Nova IMS – Information Management School

Managing Relational and Non-Relational Data

2018/2019

**No SQL Project**

Joana Ricarte, nº20170983

Tiago Marques, nº20170998

## **Sumário**

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

### **Introdução**

A Adventure Works tem a sua actividade de venda de bicicletas e produtos relacionados em várias regiões do globo. Para colmatar alguns problemas em escoamento de stock, necessário principalmente aquando lançamento de novos modelos, em que o stock de modelos antigos deve ser reduzido ao máximo, resolveu testar uma plataforma de leilões e estendê-la às várias regiões do mundo onde é feita a venda e distribuição.

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

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.

### **Aplicação em Cloud**

O princípio da Cloud assenta na criação de aplicações onde os serviços são decompostos e descentralizados, onde a comunicação entre serviços é feita ou por API’s, ou por um sistema de mensagens assíncrono.

Estas soluções estão preparadas e otimizadas para suportar a escalabilidade de um negócio. Neste caso, uma solução Cloud deverá conseguir suportar um aumento geográfico da distribuição de produtos da Adventure Works, uma reestruturação, um aumento de clientes ou até picos de adesão e utilização da aplicação.

Outra das vantagens dos serviços Cloud prende-se com o facto destes suportarem múltiplas tecnologias de armazenamento –característica conhecida como *Polyglot Persistence*.

É também de salientar a facilidade de criar formas de contornar possíveis falhas e as características de gestão automatizada e a infraestrutura (normalmente) imutável.

Os sistemas Cloud também estão preparados e necessitam de updates frequentes, de forma a manter os dados o mais atualizados e consistentes possíveis.

### **Requisitos Levantados**

Os requisitos levantados e que serviram de base para esta proposta estão listados:

|  |  |
| --- | --- |
| 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 só 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 | A informação da atualização do stock dos produtos e o preço previsto de venda deverá ser atualizada automaticamente e recorrentemente, num contexto de um determinado leilão.  Deverá também ser possível ao utilizador aceder aos vídeos e manuais atualizados dos produtos em leilão. |
| 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 na 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 não aceites cujos leilões já foram terminados.  Depois de decidido a que cliente será atribuído o produto em leilão, 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 | Poderá ser útil uma solução de pesquisa de produtos em leilão. |

**Considerações: Estilos de Arquitetura**

Existem vários tipos de arquitetura em Cloud:

* N-tier
* Web-Queue-Worker, Microservices
* CQRS
* Event-Driven Architecture

etc.

O caso proposto de exposição de produtos para licitação deverá assentar mais num modelo do tipo “Event-Driven”. Face, por exemplo, à utilização de Microservices, destaca-se por um melhor desempenho em alturas de forte congestionamento (como o que está previsto pela empresa na altura da *Black Friday*). Face a arquitetura CQRS – representada na Figura 1 – destaca-se por garantir uma melhor adaptação na mesma altura de congestionamento, em que poderá haver um grande volume de dados, sem que isso comprometa a latência.

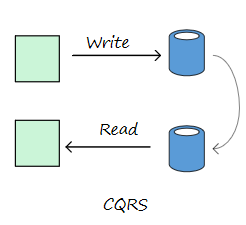


Figura 1 – Esquema da arquitetura CQRS

A arquitetura “Event Driven” recorre a um modelo “publish-subscribe”, onde os *producers* publicam eventos e os *consumers* os subscrevem - Figura 2. São, portanto, adequados a casos onde existe um grande volume de dados e é exigida uma baixa latência.

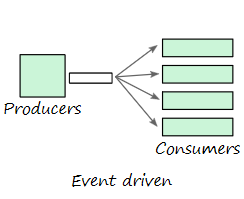


Figura 2 – Esquema da arquitetura “Event Driven”

A camada de *producers* é independente da camada de *consumers* e os eventos são entregues quase em tempo real, para que os consumidores possam dar respostas imediatas aos mesmos (neste caso, fazer uma licitação de um produto em leilão).

Os benefícios desta arquitetura prendem-se destas camadas estarem desacopladas, fazendo com que seja fácil adicionar novos consumidores ao sistema. É altamente escalável e de fácil distribuição.

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.

