Caso Web Development Specialist

Publicis Groupe

Febrero 2026

Presentado por:

Ivan Humberto Bello Sandoval

Contenido

Sección A: Gestión de Equipo y Procesos

1. **Priorización y Distribución de Solicitudes**

ID	Prioridad	Solicitud	Justificación	Asignación
B	Crítico	Bug critico checkout	Pérdida directa de ingresos. Clientes no pueden comprar.	Dev Senior 2-4 horas
C	Alto	CVE WordPress	Riesgo de seguridad. 72h de plazo.	Dev Mid + Junior 48-60 horas
A	Medio	Migración AWS	Fallas intermitentes. 2 semanas de plazo	Dev Senior 10-12 días
D	Normal	Landing page	Sin impacto crítico. 10 días disponibles.	Dev Junior 6-7 días

Tabla 1

Criterios: Se tomaron en cuenta el impacto de negocio, urgencia temporal y complejidad técnica critica, referente a los tiempos y plazos establecidos.

2. **Plan de Contingencia: Dev Senior Enfermo**

Retrasar la migración de AWS ya que contamos con un plazo de 2 semanas, el bug critico ya se resolvió el día lunes, colocar el Dev Mid a preparar una documentación o pasos a seguir para la migración de AWS para que él se familiarice en caso tal de que este escenario vuelva a suceder.

Dev Mid asume liderazgo técnico temporal. Solicitudes B/C/D no se ven afectadas ya que están en manos de Mid/Junior.

3. **Proceso de Onboarding - 30 Días.**

Semana 1 - *Fundamentos:* Setup técnico, introducción al equipo, vista previa de proyectos principales. Primer task guiado. Participación en dailies como observador.

Semana 2 - *Primeros Commits:* Tareas de complejidad baja y media, revisión de código activo y aprendizaje de deployment a staging *(local -productivo)*. Primera sesión con feedback.

Semana 3 - *Autonomía:* Tareas de complejidad media de forma más autónoma. Primer acercamiento de comunicación con clientes.

Semana 4 - *Integración:* Proyecto completo end-to-end. Primera rotación on-call con backup. Revisión de onboarding y plan de desarrollo 2-3 meses.

métricas de estudio, desarrollos y despliegues autónomos y participación activa en scrum en formulación base a los task diarios.

