

Final - Cartagram

Hecho por: Centurión, Franco; Idañez, Lucía; Lamas, Chabela

[Final DDS 20220826 - CartaGram .pdf](#)

[justificaciones.pdf](#)

Dudas

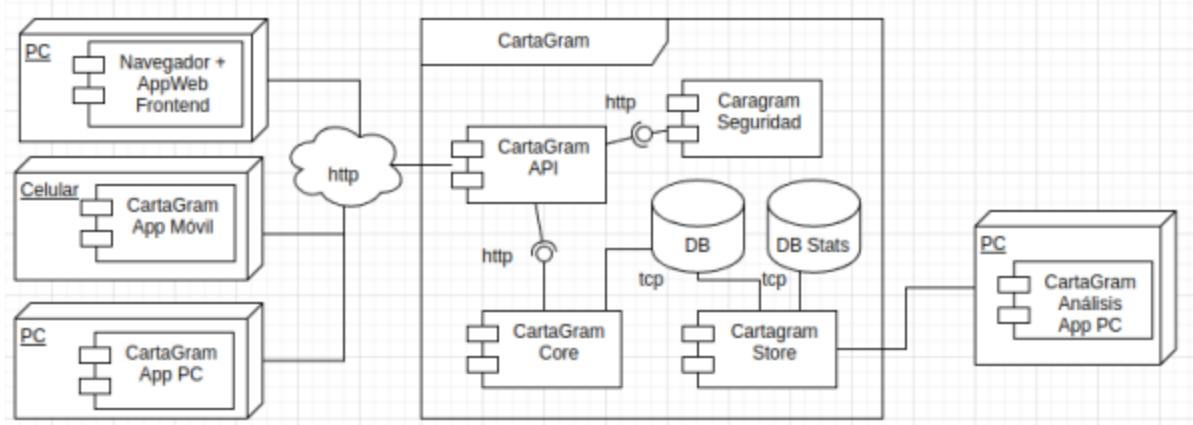


Notas

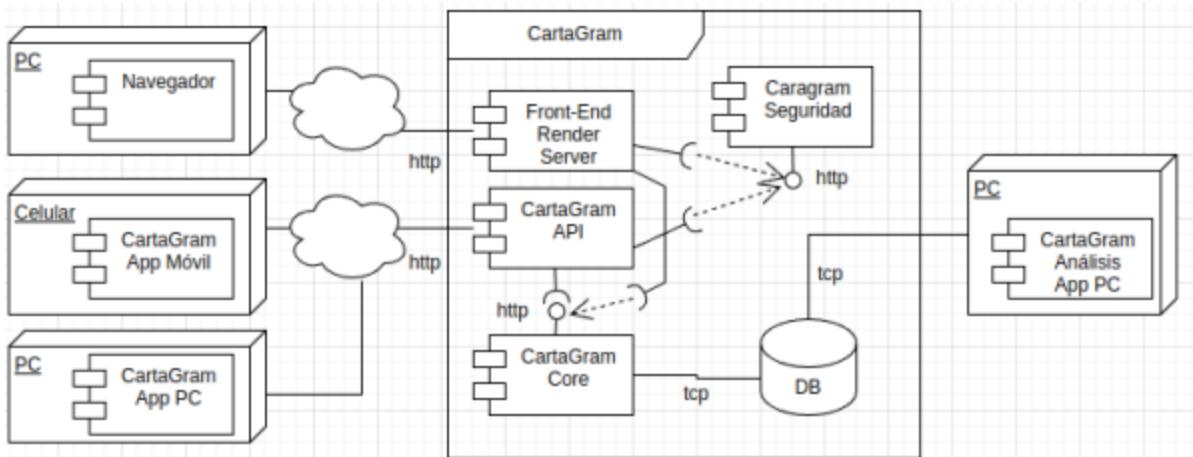
- Servidor Web+App Web+Frontend = cliente pesado

