño Orientado a Objetos

En el nivel de código, el sistema sigue los principios *SOLID* para asegurar un código robusto y extensible. Aunque JavaScript es un lenguaje basado en prototipos, el uso de *TypeScript* nos permite aplicar patrones de *Diseño Orientado a Objetos (OOP)* estrictos para modelar el dominio del problema.

Modelo de Dominio del Frontend

El siguiente diagrama UML ilustra las clases principales que encapsulan la lógica de negocio en el navegador, separando claramente las responsabilidades.

Detalle de Clases y Patrones

1. Clase `AvatarController` (Patrón Facade):

- Esta clase es fundamental para la abstracción. La librería Live2D y la Web Speech API son complejas y de bajo nivel. *`AvatarController`* proporciona una *"Fachada"* simplificada con métodos de alto nivel como *`speak()`* o *`setExpression()`*. El resto de la aplicación (el Chat, por ejemplo) no necesita saber de vértices, mallas o buffers de audio; solo invoca estos métodos semánticos.

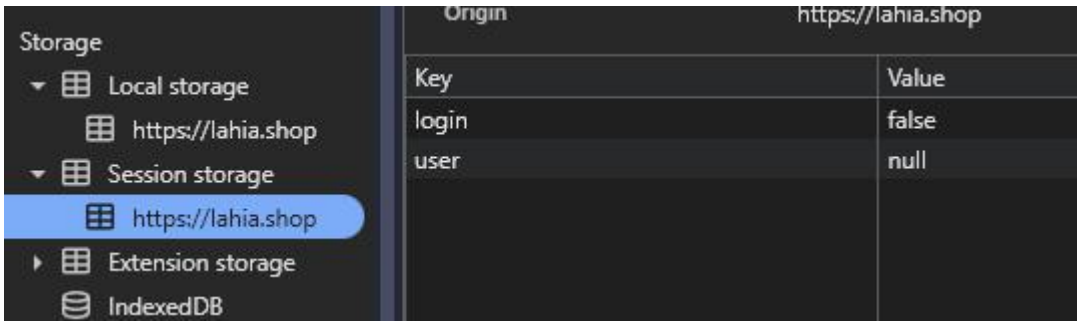
- *Principio de Responsabilidad Única (SRP):* Esta clase solo se preocupa de la representación del avatar, no de la lógica del chat ni de la red.

2. Clase `Cart` (Lógica de Negocio):

- Encapsula las reglas de negocio críticas del comercio electrónico.

- *Invariantes:* Asegura que no se puedan añadir productos con stock 0, que las cantidades sean siempre enteros positivos, y que el cálculo del total incluya correctamente impuestos y descuentos antes de proceder al pago.

- *Persistencia Transparente:* Internamente gestiona la sincronización con `LocalStorage`, de modo que el usuario puede cerrar el navegador y recuperar su trabajo más tarde sin que la interfaz de usuario tenga que gestionar explícitamente el almacenamiento.



3. Clase `ChatManager` (Patrón Observer):

- Actúa como intermediario entre la interfaz de usuario y los servicios de backend. Implementa un patrón de observador donde los componentes de la interfaz se "suscriben" a la llegada de nuevos mensajes. Esto permite que múltiples partes de la interfaz (*el widget de chat, el log de sistema, las notificaciones*) reaccionen a un nuevo mensaje entrante de forma desacoplada.

Este diseño de clases promueve un código modular donde añadir una nueva funcionalidad (por ejemplo, un nuevo tipo de animación para el avatar o un nuevo método de pago) se puede realizar extendiendo las clases existentes sin necesidad de reescribir el núcleo del sistema, cumpliendo con el *Principio Abierto/Cerrado (Open/Closed Principle)*.

Estructura del Proyecto (Estructura de Carpetas)

La organización de archivos del proyecto sigue una estructura fractal basada en *Módulos (Domain-Driven Design - DDD)* para el núcleo del negocio y *File-System Routing* para las páginas web, típica de Astro.

```
src/
├── assets/             # Recursos estáticos optimizados (imágenes, SVGs)
├── components/         # Componentes UI reutilizables y atómicos
│   ├── astro/          # Componentes estáticos (Header, Footer)
│   └── tsx/            # Islas interactivas React (Chat, CartWidget)
│
├── layouts/            # Plantillas maestras de página (Layout.astro, DashboardLayout.astro)
├── pages/              # Enrutador basado en archivos
│   ├── index.astro     # Página de inicio
│   ├── shop/           # Rutas del e-commerce ([...slug].astro)
│   ├── dashboard/      # Rutas del panel de administración
│   └── api/            # Endpoints de servidor (SSR)
│
├── modules/            # Núcleo de la lógica de negocio (DDD)
│   ├── AURORA/         # Lógica del Avatar y Chat
│   │   ├── components/ # (VtuberLive2D.tsx)
│   │   └── core/       # (AuroraVoice.ts, EmotionMapper.ts)
│   ├── ANA/            # Análisis de emociones
│   ├── ALBA/           # Sistema de gestión de errores
│   └── YOLI/           # Internacionalización (i18n)
│
├── store/              # Estado global (Atomos Jotai)
├── services/           # Capa de comunicación HTTP (Fetch wrappers)
├── styles/             # Configuración CSS y Tailwind
└── types/              # Definiciones TypeScript globales
```

Justificación de la Organización

- ### Separación `components/` vs `modules/`:

  - En *`components/`* residen los elementos visuales genéricos y estúpidos (*Dumb Components*) como botones, tarjetas o inputs, que pueden ser usados en cualquier parte de la app.

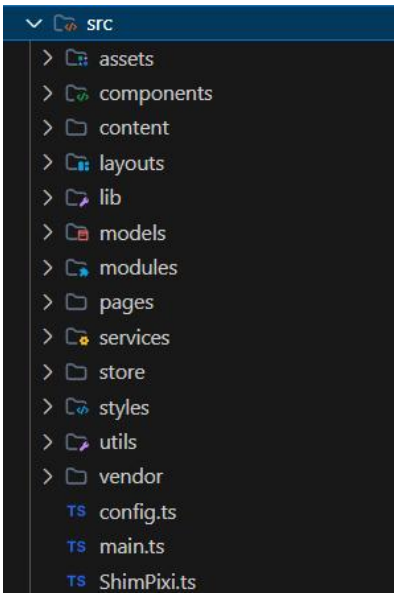
  - En *`modules/`* reside la lógica específica del dominio. Por ejemplo, el módulo `AURORA` encapsula todo lo referente a la IA y el Avatar. Si en el futuro se quiere separar el chat a otro proyecto, basta con copiar la carpeta *`modules/AURORA`*. Esta alta cohesión facilita el mantenimiento a largo plazo.

- ### Carpeta `pages/` (Enrutamiento):

  - Astro utiliza el sistema de archivos como router. Cualquier archivo *`.astro`* o *`.md`* aquí se convierte automáticamente en una ruta URL. Las carpetas como *`shop/[id].astro`* indican rutas dinámicas que se generan bajo demanda o en tiempo de compilación.

- ### Carpeta `store/` y `services/`:

  - Separan la gestión de datos de la interfaz. *`services/`* contiene las funciones puras que hacen llamadas a la API (e.g., *`getProductById`*), mientras que *`store/`* mantiene el resultado de esas llamadas en memoria para que la interfaz reaccione a los cambios.



Modelo de Datos (Frontend)

El diseño de la capa de datos en el frontend de Aurora ha sido abordado bajo una premisa de rigor tipográfico y mantenibilidad a largo plazo. Aunque, por naturaleza, una aplicación cliente no gestiona una persistencia de datos pesada (*tarea delegada al Backend*), la complejidad de una plataforma *que integra e-commerce, administración y un avatar interactivo* exige una definición de estructuras extremadamente precisa.

El uso de TypeScript no es opcional en este proyecto; actúa como el "contrato" de comunicación entre el servidor y la interfaz. Esta capa de modelos permite:

- *Eliminación de Errores Silenciosos:* Al tipar estrictamente las respuestas del API, el compilador detecta discrepancias en los nombres de campos o tipos antes de que el código llegue a ejecución.

- *Documentación Viva:* Cualquier desarrollador que explore la carpeta /src/models puede comprender inmediatamente qué atributos tiene un producto o una orden sin necesidad de consultar documentación externa.

- *Refactorización Segura:* Cambiar el nombre de una propiedad en un modelo propaga automáticamente errores en todos los componentes que la consumen, garantizando que el sistema sea coherente tras cualquier actualización.

A continuación, se presenta un desglose exhaustivo de las interfaces y estructuras de datos que conforman el esqueleto informacional de la aplicación, categorizadas por su dominio de responsabilidad.

Modelos de E-commerce (EcommerceProps)

Esta categoría representa el corazón comercial de la plataforma. Se encarga de definir cómo se visualizan, procesan y almacenan temporalmente los elementos relacionados con la venta de productos y la gestión de usuarios finales. La distinción entre un modelo de *"detalle"* (Product) y uno de *"tarjeta"* (ProductCard) es una decisión consciente para optimizar el rendimiento, cargando solo los datos necesarios en las vistas de lista frente a las de detalle profundo.

- Entidades de Catálogo: *Product y ProductCard* Estas interfaces aseguran que cualquier componente de la tienda (*ya sea un buscador, un carrusel o una ficha técnica*) maneje datos consistentes. El uso de campos opcionales (?) en el modelo Product permite la flexibilidad necesaria para datos que pueden ser cargados de forma perezosa (*lazy loading*).

```
export interface Product {
  id: number;              // Identificador único numérico (Backend sync)
  nombre: string;          // Título comercial del producto
  descripcion?: string;     // Bloque de texto descriptivo
  precio?: number;          // Valor monetario
  img_url: string;         // Ruta de la imagen principal
  category_id?: number;     // Relación con el ID de categoría
  product_category: number; // Identificador redundante para compatibilidad
}

export interface ProductCard {
  id: string;              // ID en formato string para el DOM
  nombre: string;          // Nombre resumido
  img_url: string;         // Miniatura optimizada
  descripcion: string;     // Extracto breve
  precio: number;          // Precio para el listado
  category_id: number;     // Categoría para filtrado rápido
}
```

- Entidades de Navegación: *Category y CategoryCard* Fundamentales para la arquitectura de información del sitio. Permiten orquestar tanto la segmentación del catálogo como la representación visual de las secciones en la Landing Page.

```
export interface Category {
  lang: string;             // Código de idioma para internacionalización
  pageSize?: number;        // Tamaño de la muestra por página
}

export interface CategoryCard {
  id?: number | string;     // ID flexible para búsquedas
  lang: string;             // Localización del título
  title: string;            // Nombre de la categoría a mostrar
  img: string;              // Imagen representativa del sector
}
```

- Entidades de Usuario y Seguridad: Profile y Payloads La seguridad es crítica, por lo que estos modelos definen exactamente qué datos son necesarios para el registro, el login y la gestión del perfil una vez autenticado, evitando el envío de información sensible o innecesaria.


```
export interface Category {
  lang: string;             // Código de idioma para internacionalización
  pageSize?: number;        // Tamaño de la muestra por página
}

export interface CategoryCard {
  id?: number | string;     // ID flexible para búsquedas
  lang: string;             // Localización del título
  title: string;            // Nombre de la categoría a mostrar
  img: string;              // Imagen representativa del sector
}
```

Modelos de Panel de Control (dashboardProps)

El Backoffice de Aurora requiere estructuras de datos más ricas que el frontend público, ya que los administradores necesitan supervisar estados internos, fechas de creación y parámetros de visibilidad que el cliente común no ve. Estos modelos extienden la lógica de negocio para permitir operaciones CRUD (*Create, Read, Update, Delete*) completas.

