

Entregable 1 - 26 de octubre último commit - 5%

Métricas de los requerimientos no funcionales

Documentar cada atributo no funcional con valores cuantitativos, parámetros técnicos y tecnologías a usar. Incluir ejemplos claros, abajo se documentan ejemplos de valores y tecnologías, pero ustedes deben diseñar y encontrar sus propios valores los cuales deben estar justificados por alguna teoría o práctica debidamente argumentada y analizada; realizando las extrapolaciones o cálculos que usted pueda diseñar bajo un algoritmo o método cuantitativo diseñado por el grupo mismo.

Integrantes: kskskss kskskss ksksksk

Requerimientos No Funcionales (Utilizar estos para desarrollar el Entregable 1)

Rendimiento

- El tiempo promedio de respuesta del portal web no debe exceder **2.5 segundos** en operaciones estándar.
- Las consultas cacheadas mediante Redis deben entregar resultados en menos de **400 milisegundos**.
- Los procesos de generación automática de contenido o campañas deben ejecutarse en menos de **7 segundos** para solicitudes simples y menos de **20 segundos** para ejecuciones complejas con IA.

Escalabilidad

- La arquitectura debe soportar un incremento de **hasta 10 veces** la carga base sin degradación perceptible del rendimiento.
- Kubernetes debe permitir **autoescalado horizontal** basado en CPU, memoria y número de solicitudes concurrentes.
- El sistema debe manejar simultáneamente **más de 5000 campañas activas** y **más de 300 agentes o usuarios concurrentes** distribuidos entre las subempresas.
- Las API de cualquiera de las plataformas podría llegar a alcanzar en el primer año hasta 100000 operaciones de usuario por minuto que requieren acceso a la base de datos, en horario habitual de 7am hasta 7pm. Y hasta 300 procesos no supervisados por minuto ejecutándose en background fuera de horario de oficina.

Tolerancia a Fallos

- Disponibilidad mínima del sistema de **99.9% mensual**.
- Los contenedores críticos deben reiniciarse automáticamente ante fallos o degradación de servicio.
- Redis y las bases de datos deben contar con **replicación en tiempo real** y **failover automático**.
- Implementación de **backups automáticos diarios** y retención mínima de **30 días**.

Seguridad

- Autenticación y autorización mediante **OAuth 2.0** para todos los usuarios y servicios.
- Cifrado de todas las comunicaciones entre servicios con **TLS 1.3**.
- Cifrado de datos en reposo con **AES-256** en bases de datos y almacenamiento persistente.
- Auditoría y logging centralizado con retención de **90 días**.

- Cumplimiento con estándares de protección de datos internacionales (**GDPR, CCPA**) y políticas de acceso mínimo necesario (principio de least privilege).

Hay que diseñar estrategias para cumplir con los requerimientos no funcionales, como evidencia de que la estrategia diseñada cumple los requerimientos no funcionales hay que adjuntar algún benchmark hecho a esa tecnología. Los resultados de estos benchmarks hay que escalarlos a nuestra implementación.

Performance

- Tiempo máximo permitido para una consulta estándar: 1.5 segundos.
- Tiempo máximo para resultados cacheados: 200 ms usando Redis.
- Tecnología: PostgreSQL, Redis; aquí pueden ser métricas por las tecnologías clave por separado.

¿Que es lo que más hacen los usuario? La operación más intensa es kskksksksk... Esto para cada sub empresa.

PromptAds

Pivote: el tiempo que toma un read en DB.

Ejemplo: Performance (Supuesto) Fast + API + Python + SQL Server + REST service 1000 transacciones duran 1400ms (Fuente: "link"). Cuando se obtiene la info sumariada de una campaña. Este sería un ejemplo de tiempo para PromptAds.

Si para 1000 transacciones son 1400ms entonces para 100 serían XXX.

El link debe decir las especificaciones de la máquina en la que se hizo. RAM, cores y velocidad de CPU.

Revizar los benchmark de PCMagazine.

Scalability

- Debe soportar incremento de carga de hasta 10x sin degradación.
- Kubernetes configurado con autoescalado horizontal por CPU y memoria. Justificarlo con la configuracion de K8s.

Ya sabemos cuanto hardware ocupamos para cumplir con performance, entonces...

Ejemplo: Se utilizara kubernetes para contenerización dinámica con un cluster de mínimo X servers con autoscale de tanto. Cada máquina tendrá XXX specs. Aquí está el link al archivo de configuración de kubernetes.

Reliability

- Tasa de errores máxima permitida: 0.1% de transacciones por día.
- Monitoreo con pg_stat_statements y logs centralizados.

- Es importante determinar como se monitorea y como se notifican alertas.

"Nivel de confianza en el sistema"

Para este punto ya debemos de saber en qué lo vamos a hacer.

¿En el cloud que escogí qué hay para monitorear alertas?

"Se va a usar el servicio XXX de XXX para el monitoreo de alertas críticas. Estas se van a comunicar por XXX, XXX, XXX otras por email. Se va a permitir un máximo de XXX (inventado) errores al día."

