

SEFAZ – SP  
Guia de Desenvolvimento

Preparado para

terça-feira, 19 março 2013

Versão 1.2 Final

Preparado por

Ernesto Guimaraes

Revisão

Histórico de mudanças

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Autor | Versão | Descrição |
| 20/01/2013 | Ernesto | 0.1 | Versão inicial para revisão |
| 21/01/2013 | Gleisson | 0.2 | Comentários gerais sobre o detalhamento da arquitetura |
| 22/01/2013 | Ernesto | 0.3 | Ajustes e nova visão da descrição da arquitetura |
| 23/01/2013 | Gleisson | 0.4 | Revisão da versão |
| 02/02/2013 | Ernesto | 0.5 | Inclusão de detalhes sobre o MSMQ e Service Broker |
| 06/02/2013 | Gleisson | 0.6 | Revisão de versão |
| 07/02/2013 | Ernesto | 0.7 | Ajustes de estrutura, organização das referências, adicionadas novas recomendações sobre Windows Services |
| 08/02/2013 | Ernesto | 0.8 | Inclusão de código de exemplo utilizado o serviço Service Broker |
| 08/03/2013 | Ernesto | 0.9 | Atualização do cenário de exemplo utilizando Service Broker |
| 08/03/2013 | Gleisson Bezerra | 1.0 | Revisão Final |
| 14/03/2013 | Ernesto | 1.1 | Inclusão do desenho físico |
| 19/03/2013 | Gleisson Bezerra | 1.2 | Revisão e Entrega Final do Projeto |

# Índice

[Índice 4](#_Toc351421130)

[Índice de Figuras 5](#_Toc351421131)

[Índice de Tabelas 5](#_Toc351421132)

[1 Objetivo 1](#_Toc351421133)

[2 Público Alvo 2](#_Toc351421134)

[3 Requisitos de Ambiente 2](#_Toc351421135)

[4 Siglas 2](#_Toc351421136)

[5 Visão geral de arquitetura 3](#_Toc351421137)

[6 Recomendações 6](#_Toc351421138)

[6.1 Arquitetura lógica para processamento em lote 6](#_Toc351421139)

[6.2 Recomendações para Windows Services 16](#_Toc351421140)

[6.2.1 Modelo de desenvolvimento 16](#_Toc351421141)

[6.2.2 Definir uma conta de Windows para a execução de cada serviço 17](#_Toc351421142)

[6.2.3 Certificar que os serviços que acessam pastas no servidor possuam somente a permissão necessária. 17](#_Toc351421143)

[6.2.4 Adicionar o termo “Exception” ao final do nome da classe. 17](#_Toc351421144)

[6.2.5 Desenhar o serviço sem interface com o usuário 18](#_Toc351421145)

[6.2.6 Remover todas as referências do método Assert do serviço antes de publicar em produção 18](#_Toc351421146)

[6.2.7 Implementar pelo menos os métodos OnStart e OnStop 18](#_Toc351421147)

[6.2.8 Não armazene informações sensíveis (confidencias) nos logs. 18](#_Toc351421148)

[6.2.9 Não utilizar SQL Jobs para processamento em Lote 19](#_Toc351421149)

[6.2.10 Ajuste a propriedade Autolog para “false” caso deseja gravar logs no Event Viewer 19](#_Toc351421150)

[7 Referências 20](#_Toc351421151)

[Apêndice A - Comparação entre MSMQ e Service Broker 21](#_Toc351421152)

[Apêndice B - Guia para monitoramento de Windows Services 23](#_Toc351421153)

# Índice de Figuras

[Figura 1 Arquitetura Física 3](#_Toc351421154)

[Figura 2 Sistema hipotético de processamento batch 4](#_Toc351421155)

[Figura 3 Atual Modelo lógico do processamento em Lote 6](#_Toc351421156)

[Figura 4 Proposta de modelo de arquitetura lógica 7](#_Toc351421157)

[Figura 5 Entidade Estado Arquivo no contexto da arquitetura 9](#_Toc351421158)

[Figura 6 Visão geral do cenário 11](#_Toc351421159)

[Figura 7 Detalhe do cenário - Troca de mensagens 12](#_Toc351421160)

[Figura 8 Padrão de desenvolvimento Windows Services 16](#_Toc351421161)

# Índice de Tabelas

[Tabela 1 Siglas utilizadas no documento 2](#_Toc351421162)