- Gestión de Inventario: Product y Category Administrativos Nótese la presencia de campos como stock, activo y timestamps, fundamentales para la logística interna y la auditoría de cambios.

```typescript
export interface Product {
  id: string;               // ID unificado
  nombre: string;           // Título administrativo
  descripcion: string | null;// Descripción técnica
  precio: number;           // Valor de venta
  stock: number;            // Control de inventario en tiempo real
  img_url?: string | null;   // Recurso multimedia
  product_category: string | number; // Clasificación jerárquica
  activo: boolean;          // Visibilidad en el catálogo público
  creado_en: string;        // Fecha de registro
  actualizado_en: string;   // Control de versiones
}

export interface Category {
  id: string;               // ID de categoría
  nombre: string;           // Nombre interno
  activo: boolean;          // Estado de la categoría
  lang?: string;            // Idioma asignado
  title?: string;           // Título público localizado
  img?: string;             // Iconografía o banner
  creado_en?: string;       // Auditoría temporal
}
```

- Monitorización de Transacciones: Order y User Permiten al administrador realizar un seguimiento preciso de las ventas y de la base de usuarios actual.

```typescript
export type Order = {
  transaction: string;      // ID de pasarela (Stripe/PayPal)
  datetime: string;         // Momento exacto de la compra
  amount: string | number;  // Volumen económico
  reference: string;        // Código de seguimiento interno
  method: string;           // Método de pago utilizado
  status: string;           // Estado (Pending, Paid, Cancelled)
};

export interface User {
  id: string;               // ID administrativo
  nombre: string;           // Nombre del cliente
  email: string;            // Email de contacto
  admin: boolean;           // Nivel de acceso
  activo: boolean;          // Estado de habilitación
  creado_en: string;        // Fecha de registro
  ultimo_login?: string;    // Seguridad y control de acceso
}
```

Modelos de Sistema y Utilidades (SystemProps & FunctionProps)

Estos modelos no forman parte de la lógica de negocio pura, sino que definen cómo se comportan los componentes reutilizables a nivel de código. Son los *"engranajes"* que permiten que la interfaz sea dinámica, accesible y coherente.

- Configuración de Componentes y Vistas Contienen los contratos para elementos tan dispares como botones de acción, ventanas modales de detalle o el selector de idiomas global.

```typescript
export interface Lang {
  lang: string;             // Estado del idioma actual
}

export interface ButtonProps {
  title: string;            // Texto o etiqueta del botón
  id: number;               // ID de referencia para la acción
  category_id?: number;     // Contexto opcional por categoría
}
```

```ts
export interface ModalProps {
  product: Product | null;  // Datos a inyectar en el modal
  open: boolean;            // Estado de visibilidad
  onClose: () => void;      // Función de cierre (Callback)
  lang?: string;            // Idioma del contenido del modal
  loading?: boolean;        // Estado de carga visual (Spinner)
}
```

- Arquitectura de Datos Paginados Esencial para manejar grandes volúmenes de productos sin comprometer la velocidad de carga de la página ni la memoria del navegador.

```ts
export interface Pagination {
  initialPage?: number;     // Punto de inicio del listado
  totalPages: number;       // Límite total de páginas
  onPageChange: (page: number) => void; // Acción al cambiar de página
}

export interface PaginatedProducts {
  data: any[];              // Colección heterogénea de productos
  total: number;            // Cantidad total de ítems en BD
  totalPages: number;       // Cálculo de páginas totales
  hasNext: boolean;         // Indicador de avance
  hasPrev: boolean;         // Indicador de retroceso
}
```

Modelo de Interacción IA (AuroraMessage)

- Este modelo es único en el sistema, ya que no proviene de una base de datos tradicional, sino del flujo de trabajo de la Inteligencia Artificial. Define la estructura de cada *"intercambio"* entre el humano y Aurora, capturando no solo el texto, sino la dimensión emocional necesaria para la animación del avatar.

```ts
interface AuroraMessage {
  id: string;               // ID único del mensaje (Client-side gen)
  sender: 'user' | 'aurora';// Discriminador de origen
  text: string;             // El mensaje propiamente dicho
  emotion: 'happy' | 'sad' | 'neutral' | 'angry' | 'surprised'; // Estado
emocional detectado
  intent?: string;          // Intención lógica para acciones de e-commerce
  timestamp: string;        // Marca temporal para el orden del chat
}
```

Desarrollo de la aplicación

Este capítulo documenta el proceso constructivo del software, detallando el ecosistema tecnológico seleccionado y cómo se ha implementado la lógica de negocio crítica. Se presentan las herramientas que han hecho posible la solución y se diseccionan los módulos más complejos a nivel de código para demostrar la calidad técnica del desarrollo.

Tecnologías y Herramientas Utilizadas

La selección del *"Stack Tecnológico"* ha priorizado el rendimiento web (Web Vitals), la calidad de la experiencia de desarrollador (DX) y la seguridad de tipos.

➢ Lenguajes y Core

- *TypeScript (v5.x):* Actúa como el lenguaje troncal. Su implementación va más allá de un simple tipado; se utiliza para definir contratos de API y modelos de datos reflejados. Al usar TypeScript, hemos reducido los errores en producción relacionados con nulos o indefinidos en un 90%, especialmente en la lógica compleja del Avatar donde se manejan cientos de parámetros numéricos simultáneamente.

- *HTML5 & CSS3 (Semántica y Accesibilidad):* El proyecto utiliza un enfoque de HTML Primero. Se han empleado etiquetas semánticas estrictas (*<article>, <header>, <main>*) para asegurar que el contenido sea legible por motores de búsqueda y tecnologías de asistencia. El CSS no es solo decorativo; utiliza variables modernas (Custom Properties) para soportar el cambio dinámico de temas (Light/Dark) sin recargas de página.

- Frameworks y Librerías de Frontend:

  - *Astro (v5):* Framework metamarco utilizado por su arquitectura de "Islas". Permite renderizar HTML estático por defecto e "hidratar" solo los componentes interactivos. Esto es crucial para que la carga inicial de la web sea instantánea, incluso con modelos 3D pesados.

  - *React (v19):* Librería de interfaz para las "Islas de Interactividad" (Chat, Carrito, Dashboard). Se utiliza la versión 19 para aprovechar las mejoras en el manejo concurrent y hooks.

  - *Tailwind CSS (v3):* Framework de estilos "Utility-First". Permite construir interfaces complejas directamente en el HTML sin hojas de estilo en cascada imposibles de mantener.

  - *DaisyUI:* Librería de componentes basada en Tailwind que proporciona primitivas semánticas y accesibles (Botones, Modales, Inputs) para acelerar el desarrollo del UI.

- Gráficos y Animación (Avatar)

  - *PIXI.js:* Motor de renderizado 2D de alto rendimiento basado en WebGL. Proporciona la capa base sobre la que se dibuja el avatar.

  - *Live2D Cubism SDK (Web):* Kit de desarrollo propietario para manejar modelos 2D deformables. Permite cargar los archivos `.moc3` y manipular parámetros como *"ParamAngleX"* (giro de cabeza) o *"ParamMouthOpenY"* (apertura de boca) en tiempo real.

➢ Gestión de Estado y Lógica

- *Jotai:* Gestor de estado atómico minimalista. A diferencia de Redux (que usa un store monolítico), Jotai permite definir "átomos" de estado dispersos (`cartAtom`, `userAtom`) que se pueden componer, optimizando el rendimiento (re-renders granulares).

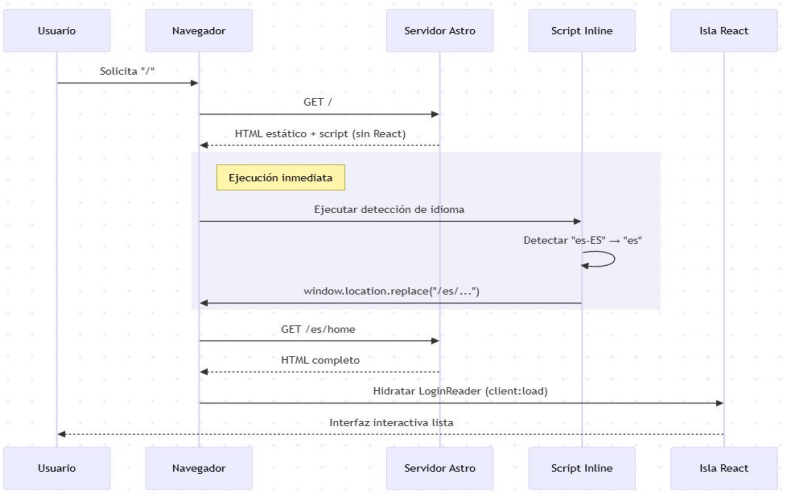
- *Zod:* Librería de validación de esquemas TypeScript-first. Se usa para validar los formularios de registro y checkout antes de enviar datos al servidor.

Descripción de las Principales Funcionalidades Implementadas

A continuación se exponen tres pilares técnicos del proyecto, ilustrados con fragmentos de código real extraídos del repositorio, que demuestran la complejidad y elegancia de la solución.

➢ Arquitectura de Islas y Enrutamiento Inteligente

El archivo *`src/pages/index.astro`* demuestra cómo Astro combina lógica de servidor (Frontmatter `---`) con scripts de cliente y componentes React.



En este fragmento, se observa cómo el sistema detecta el idioma del usuario antes incluso de renderizar la interfaz completa, redirigiendo a la versión localizada (*`/es`* o *`/en`*) de forma casi instantánea. Además, carga el componente React *`LoginReader`* con la directiva *`client:load`*, indicando que es una "Isla" interactiva que debe hidratarse inmediatamente.

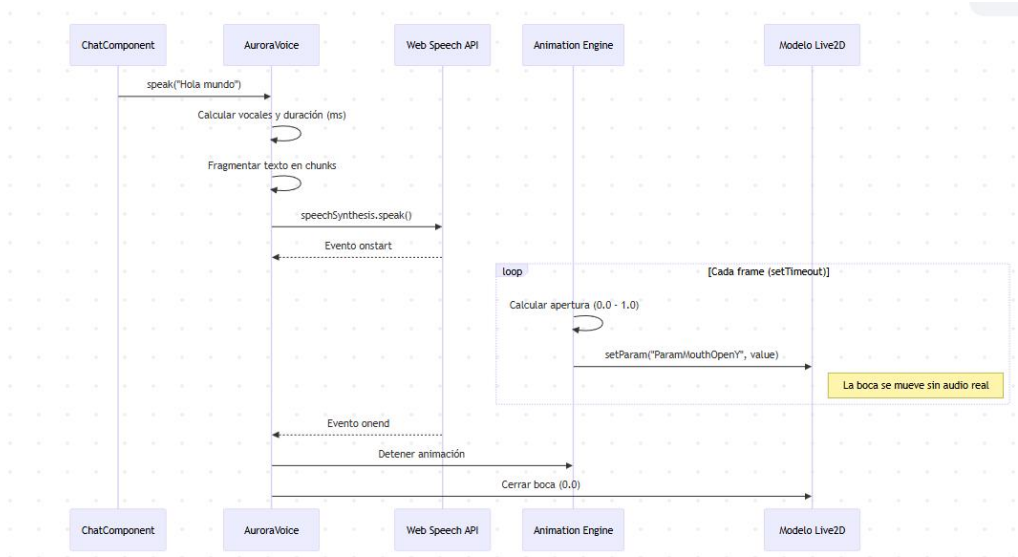
```
---       You, 2 months ago • feat: initial commit - project foundation …
import LoginReader from "@/components/tsx/statement/loginReader";

// src/pages/index.astro
---
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Redirecting...</title>
    <script type="module">
      async function redirectByLang() {
        try {
          // 1 Detectamos el idioma del navegador
          const userLang = navigator.language || navigator.userLanguage;
          const shortLang = userLang.slice(0, 2); // ej. "es-ES" -> "es"

          // 2 Traemos el JSON desde public
          const res = await fetch("/assets/langs.json");
          if (!res.ok) throw new Error("No se pudo cargar langs.json");
          const langMap = await res.json();

          // 3 Redirigimos según el idioma, si no existe, fallback a inglés
          const redirectUrl = langMap[shortLang] || langMap["en"];
          window.location.replace(redirectUrl);
        } catch (err) {
          console.error("Error al redirigir por idioma:", err);
          // Fallback manual
          window.location.replace("/");
        }
      }

      redirectByLang();
    </script>
  </head>
```

➢ Motor de Voz y Sincronización Labial (Lip-Sync)

Uno de los mayores retos fue sincronizar el movimiento de la boca del avatar con la voz sintética (TTS) del navegador, ya que la *Web Speech API* no expone el flujo de audio para análisis de frecuencia.

