Guía de ejercicios de Arquitectura

Franco Bulgarelli
Versión 2.0
Octubre 2023

Arquitecturas de Sistemas, Arquitecturas de Software Clásicas

EntregaYaYaYa

Lectura recomendada El diablo está en los detalles

Nos han pedido que desarrollemos un sistema para realizar entregas a domicilio de cualquier tipo de producto. El flujo es simple:

1. una vez realizado el *login*, seleccionaremos la opción de *Solicitar Entrega*: allí ingresaremos nuestra dirección actual (idealmente, por defecto cargada con la ubicación del dispositivo), la dirección del lugar en que se pasará a retirar el producto, el costo y el tamaño del mismo (sobre/paquete pequeño/paquete grande) y presionaremos el botón de *Continuar*;
2. Allí el sistema nos mostrará el costo adicional que deberemos pagar (calculado en base a los parámetros anteriores) y podremos seleccionar *Entregar Ya* o *Cancelar*.
3. Una vez seleccionado Entregar Ya, se generá un pedido de envío, que une *Operadore* del sistema revisará, buscará une *Despachante* disponible, y nos confirmará por mail que el envío se realizará.

Proponé y diagramá la arquitectura más simple posible para este software, y **explicá** dónde lo desplegarías (es decir, qué máquinas usarás para servir la aplicación y dónde estarán ubicadas). **Respondé** teniendo en cuenta los distintos tipos de infraestructura (*Bare Metal*, PaaS, SaaS, IaaS).

Además**, planteá** que aspectos de la arquitectura del sistema completo (y no sólo del software) falta definir e integrarlos a la arquitectura del software.

MicroPC IDE

La empresa MicroPC nos pidió que desarrollemos un entorno de desarrollo para programar su familia de *microcontroladores* y placas-computadora[1]. En ambos casos, la forma de trabajo es similar: se programa utilizando un editor de código, y cuando terminamos, *quemamos* el software en el dispositivo, lo que consiste en transferir el código compilado al mismo, para que pueda funcionar. Esta transferencia se realiza a través de un puerto USB o Serial (Protocolo RS232).

Adicionalmente, el entorno de desarrollo debe contar con un simulador y un debugger paso a paso, que nos permitirá simular las entradas y salidas del dispositivo.

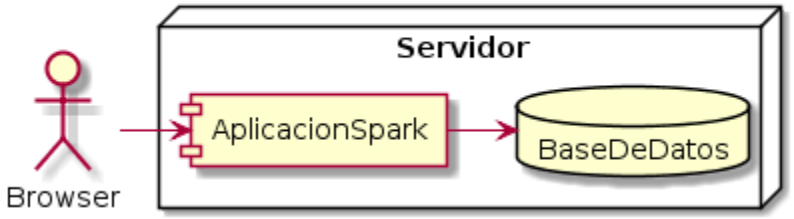
Por último, MicroPC desea contar con un *Marketplace*, es decir, un espacio online en el cual empresas y particulares pueden ofrecer (y eventualmente, comprar y vender) pequeños programas llamados *snippets* ya probados listos para quemar o cargar en el editor y modificar.

Proponé la arquitectura de este sistema **y comunicala** utilizando un diagrama de despliegue.

Escala, Arquitectura Organizacional

Rekomendashi, Fase 2

Lectura recomendada: *Breve listado de componentes arquitecturales*



¡Rekomendashi volvió! ¡En forma de un emprendimiento exitoso! Nos han invertido, y ahora tenemos ya no decenas, sino miles de usuaries. Y esto nos está trayendo algunos problemas:

1. Tuvimos que agregar más memoria y capacidad de procesamiento al servidor desde el que estamos sirviendo a Rekomendashi. Pero en nuestro proveedor de nube esto nos está saliendo bastante caro: por el costo con el que pagamos los 4 cores y 16GB de memoria, podríamos pagar 6 máquinas de 1 core y 4GB de memoria cada una
2. La carga está aumentando todos los meses, y cada vez que redimensionamos la máquina tenemos que apagar al sistema, lo que se traduce en downtime. Además, el proveedor siempre nos tira un warning sobre los riesgos de esta operación, y nos recomienda realizar un backup de la base (que está en el mismo servidor).
3. A veces el servidor se cae (estamos investigando si se trata de un *memory leak*), con las evidentes quejas de les clientes.
4. Por último, cada vez tenemos más imágenes de piezas de sushi en la página, de mejor calidad, y esto está aumentando drásticamente los tiempos de carga de página.