Sección B: Arquitectura y Desarrollo Técnico

2. Arquitectura Plataforma Farmacéutica

1. **Diagrama de Arquitectura**

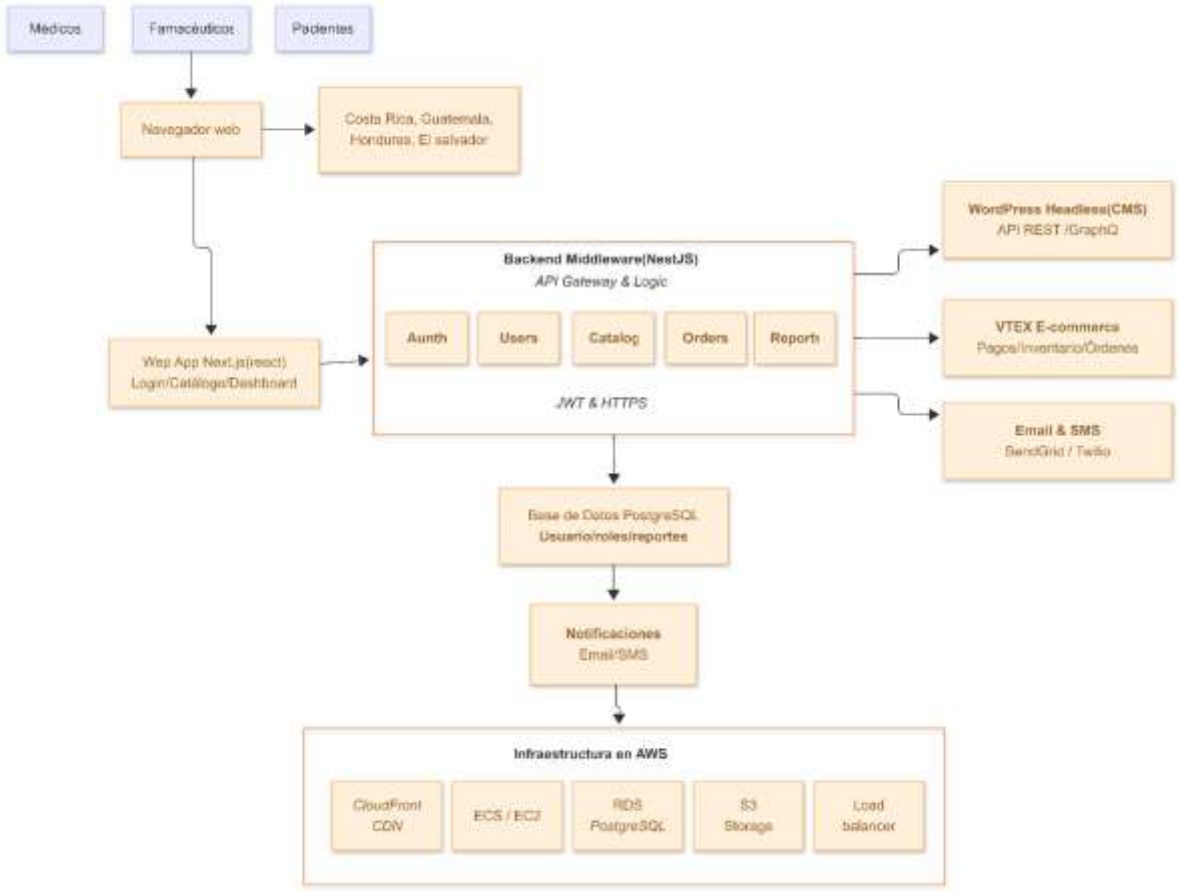

Figura 1

Link: https://mermaid.ai/view/8bf7386a-9444-461b-ab5a-8d8c54d6b45f

2. **Stack Tecnológico Recomendado**

Capa	Tecnología	Justificación
Frontend	Next.js 14 + TypeScript	SSR/SSG para SEO, es escalable y clave para contenidos de catálogo y dashboard
CMS	WordPress Headless + WPGrap	Gestiona contenidos complejos y frontend moderno
Backend	Node.js + Express	Reutilización de código TS, experiencia personal extensa y escalable
Base de Datos	PostgreSQL (AWS RDS)	Mejor para datos relacionales e ideal para usuarios, roles, órdenes y reportes
E-commerce	VTEX	Integración requerida por cliente, PCI-DSS compliant
Cloud	AWS	Familiaridad personal, servicios maduros, multi-región, balanceo y tolerancia a fallos (load -balance)
CDN	CloudFront	Integración nativa con S3, ACM SSL, Route 53 y cacheo optimizado
Cache	Redis (ElastiCache)	Sesiones, object cache, queries frecuentes
Notificaciones	SES + SNS + Twilio	Emails transaccionales y SMS a médicos/pacientes

Tabla 2

3. **Patrones de Diseño**
    1. **Backend for Frontend:** Hace llamadas **a wordpress, VTEX, y DB** exponiendo API simplificada de producto como directorio /api/producto/:id y restricciones (descripción) + VTEX (precio) + BD (restricciones de iva por pais).
    2. **Repository Pattern:** Trae acceso de datos a los repositorios importando **UserRepository, OrderRepository** donde se facilita el testing y cambios de BD.
    3. **API Gateway Pattern:** Centraliza autenticación, autorización y logging y reduce el acoplamiento de servicios en este caso VTEX Y Wordpress.

4. **Puntos Críticos de Seguridad**
    1. **Inyecccion:** Propagacion de codigo malicioso en XSS. CSP headers, exponiendo configuraciones.
    2. **Autenticación:** Login seguros com MFA y RECAPCHAT para evitar secciones contiguas o backdoors.
    3. **Encriptar:** Encriptar archivos sensibles como .env y archivos presentes en wp como es wp_config y tokens expuestos en directorios publicos.
    4. **DDoS:** AWS WAF en CloudFront. Rate limiting por IP/usuario. Auto-scalin.
    5. **Token:** Implementar tokens para no almacenar datios de tarjetas en las pasarelas de pago.
    6. **Regulación Multisitio:** Tabla de restricciones por país/producto. Validación en checkout. Workflow de recetas médicas con verificación.

Sección C: Infraestructura Cloud y Servidores

4. Migración WordPress Nutresa a AWS

1. **¿AWS o GCP? Justifica y lista servicios especificos a usar**
   AWS ya que reduce el riesgo de migracion critica y tiene mejor soporte para cms como Wordpress, integrado con S3/ACM/Route 53.

