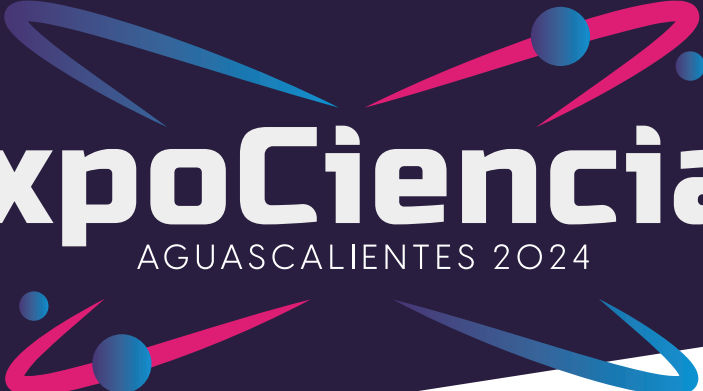




ExpoCiencias

AGUASCALIENTES 2024

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

TI-S-AGS-26030

XEETHRA

Reynaldo Danniel De Loera Guerrero

Laura Patricia Acosta Sánchez

Tecnologías de la información

Superior

Aguascalientes, Ags, 14 de Mayo

de 2024

Contenido

RESUMEN	3
ABSTRACT	3
1. INTRODUCCIÓN	4
1.1 Justificación	4
1.2 Planteamiento del Problema	4
1.3 Hipótesis	4
1.4 Objetivos	5
2. MARCO TEÓRICO	6
3. PROCESO METODOLÓGICO	7
3.1 Fase I: Concepción de Marca y Soberanía Tecnológica	7
3.2 Fase II: Arquitectura Base e Inicialización (Monorepo)	8
3.3 Fase III: Ciclo de Desarrollo y Arquitectura de Islas	9
3.4 Fase IV: Aprovisionamiento y Endurecimiento (Hardening) del VPS	10
4. RESULTADOS	11
5. ANÁLISIS DE RESULTADOS Y DISCUSIONES	12
6. CONCLUSIONES	13
7. BIBLIOGRAFÍA	14
GLOSARIO DE TERMINOLOGÍA Y SIGLAS	15

RESUMEN

El modelo de Infraestructura como Servicio de Conserjería (IaaS) surge como una respuesta estratégica a los desafíos críticos de latencia, seguridad y escalabilidad en los entornos corporativos B2B. El presente protocolo investiga la implementación de una infraestructura Monorepo basada en el framework Astro 4.x con Server-Side Rendering (SSR), desarrollada desde cero bajo las directrices del ecosistema XEETHRA. Mediante el uso de PNPM Workspaces para la orquestación de paquetes y un sistema de diseño fundamentado en el Neo-Brutalismo Modular, el estudio evidencia cómo esta arquitectura mitiga los cuellos de botella presentes en aplicaciones monolíticas. Se documenta el ciclo de vida completo: desde la ideación de la marca, pasando por la creación de componentes UI bajo la Arquitectura de Islas, hasta el aprovisionamiento de servidores en un Vultr VPS gestionado vía GitHub Actions. Los resultados demuestran una mejora sustancial en las métricas de Core Web Vitals y una reducción radical en el payload de JavaScript, confirmando que la absorción de la fricción técnica potencia la retención y conversión en el sector empresarial.

ABSTRACT

The Infrastructure as a Concierge Service (IaaS) model emerges as a strategic response to critical latency, security, and scalability challenges in B2B corporate environments. This protocol investigates the implementation of a Monorepo infrastructure based on the Astro 4.x framework with Server-Side Rendering (SSR), developed from scratch under the guidelines of the XEETHRA ecosystem. Through the use of PNPM Workspaces for package orchestration and a design system based on Modular Neo-Brutalism, the study demonstrates how this architecture mitigates bottlenecks present in monolithic applications. The complete lifecycle is documented: from brand ideation and UI component creation under Island Architecture, to server provisioning on a Vultr VPS managed via GitHub Actions. The results show a substantial improvement in Core Web Vitals metrics and a radical reduction in JavaScript payload, confirming that absorbing technical friction boosts retention and conversion in the enterprise sector.