[Tabela 2 Principais entidades de arquitetura 8](#_Toc351421163)

[Tabela 3 Exemplo de código do *Service Broker* para envio e recebimento de mensagens 15](#_Toc351421164)

[Tabela 4 Contextos de segurança de um Windows Services 17](#_Toc351421165)

[Tabela 5 Comportamentos de um Serviço 18](#_Toc351421166)

[Tabela 6 Tabela de Referências 20](#_Toc351421167)

1. Objetivo

Este documento é parte integrante do projeto de **Elaboração de Arquiteturas de Software – FASE 1**, desenvolvido segundo o Escopo de Trabalho e Formato de Entregas especificados no documento de Visão Escopo (SEFAZ - ITAP Arquiteturas de Software - Visão e Escopo v1.1).

O projeto visa estabelecer uma referência para o desenvolvimento padronizado de sistemas que permita o planejamento da adoção de práticas para reduzir riscos de segurança e aumentar a disponibilidade de serviços da instituição.

Dentro desta proposta foram priorizados os aspectos referentes as seguintes frentes de trabalho:

* **PLATAFORMA .NET**: Diretrizes para organização e construção de componentes que devem servir todas as arquiteturas específicas;
* **SERVIÇOS WEB**: Arquitetura específica, hospedagem e monitoramento de operações;
* **PROCESSAMENTO BATCH**: Arquitetura para aplicações responsáveis por processamentos em lote de forma assíncrona;
* **BANCO DE DADOS**: Revisão de práticas de desenho físico e ferramentas de apoio.

Não foram abordados, nesta fase, a camada de apresentação de **Aplicações, *Providers* de Acesso a Dados e Mapeamento Objeto-Relacional**. Assuntos reservados para as fases seguintes.

Este guia diz respeito à frente de trabalho de Processamento Batch e tem como objetivo orientar o desenvolvedor da plataforma Microsoft .NET com recomendações para o desenvolvimento de deste tipo de aplicação**.** Entende-se como processamento em lote o processo que ocorre em background, ou seja, sem intervenção direta do usuário e tem como objetivo de processar e validar dados externos e enviá-los a uma entidade destino.

Além da proposta de recomendações em relação à tecnologia, propõe-se também uma arquitetura de referência para ser utilizada no desenvolvimento de novos serviços.

Com isso, os objetivos específicos deste documento:

* Propor uma nova arquitetura de referência para processamento de Batch;
* Propor recomendações gerais em relação ao desenvolvimento de Windows Services
* Descrever as funcionalidades do MSMQ e SQL Service Broker, com o objetivo de orientar na decisão de qual tecnologia utilizar;

Na elaboração deste documento, foram levantados três domínios em relação ao processamento em Lote: Execução, Agendamento e Monitoramento. A arquitetura proposta visa atender os pontos relacionados com a execução.

1. Público Alvo

Este guia deve ser utilizado como referência por arquitetos e desenvolvedores especializados em projeto e codificação de aplicações para serem distribuídas como serviço Windows (*Windows Services Application*), desenvolvidos em C#, na plataforma Microsoft .NET 4.0.

1. Requisitos de Ambiente

Para a utilização das práticas adotadas neste guia, é considerada a execução dos serviços no seguinte cenário:

* **Tecnologias para processamento em fila**: MSMQ ou SQL Service Broker
* **Versão do Framework**: Net Framework 4.0
* **Versão do Banco de Dados**: SQL Server 2008 SP1 e SQL Server 2012
* **Sistema Operacional**: Windows Server 2008

1. Siglas

|  |  |
| --- | --- |
| Sigla | Descrição |
| SSiS | SQL Server Integration Services |
| DTC | Distributed Transaction Coordinator |
| DBA | Administradores do Banco de Dados |
| MSMQ | Microsoft Messaging Queuing |
| SCOM | System Center Operations Manager |

Tabela 1 Siglas utilizadas no documento

1. Visão geral de arquitetura

A figura a seguir representa a disposição dos servidores para o processamento em lote. Estes, por sua vez, fazem diversas validações do dado em serviços encontrados na DMZ (Web Service) ou outros sistemas internos. As recomendações direcionadas a comunicação e construção de serviços Web são encontradas no guia **SEFAZ – SP Guia de Desenvolvimento para Serviços Web***.*

