Nova IMS – Information Management School

Managing Relational and Non-Relational Data

2018/2019

**No SQL Project**

Joana Ricarte, nº20170983

Tiago Marques, nº20170998

Sara Matos, nº

## **Sumário**

Realizado no âmbito da disciplina “Managing Relational and Non-Relational Data”, este relatório pretende sugerir a melhor aplicação para um caso de extensão de plataformas de leilões de produtos à escala global, de uma empresa fictícia de venda de produtos de desporto – Adventure Works.

### **Introdução**

O nosso caso de estudo prende-se com a decisão de uma empresa de venda de produtos de desporto de construir uma plataforma de leilões de alguns dos seus produtos, para maximização de lucro e escoamento de stock.

A empresa tem distribuição em várias regiões do globo e esta aplicação pretende alcançar os clientes em todas as regiões onde os produtos são distribuídos.

O objetivo deste trabalho é apresentar uma solução aplicacional que vá ao encontro das necessidades de negócio e tecnológicas definidas.

Para isso, este relatório foca-se na apresentação de uma solução baseada em arquitetura, computação e modos de armazenamento de aplicações em Cloud, fugindo às aplicações web convencionais, que pensamos não se serem adequadas às necessidades descritas no case study para a Adventure Works.

### **Aplicação em Cloud**

O princípio da Cloud assenta na criação de aplicações onde os serviços são decompostos e descentralizados.

A comunicação entre serviços é feita ou por API’s, ou por um sistema de mensagens assíncrono.

A preparação destas soluções para suportar a escalabilidade de um negócio é uma das suas grandes vantagens. Neste caso, uma solução Cloud deverá conseguir suportar um aumento geográfico da distribuição de produtos da companhia, uma reestruturação, um aumento de clientes ou até picos de adesão e utilização da aplicação.

Outra das vantagens de uma tecnologia Cloud prende-se com o facto desta suportar múltiplas tecnologias de armazenamento – característica denominada *Polyglot Persistence*.

O processamento de dados deste tipo de soluções é feito de forma assíncrona e paralela e normalmente estão preparados para lidar com falhas (MTTR).

É também de salientar as características de auto-gestão automatizada e a infraestrutura normalmente imutável.

Para além disso, pressupõe-se que um sistema em cloud esteja constantemente a ser sujeito a updates frequentes, mantendo os dados o mais consistentes possíveis.

### **Caso de Estudo – Requisitos Levantados**

Como já referido, a empresa em estudo tem produção e revenda em todo o mundo e, no sentido de otimizar o processo, foram delineados alguns requisitos de execução:

|  |  |
| --- | --- |
| Formato de dados | Utilização de Objetos JSON para o catálogo de produtos;  Garantir a consistência do inventário de produtos no decorrer das licitações/ leilões |
| Tamanho dos Dados | Capacidade de fornecer ao utilizador a descrição do produto, a quantidade em stock, manuais, vídeos, preço previsto. |
| Escalabilidade e Estrutura da solução | Necessário haver capacidade suficiente para armazenar os dados dos produtos distribuídos localmente. Para otimização e melhoramento da performance, deverá ser considerada a partição dos dados por localização geográfica/região. |
| Consistência | Os utilizadores online irão ser informados que a última oferta poderá não estar disponível. Não deverão, no entanto, ser aceites atrasos acima dos 30s. O vencedor do leilão apenas será decidido depois do leilão terminar. |
| Concorrência | Se existirem vários utilizadores a fazer a mesma aposta, o vencedor corresponderá ao utilizador que tiver um registo data-hora mais recente. |
| Fluxo dos Dados | Deverá ser desenvolvido um ETL que atualize recorrentemente o stock dos produtos e o preço previsto de venda, num contexto de um determinado leilão. Deverá também ser possível ao utilizador aceder aos vídeos e manuais atualizados dos produtos. |
| Desempenho e Escalabilidade | Os utilizadores deverão poder colocar as suas licitações, independentemente da região onde se encontram, ou dos picos de afluência (como por exemplo, a *Black Friday*) |
| Reliability | A aplicação deverá estar disponível a maioria do tempo, mesmo em períodos em que alguns servidores se possam ir abaixo. |
| Limites | Por uma questão de conveniência, deverá ser realizada uma limpeza às licitações cujos leilões já foram terminados. Como já foi decidido a que cliente será atribuído o produto em leilão e como já terá sido retirado o produto em stock, o histórico das licitações torna-se irrelevante para o funcionamento da aplicação. |
| Licenciamento | Não existem preferências de licenciamento |
| Custo Geral | Devido à variação de tráfego que poderá ocorrer ao longo do ano, pretende-se uma solução que permita gerir os serviços onde a equipa técnica possa ter controlo/ ou acompanhar os recursos utilizados |
| Segurança | Apenas utilizadores com morada validada poderão fazer licitações, e só a produtos distribuídos na sua região. |
| Outras Funcionalidades | Poderia ser útil uma solução de pesquisa de produtos em leilão. |