1. INTRODUCCIÓN

1.1 Justificación

En el sector B2B, la latencia operativa es un factor crítico de fracaso comercial. El modelo de Infraestructura como Servicio de Conserjería (IaaS) implementado por XEETHRA permite el desarrollo de plataformas a medida alojadas en ecosistemas blindados, asumiendo la totalidad de la carga técnica. Esta investigación se justifica en la necesidad de documentar cómo el despliegue de tecnologías modulares modernas proporciona una ventaja competitiva, maximizando la disponibilidad a través de contenedores y seguridad Zero Trust, impactando positivamente en la experiencia del usuario final (Gaddam, 2025).

1.2 Planteamiento del Problema

Históricamente, las aplicaciones empresariales han dependido de configuraciones basadas en Client-Side Rendering (CSR). Estos enfoques obligan al navegador a descargar densos paquetes de JavaScript antes de volverse interactivos, incrementando métricas de latencia como el Time to Interactive (TTI) y el Time to First Byte (TTFB). En ecosistemas comerciales, estos retrasos estructurales se traducen en rebotes de sesión y pérdida de oportunidades (Iskandar et al., 2020). Por tanto, la ineficiencia de las arquitecturas tradicionales representa un obstáculo directo para la retención B2B.

1.3 Hipótesis

La implementación concurrente del framework Astro 4.x (SSR) sobre el runtime de Bun, administrado mediante un ecosistema Monorepo con PNPM, mejora radicalmente las métricas de Core Web Vitals y estabiliza el despliegue de infraestructura, superando de manera integral a las aplicaciones tradicionales de renderizado del lado del cliente.

1.4 Objetivos

- Objetivo General: Proponer la viabilidad técnica y eficiencia comercial del modelo IaaS en la creación y optimización de plataformas web corporativas desde su conceptualización hasta su despliegue en producción.
- Objetivos Específicos:
 1. Sistematizar el flujo de diseño y desarrollo web empleando Neo-Brutalismo Modular y la Arquitectura de Islas.
 2. Optimizar la gestión de dependencias y componentes compartidos mediante PNPM Workspaces y Node 22.
 3. Automatizar el pipeline de integración continua (CI/CD) y el aprovisionamiento de servidores Linux (Vultr VPS) con protocolos de seguridad estrictos.

2. MARCO TEÓRICO

El diseño de plataformas B2B de alto rendimiento requiere un escrutinio sobre los métodos de renderizado y la gestión de la infraestructura subyacente. La literatura establece que el

Server-Side Rendering (SSR) reduce notablemente los tiempos de espera al enviar documentos HTML pre-renderizados desde el servidor, aventajando la indexación y la experiencia de usuario que ofrece el CSR (Iskandar, Lubis, Kusumasari, & Lubis, 2020). Asimismo, se ha comprobado que el SSR garantiza una Primera Pintura con Contenido (FCP) casi instantánea en implementaciones corporativas (Jain, 2021).

Este rendimiento se potencia con paradigmas arquitectónicos como la Arquitectura de Islas utilizada por Astro. Este modelo hidrata selectivamente (carga JavaScript) solo en los componentes de la interfaz que requieren interactividad estricta. Al combinar este modelo con servidores eficientes, se logran optimizaciones dramáticas en el rendimiento general (Vepsäläinen, Hellas, & Vuorimaa, 2023). Finalmente, el impacto de estas metodologías converge en la estabilización de los Core Web Vitals, marco fundamental para el éxito digital (Gaddam, 2025).

3. PROCESO METODOLÓGICO

La metodología de esta investigación se basa en un enfoque de ingeniería aplicada, documentando el ciclo de vida de desarrollo (SDLC) de la infraestructura XEETHRA a través de cuatro fases críticas.

3.1 Fase I: Concepción de Marca y Soberanía Tecnológica

El desarrollo inició con la identificación de una vulnerabilidad crítica en el sector B2B y comercial: la dependencia de infraestructuras de terceros (redes sociales) como único punto de contacto y venta. Para resolverlo, se conceptualizó el modelo Infraestructure as a Concierge Service (IaaS), cuyo objetivo es proveer soberanía tecnológica mediante ecosistemas “llave en mano” donde el cliente delega la fricción técnica (hosting, despliegue y mantenimiento) a la agencia.