**Considerações: Tecnologias**

Por entre todos os serviços de computação disponíveis dentro do universo Cloud (uns garantem mais liberdade/ controlo/ portabilidade/ flexibilidade que outros), considerou-se uma abordagem FaaS (*function as service*), que poderá ser mais adequada para o presente caso de análise.

Este tipo de serviço permite um menor esforço na 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*).

**Considerações: 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
  + Leilão (Número, Produto, Região, Data-Hora de Início, Data-Hora de Fim, Estado)
* Dados recolhidos do sistema aplicacional
  + Licitações (cliente, valor da licitação, produto, Número de leilão, *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.

|  |  |
| --- | --- |
| *Key* | *Value* |
| *11000* | *United States* |

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.

*{*

*“Name”: “HL Mountain Seat Assembly”,*

*“Description”: “SA-M687”,*

*“Stock”: “500”,*

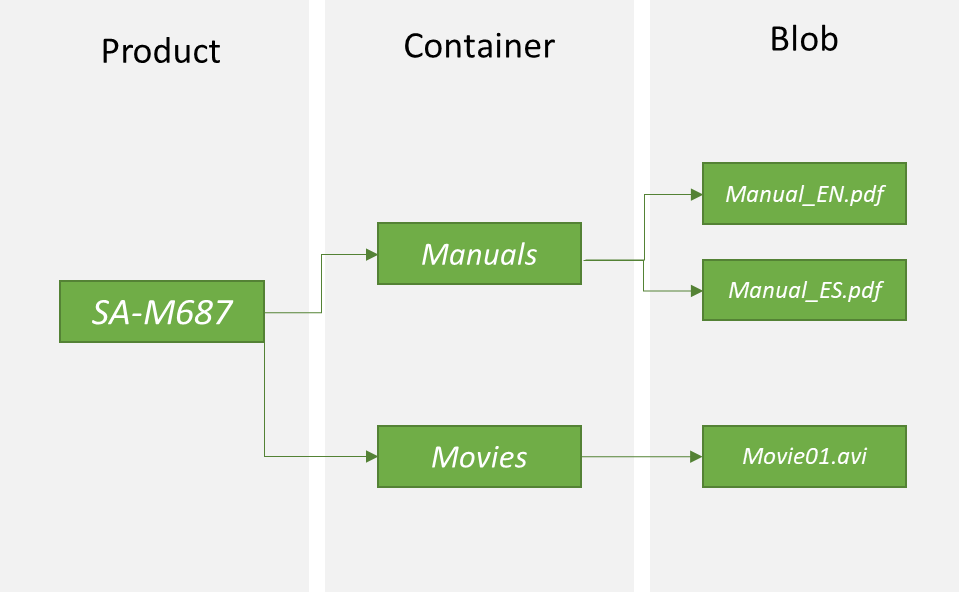
*“Price”: “196.92”,*

*“Region”: “United States”*

*}*

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.



Leilão

Dados de leilão irão ser guardados em estruturas de armazenamento do tipo *Document*, fazendo partição por região para otimização de leitura e processamento.

*{*

*“Number”: “2014003020”,*

*“Product”: “SA-M687”,*

*“Region”: “500”,*

*“StartingAt”: “2014-11-15 00:00:00”,*

*“EndingAt”: “2014-11-25 23:59:59”,*

*“Status”: “Terminated”,*

*“Winning Bid Amount”:156.00,*

*“BidsCount”: 57*

*}*

Estes dados também deverão ser organizados por região.

Licitações de Clientes

Estes dados também terão estrutura de armazenamento do tipo *Document*. Considera-se que os dados de licitações não são para manter, a não ser se a licitação tiver originado uma compra. Aquando o fecho de um leilão, deverão ser analisadas todas as licitações. A licitação com valor mais elevado e, no caso de haver várias com esse mesmo valor, a licitação mais antiga, deverá ser a licitação vencedora.

O sistema deverá ser capaz de mudar o estado do leilão para “terminated” e acrescentar o valor da licitação vencedora, bem como o total e licitações.

No final, as licitações não vencedoras deverão ser apagadas.