Proponé y diagramá una arquitectura que resuelva estas problemáticas

---

[1] Computadora que está conformada por una sólo plaqueta; las más famosas son las RaspberryPI, Arduino y Microbit

SheRa Enterprise

Lectura recomendada: *Breve listado de componentes arquitectónicos*

¿Se acuerdan de SheRa? Bueno, ¡resulta que no era sólo un *issue-tracker*! Ahora SheRa cuenta con muchas nuevas funcionalidades y módulos para empresas:

- seguimiento de incidentes (issue-tracking, bueno...)
- planificación ágil utilizando tarjetas Kanban, similares a las de Trello o Github Projects
- facturación de proyectos
- revisión de código, estilo Pull-Request
- integración con el control de versiones
- administración de equipos de trabajo
- y por si fuera poco, planea contar con soporte para seguimiento de ventas, clientes y personal

Para dar soporte a todas estas funcionalidades y a las decenas de clientes, ahora cuenta con un equipo de 50 desarrolladeres, liderado por una sóla persona: Analía. Sin embargo la arquitectura no ha cambiado ni un poquito desde la primera iteración: es un servidor monolítico escrito en Java, Spark y JPA. Y todes modifican ese código: es normal que haya cambios frecuentes y conflictos en el control de versiones, la compilación se ha vuelto lenta, y cualquier modificación implica redesplegar todo.

Proponé y diagramá una arquitectura que resuelva estas problemáticas, y **planteá** cómo organizarías al equipo en torno a ella. Además, ¿de qué manera impactaría en la solución que Shera se comercializara como un SaaS o un producto on-premise?

Performance, Tolerancia a Fallos

Cyberfriday

Lectura recomendada: *Breve listado de componentes arquitecturales*

¡Se nos vino el *cyberfriday*! Tenemos una aplicación de *ecommerce*, en el cual hay un árbol de categorías de profundidad arbitraria, ejemplo:

- Electrodomésticos (L1)
    - Living (L2)
        - Televisores
    - Cocina (L2)
        - Heladeras
        - Hornos (L3)
            - Eléctricos
            - A gas (L4)

En la home mostramos los dos primeros niveles del árbol de categorías y un par de promociones de nuestros patrocinadores más importantes.

El primer nivel de categorías no se puede seleccionar, solo es a modo de agrupamiento. Cuando seleccionamos el segundo nivel nos llevará a una nueva pantalla que mostrará el título de la categoría y los 8 productos de oferta más comprados en la última hora.

El problema es que la carga de estas pantallas está tardando muchísimo tiempo. ¿Por qué puede ser? ¿Cómo podría solucionarse?

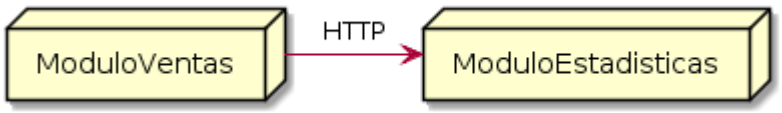
Proponé y diagramá una arquitectura que resuelva estas problemáticas.

Librerías Bustos Domecq

El equipo de arquitectura de 2Diseños tiene un nuevo desafío: Librerías Bustos Domecq tiene un serio problema de performance con el sistema que Mislav, la consultora anterior, les construyó.

Este sistema les permite gestionar las ventas, controlar el stock y generar estadísticas **en tiempo real** de las más de 80 sucursales en la región. Como además Librerías Bustos Domecq vende a través de MercadoAbierto (popular sitio de compras y ventas online) y NegocioNuboso (sitio para crear y gestionar tiendas virtuales), este sistema está integrado con las APIs de ambos portales, que en la práctica funcionan como dos sucursales más.