   Servicios AWS: EC2 t3.medium (~$30), RDS MySQL Multi-AZ (~$50), S3 (~$3), CloudFront (~$15-20), ElastiCache Redis (~$15), Route 53 (~$1), ACM (gratis), EBS (~$3). Total: $165-185/mes

   Servicios AWS:

- EC2 ejemplo de instancia: t3.medium
- RDS MySQL (Multi-AZ)
- S3
- CloudFront
- Route 53 CDN

- EBS

2. **Describe el proceso de migración (minimo 6 pasos clave).**

    1. Identificar la estructura de aplicativo WordPress version, plugins, PHP, dependencias externas
    2. Arquitectura adquirida VPC, RDS Multi-AZ, EC2 LEMP stack, S3, ElastiCache, CloudFront
    3. Importar BD a RDS y ZIP del sitio a S3, instalar WP ALL MIGRATIONS (opcional) copiar a un bucket y reutilizar la misma arquitectura, probar en staging
    4. Configuracion del nuevo entorno, apache o nginx
    5. Reubicacion del sitio desde el S3
    6. Configuraciones pertinentes al DB.

3. **Estrategia para cero downtime y plan de rollback**

    Downtime Deployment: Servidor actual (inicial) activo durante preparación de AWS (final). Switch via DNS.

    TTL Bajo: 300s configurado 48h antes. Propagación máxima 5 minutos. Modo Lectura: < 15 min durante sync final. Usuarios ven contenido, no pueden postear .

    Plan de Rollback: Revertir DNS a IP vieja (5 min). Servidor viejo activo como backup. Ventana de decisión: primeras 2 horas. Criterios: error rate > 5%, performance degradada, funcionalidad rota.

4. **Proporciona configuración básica de Nginx como reverse proxy con SSL (Let's Encrypt).**

    Configuración completa de Nginx con SSL Let's Encrypt en archivo nutresa.conf del repositorio.

5. **Si el sitio responde error 502 post-migración, describe tu proceso de diagnóstico (5 pasos)**

    Diagnóstico Error 502:

Paso 1: Verificar PHP-FPM corriendo: systemctl status php8.1-fpm.
Paso 2: Verificar socket/puerto correcto en Nginx y PHP-FPM configs.
Paso 3: Revisar logs de Nginx (error.log) buscando connection failed.
Paso 4: Verificar límites PHP-FPM (pm.max_children), RAM, CPU.
Paso 5: Test directo PHP (info.php), revisar wp-config.php DB_HOST, verificar security groups RDS, enable WP_DEBUG.

Sección D: CMS y E-commerce

5.1. Diagnóstico WordPress Lento (12s TTFB)

1. **Describe proceso de diagnóstico (herramientas a usar).**

   Herramientas de diagnóstico: GTmetrix, PageSpeed Insights, Query Monitor plugin, Chrome DevTools.

   Optimizaciones por Impacto:

   1. Implementar Cache (IMPACTO ALTO): WP Rocket o W3 Total Cache. Page + object + browser cache. Reducción TTFB de 12s a 0.5-1s. Costo: $0-50/año.

   2. Reducir Plugins (IMPACTO ALTO): De 35 a <15 plugins esenciales. Usar Query Monitor para identificar queries lentas. Reducción 50-70% en queries.

   3. Optimizar Imágenes (IMPACTO MEDIO): Imagify/ShortPixel batch compression. Lazy loading nativo. WebP format. Página de 8MB baja a 2-3MB.
   4. Optimizar BD (IMPACTO MEDIO): WP-Optimize para limpiar revisions, spam, transients. **OPTIMIZE TABLE.** Agregar índices a queries lentas.

5 .Upgrade Hosting (IMPACTO ALTO si budget permite): Migrar a 4GB+ RAM con CPUs dedicados. Hosting compartido 2GB es muy limitado.

¿Qué harias primero con presupuesto limitado?

Presupuesto Limitado*:* Optimizo la utilización de plugins pagos, memoria e imágenes que estén consumiendo recursos innecesarios.

5.2. Comparativa E-commerce

	VTEX	Shopify	WooCommerce
Costos	$1,500-5,000/mes Alto inicial	29-299/mes + apps Medio predecible	Gratis core $500-2,000/mes hosting
Escalabilidad	Muy alta Enterprise-grade	Alta Shopify Plus disponible	Media Depende de hosting
Personalización	Alta via APIs Headless nativo	Media Templates + Liquid	Muy alta PHP + hooks infinitos
Multi-país	Excelente Multi-moneda nativo	Bueno Con Shopify Markets	Básico Requiere plugins

Recomendación para farmacéutica (50 SKUs, regulaciones complejas):
VTEX. Justificación: Regulaciones farmacéuticas requieren workflows complejos (verificación de recetas, restricciones por país, trazabilidad).