Arquitectura

Alternativa A



Alternativa B



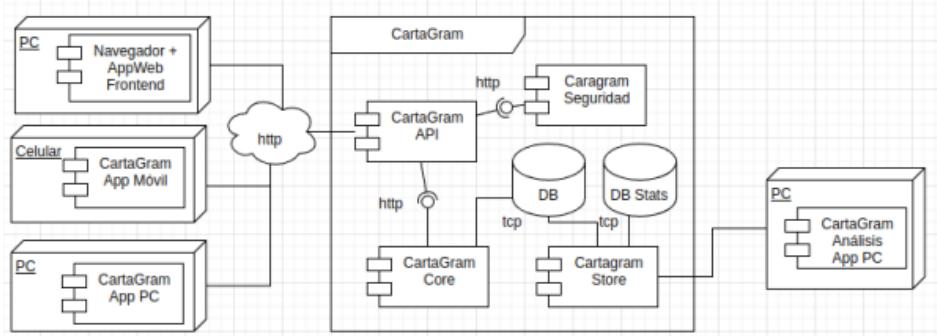
- CartaGram Core es nuestro componente, el que implementa las reglas de negocio del contexto.
- Dentro de él se ejecutan los bots.
- CartaGram API es la API REST que es apuntada por componentes que se ejecutan en clientes externos.
- CartaGram Seguridad y CartaGram Core exponen una API REST para comunicarse con los otros componentes que se ejecutan en la infraestructura de la empresa.

- FrontEnd Render Server es una aplicación web cuyo motor de templates se ejecuta en el servidor.
- CartaGram Seguridad autentica y autoriza mensajes, verifica políticas, recupera contraseñas, etc..
- CartaGram Store toma los datos de operatoria diaria (DB) y los coloca en una base NoSQL documental para agilizar los análisis de datos
- CartaGram Análisis es una App utilizada por los miembros de la empresa para analizar los datos de uso y generar estadísticas, modelos, etc...

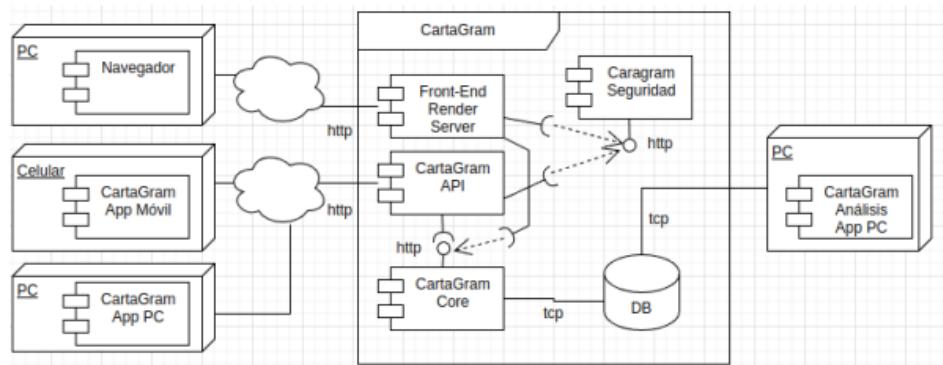
1. Especifique los métodos que considere más importantes del componente “CartaGram API”. Utilice el nivel de detalle que considere adecuado y haga explícitas todas sus suposiciones.

2. Explique las diferencias en la facilidad de mantenimiento de las alternativas A y B

Alternativa A



Alternativa B



(1.2) Diferencias de facilidad de mantenimiento entre Alternativa A y B.

1. Alternativa A

1. *CartaGram API* expone una única interfaz para las 3 terminales de cliente: Desktop, Web y Mobile.
2. *Cartagram API* utiliza interfaces expuestas desde *CartaGram Core* y *CartaGram Seguridad*, sólo 2 dependencias.
3. Un cambio en las interfaces de *Cartagram Core* o *CartaGram Seguridad* impacta sólo en 1 componente: *CartaGram API*.
4. Existe una DB NO-SQL (*DB Stats*) utilizada por el componente *CartaGram Análisis App PC* que resulta útil para no sobrecargar la DB operativa con operaciones de lectura y análisis.