El sistema normalmente funciona sin sobresaltos, pero ocasionalmente y ante grandes volúmenes de ventas en cortos períodos (por ejemplo, cuando se lanzan nuevas entregas de sagas exitosas o expansiones de juegos de mesa largamente esperadas) el sistema colapsa. Después de bastante análisis se llegó a la conclusión de que el problema está en la interfaz entre el módulo de ventas y el módulo de estadísticas, que fueron modelados como dos microservicios:



Aparentemente, con cada venta, el módulo de estadísticas persiste en una base de datos información altamente desnormalizada, y el proceso completo demora demasiado tiempo, bloqueando al módulo de ventas y haciendo que funcione mucho más lento. En algunas ocasiones, incluso, el módulo de estadísticas se ha caído, provocando que las ventas fallen y la saturación del disco por el crecimiento descontrolado de los logs de error. ¿Qué se puede hacer?

Proponé y diagramá una arquitectura que resuelva estas problemáticas, además de anotar cualquier otra consideración que ayude a mitigar este comportamiento catastrófico.

Arquitectura y poder

Lectura recomendada [Feminismo de datos (introducción y capítulo 1)]

A.F. Analytics

Como parte de una política pública de digitalización, un estado nacional (cuya identidad no revelaremos) ha encargado la construcción de A.F., un sistema software para eliminar completamente los trámites tradicionales en papel. Una vez que A.F. se encuentre en producción, el 100% de las gestiones de seguridad social, planes de acceso al crédito, tenencia de armas, pasaportes, solicitud de residencia y ciudadanía, presentaciones judiciales, boletos estudiantiles (por sólo nombrar algunas) serán virtuales.

La noticia ha sido ampliamente aplaudida por el periodismo especializado que ha destacado el ahorro que significará en infraestructura edilicia, papel y mano de obra, además de que evitará la pérdida de información y agilizará drásticamente procesos altamente ineficientes. Más allá de algunas voces aisladas, la sociedad también ha recibido la novedad con expectativa y amplios márgenes de aprobación.

El proyecto se desarrollará a lo largo de 18 meses y demandará de un equipo altamente capacitado. Justamente por eso, nuestra consultora, 2Diseños, ha sido recomendada por el Ministerio de Futuro Digital y seleccionada para su construcción.

 [2]

(Foto reciente de nuestro equipo en el segmento de tecnología de un diario matutino)

La lista de requerimientos es extensa. Algunos de ellos son:

1. Signup y login, el cual se resolverá utilizando documento de identidad o pasaporte (*numérico*), sexo (*hombre/mujer*)
2. Búsqueda por nombre, selección, inicio, carga y seguimiento de trámites.
3. Panel de control administrativo, en el cual se podrán obtener métricas, desagregadas por rango etario
4. Generación de un código QR de acceso directo a ciertos trámites, como formularios de aduana y declaraciones juradas. Estos códigos QR estarán impresos en carteles de aeropuertos y estaciones fluviales.

Además, A.F. debe ser capaz de manejar un gran volumen de solicitudes y transacciones, ya que se espera que maneje todas las gestiones mencionadas para toda la población del país. Por este motivo se está realizando una evaluación de costos de proveedores de infraestructura como Amazon, Azure y Google Cloud, entre otros.

El sistema debe funcionar en teléfonos celulares y distribuirse únicamente como una aplicación nativa para Android y iPhone, completamente en español. Cada habitante en edad adulta deberá contar con la aplicación instalada, dado que los trámites son privados, seguros, personales e intransferibles.

Por último, nos han adelantado que habrá luego una siguiente fase, en la que se cruzarán los datos con otros sistemas del estado y se construirán modelos de inteligencia artificial para realizar recomendaciones de

---

[2] Imagen generada automáticamente por https://www.freepik.com ante la consulta "*team of successful programmers*"

trámites y predicciones para políticas de salud y seguridad, como prevención del delito y embarazo adolescente. Algunos de los potenciales proveedores de infraestructura tecnológica han mostrado su interés en colaborar gratuitamente en el diseño de estos algoritmos y han mencionado que podrían llegar a ofrecer descuentos en sus productos en caso de ser elegidos.

Como parte del equipo de arquitectura de 2Diseños, ***discutí*** *en grupo este caso de estudio y* ***elaborá*** *tus comentarios al respecto.¿Qué recomendaciones, objeciones o acciones propondrías?*[3]

[3] Como punto de partida, te proponemos analizar la situación según los ejes de androcentrismo, binarismo, capacitismo, colonialismo, racismo, extractivismo y meritocracia.