*{*

*“Bidder”: “2014003020”,*

*“Amount”: “SA-M687”,*

*“Timestamp”: “2014-11-19 15:17:48”,*

*“Auction”: “2014003020”*

*}*

**Proposta de Arquitetura**

Tendo em conta os requisitos levantados, foi criada a estrutura representada na Figura 3.

Quando o utilizador efetua login na aplicação, as suas credenciais deverão ser validadas e o seu acesso permitido ou bloqueado. Recorrendo ao serviço de Active Directory do Azure (ou outro equivalente) é possível controlar esses acessos à aplicação.

Independentemente da forma da aplicação e linguagem utilizada, esta deverá, aquando o acesso de um utilizador, apresentar os dados dos produtos disponíveis para licitação, a sua descrição, bem como os dados mais “pesados”, como manuais e vídeos (dados menos dinâmicos, que podem ser disponibilizados via Azure Blob Storage).

A aplicação também deverá ter componente de autoscaling e estar quase sempre disponível – Necessidades cujo serviço de Web App do Azure poderá colmatar.

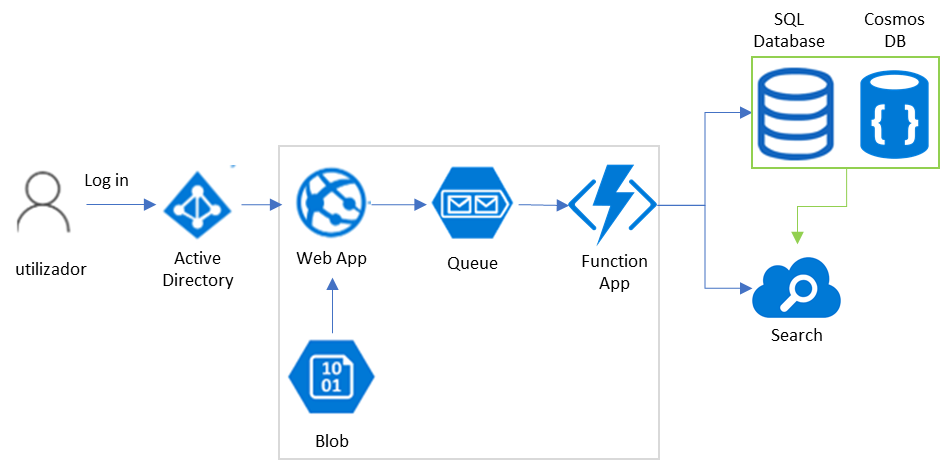
Considera-se relevante que um utilizador tenha a possibilidade de procurar um produto ou uma determinada característica do mesmo, deve-se recorrer ao Search no Azure, para configuração de um motor de busca eficiente. Desta forma, em casos de muitos produtos em leilão, será mais fácil ao utilizador usar o produto pretendido.

As várias licitações poderão ser processadas numa queue.

O módulo Function App poderá ser importante para confirmação das licitações, gerir jobs, como por exemplo limpeza de dados antigos não necessários.

Os dados, distribuídos pelo SQL Database e o Cosmos DB, deverão estar atualizados e disponíveis. O módulo de SQL Database em Azure deverá permitir uma sincronização híbrida com a(s) base(s) de dados on premisse, para permitir consistência dos dados no decorrer das várias atividades da organização.

Cosmos DB irá importar recursivamente e disponibilizar a informação dos ficheiros JSON com informação do produto relevante para o leilão. Irá também gerir os dados dos documentos (recurso à API para Mongo DB)



Figura

No sentido de garantir a disponibilidade da aplicação, sugere-se também a replicação de dados para infraestruturas noutras regiões, para que, mesmo que numa região existam condicionantes, os dados e a aplicação possam continuar a ser disponibilizados e utilizados.

Poderá também ser necessário implementar uma solução de gestão de tráfego na atividade de licitação de produtos, uma vez que se prevê uma forte afluência à mesma durante o período de disponibilização. Tal necessidade poderá ser colmatada com o Azure Traffic Manager.

### **User Stories**

Nesta secção pretende-se fazer uma análise funcional prévia, recorrendo a algumas user stories, com o objetivo de melhorar a compreensão dos conceitos, requisitos levantados.