VTEX maneja un acoplamiento mejor con APIs robustas para integraciones custom. Multi-país nativo esencial para Costa Rica, Guatemala, Honduras, El Salvador. Aunque más caro, vale la pena para dominio tan regulado.

WooCommerce requeriría mucho desarrollo custom aunque su ventaja es que es que no pagas más referente a VTEX si el presupuesto es limitado esta sería la mejor opción .

Shopify no está optimizado para regulaciones complejas.

Sección E: Comunicación y y Gestión de Stakeholders

Redeccion de email:

Estimado/a [Nombre del cliente]:

Gracias por contactarnos. Entiendo la urgencia de mejorar tanto la **velocidad** como la **seguridad** de su sitio web, ya que ambos factores impactan directamente en la experiencia del usuario y en los resultados del negocio.

Para poder brindarle una **cotización precisa** y un **timeline realista**, agradecería contar con la siguiente información:

- ¿Qué funcionalidades específicas presentan lentitud? (páginas principales, checkout, búsqueda, panel de usuario, etc.)
- ¿Han identificado algún patrón? (horarios específicos, picos de tráfico, campañas recientes).
- ¿Cuál es la plataforma actual? (WordPress, desarrollo a medida, tipo de hosting: compartido, VPS, cloud).
- En cuanto a seguridad, ¿han tenido incidentes recientes o se trata de una medida preventiva?

Plan de acción propuesto

Fase 1 Auditoría técnica (3 a 5 días)
Análisis completo de performance y seguridad para identificar cuellos de botella, vulnerabilidades y oportunidades de mejora.

Fase 2 Optimizaciones rápidas (1 a 2 semanas)
Implementación de mejoras de alto impacto (“quick wins”): cacheo, CDN, optimización de imágenes y recursos.

Fase 3 Mejoras estructurales y seguridad (2 a 3 semanas)
Configuración de firewall, SSL avanzado, backups automáticos, monitoreo y ajustes de arquitectura para estabilidad a largo plazo.

Sobre el timeline de 1 semana

Es posible lograr **mejoras visibles en velocidad en 5–7 días** mediante optimizaciones básicas (cache y CDN). Sin embargo, una solución completa,

estable y segura requiere **3 a 4 semanas** para garantizar resultados sostenibles.

¿Prefieren priorizar mejoras inmediatas o avanzar con una solución integral a largo plazo?

Quedo atento para coordinar una **llamada de 30 minutos esta semana** y revisar los detalles.

Saludos cordiales,
Iván Bello

Publicis Groupe
Web Development Specialist

Parte B - Respuesta al CFO:

Entiendo su preocupación respecto al costo de la infraestructura actual. Si bien el hosting actual es más económico, presenta limitaciones que impactan directamente en el negocio.

Un **tiempo de carga elevado** afecta la conversión y la retención de usuarios: estudios indican que hasta el **40% de los usuarios abandona un sitio** si tarda más de lo esperado en cargar, lo que se traduce en pérdida directa de ventas.

La nueva infraestructura propuesta incluye:

- Servidores dedicados con **99.9% de uptime**
- **CDN global** para mejorar tiempos de carga
- **Backups automáticos diarios**
- Seguridad de nivel enterprise contra ataques y caídas

Desde una perspectiva de **retorno de inversión (ROI)**, reducir el tiempo de carga de **8 segundos a 2 segundos** puede aumentar la tasa de conversión entre **20% y 35%**, según estudios de Google.
Con el volumen de tráfico actual, esto representaría aproximadamente **$X,XXX adicionales en ingresos mensuales**, permitiendo que la inversión se recupere en un plazo estimado de **2 a 3 meses**.

Estrategia de comunicación recomendada

Dado que el enfoque del CFO es financiero, se sugiere presentar:

- Comparativa **costo vs. beneficio** (Excel)
- Casos de éxito de empresas similares
- Proyección de ingresos con y sin mejora de performance
- Costo estimado del downtime (pérdida por hora de sitio inactivo)

Información adicional a preparar

- Benchmark de competidores (nivel de inversión tecnológica)
- Análisis actual de tráfico y conversión
- Proyección financiera a 12 meses con y sin upgrade
- Referencias de clientes satisfechos

Quedo atento para ampliar cualquiera de estos puntos.