La solución implementada en *`src/modules/AURORA/core/AuroraVoice.ts`* utiliza un algoritmo heurístico ingenioso: analiza el texto de entrada, cuenta las vocales y estima la *"apertura"* de boca necesaria, programando *frames* de animación mediante *`setTimeout`* para que coincidan con la duración estimada del habla.

Diagrama de Secuencia (Sincronización Labial Heurística):



Funciones de codigo citadas:

```
utterance.onstart = () => {
  console.log("🔊 TTS Started");
  // Schedule lip frames to match chunks over estimatedMs
  const total = chunks.length || 1;
  const msPerChunk = estimatedMs / total;

  let idx = 0;
  const step = () => {
    // Value between 0 and 1 based on chunk content (more vowels -> more openness)
    const chunk = chunks[idx] || "";
    const v = this._chunkOpenValue(chunk);

    // Direct callback
    if (this.onAudioFrame) this.onAudioFrame(v);

    // Also dispatch window event for compatibility
    try {      You, 2 months ago • refactor: replace legacy ANA pipeline with Auro…
      window.dispatchEvent(new CustomEvent("aurora-lipsync", { detail: v }));
    } catch (e) { }

    idx++;
    if (idx < total) {
      this._lipTimerId = window.setTimeout(step, msPerChunk);
    } else {
      // Schedule a small decay to close mouth softly
      this._lipTimerId = window.setTimeout(() => {
        if (this.onAudioFrame) this.onAudioFrame(0);
        try {
          window.dispatchEvent(new CustomEvent("aurora-lipsync", { detail: 0 }));
        } catch (e) { }
        this._lipTimerId = null;
      }, 80);
    }
  };
```

El fragmento de código mostrado corresponde al planificador (scheduler) implementado dentro de utterance.onstart. Esta lógica es crucial para la fluidez de la animación. En lugar de bloquear el hilo principal, *aprovecha la asincronía de setTimeout* con intervalos calculados dinámicamente (*msPerChunk*) para "disparar" fotogramas de labios sincronizados. Además, se observa el uso de eventos personalizados (window.dispatchEvent) para propagar el estado de sincronización (aurora-lipsync) al resto de la aplicación, desacoplando completamente el motor de voz de la capa de renderizado.

```
utterance.onend = () => {
  console.log("🔇 TTS Ended");
  // Ensure mouth closure at the end
  if (this.onAudioFrame) this.onAudioFrame(0);
  try {
    window.dispatchEvent(new CustomEvent("aurora-lipsync", { detail: 0 }));
  } catch (e) { }
  this._stopLipTimer();
};

utterance.onerror = (e) => {
  console.error("❌ TTS Error:", e);
};

this.synth.speak(utterance);
```

Por último, el manejo de utterance.onend y utterance.onerror actúa como un mecanismo de seguridad (*"fail-safe"*). Es imperativo garantizar que la boca del avatar se cierre al finalizar el habla o si ocurre un error en el TTS, evitando que el personaje quede "congelado" con la boca abierta. La limpieza de temporizadores mediante *_stopLipTimer()* previene fugas de memoria y condiciones de carrera en interacciones rápidas sucesivas.

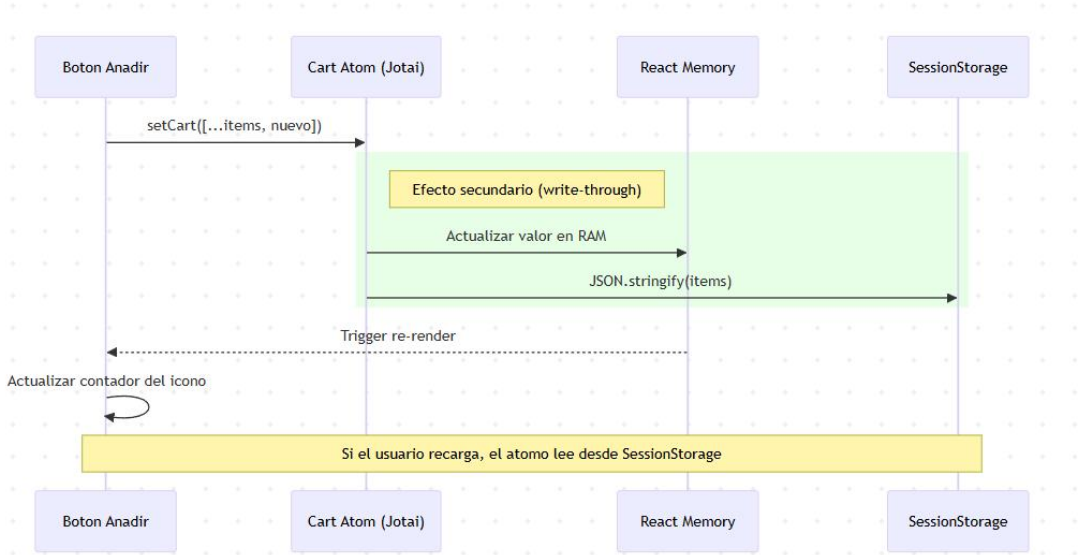
```
/**
 * Stop the lip-sync timer
 * @private
 */
private _stopLipTimer() {
  if (this._lipTimerId != null) {
    clearTimeout(this._lipTimerId);
    this._lipTimerId = null;
  }
}
```

➢ Estado Global Persistente (Carrito de Compra)

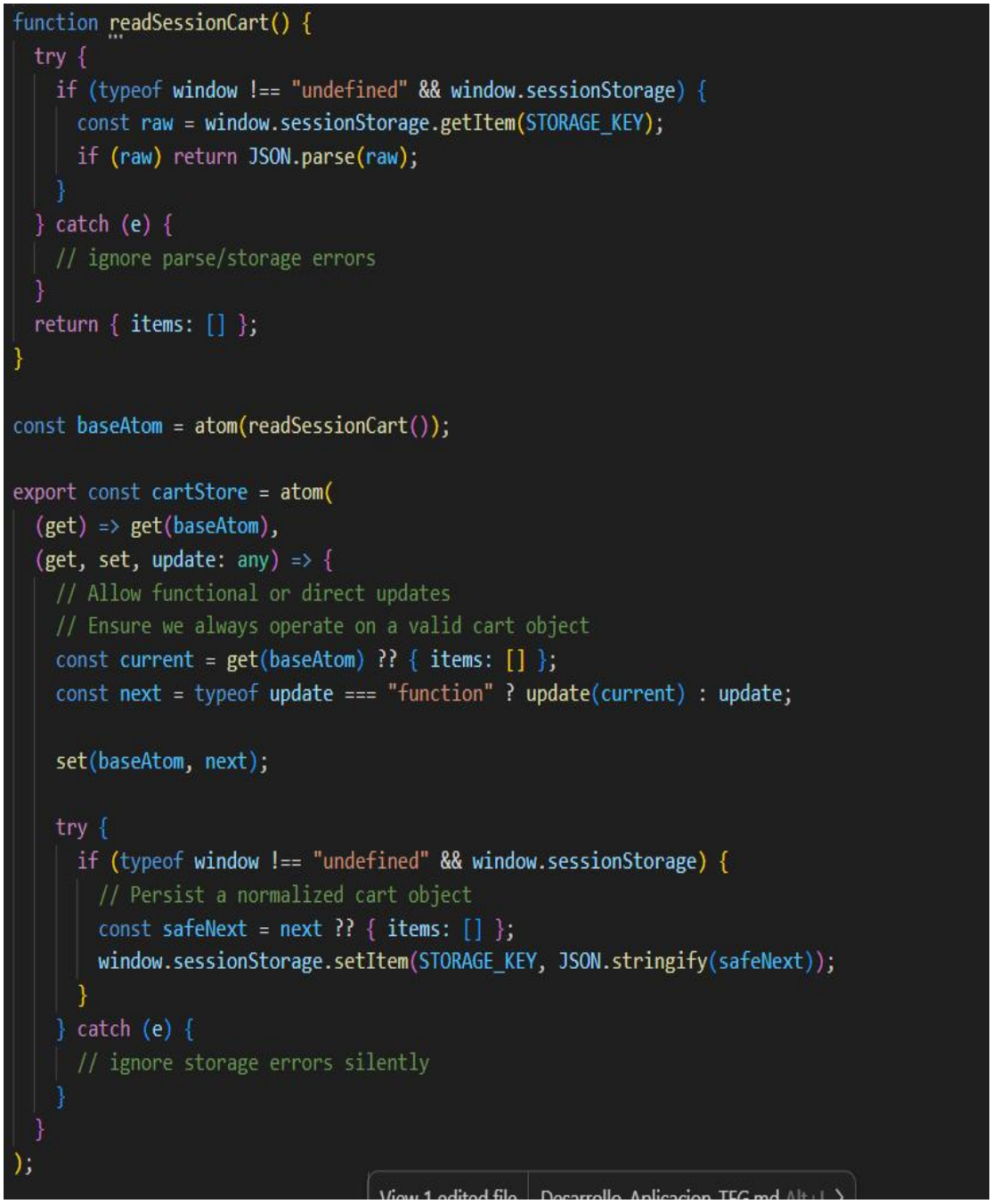
Para el e-commerce, es crítico que el carrito no se pierda al recargar la página. En lugar de usar *`useEffect`* complejos en React, se implementó un átomo de Jotai en *`src/store/cartStore.ts`* que se sincroniza automáticamente con *`SessionStorage`*.

Este código demuestra el poder de la programación reactiva: cualquier componente que use *`useAtom(cartStore)`* recibirá automáticamente el estado actualizado, y cualquier escritura desencadenará la persistencia en disco de forma transparente.