2. Alternativa B

1. Expone 2 interfaces diferentes para los 3 terminales cliente (Desktop, Web y Mobile): *Front-end Render Server* y *CartaGram API*.
 2. Ambos componentes a su vez dependen de otras dos interfaces expuestas por *CartaGram Seguridad* y *CartaGram Core*.
 3. Un cambio en las interfaces expuestas por *CartaGram Seguridad* o *CartaGram Core* afectaría como mínimo 2 componentes.
 4. El componente *CartaGram Análisis App PC* se conecta directamente con la DB operativa: esto puede llegar a producir problemas de performance al estar compitiendo contra las peticiones de los usuarios finales de la aplicación.
- Considerando estos aspectos concluyo que la *Alternativa A* es más sencilla de mantener.

3. Dado el siguiente escenario, indicar el atributo de calidad que considere que está implicado en el problema detallado, definirlo y comparar cómo impactaría la elección de las alternativas A o B.

| | |
|-----------|--|
| Estímulo | CartaGram Análisis realiza una consulta que procesa un gran volumen de datos |
| Ambiente | Producción |
| Respuesta | El servicio de mensajería no sufre ninguna degradación en su nivel de servicio |

RTA:

Entiendo que el escenario planteado en la consigna está relacionando con estos atributos de calidad.

1. Mantenibilidad: capacidad que tiene un producto de software de ser modificado evolutivamente con correcciones.
 1. Analizabilidad: facilidad con la que se puede evaluar un cambio en el software, diagnosticar deficiencias, fallos, etc.
2. Fiabilidad: capacidad de desempeñar las funciones definidas bajo condiciones y tiempos acotados.
 1. Disponibilidad: capacidad de estar operativo cuando se lo requiere.
3. Eficiencia de desempeño: desempeño relativo a la cantidad de recursos usados bajo ciertas condiciones
 1. Capacidad: grado en el que los límites máximos del sistema cumplen con los requisitos.

Comparativa de Alternativa A y B bajo estos atributos.

1. Mantenibilidad (Analizabilidad)
 1. Alternativa A y B: ambas cumplen con este atributo de calidad dado que ambas alternativas poseen un componente dedicado al análisis de datos de la aplicación. Esto permite tomar medidas correctivas más fácilmente.
2. Fiabilidad (Disponibilidad)
 1. Alternativa A: al contar con una base de datos NO-SQL dedicada exclusivamente a la información estadística de la aplicación, todas las operaciones de análisis se pueden hacer sin