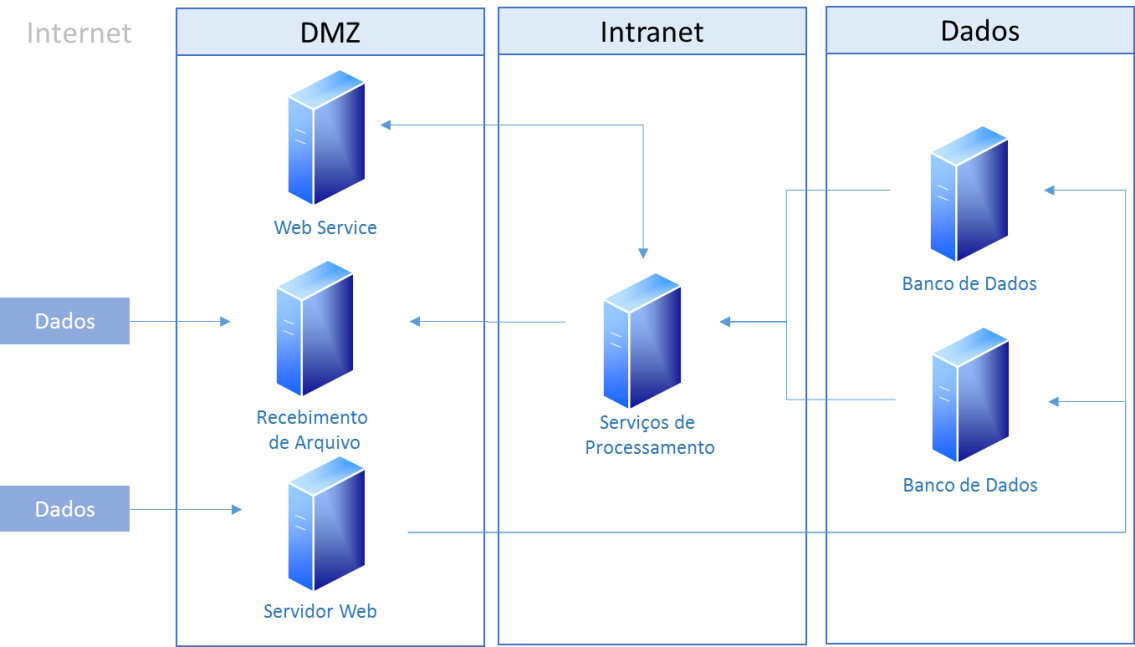


Figura 1 Arquitetura Física

Além disso, o mesmo dado pode ser encontrado para ser consumido em uma pasta compartilhada em um file system na DMZ. Neste cenário atual, o serviço que acessa a mesma e executa a cópia do arquivo para a intranet utiliza uma conta de serviço com permissões para executar tal ação.

Da mesma maneira que informações podem ser enviadas através de um site web e salvo direto no banco de dados. A existência deste tipo de comunicação deve ser avaliada e a análise desta deve ser endereçada pelo guia para serviços Web citado acima.

Neste cenário, as recomendações de arquitetura deste guia limitam-se a zona da intranet e ao processamento do arquivo em lote já recebido.

Um dos principais fatores que afetam o design de uma aplicação distribuída é a forma com que é desenhada a infraestrutura de comunicação para cada componente e como estes devem comunicar entre si. Na maioria das situações, surge a necessidade de se decidir entre implementar uma comunicação **direta** (como chamadas de métodos entre componentes) ou baseada em **mensagens**.

Dentre as vantagens encontradas na comunicação baseada em mensagens, pode-se citar a possibilidade de **desacoplar** os componentes da aplicação que não apenas visa melhorar a manutenção, como também modifica as estratégias de *deploy*. Contudo, a comunicação via mensagens traz novos desafios que precisam ser considerados como performance, confiabilidade e segurança e outros desafios que necessitam de avaliação.

Desenhar os componentes de forma **desacoplada** é um motivador para a distribuição em camadas independentes. O desenho almeja o máximo de coesão com o mínimo de acoplamento, onde componentes possuem suas responsabilidades e seguem o princípio que um não conhece os detalhes de implementação de outro. Nesta forma, é possível separa-los em camadas físicas distintas, já que independem entre si.

Com a separação física das camadas, a comunicação entre os componentes precisa ser avaliada e pode ser classificada em duas maneiras: **síncrona**, normalmente utilizada onde existe a necessidade de espera pelo retorno do método chamado. E **assíncrona**, onde este retorno não é necessário e é desejo ter componentes menos acoplados com possibilidade de maior desempenho no processamento.