**Estilos de Arquitetura**

Existem vários tipos de arquitetura em Cloud – N-tier, Web-Queue-Worker, Microservices, CQRS, Event-Driven Architecture, etc.

No entanto, neste relatório propõe-se o estilo Event Driven Architecture face, por exemplo à utilização de Microservices (por poder não ser o ideal em alturas de forte congestionamento, como por exemplo, o impactante período da *Black Friday*) ou à utilização da arquitetura CQRS – representada na Figura 1 – que, apesar de separar as operações de leitura e escrita, poderá não estar otimizada para períodos de forte afluência – grande volume de dados e necessária pouca latência).

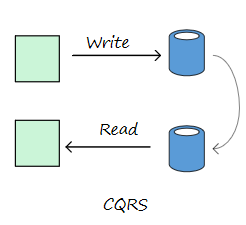


Figura – Esquema da arquitetura CQRS

O estilo Event Driven Arquitecture é o mais adequado ao cenário de licitações de produtos. A marcação de uma licitação não é complexa em termos de estrutura de dados, mas é exigente em termos de tempo de resposta e volume de dados. Recorre a um modelo “publish-subscribe”, onde os produtores publicam eventos e os consumidores os subscrevem. São portanto adequados a casos onde existe um grande volume de dados e é exigida uma baixa latência.

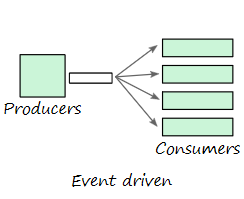


Figura – Esquema da arquitetura “Event Driven”

A camada de produtores é independente da camada de consumidores e os eventos são entregues quase em tempo real, para que os consumidores possam dar respostas imediatas aos mesmos.

Os benefícios desta arquitetura prendem-se com o facto de os produtores e os consumidores estarem desacoplados. Não existem integrações point-to-point, pelo que é fácil adicionar novos consumidores ao sistema. É altamente escalável e de fácil distribuição.

Já os subsistemas têm *views* independentes da *stream* de eventos. Ao utilizar este modelo, dever-se-á garantir que a aplicação garante que os eventos são corretamente apresentados e que estes são processados por ordem, apenas uma vez.

**Tecnologias**

Por entre todos os serviços de computação disponíveis dentro do universo Cloud (uns garantem mais liberdade/ controlo/ portabilidade/ flexibilidade que outros), consideramos quem uma abordagem FaaS (function as service), poderá ser mais adequada para o presente caso de análise. Para além de eliminar a preocupação de ter um hosting environment, também não é necessário ter a preocupação da gestão dos diferentes recursos de suporte à computação. Estes serviços normalmente estão inseridos numa arquitetura serverless e são facilmente adaptáveis ao tráfego que esses mesmos serviços suportam num determinado momento. (por exemplo, recorrendo ao *Azure Funtions*).