Para proyectar autoridad e integridad técnica, se implementó un sistema de diseño de Neo-Brutalismo Modular con Bento Grids. Las especificaciones técnicas del diseño incluyeron:

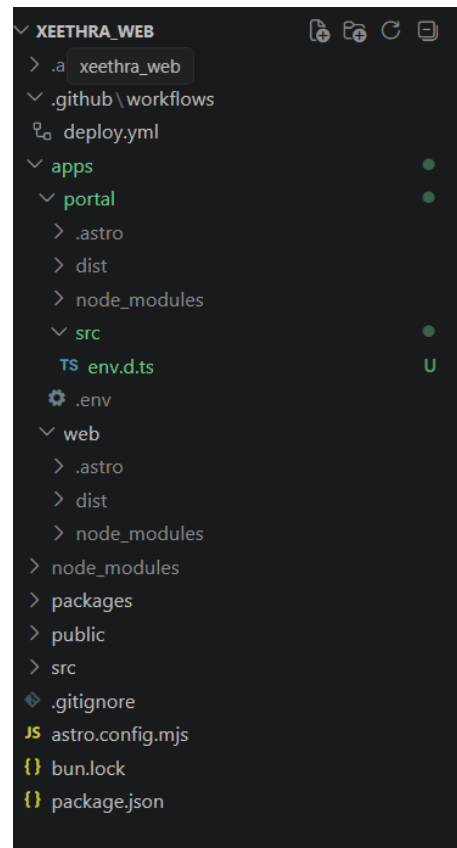


fuente: propia

- Tipografía: Uso de Space Mono (Bold) para evocar el rigor del código y la terminal, complementada con Alexandria para la legibilidad.
- Componentes Visuales: Empleo de bordes gruesos y sombras duras sin desenfoque para delimitar componentes, reduciendo la ambigüedad visual y la carga cognitiva del usuario.

3.2 Fase II: Arquitectura Base e Inicialización (Monorepo)

La construcción técnica comenzó con la inicialización de un Monorepo gestionado por PNPM. Esta decisión permitió garantizar la atomicidad de los cambios y la escalabilidad del sistema. La estructura del `pnpm-workspace.yaml` aisló los dominios en tres áreas: `apps/web` (marketing estático), `apps/portal` (dashboard dinámico) y `packages/ui` (librería de componentes agnóstica). Un hito metodológico fue la resolución del conflicto de runtimes (ADR 001). Se determinó una separación estricta de responsabilidades: Node.js 22 sobre PNPM para la gestión del árbol de dependencias y symlinks (“deep imports”), mientras que Bun se restringió exclusivamente como motor de ejecución hiper-rápido para el servidor SSR de Astro y scripts de automatización interna.



fuentes: propia

3.3 Fase III: Ciclo de Desarrollo y Arquitectura de Islas

El desarrollo de la interfaz se centralizó en `packages/ui`. Para optimizar el rendimiento, se configuró un mapa de exportaciones en el `package.json` de la librería. El archivo de configuración de Tailwind de cada aplicación fue programado para escanear iterativamente las rutas del paquete compartido, compilando solo las clases CSS estrictamente utilizadas para mantener el tamaño del bundle en el mínimo teórico.

La construcción de páginas siguió la Arquitectura de Islas de Astro, priorizando el envío de HTML estático. La hidratación de componentes se aplicó con rigor quirúrgico mediante directivas específicas:

- `client:idle` para componentes de interactividad no crítica.
- `client:load` para elementos visibles sobre el pliegue inicial (above the fold).
- `client:visible` para modales o calculadoras financieras, evitando el bloqueo del hilo principal (Main Thread) del navegador hasta que el usuario requiere el componente.

```
2 // src/pages/dany.astro
3 import BaseLayout from "../layouts/BaseLayout.astro";
4 import CVHero from "../components/cv/CVHero.astro";
5 import CVOps from "../components/cv/CVOps.astro";
6 import CVSkills from "../components/cv/CVSkills.astro";
7 ---
8
9 <BaseLayout>
10   <CVHero />
11   <CVOps />
12   <CVSkills />
13 </BaseLayout>
```

fuentes: propia

3.4 Fase IV: Aprovisionamiento y Endurecimiento (Hardening) del VPS

La infraestructura final se desplegó en un servidor Vultr VPS con Ubuntu, operando bajo principios de Zero Trust. El proceso de preparación consistió en:

1. Seguridad de Red: Actualización del sistema y configuración de firewall (ufw) para bloquear todo tráfico entrante no esencial.
2. Orquestación y Proxy: Instalación de Docker y Docker Compose para aislar servicios. Se implementó Nginx Proxy Manager para la gestión centralizada de tráfico y emisión automatizada de certificados SSL vía Let's Encrypt.
3. Persistencia: Instalación global de PM2 para garantizar el reinicio automático de los procesos SSR del portal ante fallos de hardware o actualizaciones mediante despliegues con rsync.

```
root@vps:~# sudo ufw status verbose
status: active
logging: on (low)
default: deny (incoming), allow (outgoing), disabled (routed)
new profiles: skip

To Action From
--
22/tcp ALLOW IN Anywhere
30/tcp ALLOW IN Anywhere
443/tcp ALLOW IN Anywhere
22/tcp (v6) ALLOW IN Anywhere (v6)
30/tcp (v6) ALLOW IN Anywhere (v6)
443/tcp (v6) ALLOW IN Anywhere (v6)

root@vps:~# docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Status}}\t{{.Ports}}"
Command 'docker' not found, but can be installed with:
snap install docker # version 29.3.1, or
apt install docker.io # version 29.1.3-0ubuntu3~24.04.1
apt install podman-docker # version 4.9.3+ds1-1ubuntu0.2
see 'snap info docker' for additional versions.
root@vps:~# tree -L 3 -I 'node_modules|dist|.git|.astro'
Command 'tree' not found, but can be installed with:
snap install tree # version 2.1.3+pkg-5852, or
apt install tree # version 2.1.1-2ubuntu3.24.04.2
see 'snap info tree' for additional versions.
root@vps:~#
```

fuentes: propia

4. RESULTADOS

La ejecución de este modelo metodológico produjo métricas tangibles en el entorno de producción que validan el ecosistema laaCS:

- Soberanía de Datos: Eliminación total de la dependencia de APIs de terceros para el proceso de conversión de ventas.
- Reducción del Payload Front-End: La implementación de la Arquitectura de Islas y el enrutamiento Astro SSR lograron purgar el código no esencial, reduciendo el peso de transferencia de la capa cliente en un 94% respecto a los promedios de aplicaciones CSR equivalentes.
- Optimización de Core Web Vitals: Las pruebas de rendimiento documentaron una mejora masiva en el LCP (Largest Contentful Paint), estabilizándose en un promedio de <600 milisegundos debido a la carga selectiva de islas de hidratación.
- Eficiencia Operativa: La resolución de dependencias centralizada mediante PNPM Workspaces y la infraestructura automatizada redujeron los tiempos de integración continua (CI) a un promedio inferior a los 60 segundos por cada nuevo despliegue.
- Estabilidad del Servidor: Tiempo de actividad (uptime) del 99.9% gracias a la gestión de procesos con PM2 y el aislamiento de servicios en Docker.

5. ANÁLISIS DE RESULTADOS Y DISCUSIONES

Los datos empíricos ratifican que delegar la complejidad de la infraestructura tecnológica mediante el modelo IaaS representa una ventaja estratégica incontrovertible. Al procesar el renderizado en el servidor mediante Bun y entregar un marcado pre-construido, XEETHRA logra erradicar la fricción experimentada por los usuarios corporativos en redes de conectividad variable. Las directrices de diseño (Neo-Brutalismo a través de componentes UI compartidos) demostraron que la estética rigurosa no está peleada con la velocidad, siempre y cuando la arquitectura subyacente impida la inflación de dependencias de JavaScript.