Availability

- Disponibilidad mínima: 99.9% mensual. En el diseño de infraestructura debe lograr verse como se logra esto, podría ir en el diagrama de arquitectura, pero sería mejor uno de infraestructura.
- Redis y bases de datos con failover y replicación.
- Considere load balancers.

Esto se refiere al porcentaje del tiempo que el sistema debería de estar disponible. 98% o 99,7% etc. Existe un ejemplo de esto en el Excel.

Security

- Autenticación: OAuth 2.0 y/o OpenID Connect.
- Cifrado TLS 1.3 en comunicación y AES-256 en reposo.

¿Cómo voy a hacer el cifrado y la autenticación?

Primero decidir si va a ser OAuth o OpenID Connect

REST con SSL

Para cada base de datos averiguo qué cifrado ofrece y selecciono la opción que más me parezca. Mongo utiliza XXX cifrado. SQL Server XXX cifrado...

Maintainability

- Código modular siguiendo DDD y separación de dominios.
- Documentación en readme.md y comentarios claros en código.

¿Cómo le voy a dar mantenimiento al sistema durante y después?

¿Durante el desarrollo? Sistema de tickets, investigar sobre GitFlows, PRs, branching, release process ("van a utilizar branches conversión promptv{") habrá un release cada 3 semanas, procedimiento de hotfixes.

¿Después de desarrollo? Buscar niveles de soporte L1, L2 y L3. No queremos tener problemas en L3. L1: Se le ofrece al usuario manuales y un RAG o bot en WhatsApp que le sirva como asistente a los usuarios. L2: Se le ofrece la disponibilidad al usuario por email para que el equipo de support le brinde apoyo. L3: Se ofrece medio y tiempo. Un encargado del equipo técnico se conectará con el usuario para brindarle apoyo.

Interoperability

- Integración de APIs REST y MCP servers entre subempresas y servicios externos.

Compliance

- Cumplimiento de GDPR en gestión de datos sensibles.

"Todo pago o transferencia se hace con servicio de terceros (paypal, ...)" Owasp para pruebas de seguridad web (vulnerabilidades). "Se aceptarán 0 pruebas críticas, XX leves, XX comunes, etc."

Extensibility

En futuras versiones el alcance de esta arquitectura se puede extender conectando REST APIs, creando nuevos MCP Servers o incluso acoplando nuevos dominios.

Domain driven design

PromptContent

Este dominio se encarga de la creación de contenido textual, audiovisual e imágenes para campañas publicitarias en redes sociales y páginas web. Este contenido es creado de manera automatizada por IA para distintos públicos meta. El dominio PromptAds utiliza este contenido para diseñar y crear campañas.

Integración con plataformas externas como Canva, Adobe, Meta Business Suite y OpenAI API, etc

Domain list

- Text Generation
- Image Generation
- Video Generation
- Campaign generation
- Culture campaign generation
- Cloud
- Creative Briefing
- SEO & Adaptation
- External Publishing

PromptAds

Este dominio se encarga del diseño, segmentación y publicación de campañas publicitarias en redes sociales, email marketing, SMS, LinkedIn e influencers. También se encarga del análisis en tiempo real del rendimiento de estas campañas. Las campañas son generadas de manera automática a partir de datos de públicos meta y objetivos de venta.

Integración con plataformas externas como Google Ads, Meta Ads, TikTok for Business, Mailchimp y plataformas de CRM.

Domain list


- Campaign Planning: Encargado de diseñar la campaña publicitaria a partir del objetivo definido por el cliente. Define los KPIs, canales, formatos y duración de la campaña.
- Targeting: Selecciona el público objetivo al que se mostrará la campaña, segmentando por datos demográficos, comportamiento, intereses y otras variables.

- **Bidding and Budgeting:** Establece las reglas de puja y presupuesto. Define cuánto se invertirá por canal o audiencia y cómo se distribuirá la oferta en cada plataforma.
- **Activation Manager:** Se encarga de activar y publicar la campaña en las plataformas externas (Google Ads, Meta Ads, TikTok, etc.) utilizando listas de control de acceso (ACL) y herramientas de integración.
- **Insights:** Recopila y presenta métricas clave de desempeño como clics, impresiones, CTR, tasa de conversión y otros indicadores de efectividad en tiempo real.
- **Attribution:** Asigna crédito de conversión a los distintos canales o interacciones que contribuyeron al cierre de una venta, permitiendo mejorar futuras campañas.
- **Experimentation:** Permite probar variaciones de campañas, audiencias o mensajes mediante pruebas A/B u otros métodos para optimizar resultados.
- **Compliance:** Supervisa el cumplimiento de regulaciones locales e internas. Aplica reglas por país o cliente, impone límites de gasto y registra cambios con trazabilidad (quién, qué, cuándo).

PromptCrm

Este dominio se encarga principalmente del monitoreo de leads. Estos se registran y clasifican automáticamente con sus respectivos datos de proveniencia. Para la interacción con el usuario el dominio ofrece chatbots, voicebots y flujos automatizados de atención. Todo esto es proporcionado por la implementación de IA.

Integración con plataformas como HubSpot, Salesforce, Zendesk, WhatsApp Business API, entre otras

 DDD diagram