10 / 16

justificaciones.md

2023-11-13

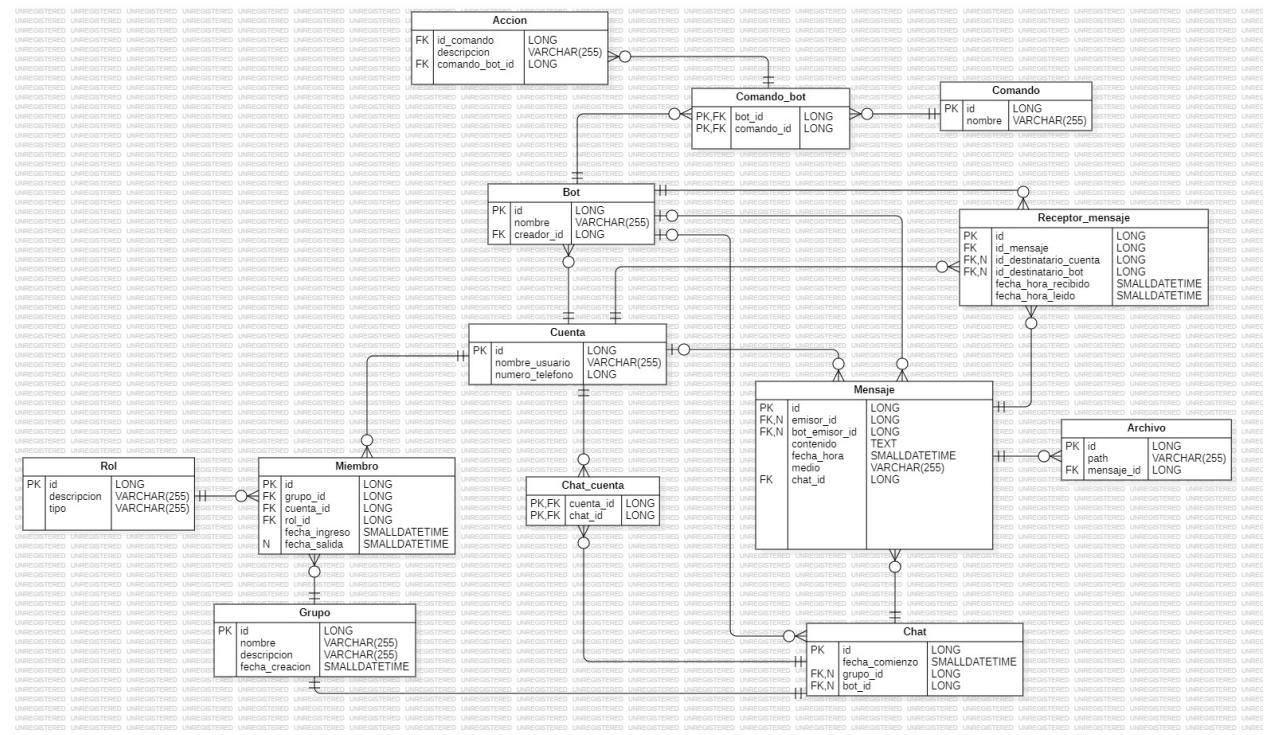
- afectar las operaciones de los usuarios finales ya que el análisis no compite contra las peticiones de los usuarios finales.
2. Alternativa B: al no contar con una base de datos dedicada y al utilizar la misma base de datos que los usuarios finales, las operaciones de análisis pueden llegar a competir también contra las peticiones de los usuarios finales. Esto puede generar problemas de disponibilidad.
 3. Eficiencia de desempeño (Capacidad)
 1. Alternativa A: como tiene una base de datos NO-SQL dedicada al análisis, esta alternativa resulta más resistente a la degradación de rendimiento.
 2. Alternativa B: como la base de datos que utiliza es la misma que la de los usuarios finales, el análisis de datos puede llegar a degradar el rendimiento de los usuarios finales.

Persistencia

Diseñar el modelo de datos del punto anterior para poder persistir en una base de datos relacional, indicando las entidades con sus respectivos campos, claves primarias, las foráneas, cardinalidad, modalidad y las restricciones según corresponda.

Justificar:

- Qué elementos del modelo es necesario persistir.
- Cómo resolvió los impedance mismatches.
- Las estructuras de datos que deban ser desnormalizadas, si corresponde.



Decisión de diseño

- Por cuestiones de trazabilidad, decidimos guardar en el mensaje el medio por el cual fue enviado. Los medios los definimos en un ENUM y los embebemos en la entidad Mensaje
- Debido a que por cuestiones de extensibilidad, los tipoRol son un enum. Decidimos embeber esta clase en la tabla de Rol. Los roles que consideramos a priori son ADMINISTRADOR y NORMAL. No consideramos esquema de permisos ya que a priori no se menciona especificidad alguna.
- Los roles solo aplican a las cuentas cuando ingresan a un grupo. Por esto es que pusimos el rol_id en el Miembro.
- No realizamos una tabla de permisos ya que en este dominio sería realizar estructuras extras
- Consideramos que los comandos de los chatbots son únicos y propios de cada uno
 - Las acciones que tienen los comandos las modelamos como referencia a clase, dado que en nuestro mundo de objetos las acciones serían stateless.
- Con respecto a **Comando**, nosotros habíamos pensando que ya estaban precargados/predeterminados por un admin, entonces sería como que yo puedo ir seleccionado comandos para mi bot que únicamente sean los precargados anteriormente y, a su vez, cada comando puede ejecutar una o varias acciones (Aclaro que tranquilamente las acciones podrían ser una lista de strings mapeados, lo cual sería como un ElementCollection, sin embargo tuvimos en cuenta que como puede haber una gran cantidad de comandos tal vez sea mejor darles su propia tabla)