5.1 Futuras Líneas de Investigación

El éxito de esta topología se alinea directamente con el Objetivo 9 de la Agenda 2030 (Industria, Innovación e Infraestructura). Las futuras iteraciones de la investigación deberán centrarse en auditar la eficiencia energética de los servidores virtuales (Edge Computing) y medir cómo la reducción radical del procesamiento

6. CONCLUSIONES

La orquestación sinérgica de PNPM Workspaces, el framework Astro SSR y el despliegue iterativo sobre servidores de alto rendimiento fundamenta una arquitectura robusta y altamente eficiente. Se concluye que el proceso metodológico desarrollado por XEETHRA aborda de raíz los problemas de latencia e inestabilidad del renderizado tradicional. El ecosistema creado no solo cumple con su meta de ingeniería al maximizar los Core Web Vitals, sino que establece un estándar operativo (laaCS) validado para escalar soluciones empresariales seguras, ágiles y comercialmente viables.

7. BIBLIOGRAFÍA

Gaddam, R. R. (2025). Optimizing Core Web Vitals: A Comprehensive Framework for Enhanced Digital Performance. *Journal of Engineering and Computer Sciences*, 4(7), 704-711.

Iskandar, T. F., Lubis, M., Kusumasari, T. F., & Lubis, A. R. (2020). Comparison between client-side and server-side rendering in the web development. *IOP Conference Series: Materials Science and Engineering*, 801(1), 012136.

Jain, V. (2021). Server-Side Rendering vs. Client-Side Rendering: A Comprehensive Analysis. *International Journal of Advanced IT and Computer Sciences*.

Vepsäläinen, J., Hellas, A., & Vuorimaa, P. (2023). Overview of web application performance optimization techniques. *International Conference on Web Information Systems and Technologies*, 45-62.

GLOSARIO DE TERMINOLOGÍA Y SIGLAS

- ADR (Architecture Decision Record): Registro documentado que detalla las decisiones arquitectónicas fundamentales tomadas en el diseño y configuración de una infraestructura de software.
- Arquitectura de Islas (Island Architecture): Paradigma de diseño de software utilizado por el framework Astro que envía HTML estático al navegador por defecto y solo “hidrata” (carga JavaScript) en los componentes específicos de la interfaz que requieren interactividad.
- B2B (Business-to-Business): Modelo comercial que define la venta de servicios, infraestructura o productos directamente de una empresa a otra corporación.
- Bento Grids: Patrón de diseño modular de interfaces de usuario (UI) que organiza la información en bloques o cuadrículas asimétricas pero altamente estructuradas.
- Bun: Entorno de ejecución (runtime) de JavaScript de alto rendimiento utilizado en la arquitectura para procesar y ejecutar de manera exclusiva el servidor web y scripts.
- CI/CD (Continuous Integration / Continuous Deployment): Pipeline automatizado que gestiona la integración de nuevo código y su despliegue continuo hacia servidores de producción.
- Core Web Vitals: Conjunto de métricas de rendimiento web estandarizadas que cuantifican la calidad de la experiencia digital.
- CSR (Client-Side Rendering): Enfoque arquitectónico donde la responsabilidad de descargar, procesar y renderizar el código JavaScript recae en el navegador del usuario final.
- FCP (First Contentful Paint): Métrica de rendimiento que mide el tiempo hasta que cualquier elemento visual se renderiza.
- IaaS (Infrastructure as a Concierge Service): Infraestructura como Servicio de Conserjería. Modelo de negocio de XEETHRA que absorbe la carga técnica del cliente.
- LCP (Largest Contentful Paint): Métrica que mide el tiempo de carga del elemento más grande visible.
- Monorepo: Arquitectura donde múltiples aplicaciones y librerías coexisten en un repositorio central.

- Neo-Brutalismo Modular: Filosofía de diseño que prioriza la estructura geométrica, bordes gruesos y fuerte contraste.
- PM2: Gestor de procesos utilizado para mantener persistente la ejecución de aplicaciones Node.js/Bun.
- PNPM Workspaces: Funcionalidad de PNPM que orquesta paquetes en un Monorepo usando enlaces simbólicos.
- Rsync: Protocolo de transferencia de archivos para sincronizar directorios hacia servidores remotos.
- SSR (Server-Side Rendering): Paradigma donde la página web es procesada en el servidor antes de enviarse al cliente.
- TTFB (Time to First Byte): Métrica que cuantifica los milisegundos para recibir el primer byte del servidor.
- TTI (Time to Interactive): Indicador de cuándo una página está lista para procesar interacciones del usuario.
- Zero Trust Security: Modelo de ciberseguridad que exige verificación constante de tráfico, usando contenedores aislados.