Saludos cordiales,
Iván Bello

Publicis Groupe
Web Development Specialist

Sección F: Liderazgo y Gestión de Equipo

Plan de acción paso a paso

Paso 1 Conversación 1:1 privada y directa (Semana 1)

- Preparar datos objetivos: ejemplos específicos de tareas retrasadas con fechas.
- Conversación amigale pero clara: **"***He notado que los últimos 3 sprints has entregado con retraso. ¿Hay algo que esté afectando tu trabajo?"*

- Escuchar activamente: pueden existir problemas personales, técnicos o de proceso.
- Establecer expectativas claras: *“Necesito que las tareas se entreguen a tiempo y con calidad. ¿Qué necesitas de mí?”*
- Crear un plan de mejora, con deadlines específicos.

Paso 2 Monitoreo cercano y soporte (Semanas 2–4)

- Check-ins diarios breves en post-daily.
- Seciones de progranació para entender dónde está lo técnico.
- Asignar tareas más pequeñas para crear momentum de éxito.
- Ofrecer recursos: capacitación, mentorias y herramientas.

Paso 3 Evaluación y ajuste (Semana 4)

- Segunda conversación 1:1 formal.
- Revisar progreso contra el plan establecido.
- Si hay mejora: reconocer avances y continuar con el soporte.
- Si no hay mejora: conversación más seria sobre consecuencias.

Paso 4 Decisión y acción (Semanas 6–8)

- Si continúa sin mejora: iniciar PIP (Plan de mejora del desempeño) formal con RRHH.
- Si hay mejora parcial: extender el plan y re-evaluar.
- Si hay mejora total: cerrar el plan exitosamente.

Documentación

- **Email posterior a cada conversación, resumiendo acuerdos y plazos.**
- **Registro en herramienta con correo:**
    - Fecha
    - Tema discutido
    - Compromisos

    - Próximos pasos
- Screenshots o evidencia de tareas retrasadas.
- Notas de cada check-in semanal.

Importante: Nunca tomar acciones sin documentación escrita.

Cuándo escalar a RRHH

- Después de 2 conversaciones formales *(4–6 semanas)* sin mejora significativa.
- Si aparecen problemas de conducta (actitud negativa, falta de respeto).
- Si hay sospecha de problemas personales graves (salud mental, adicciones) que requieran soporte profesional.
- Antes de cualquier decisión de terminar contrato (RRHH debe estar involucrado desde el inicio del PIP).

Protección de la operación

Paralelamente al proceso de mejora:

- **Documentar el sistema legacy:**
    - Pair programming sessions grabadas
    - Wiki detallada
    - Diagramas de arquitectura
- Cross-training: un Dev Mid comienza a familiarizarse con el sistema legacy (shadowing en tasks).
- Mitigación del *bus factor*: cada sesión de conocimiento debe quedar documentada.
- Si la situación se vuelve crítica: contratar un freelancer part-time como backup mientras se resuelve la situación.

Nunca dejar que conocimiento crítico esté en una sola persona sin respaldo.

Sección G: Pensamiento Crítico

Recomedacion : Mantenter y migrar de ser necesario

Mantener la arquitedtura monolitica wordpress pero migrar gradualemnte sin alterar el trafico y el sentido critico de cada sitio CMS eadless + JAMstack.

Esto sugiere un costo Migrar 60 sitios a JAMstack = 6-12 meses de desarrollo full-time ($300K-600K).

Mantener WordPress = mantenimiento actual $50K/año.

Enfoque gradual = $100K año 1 (5-10 sitios), $150K año 2, escalable según ROI.

El riego que supone estas migraciones es alto ya que si falla alguno sin conocer la arquitectura completo podemos entrar a eventos críticos y el sitio podría sufrir alteraciones en su funcionamiento.

Punto importante para atender es con que equipo disponemos ya que hay que contar y que skills cuenta con cada cargo, en este caso infraestructura.

No saltar a "lo nuevo" por estar de moda. Evaluar con pilotos pequeños, medir ROI real, y escalar solo si comprobamos valor. Mantener lo que funciona (WordPress) mientras innovamos donde tiene sentido

“Recordar que estos costos son estimados y no suponen un entorno real exacto.”

Fin del documento. El código funcional y configuraciones detalladas están disponibles en el repositorio Git adjunto :

https://github.com/IvanBlunar/Git_Caso_Web_Development_Specialist