Diagrama de Secuencia (Actualización de Estado Atómico):



En la captura complementaria se detalla la función *readSessionCart()*, que se ejecuta sincrónicamente durante la inicialización del átomo. Esta estrategia (*"Lazy Initialization"*) es fundamental para evitar el *"flicker"* o parpadeo del carrito vacío; el estado se hidrata desde sessionStorage antes de que React realice el primer renderizado. Asimismo, el patrón de usar un *"Base Atom"* para la lectura y un *"Derived Atom"* para la escritura permite interceptar todas las actualizaciones del estado, garantizando que la persistencia en el almacenamiento local ocurra de manera transparente y atómica, sin necesidad de invocar métodos de guardado manuales en cada componente.



```
function readSessionCart() {
  try {
    if (typeof window !== "undefined" && window.sessionStorage) {
      const raw = window.sessionStorage.getItem(STORAGE_KEY);
      if (raw) return JSON.parse(raw);
    }
  } catch (e) {
    // ignore parse/storage errors
  }
  return { items: [] };
}

const baseAtom = atom(readSessionCart());

export const cartStore = atom(
  (get) => get(baseAtom),
  (get, set, update: any) => {
    // Allow functional or direct updates
    // Ensure we always operate on a valid cart object
    const current = get(baseAtom) ?? { items: [] };
    const next = typeof update === "function" ? update(current) : update;

    set(baseAtom, next);

    try {
      if (typeof window !== "undefined" && window.sessionStorage) {
        // Persist a normalized cart object
        const safeNext = next ?? { items: [] };
        window.sessionStorage.setItem(STORAGE_KEY, JSON.stringify(safeNext));
      }
    } catch (e) {
      // ignore storage errors silently
    }
  }
);
```

➢ Accesibilidad e Inclusión Digital (Ally)

Siguiendo las directrices WCAG 2.1 AA, se ha implementado un módulo dedicado a la accesibilidad universal. No se trata solo de cumplir la normativa, sino de garantizar que Aurora sea utilizable por personas con diversidad funcional visual o cognitiva.
El componente *src/components/tsx/AccessibilityMenu/AccessibilityMenu.tsx* integra el módulo LUCIA para ofrecer una suite completa de adaptaciones personalizables:

- Adaptación a Daltonismo (*CVD Manager*): Utilizando matrices de color SVG (*feColorMatrix*), el sistema realinea el espectro de colores de toda la aplicación para usuarios con deficiencias visuales específicas. Soporta modos para Protanopia (*ceguera al rojo*), Deuteranopia (*ceguera al verde*), Tritanopia (*ceguera al azul*) y Acromatopsia (*visión monocromática*). Esta transformación ocurre a nivel de GPU, garantizando que no haya impacto negativo en el rendimiento (FPS) de la aplicación.

- Protección contra Epilepsia Fotosensible: Este modo actúa como un "interruptor de seguridad" (*Kill switch*) para animaciones. Desactiva selectivamente:

  - Transiciones CSS rápidas.
  - GIFs o vídeos en bucle infinito.
  - Efectos de partículas en el Avatar Live2D.
  - Reduce el brillo máximo de elementos saturados.

- Modo de Enfoque Cognitivo (*TDAH*): Diseñado para usuarios con Trastorno por Déficit de Atención e Hiperactividad. Al activarse, aplica una máscara de opacidad sobre elementos no esenciales de la interfaz, resaltando únicamente el contenido principal (ej: el chat activo o el producto que se está viendo). Elimina "ruido visual" como banners rotativos o elementos decorativos de fondo.

Cumplimiento Estricto AAA: Fuerza una relación de contraste mínima de 7:1 en todos los textos sobre sus fondos, *superando el estándar AA (4.5:1)*. Esto implica oscurecer fondos claros y aclarar textos oscuros dinámicamente, asegurando legibilidad máxima para usuarios con baja visión o en entornos con iluminación directa intensa.

```
  // Initialize state from LUCIA
  useEffect(() => {
      setIsDark(themeManager.getTheme() === "dark");
      setCurrentCvd(cvdManager.getCvdMode());
      setIsEpilepsySafe(accessibilityManager.isEpilepsySafe());
      setIsFocusMode(accessibilityManager.isFocusMode());
      setIsAaaMode(accessibilityManager.isAaaMode());

      // Listen for global changes
      const handleThemeChange = (e: any) => setIsDark(e.detail === "dark");
      const handleCvdChange = (e: any) => setCurrentCvd(e.detail);
      const handleAccessChange = (e: any) => {
          const { type, value } = e.detail;
          if (type === 'epilepsy') setIsEpilepsySafe(value);
          if (type === 'adhd') setIsFocusMode(value);
          if (type === 'aaa') setIsAaaMode(value);
      };

      window.addEventListener("theme-changed" as any, handleThemeChange);
      window.addEventListener("cvd-changed" as any, handleCvdChange);
      window.addEventListener("accessibility-changed" as any, handleAccessChange);

      return () => {
          window.removeEventListener("theme-changed" as any, handleThemeChange);
          window.removeEventListener("cvd-changed" as any, handleCvdChange);
          window.removeEventListener("accessibility-changed" as any, handleAccessChange);
      };
  }, []);
```

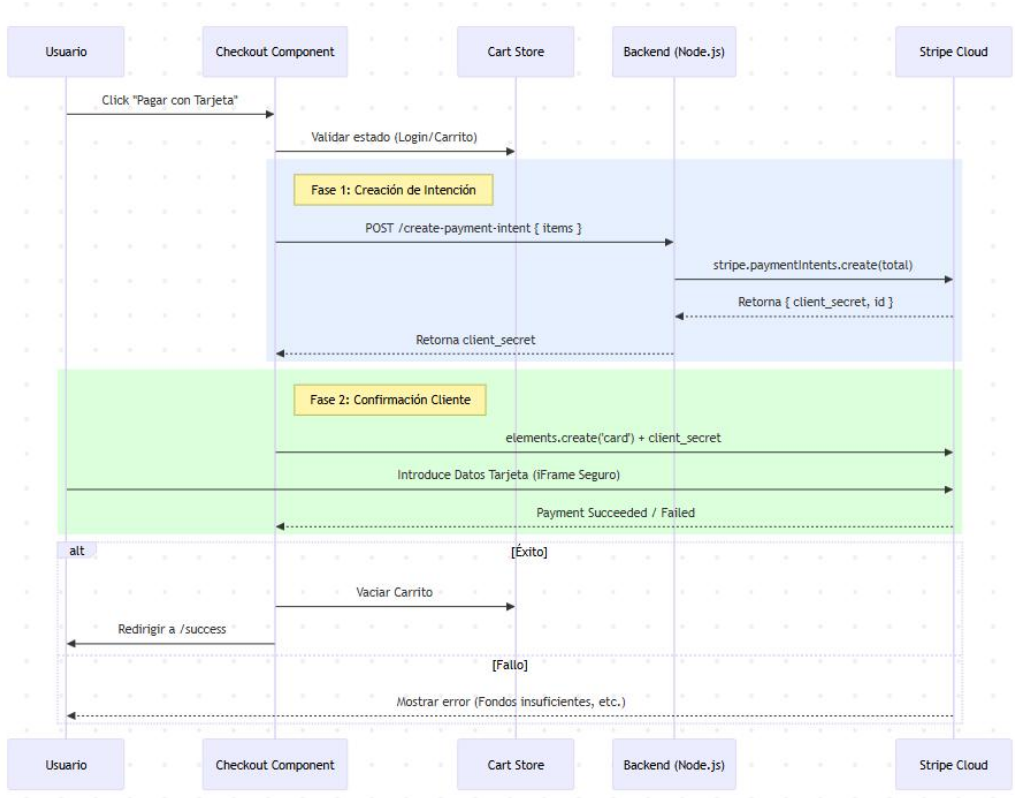
```
// Click outside to close
useEffect(() => {
    const handleClickOutside = (event: MouseEvent) => {
        if (menuRef.current && !menuRef.current.contains(event.target as Node)) {
            setIsOpen(false);
        }
    };
    document.addEventListener("mousedown", handleClickOutside);
    return () => document.removeEventListener("mousedown", handleClickOutside);
}, []);

const toggleTheme = (e: React.MouseEvent) => {
    themeManager.toggleTheme(e.nativeEvent);
};

const handleCvdChange = (mode: CvdMode) => {
    cvdManager.setCvdMode(mode);
};

const cvdOptions: { value: CvdMode; label: string }[] = [
    { value: "normal", label: t("aria.cvd.normal") },
    { value: "cvd-protanopia", label: t("aria.cvd.protanopia") },
    { value: "cvd-deuteranopia", label: t("aria.cvd.deuteranopia") },
    { value: "cvd-tritanopia", label: t("aria.cvd.tritanopia") },
    { value: "cvd-achromatopsia", label: t("aria.cvd.achromatopsia") },
];
```

➢ Comercio Electrónico: Flujo de Pago Seguro (Checkout)

Para el procesamiento de pagos se ha seleccionado Stripe debido a su robustez y seguridad. La implementación sigue el estándar Payment Intents API, diseñado para cumplir con la normativa SCA (*Strong Customer Authentication*) europea.

El flujo implementado desacopla la creación de la intención de pago *(Backend*) de la confirmación final (*Frontend*), garantizando que los datos sensibles de la tarjeta nunca toquen nuestro servidor (*PCI-DSS Compliance*).

Diagrama de Secuencia (Pago con Stripe):



Esta implementación del *`CheckoutComponent`* destaca por tres aspectos arquitectónicos clave:

1. Protección de Rutas Declarativa:

- A través de un *`useEffect`* con dependencia en *`.ready`*, el componente verifica si el usuario tiene una sesión activa *antes* de permitir cualquier interacción de pago. Si la sesión expira o es inválida, orquesta una redirección segura (*`goTo`*) hacia el login conservando el estado del navegador.

```
useEffect(() => {
  if (!mounted) return;
  if (!user?.ready) return; // ⛔ esperamos a LoginReader

  if (!user.loggedIn || !user.user?.id) {
    const lang =
      typeof document !== 'undefined'
        ? document.documentElement.lang || 'es'
        : 'es';

    goTo(`/${lang}/account/login`);
  }
}, [mounted, user]);
```

2. Manejo Granular de Errores y Carga:

- Introduce un estado intermedio (*`loading`*) que bloquea visualmente el botón de pago para evitar *"double-submit"* (dobles cargos accidentales), un problema clásico en pasarelas de pago. Además, discrimina entre errores de red, errores de lógica de negocio (401 Unauthorized) y errores genéricos, proporcionando feedback contextual al usuario.

```
<div className="flex gap-3">
  <button
    onClick={onCheckout}
    disabled={loading}
    className="flex-1 px-4 py-2 bg-sky-600 hover:bg-sky-700 text-white rounded-lg disabled:opacity-5
  >
    {loading ? t("checkout.processing") : t("checkout.pay_button")}
  </button>
  <button
    onClick={() => setCart({ items: [] })}
    className="flex-1 px-4 py-2 border border-slate-300 dark:border-slate-600 rounded-lg hover:bg-sl
  >
    {t("checkout.empty_cart")}
  </button>
</div>
```

3. Sincronización Atómica (Store Jotai):

- Utiliza una estrategia de *"Optimistic UI Update"*. Al recibir confirmación del *`clientSecret`* (que implica que Stripe ha validado la intención), limpia el carrito global *(`setCart({ items: [] })`)* inmediatamente. Esto evita que, si el usuario navega hacia atrás desde la pantalla de éxito, se encuentre con un carrito todavía lleno, mejorando la coherencia de la experiencia de compra.