***User Story 1***

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

*Comportamento Esperado: O utilizador vê o seu acesso negado à aplicação. O utilizador recebe mensagem de erro.*

***User Story 2***

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

*Comportamento Esperado: É pedida, através da aplicação uma pesquisa à morada do utilizador (dados em Azure SQL DataBase). Utilizador deverá receber mensagem a avisar de morada não confirmada.*

***User Story 3***

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

*Comportamento Esperado: Utilizador deverá receber mensagem a avisar da não disponibilidade de produtos.*

***User Story 4***

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

*Comportamento Esperado: Utilizador deverá ter acesso à home da aplicação, e a uma lista de produtos disponíveis para leilão. Poderá também ter acesso à lista de produtos já comprados em leilão, se existirem.*

***User Story 5***

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

*Comportamento Esperado: Aplicação vai buscar dados a Cosmos DB, dados deverão ser recentes, embora atrasos até 30 segundos sejam aceitáveis. Deverá surgir uma mensagem ao utilizador a referir o potencial atraso do estado mais atualizado. O utilizador, ao confirmar essa mensagem deverá poder ver os dados de catálogo do produto a licitar.*

***User Story 6***

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

*Comportamento Esperado: A aplicação acede ao ficheiro do manual via Blob Storage e abre automaticamente o ficheiro.*

***User Story 7***

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

*Comportamento Esperado: A aplicação apresenta uma lista de vídeos associados ao produto. Ao selecionar um vídeo específico, o utilizador consegue reproduzir o vídeo na aplicação.*

***User Story 8***

*Descrição: Um utilizador, nas condições de 4), seleciona, dentro de um dos artigos em leilão, a opção de licitação.*

*Comportamento Esperado: Aplicação deverá mostrar nome do produto, última licitação e campo de input para colocação do valor de licitação do próprio utilizador. Deverá também existir a opção de refrescamento.*

***User Story 9***

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

*Comportamento Esperado: Campo não é gravado e mensagem de erro surge.*

***User Story 10***

*Descrição: Utilizador refresca a aplicação passados mais de 30 segundos da sua licitação.*

*Comportamento Esperado: Valores de licitação foram alterados, para valor igual ao superior ao licitado pelo utilizador.*

***User Story 11***

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

*Comportamento Esperado: Utilizador recebe mensagem a indicar que a sua licitação é presentemente a considerada. Nova licitação é permitida.*

***User Story 12***

*Descrição: Período de licitação encerra, utilizador em condições de 11.*

*Comportamento Esperado: Utilizador recebe notificação de que o produto foi vendido a si próprio.*

***User Story 13***

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

*Comportamento Esperado: Apesar de todos as licitações ficarem registadas, a licitação considerada é a que tiver registo data-hora mais antigo, para o mesmo valor de licitação.*

***User Story 14***

*Descrição: O período de licitação de um produto acaba.*

*Comportamento Esperado: Estado do leilão é atualizado para “Terminado”, utilizador, valor e registo data-hora da licitação com valor mais alto ou, se existirem vários valores, com registo data-hora mais antigo.*

***User Story 15***

*Descrição: Região da instância na qual estão assentes os recursos Azure sofre algum tipo de constrangimento e fica inacessível.*

*Comportamento Esperado: O utilizador não sente impacto na sua experiência. A replicação de dados para uma estrutura igual numa região “stand-by”, vai permitir recuperar dados e garantir o normal funcionamento da aplicação.*

### **Conclusões**

Apesar da solução FAAS apresentada estar em linha com os requisitos de negócio definidos para a estratégia da Adventure Works. Esta necessitaria de ser complementada com o desenvolvimento de uma aplicação para POC e, idealmente, proceder-se a uma bateria de testes no sentido de poder especificar mais a solução aplicacional. Validação da escalabilidade, disponibilidade, resiliência e segurança são factores muito importantes de serem garantidos.

O factor económico também é sempre um factor importante a considerar. Apesar de soluções FAAS serem soluções com custos de opração mais reduzidos, seria necessário avaliar e validar os custos de uma implementação semelhante à sugerida.

### **Bibliografia**

Aulas de Managing Relational and Non-Relational Data, EDSA, 2018, NOVA IMS

<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>

<https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/app-service-web-app/basic-web-app>