**Armazenamento de Dados**

Neste caso de estudo vão existir duas fontes de dados principais e distintas:

* Base de Dados Relacional da organização
  + Dados de utilizadores e endereços de faturação
  + Descrição do produto, stock, preço, região de distribuição
  + Manual do Produto e Vídeo
* Dados recolhidos do sistema aplicacional
  + Licitações (cliente, valor da licitação, *timestamp*)

Dados de Utilizadores e Endereços de Faturação

Estes dados irão ser guardados numa estrutura de armazenamento Key-Value, que associa um valor chave (clientID), um valor de região da morada de faturação.

Descrição, Stock Disponível, Preço e Região de Distribuição do Produto

Na estrutura de armazenamento do tipo *Document*, é guardado um conjunto de informação relacionada com uma key, normalmente listas ou child collections. Os dados nos campos podem ser codificados da mais variada forma (XML, YAML, JSON, BSON), ou como texto inteiro. Este tipo de armazenamento permite à aplicação filtrar e procurar os dados, usando os valores desses campos.

Guardar os dados do produto por região de distribuição é algo que poderá ser feito em document databases.

Manuais e Vídeos de Catálogo do Produto

A estrutura de Object Data Stores está otimizada para guardar e devolver grandes ficheiros binários: Imagens, Ficheiros, vídeo, Audio Streams, etc). Os objetos neste tipo de armazenamento contêm a não só os dados em si, mas também a metadata e um ID único para facilitar o acesso ao conteúdo do mesmo. Alberga grandes quantidades de dados não estruturados.

Licitações de Clientes

A estrutura de armazenamento do tipo Time Series está otimizada para conjuntos de dados organizados no tempo. Suporta um elevado número de escritas, e reúne grandes quantidades de dados em tempo real, oriundos de um grande número de fontes. Neste tipo de armazenamento, é rara a existência de *updates* e os *deletes* são feitos em *bulk operations*. Normalmente os dados escritos são pequenos em tamanho, mas grandes em volume, pelo que o total de dados pode crescer rapidamente.

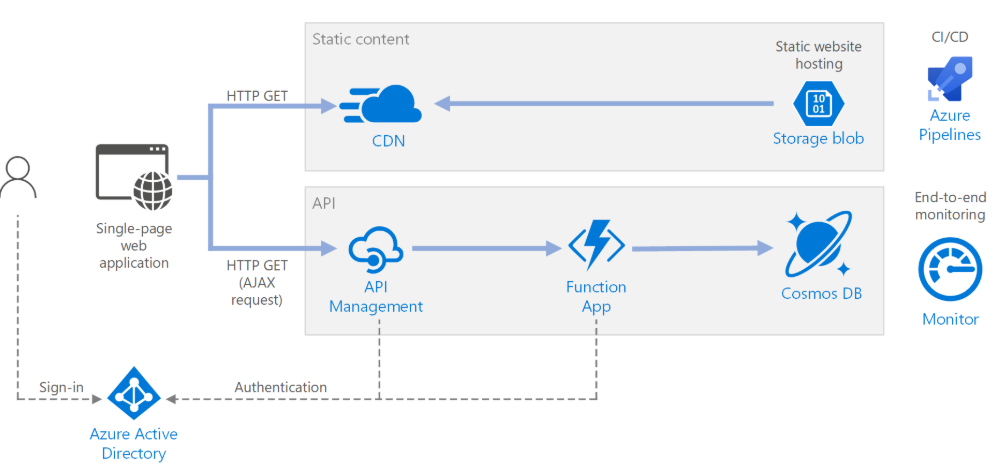
**Princípios de Design para aplicações Azure**