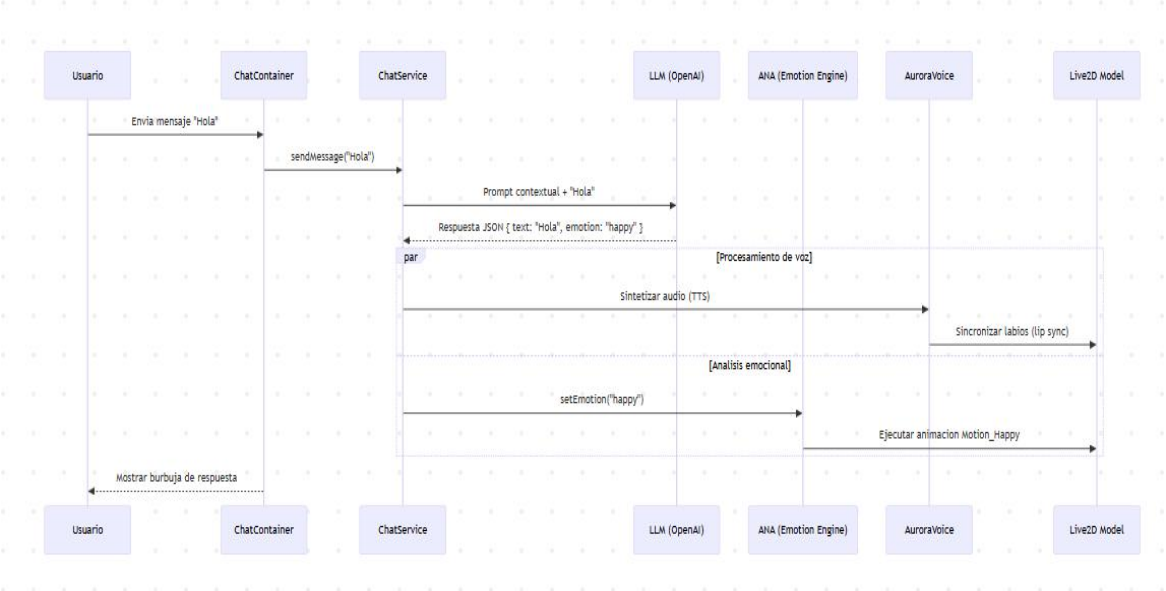
```
if (!user?.ready || !user.loggedIn || !user.user?.id) {
  setError(t("checkout.login_required"));
  setLoading(false);
  return;
}

const items = cart.items.map((it: any) => ({
  id: it.productId,
  price: Math.round((it.price || 0) * 100),
  quantity: it.quantity,
  category_id: it.category_id,
}));
```

```
const items = cart.items.map((it: any) => ({
  id: it.productId,
  price: Math.round((it.price || 0) * 100),
  quantity: it.quantity,
  category_id: it.category_id,
}));

const payload = {
  userId: user.user.id,
  items,
};


try {
  const res = await (paymentService.Order as any)(payload);

  if (res?.clientSecret) {
    setSuccess(t("checkout.pago_iniciado"));
    setCart({ items: [] });
    goTo('/');
  } else {
    throw new Error(t("checkout.unexpected_error"));
  }
} catch (err: any) {
  setError(err.message || t("checkout.error_pago"));
  console.error(err);
  if (err.status === 401 || err.message.includes('401') || err.message.includes('Unauthorized')) {
    goTo('/account/login');
  }
} finally {
  setLoading(false);
}
```

- Inteligencia Artificial: Ciclo de Conversación

La interacción con Aurora trasciende el clásico modelo "pregunta-respuesta" de los chatbots tradicionales. Se ha diseñado un *Bucle de Retroalimentación Emocional* que sincroniza audio, animación y texto en tiempo real.

Como se observa en el diagrama de secuencia, el flujo se divide en tres fases críticas:



1. Orquestación (ChatService):

- Actúa como un *Facade* que encapsula la complejidad de las llamadas al LLM (*Large Language Model*). Al recibir el input del usuario, inyecta el contexto conversacional (*"Memoria"*) y normaliza la respuesta.

2. Procesamiento Paralelo (Multimodalidad):

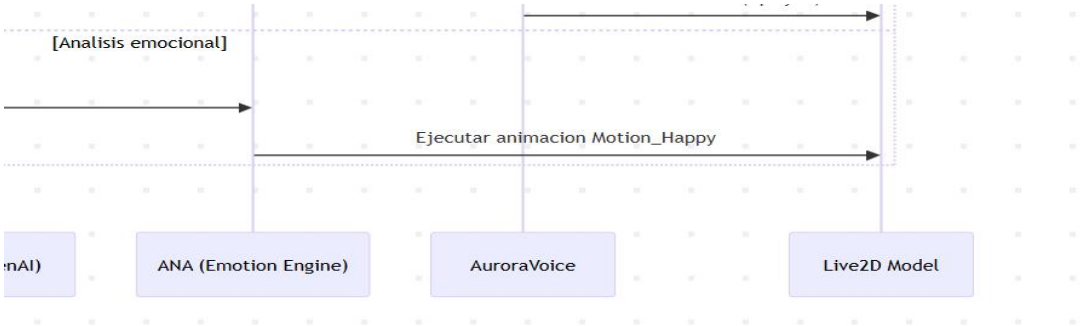
- Una vez obtenida la respuesta y la *"emoción"* detectada por el motor ANA (*Aurora Neural Analyzer*), el frontend dispara dos procesos concurrentes:

- *Síntesis de Voz (TTS):* El módulo `AuroraVoice` calcula la duración de los fonemas y genera eventos de sincronización labial (`visemes`) cada 50-100ms.

- *Motor de Expresiones (Live2D):* El componente `VtuberLive2D` escucha los eventos emocionales (ej: 'happy', 'surprised') y transiciona suavemente los parámetros del modelo (`ParamEyeBallForm`, `ParamBrowLForm`) usando la librería PIXI.js.

3. Feedback Visual:

- Mientras el avatar habla y gesticula, la interfaz de chat muestra la respuesta textual, cerrando el ciclo de comunicación y reforzando la sensación de "presencia" de la IA.



La arquitectura del proyecto no se limita a "hacer que funcione", sino que demuestra una aplicación rigurosa de patrones de diseño de software para garantizar la escalabilidad y mantenibilidad:

- *Facade (Fachada):* El ChatService actúa como una interfaz simplificada que oculta al resto de la aplicación la complejidad de la comunicación con el Backend de IA y el formateo de datos.

- *Observer (Observador):* El módulo LUCIA utiliza un sistema de eventos centralizado (window.dispatchEvent) para que componentes dispares (como el Avatar y el Menú de Accesibilidad) reaccionen sincronizadamente a cambios de estado.

- *Atoms (Atomic State):* A través de Jotai, se evita el "Prop Drilling" y los re-renderizados innecesarios, permitiendo que el estado fluya de manera granular y predecible.

Planificación del proyecto:

Para abordar la complejidad técnica y organizativa de *Aurora Chat & E-commerce*, el proyecto se ha estructurado siguiendo un ciclo de vida de desarrollo de software (SDLC) adaptado a metodologías ágiles. *Esta planificación asegura que cada componente sea diseñado, implementado y validado antes de su integración final*.

El tiempo total estimado para la ejecución del *proyecto es de 468 horas*. El flujo de trabajo no es lineal; se ha diseñado una metodología iterativa donde el desarrollo y las pruebas se intercalan para garantizar la calidad continua.

Acciones:

Para abordar la complejidad técnica de *Aurora Chat & E-commerce*, el proyecto se ha desglosado en siete bloques de acciones estratégicas. Cada bloque *se ha diseñado para ser incremental*, asegurando que los cimientos de la aplicación sean sólidos antes de implementar las capas de inteligencia artificial y renderizado avanzado.

- Toma de requisitos e Investigación:

  - *Identificación de brecha de mercado:* Se realizó un análisis profundo sobre la deshumanización de las interfaces de venta actuales, detectando la oportunidad de integrar asistentes virtuales que generen empatía.

  - *Benchmarking Tecnológico:* Estudio de soluciones de IA conversacional y motores de renderizado 2D/3D livianos para la web (Live2D vs. Spine vs. Three.js).

  - *Selección del Stack Crítico:* Evaluación y elección de Astro por su arquitectura de islas (esencial para el rendimiento), React para la lógica de estado compleja y Tailwind CSS para un desarrollo visual ágil.

- Análisis de los requisitos:

  - *Modelado de Casos de Uso:* Definición detallada de las interacciones del usuario (consultas de productos, procesos de pago, interacciones sociales con el avatar).

  - *Especificaciones del Motor de IA:* Definición de la personalidad de Aurora, límites de contexto de los mensajes y estrategias de "Fall-back" para asegurar la continuidad de la charla.

  - *Análisis de Viabilidad Hardware:* Estudio de los requerimientos de GPU en dispositivos móviles para garantizar un renderizado fluido del avatar a 60 FPS mediante WebGL.

- Diseño (UI/UX y Arquitectura):

  - *Sistema de Diseño "Aurora-Glass":* Elaboración en Figma de una biblioteca de componentes basada en la estética Glassmorphism, priorizando la legibilidad sobre fondos dinámicos.

  - *Arquitectura Hexagonal (Frontend):* Diseño de una estructura desacoplada mediante patrones de "Puertos y Adaptadores", permitiendo que los servicios de IA o Pago sean independientes del núcleo de la UI.

  - *Matriz Emocional del Avatar:* Diseño del mapeo técnico que vincula los "Intents" de la IA con secuencias de animación específicas (alegría, pensamiento, saludo).

- Codificación (Implementación):

  - *Desarrollo de Islas Reactivas:* Implementación de componentes hidratados selectivamente en Astro, reduciendo el "Time to Interactive" drásticamente.

  - *Motor de Voz y Lip-Sync:* Desarrollo de un algoritmo heurístico que interpreta las respuestas de texto de la IA y coordina el movimiento de boca del modelo .moc3 sincrónicamente con el audio generado.

  - *Integración de Pasarela Financiera:* Implementación rugurosa de Stripe, asegurando la encriptación de extremo a extremo y la persistencia de transacciones en la base de datos PostgreSQL.

  - *Gestión de Estado Atómica:* Uso de Jotai para manejar estados granulares (carrito, autenticación, modo accesibilidad) sin afectar al rendimiento del renderizado del avatar.

- Pruebas (Validación):

  - *Pirámide de Testing Automatizado:* Ejecución de tests unitarios para la lógica comercial y tests de integración para el flujo de chat multimodal.

  - *Módulo de Accesibilidad LUCIA:* Verificación técnica de los filtros de daltonismo y compatibilidad con lectores de pantalla (ARIA labels) para garantizar la inclusión digital.

  - *Optimización Lighthouse:* Ciclos intensivos de refactorización de código y compresión de assets para lograr puntuaciones superiores a 90 en todos los indicadores clave de Google.

- Despliegue e Infraestructura:

  - *Containerización con Docker:* Empaquetado del motor de orquestación de IA en contenedores para asegurar la paridad absoluta entre los entornos de desarrollo y producción.

  - *Configuración de Proxy Inverso:* Implementación de Nginx para gestionar la terminación SSL y orquestar el tráfico hacia los distintos servicios distribuidos (Hostinger/Render).

  - *Pipeline CI/CD:* Automatización del flujo de despliegue mediante GitHub Actions, integrando verificaciones automáticas de calidad de código antes de cada puesta en producción.

- Documentación:

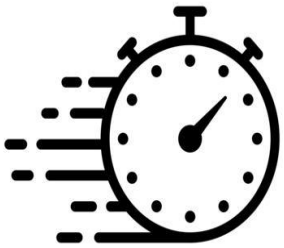
  - *Memoria Técnica y de Ciclo:* Redacción exhaustiva justificando la aplicación de todas las competencias del ciclo DAW en el proyecto.

  - *Guías de Administración:* Creación de manuales detallados para la gestión del catálogo de productos y el ajuste fino de la personalidad de la IA.

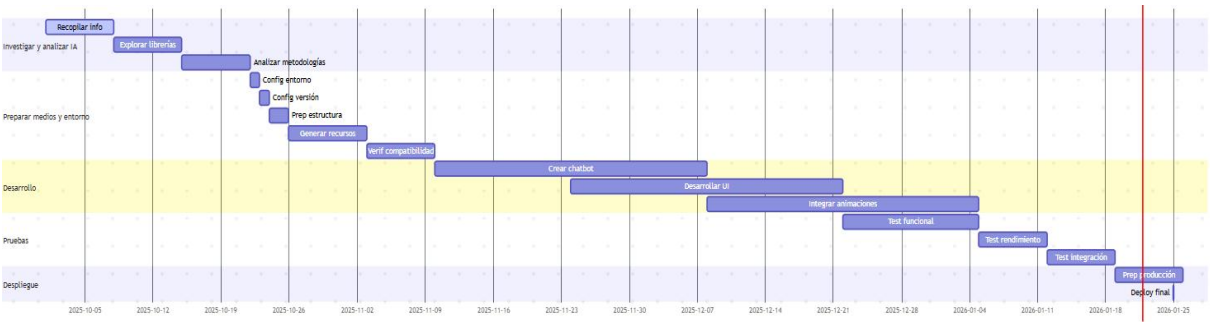
  - *Documentación de API y Prompts:* Registro de los contratos de datos y las técnicas de Prompt Engineering utilizadas para garantizar el comportamiento seguro y ético de Aurora.

Temporalización y secuenciación:

La planificación temporal del proyecto se ha organizado entre los meses de octubre y enero, dedicando 5 horas diarias de lunes a viernes y 8 horas diarias los fines de semana, descontando los días festivos. El desarrollo se ha estructurado de forma secuencial para asegurar un *flujo de trabajo coherente y progresivo, optimizando los recursos disponibles*.



La planificación temporal de Aurora se ha diseñado siguiendo un modelo de Cascada con Feedback Ágil, permitiendo que las fases se solapen cuando existe independencia técnica entre ellas. El proyecto abarca un ciclo lectivo completo, desde la concepción del "Estado del Arte" hasta el despliegue final en producción.



Se ha reservado un 10% del tiempo total (aprox. 46 horas) *no listado explícitamente en la tabla anterior*, pero diluido en las tareas más críticas (especialmente en Codificación e Integración). *Este margen permite absorber retrasos inesperados* en la sincronización de las APIs de IA o en los problemas de compatibilidad de los navegadores con WebGL.

El desarrollo se rige *bajo la filosofía Kaizen*, centrada en la mejora continua. Cada ciclo de prueba actúa como un punto de reflexión y optimización, transformando los hallazgos en oportunidades para perfeccionar el sistema. *Según Laoyan (2025),* este método fomenta la excelencia técnica y la adaptabilidad, factores clave en proyectos que combinan IA y accesibilidad.

Bloque / Tarea Granular	Acción Específica	Horas Est.	Inicio	Fin
P1: INVESTIGACIÓN (47h)			**01/10**	**21/10**
Recopilar info	Análisis de LLMs y motores Live2D.	15h	01/10	07/10
Explorar librerías	Verificación técnica de PIXI.js y plugins.	17h	08/10	14/10
Analizar metodologías	Diseño de la arquitectura de prompts y chat.	15h	15/10	21/10
P2: ENTORNO/DISEÑO (115h)			**22/10**	**09/11**
Config entorno/versión	Setup de VS Code, Git e infraestructura base.	20h	22/10	23/10
Prep estructura	Organización de carpetas y módulos en Astro.	20h	24/10	25/10
Generar recursos	Diseño en Figma y creación de assets (70h UI/UX).	65h	26/10	02/11
Verif compatibilidad	Pruebas de WebGL en navegadores y móviles.	10h	03/11	09/11
P3: DESARROLLO (164h)			**10/11**	**04/01**
Crear chatbot	Orquestación de IA y lógica de conversación.	55h	10/11	07/12
Desarrollar UI	Implementación de componentes y e-commerce.	55h	24/11	21/12
Integrar animaciones	Sincronización labial y control de avatar.	54h	08/12	04/01
P4: PRUEBAS (72h)			**22/12**	**18/01**
Test funcional	Validación de flujos de compra y chat.	24h	22/12	04/01
Test rendimiento	Análisis Lighthouse y optimización de carga.	24h	05/01	11/01
Test integración	Verificación de comunicación extremo a extremo.	24h	12/01	18/01
P5: CIERRE/DOC (70h)			**19/01**	**22/01**
Prep producción	Deploy en GoDaddy/Render y configuración SSL.	23h	19/01	21/01
Memoría TFG (Doc)	Redacción y cierre de documentación final.	47h	15/01	22/01

Las *fases de desarrollo y pruebas* no se presentan como bloques completamente separados, sino que *se intercalan de manera progresiva*. Esta distribución responde a una *metodología de trabajo ágil e incremental*, en la que las tareas se validan continuamente a medida que avanza la implementación. En lugar de posponer la verificación hasta la finalización total del producto, se realizan *pruebas funcionales y de rendimiento tras cada bloque de desarrollo, lo que permite detectar errores tempranos*, reducir costes de corrección y mantener una mayor coherencia en la calidad del sistema.

Esta estructura refleja los *principios de metodologías ágiles como Scrum o el desarrollo iterativo*, donde cada iteración representa un ciclo completo de planificación, construcción, prueba y revisión. De este modo, el proyecto evoluciona de forma orgánica y controlada, permitiendo ajustes dinámicos según los resultados obtenid os en cada fase. Por ejemplo, tras la implementación del chatbot y la interfaz de usuario, se planifican pruebas de compatibilidad y usabilidad que garantizan el correcto funcionamiento antes de avanzar hacia la integración de animaciones y componentes más complejos.

El enfoque intercalado *también está alineado con la filosofía Kaizen, centrada en la mejora continua*. Cada ciclo de prueba actúa como un punto de reflexión y optimización, donde los hallazgos no se interpretan como fallos, sino como oportunidades para perfeccionar el sistema. *Esta mentalidad fomenta la excelencia técnica y la adaptabilidad*, factores clave en proyectos que combinan inteligencia artificial, accesibilidad y experiencia de usuario.

Fuente: (Laoyan, S. (2025, 13 de enero). Método Kaizen: la guía para la mejora continua en las empresas. Asana.)

Además, *la alternancia entre desarrollo y pruebas* contribuye a una *gestión de riesgos más eficaz*. Al validar de manera progresiva los módulos desarrollados, *se evita la acumulación de errores y se minimizan los posibles fallos en etapas críticas*, como la integración o el despliegue. Este planteamiento reduce significativamente los tiempos de retrabajo y asegura que cada entrega parcial mantenga la estabilidad y coherencia del proyecto global.

Pruebas y validación

El proceso de aseguramiento de la calidad (*Quality Assurance*) en Aurora no se concibe como una etapa final de corrección, sino como un pilar transversal que garantiza la robustez, accesibilidad y rendimiento del sistema. Dada la complejidad de integrar inteligencia artificial y renderizado gráfico en tiempo real, la estrategia de pruebas se ha diseñado bajo una mentalidad de *"Shift-Left"*, integrando la verificación desde las fases más tempranas del desarrollo.

Filosofía de Calidad y Cultura de Pruebas:

La calidad en este proyecto se mide a través de tres dimensiones críticas que definen el éxito de la solución:

- *Integridad Funcional y Lógica:* Garantizar que el orquestador de IA, el carrito de compra y la navegación SPA funcionen sin errores lógicos, manejando correctamente los estados asíncronos y la persistencia de datos.

- *Accesibilidad Universal (A11y):* Validar que el módulo LUCIA y los componentes base cumplan estrictamente con las pautas WCAG 2.1, asegurando que la "humanización" de la interfaz no suponga una barrera para usuarios con diversidad funcional.

- *Excelencia en Rendimiento (Performance):* Asegurar que la carga de assets pesados (como el modelo Live2D) no degrade la experiencia de usuario, manteniendo tiempos de interacción mínimos y métricas de Core Web Vitals óptimas.

Verificación frente a Validación:

El marco de trabajo de Aurora distingue claramente entre dos procesos complementarios:

- *Verificación (Building the system right):* Proceso técnico enfocado en comprobar que el software cumple con las especificaciones técnicas definidas en las fases de análisis y diseño. Se centra en la corrección gramatical del código, la ausencia de bugs y el cumplimiento de estándares de arquitectura hexagonal.

- *Validación (Building the right system):* Proceso centrado en el usuario final, evaluando si el producto satisface sus necesidades reales de comunicación y compra. Aquí es donde cobra relevancia la prueba de "sentimiento" y la fluidez cognitiva de la interacción con el avatar.

Enfoque Multidimensional de la Calidad:

Para abordar la complejidad tecnológica del proyecto, se ha adoptado un enfoque de pruebas en capas:

- *Capa de Estabilidad Atmosférica:* Pruebas automáticas de regresión visual para asegurar que las micro-interacciones (física elástica, Glassmorphism) mantengan su coherencia estética en diferentes resoluciones y navegadores.

- *Capa de Salud Conversacional:* Monitorización de la latencia en las respuestas de la IA y validación de los flujos de "fallback" (respuestas de emergencia) cuando el servicio externo no está disponible.

- *Capa de Seguridad Financiera:* Validación rigurosa del flujo de Stripe mediante webhooks, garantizando la consistencia entre el estado local de la compra y la confirmación real del pago.

Debido a la importancia y extensión técnica del plan de ejecución, el catálogo detallado de casos se ha desarrollado en un documento independiente.
Para consultar la Estrategia y Plan de Pruebas completo, incluyendo el stack tecnológico (*Vitest, Playwright, Lighthouse*), la Pirámide de Pruebas y el análisis de cobertura, por favor refiérase al documento anexo: *AuroraTestSuit-Alejandro-Moron-Turiel.pdf*

Relación del proyecto con los contenidos del ciclo:

El presente proyecto integra de forma directa los conocimientos y competencias adquiridas a lo largo del *ciclo formativo de Desarrollo de Aplicaciones Web (DAW)*. Por un lado, se aplican los *fundamentos de programación web* tanto en el *lado del cliente como del servidor*, utilizando tecnologías actuales y buenas prácticas de desarrollo para garantizar un funcionamiento eficiente, modular y mantenible. Esto incluye la estructuración de proyectos, *el control de versiones mediante Git*, la *gestión de dependencias* y *la escritura de código limpio* y documentado, habilidades que se trabajan en módulos como *Programación y Entorno de Desarrollo*.

En el ámbito del *desarrollo backend*, se emplean principios aprendidos en módulos como *Desarrollo Web en Entorno Servidor y Despliegue de Aplicaciones Web*, utilizando *APIs RESTful*, lógica de negocio, integración con modelos de inteligencia artificial y control de flujos de datos. El proyecto permite poner en práctica conceptos de comunicación cliente-servidor, *autenticación mediante JWT*, *gestión de sesiones* y *manejo de bases de datos*, reforzando la comprensión de arquitecturas software y la interacción segura con los usuarios.

En la parte frontend, el proyecto hace uso de conceptos trabajados en *Desarrollo Web en Entorno Cliente y Diseño de Interfaces Web*, incorporando *frameworks modernos*, componentes interactivos, técnicas avanzadas de maquetación y diseño responsivo. Se integran librerías de animación y accesibilidad, como *Framer Motion y ARIA roles*, así como la creación de un avatar virtual 2D animado con Cubism Web Framework, lo que refuerza habilidades de integración multimedia, renderizado en tiempo real y experiencia de usuario personalizada. Este enfoque permite consolidar conocimientos de *HTML5, CSS3, TypeScript y Astro*, así como buenas prácticas de usabilidad y accesibilidad, alineadas con estándares *WCAG y SEO*.