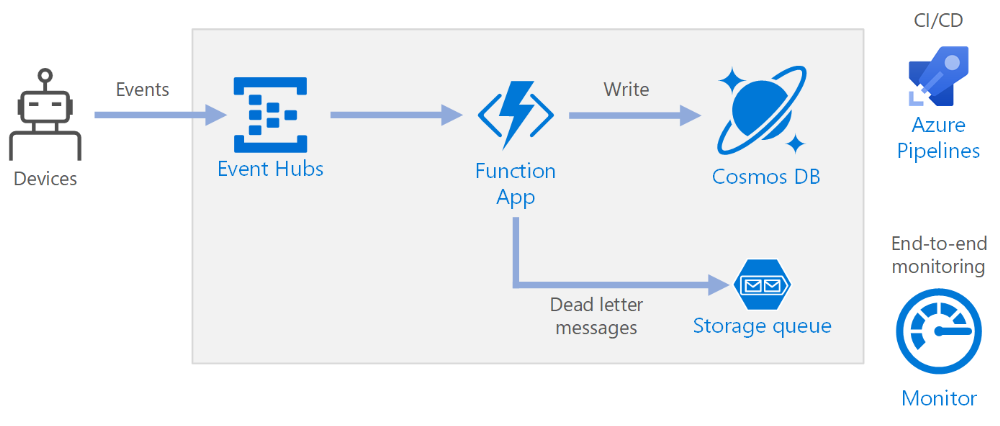
* Recuperação automática se existirem falhas
* Construção de pontos de redundância para diminuir impacto das falhas.
* Minimizar a coordenação entre serviços da aplicação para atingir a escalabilidade
* Construir uma aplicação tendo em conta a sua escalabilidade horizontal.
* Recorre ao particionamento no armazenamento.
* Desenhar a aplicação para a equipa de Operações.
* Sempre que possível, utilizar serviços PaaS ou FaaS.
* Adequar o armazenamento de dados ao caso que se está a tratar.
* Desenvolver a aplicação a pensar na sua evolução.
* Qualquer decisão de design deverá ser suportada por um requisito de negócio.

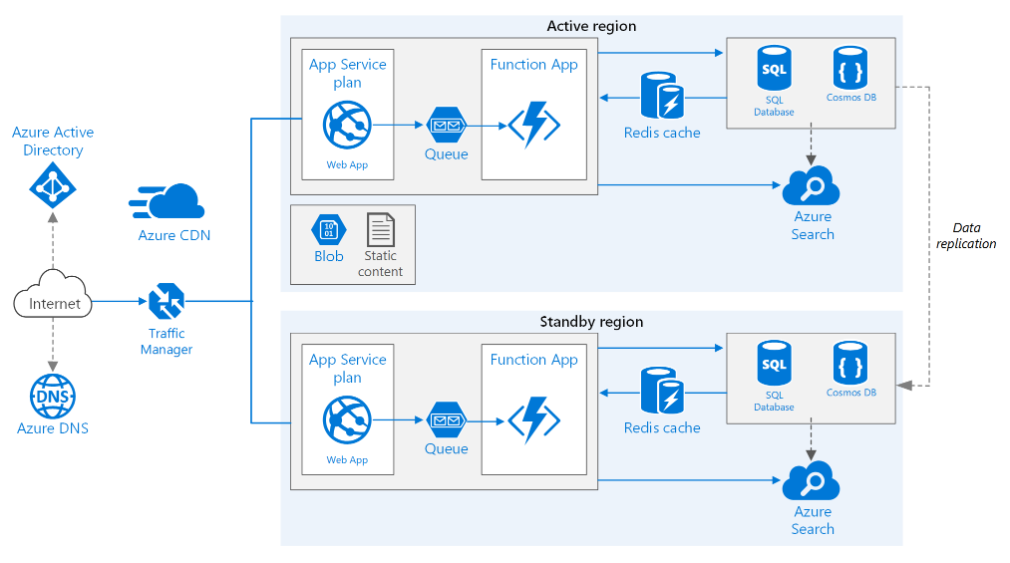
**Pilares de Qualidade:**

Para uma aplicação cloud ter sucesso, é necessário respeitar os seguintes pontos:

* Escabilidade, principalmente horizontal (para se poder adicionar instâncias a um recurso, sempre que o load é maior). Escalar uma aplicação horizontalmente também se revela ser mais barato do que escalar verticalmente.
* Disponibilidade, que reflete o tempo que um sistema está operacional. Consideram-se satisfatórios com valores percentuais acima de 99%. Na maioria dos casos, para manter um sistema com altos níveis de disponibilidade, é preciso recorrer a procedimentos de auto-diagnóstico e auto-recovery.
* Resiliência, abilidade do sistema recuperar de falhar e continuar a funcionar
* Gestão, através de uma automatização dos deployments, para excluir erros humanos. Deverão ser processos rápidos e rotineiros, para que não abrandem as novas releases. Deve também ser importante fazer rollback, se houverem problemas.
* Segurança, desde validação de identidade, a proteção da infraestrutura, da aplicação e dos dados em si.







### **User Stories**

Nesta secção detalhamos a aplicação e o seu modo de funcionamento recorrendo a algumas user stories. O objetivo desta secção pretende melhorar a compreensão dos conceitos e requisitos acima detalhados e servir de suporte para um eventual POC.

User Story 1-

Um utilizador não identificado tenta aceder à aplicação de leilões de produtos.

User Story 2 –

Um utilizador autenticado tenta aceder à aplicação, mas não tem morada de billing confirmada.

User Story 3-

Um utilizador autenticado tenta aceder à aplicação, tem morada autenticada, mas não tem nenhum artigo disponível para leilão na sua região.

User Story 4 –

Um utilizador autenticado tenta aceder à aplicação, tem morada autenticada, e tem artigos disponíveis para leilão.

User Story 5 –

Um utilizador nas condições de 4) procura por um item específico para poder consultar a sua informação.

User Story 6 –

Um utilizador nas condições de 4) seleciona, num item específico, o link do manual do produto selecionado, para download do mesmo.

User Story 7 –

Um utilizador, nas condições de 4) seleciona, num item específico disponível para leilão, o url do vídeo corresponte e visualiza o mesmo vídeo na aplicação.

User Story 8 –

Um utilizador, nas condições de 4), depois de selecionar um dos items em stock e disponíveis para leilão na sua zona para licitação.

Deverá ver mensagem de aviso sobre inconsistências dos dados (latência não superior a 30s)

User Story 9 –

Utilizador inicia a licitação e tenta colocar um valor abaixo ou igual ao valor atual de venda do produto em leilão.

User Story 10 –

Utilizador refresca a aplicação passados mais de 30 segundos da sua licitação, para confirmar que os valores de venda do produto em leilão foram alterados.

User Story 11 –

Utilizador tem a sua licitação associada ao preço momentâneo de venda de um produto em leilão e consulta a licitação do produto.

Utilizador recebe mensagem a indicar que a sua licitação é presentemente a considerada.

User Story 12 –

Período de licitação encerra, utilizador em condições de 11) recebe informação de que o produto foi vendido a si.

User Story 13 –

Vários utilizadores, na mesma região, licitam o mesmo valor praticamente ao mesmo tempo.

User Story 14 -

O período de licitação de um produto acaba e vários utilizadores, na mesma região, submeteram licitações com o mesmo valor.

### **Bibliografia**

<https://docs.microsoft.com/en-us/azure/architecture/guide/>

<https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/index>

<https://docs.microsoft.com/en-us/azure/architecture/guide/technology-choices/data-store-overview>

<https://docs.microsoft.com/en-us/azure/architecture/guide/design-principles/index>

<https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/event-driven>

<https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data>

<https://docs.microsoft.com/en-us/azure/architecture/guide/technology-choices/data-store-comparison>

TO DO: Falta descrever a forma como os dados deverão estar organizados.