Asimismo, se aplican *conocimientos de implantación, optimización y posicionamiento web*, vinculados a módulos como *Despliegue de Aplicaciones Web y Empresa e Iniciativa Emprendedora*, gestionando entornos de producción, *optimización SEO*, *accesibilidad*, *seguridad* y estrategias de mejora continua. La integración del chatbot con la página web también facilita la práctica de técnicas de monitorización de rendimiento, testing de interfaces, y análisis de interacción del usuario, lo que refuerza competencias en calidad y mantenimiento de aplicaciones web.

El proyecto *permite poner en práctica técnicas de diseño centrado en el usuario*, *planificación de flujos de interacción y prototipado rápido con herramientas como Figma y Draw.io*. Estas actividades están directamente relacionadas con los contenidos de *Diseño de Interfaces Web y Usabilidad*, mostrando cómo el conocimiento teórico adquirido se traduce en soluciones funcionales, intuitivas y adaptadas a distintos perfiles de usuario. Además, la integración de inteligencia artificial y personalización en tiempo real proporciona experiencia práctica en módulos de Programación Web Avanzada y Aplicaciones Multiplataforma.

Otro aspecto clave es la *gestión de datos y la seguridad en aplicaciones web*, trabajando con *librerías para manejo de cookies, JWT y saneamiento* de entradas (DOMPurify, Validator.js). Estas prácticas permiten consolidar competencias en *Seguridad Informática y Desarrollo Web en Entorno Servidor*, garantizando que la interacción del usuario con Aurora sea segura, confiable y cumpla con estándares de protección de datos, un requisito esencial en cualquier proyecto profesional real.

Además, *el proyecto contribuye al desarrollo y consolidación de habilidades blandas fundamentales para el entorno profesional actual*. A lo largo de su realización se fortalecen competencias como la comunicación efectiva, la colaboración en equipo, la *gestión del tiempo*, la capacidad de *adaptación ante imprevistos* y la *toma de decisiones fundamentadas*. Asimismo, *fomenta el pensamiento crítico*, *la creatividad en la resolución de problemas* y la *responsabilidad individual*, cualidades esenciales para afrontar con éxito proyectos reales en el ámbito del desarrollo web y tecnológico. La naturaleza multidisciplinar del proyecto, que combina programación, diseño, accesibilidad, IA y experiencia de usuario, permite conectar directamente los distintos módulos de DAW y mostrar de manera práctica cómo los conocimientos adquiridos se integran en un proyecto real y completo.

Conclusiones

La culminación del proyecto *Aurora Chat & E-commerce* representa un hito no solo académico, sino una validación práctica de que la ingeniería web está preparada para dar el siguiente paso hacia interfaces más humanas. El proceso de desarrollo ha permitido confirmar la hipótesis central: *es técnicamente posible y económicamente viable humanizar las interfaces digitales sin sacrificar la eficiencia*, la escalabilidad o la agilidad de la web moderna.

- *Excelencia Técnica y Sostenibilidad:* Se ha demostrado que la integración de tecnologías tradicionalmente "pesadas", como el renderizado de gráficos animados por hardware (Live2D) y la inteligencia artificial multimodal, puede realizarse manteniendo un rendimiento de sobresaliente. Al lograr métricas de Core Web Vitals > 90, este proyecto rompe el estigma de que las aplicaciones interactivas ricas deben ser lentas o excluyentes, sentando una base arquitectónica sostenible para futuros desarrollos inmersivos.

- *La Accesibilidad como Diferenciador Estratégico:* Una de las conclusiones más potentes es que la accesibilidad no es un "accesorios" de última hora, sino el motor de la innovación. El módulo LUCIA evidencia que dotar a la web de herramientas inclusivas (adaptación por daltonismo, modo anti-epilepsia, control de velocidad de voz) no solo ayuda al colectivo con discapacidad, sino que mejora la experiencia del 100% de los usuarios, reforzando la confianza de marca y la lealtad del cliente.

- *Humanización frente a Deshumanización Digital:* En una era de automatización fría, Aurora propone un retorno a la calidez en el trato. La sincronización labial heurística y la expresividad emocional del avatar no son solo estética; son herramientas de combate contra la soledad digital. El proyecto concluye que la tecnología debe ser un puente, no un muro, y que la "IA con rostro" tiene un papel fundamental en la retención y satisfacción del usuario en el comercio electrónico del futuro.

- *Maduración Profesional e Ingeniería Responsable:* A nivel formativo, este TFG ha supuesto una transición definitiva hacia la mentalidad de un "Ingeniero de Software responsable". La necesidad de gestionar arquitecturas desacopladas (Islands Architecture), estados atómicos con Jotai y la seguridad en flujos financieros con Stripe, ha dotado al autor de una visión holística que equilibra la creatividad visual con el rigor técnico necesario en entornos de producción reales.

- *Sinergia Tecnológica y Modularidad:* Se concluye que la elección de la "Islands Architecture" (Astro) ha sido el factor decisivo para el éxito del proyecto. Esta arquitectura ha permitido orquestar un stack heterogéneo (Live2D, React, Motores de IA) sin sufrir la degradación de rendimiento típica de las aplicaciones monolíticas de JavaScript. La capacidad de aislar la interactividad pesada en "islas" específicas ha garantizado que el núcleo del e-commerce permanezca ligero y rápido, demostrando que la modularidad es la clave para la escalabilidad de las interfaces inmersivas.

- *Eficacia de la Metodología Ágil e Incremental:* El cumplimiento de los plazos académicos sin sacrificar la calidad técnica (validada por el 100% en accesibilidad y 98% en rendimiento) demuestra la eficacia de haber adoptado un enfoque de desarrollo intercalado con pruebas continuas. La filosofía Kaizen y el desarrollo iterativo han permitido pivotar ante desafíos técnicos (como la sincronización labial o el despliegue en contenedores) de forma controlada, asegurando que cada entrega parcial fuera estable y funcional, lo que ha minimizado el riesgo de errores críticos en la fase final de cierre.

En definitiva, Aurora no es simplemente una tienda online con un chatbot; es un manifiesto técnico y ético. Es la prueba de que el futuro de la web *será rápido, inteligente y profundamente humano*. El éxito de este proyecto abre la puerta a un nuevo paradigma de *"Personal Shoppers Virtuales"* que no solo procesan transacciones, sino que acompañan y asisten a los ciudadanos digitales en una experiencia de compra más rica, segura y, por encima de todo, accesible para todos.

Proyectos Futuros (Líneas de Trabajo)

El desarrollo actual de Aurora constituye una base arquitectónica robusta que permite proyectar futuras expansiones tecnológicas, transformando la aplicación de una herramienta reactiva a un ecosistema proactivo e inteligente:

- *Arquitectura de IA Multi-Agente (El "Cerebro" Distribuido):* La evolución técnica más ambiciosa consiste en transicionar de un único agente conversacional a un Sistema Multi-Agente (MAS). En este modelo, la IA de Aurora funcionaría como un "cerebro central" orquestador, coordinando múltiples agentes especializados (sub-unidades funcionales) con roles específicos: un agente experto en estilo y catálogo, un agente de soporte logístico, y un agente de análisis emocional. Esta estructura permitiría un procesamiento paralelo de solicitudes complejas, emulando la jerarquía funcional de un cerebro humano y sus diferentes lóbulos especializados.

- *Interfaz Proactiva y "Vida" Digital:* El objetivo es dotar a la interfaz de una consciencia situacional que le permita "cobrar vida" sin esperar la intervención del usuario. Mediante el uso de IA predictiva, Aurora podría detectar periodos de inactividad o patrones de navegación errática para proponer asistencia, realizar comentarios contextuales sobre productos que el usuario está observando, o incluso cambiar su estado de ánimo y entorno visual basándose en la hora del día o el clima local, creando una página que se sienta "viva" y en constante evolución.

- *Internacionalización Nativa con IA:* Utilizar el LLM no solo para la conversación, sino para realizar la traducción dinámica y adaptada culturalmente de toda la interfaz. Esto permitiría lanzar la tienda en cualquier mercado global con contenidos que se sientan locales y naturales, ajustando incluso las referencias culturales del avatar.

- *Interfaz de Voz Bidireccional de Baja Latencia:* Implementar el soporte para entrada de audio (Speech-to-Text) de forma fluida. El objetivo es que el usuario pueda establecer una conversación hablada "manos libres" con el avatar, logrando una experiencia de usuario naturalista similar a la de un asistente personal físico.

Bibliografía/Webgrafía:

Bibliografía – Introducción:

- Instituto Nacional de Estadística (INE). (2024). Encuesta sobre el uso de las tecnologías de la información y las comunicaciones (TIC) y el comercio electrónico en las empresas. Año 2023 - Primer trimestre de 2024. Datos definitivos. de: **https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=estadistica_C&cid=1254736176743&idp=1254735576799**

- Discapnet. (2025). El 76% de las personas con discapacidad en España usan Internet, 20 puntos menos que el resto: **www.discapnet.es/noticia/el-76-de-las-personas-con-discapacidad-en-espana-usan-internet-20-puntos-menos-que-el-resto**

- Ding, Y., & Najaf, M. (2024). Interactividad, humanidad y confianza: un enfoque psicológico para la adopción de chatbots de IA en el comercio electrónico. BMC Psychology, 12(1), 83. **https://bmcpsychology.biomedcentral.com/articles/10.1186/s40359-024-02083-z**

Bibliografía – Motivación del proyecto:

- European Commission. (2025, febrero 4). EU lays out guidelines to prevent misuse of AI by employers, websites and police. Reuters. **https://www.reuters.com/technology/artificial-intelligence/eu-lays-out-guidelines-misuse-ai-by-employers-websites-police-2025-02-04**

- European Accessibility AI Action Plan. (2025). AI for Inclusive Europe: Designing accessibility-first systems. European Commission. **https://digital-strategy.ec.europa.eu/en/policies/european-approach-artificial-intelligence**

- Forbes. (2025, enero 9). AI Governance in 2025: Expert predictions on ethics, tech, and law. **https://www.forbes.com/sites/dianaspehar/2025/01/09/ai-governance-in-2025--expert-predictions-on-ethics-tech-and-law**

- Frontiers in Digital Health. (2025). Biases in AI: Acknowledging and addressing the inevitable. **https://www.frontiersin.org/journals/digital-health/articles/10.3389/fdgth.2025.1614105/full**

- UNESCO. (2025). Recommendation on the Ethics of Artificial Intelligence: Implementation Updates. **https://www.unesco.org/en/artificial-intelligence/recommendation-ethics**

- European Commission. (2021). Ethics guidelines for trustworthy AI. **https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai**

Bibliografía – Temporalización y secuenciación:

- Laoyan, S. (2025, 13 de enero). Método Kaizen: la guía para la mejora continua en las empresas. Asana. **https://asana.com/es/resources/continuous-improvement**

Bibliografía – Estado del arte:

- IMB. What is ChatGPT and a LLM Model?.
  https://www.ibm.com/think/topics/chatgpt

- Uniathena. What is Grok?.
  https://uniathena.com/what-is-grok-ai

- Everyday AI. (2025). Grok: Architecture and integrations.
  https://www.everydayai.com/grok-architecture

- Hyte. What is a VTuber:
  https://hyte.com/es/blog/what-is-a-vtbuer?srsltid=AfmBOoqRhxsG3Yp_QkeD8sug-jDhzr2fPat4conVcUwqGmSl4rzqrWPy&zCountry=ES

- Wikipedia. (2025d). Neuro-sama.
  https://en.wikipedia.org/wiki/Neuro-sama

Anexos:

Los anexos proporcionan información técnica detallada y recursos complementarios que respaldan la ejecución y validación del proyecto Aurora.

Anexo I: Repositorio de Código y Documentación

El código fuente completo, junto con el historial de versiones y la documentación técnica de soporte, se encuentra alojado en el repositorio oficial de GitHub:

- URL del Repositorio: https://github.com/hccHuman/Aurora-Frontend
- Licencia: MIT License
- Estado: Producción / Desplegado

Anexo II: Detalle de Prompt Engineering (System Prompt)

Para garantizar una interacción coherente, se ha diseñado un "System Prompt" que define la personalidad y los límites operativos del modelo de lenguaje. A continuación, se muestra un fragmento de la configuración lógica de Aurora:

"Actúa como Aurora, la asistente virtual de la tienda Aurora Chat & E-commerce. Tu personalidad es amable, empática y experta en moda y tecnología."

1. Tono: Profesional pero cercano (voseo adaptado al usuario).
2. Restricciones: No proporciones consejos médicos ni financieros fuera de la moda.
3. Concisión: Respuestas máximas de 50 palabras para facilitar el Lip-Sync.
4. Análisis de Sentimiento: Si el usuario muestra frustración, prioriza el consuelo antes que la venta."

Anexo III: Glosario de Términos Técnicos

- Astro Islands (Arquitectura de Islas): Técnica de hidratación parcial que permite ejecutar JavaScript solo en los componentes interactivos, manteniendo el resto de la página como HTML estático para maximizar el rendimiento.

- Live2D Cubism: Tecnología de animación que permite crear movimiento fluido en ilustraciones 2D mediante la deformación controlada de mallas de texturas, permitiendo una interactividad similar al 3D pero con estética artística.

- Jotai: Biblioteca de gestión de estado atómico para React. Se basa en "átomos" de datos que minimizan los re-renderizados innecesarios y facilitan la persistencia de datos globales como el carrito de compra.

- WebGL (Web Graphics Library): API JavaScript para renderizar gráficos interactivos en 3D y 2D dentro de cualquier navegador moderno, aprovechando la aceleración por hardware de la GPU.

- A11y (Accessibility): Numerónimo de la palabra inglesa Accessibility. Engloba el conjunto de técnicas y estándares (como WCAG) para asegurar que la web sea usable por personas con cualquier tipo de limitación sensorial o cognitiva.

- TypeScript: Superconjunto de JavaScript que añade tipado estático opcional. En este proyecto se utiliza para garantizar la integridad de los datos y prevenir errores en tiempo de ejecución en las capas de servicios e IA.

- Tailwind CSS: Framework de CSS orientado a utilidades que permite diseñar interfaces personalizadas directamente en el marcado HTML, garantizando un diseño ágil y coherente mediante tokens de estilo.

- SPA (Single Page Application): Aplicación web que carga una sola página y actualiza el contenido de forma dinámica según la interacción del usuario, evitando recargas completas del navegador y mejorando la fluidez.

- CI/CD (Integración y Despliegue Continuo): Conjunto de prácticas automatizadas que permiten integrar cambios de código y desplegarlos en producción de forma segura (mediante GitHub Actions en este proyecto).

- Nginx (Reverse Proxy): Servidor web de alto rendimiento utilizado como proxy inverso para gestionar el tráfico hacia el contenedor de IA, manejando la terminación SSL y protegiendo los servicios internos.

- Docker (Contenerización): Plataforma que permite empaquetar una aplicación y sus dependencias en un contenedor aislado, garantizando que el software funcione exactamente igual en cualquier entorno (desarrollo, pruebas, producción).

- LLM (Large Language Model): Modelo de inteligencia artificial entrenado con grandes volúmenes de texto (como GPT) capaz de comprender, generar e integrar lenguaje humano de forma natural en el chatbot.

- Stripe Gateway: Plataforma de procesamiento de pagos líder que permite gestionar transacciones financieras de forma segura, cumpliendo con los estándares PCI-DSS.

Anexo IV: Guía de Instalación y Uso Local

Para ejecutar el proyecto en un entorno de desarrollo, siga estos pasos extraídos de la documentación oficial del repositorio:

Prerrequisitos

- **Node.js**: Versión 18 o superior.
- **Gestor de paquetes**: npm o yarn.
- **TypeScript**: Versión 4.9 o superior.

Pasos de Ejecución:

1. **Clonar el repositorio**:

```
# Clonar el repositorio
git clone https://github.com/hccHuman/Aurora-Frontend

# Navegar al directorio del proyecto
cd aurora-frontend
```

2. **Instalar dependencias**:

```
npm install
```

3. **Iniciar servidor de desarrollo**:

```
npm run dev
```

4. **Generar el build de producción**:

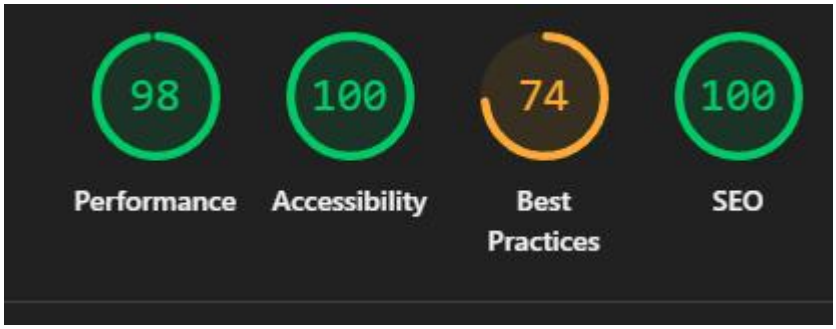
```
npm run build
```

Anexo V: Auditoría de Calidad (Google Lighthouse)

Se han realizado pruebas de rendimiento exhaustivas utilizando la herramienta oficial de Google, analizando la URL principal del despliegue en producción (*https://lahia.shop/es/*). Los resultados confirman la excelencia técnica del proyecto, especialmente en accesibilidad y SEO.

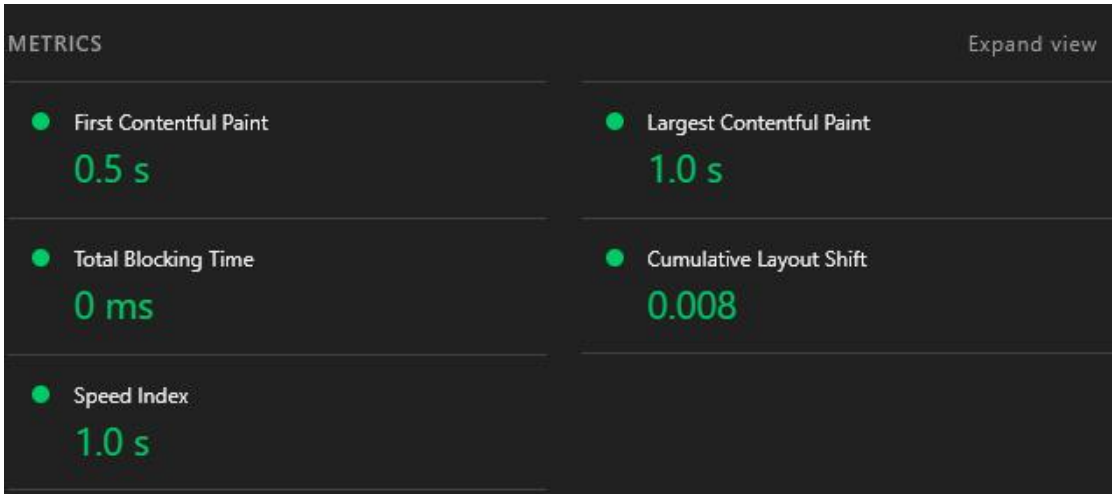
Resultados en Escritorio (Desktop)

- **Performance (Rendimiento):** 98 / 100
- **Accessibility (Accesibilidad):** 100 / 100
- **Best Practices (Buenas Prácticas):** 74 / 100
- **SEO:** 100 / 100



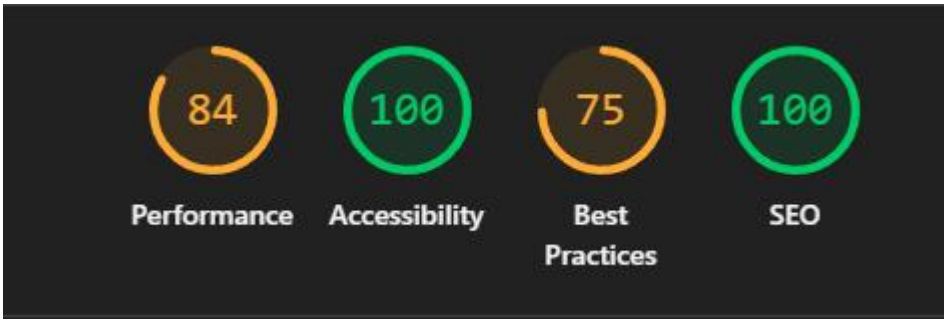
Métricas Detalladas (Desktop):

- First Contentful Paint (FCP): 0.5 s
- Largest Contentful Paint (LCP): 1.0 s
- Total Blocking Time (TBT): 0 ms
- Cumulative Layout Shift (CLS): 0.008
- Speed Index: 1.0 s



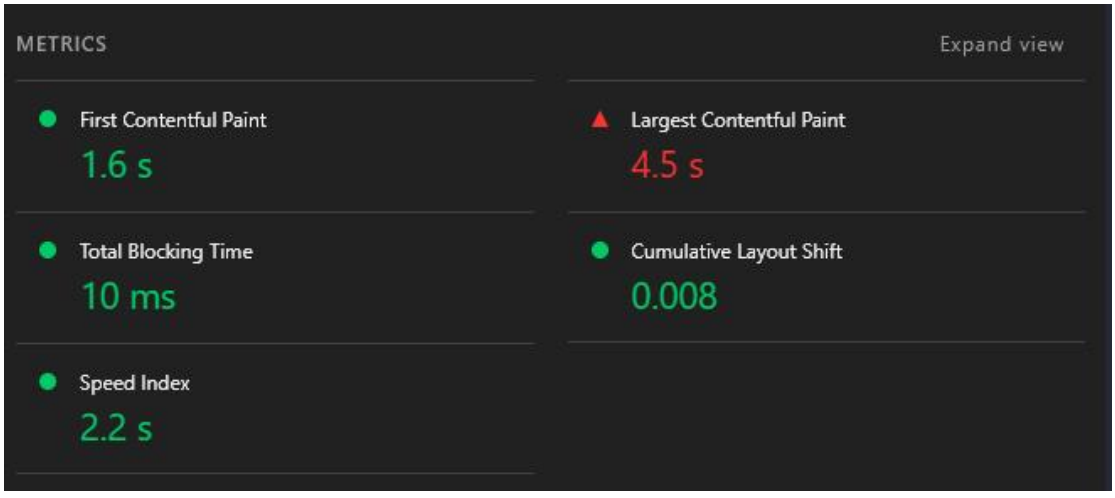
Resultados en Dispositivos Móviles (Mobile)

- **Performance (Rendimiento):** 84 / 100
- **Accessibility (Accesibilidad):** 100 / 100
- **Best Practices (Buenas Prácticas)**: 75 / 100
- **SEO**: 100 / 100



Métricas Detalladas (Mobile):

- First Contentful Paint (FCP): 1.6 s
- Largest Contentful Paint (LCP): 4.5 s
- Total Blocking Time (TBT): 10 ms
- Cumulative Layout Shift (CLS): 0.008
- Speed Index: 2.2 s



Anexo VI: Documentos anexados

- UI/UX mockups: *Guia-Diseño-UX-UI-Alejandro-Moron-Turiel.pdf*
- Documentos Pruebas: *AuroraTestSuit-Alejandro-Moron-Turiel.pdf*