Para ilustrar, temos o sistema abaixo com finalidade de executar processamento em lote de arquivos textos e enviá-los a um banco de dados.

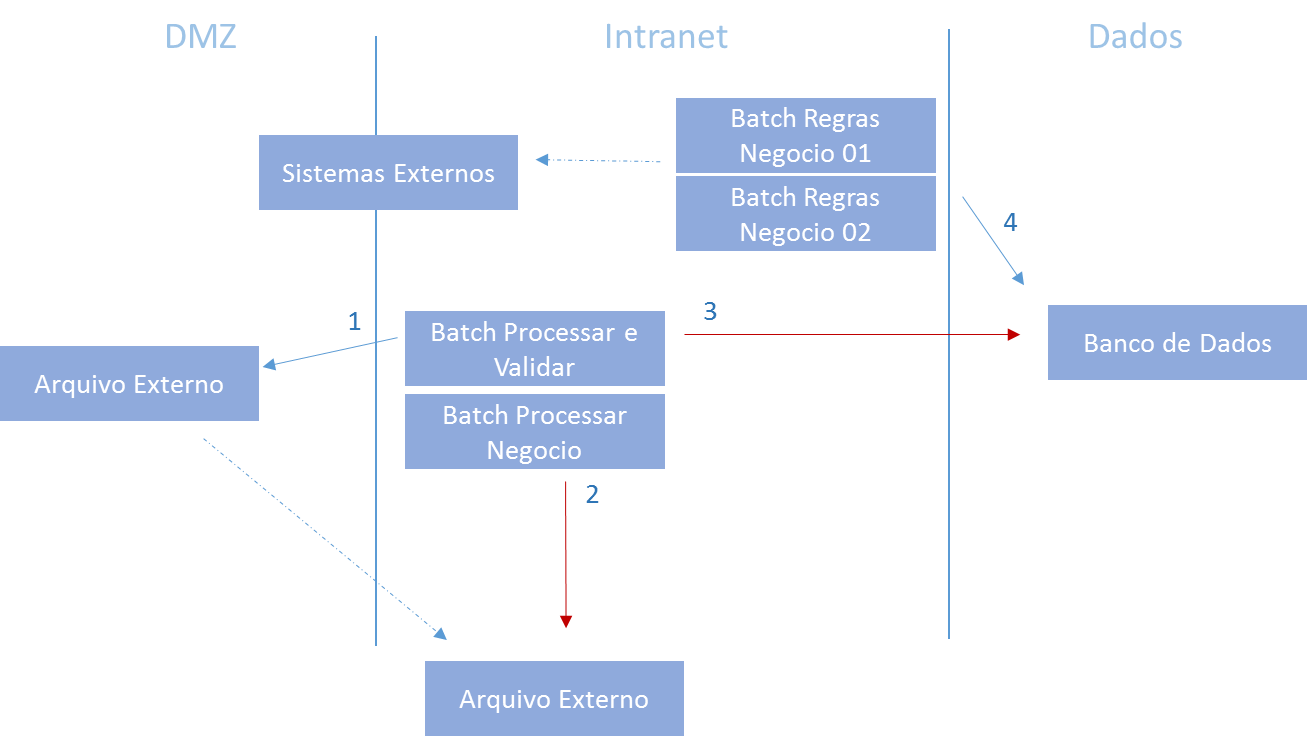


Figura 2 Sistema hipotético de processamento batch

Em uma visão macro, o processo inicia com a disponibilização os arquivos, de tamanhos variados, em uma pasta na rede. O primeiro serviço (1) é responsável por move-lo para a pasta local e executa com algumas validações simples.

Os arquivos aprovados na validação são movidos para uma segunda pasta onde serão validados baseado inúmeras regras de negócio (2) que podem ser quanto a lógica ou validado em outros sistemas através de conexão com banco de dados ou serviços Web.

Os registros aprovados são agrupados e enviados ao banco de dados (3) para serem processados por um conjunto de outros serviços. (4). Concluída esta etapa, o dado está pronto para ser consumido.

Neste macro cenário apresentado, o objetivo do negócio foi atendido. Os arquivos disponibilizados por um sistema externo foi processado, validado e incluído em seu destino final. Contudo, algumas questões sobre a arquitetura precisam ser endereçadas:

1. Como posso aumentar a capacidade de processamento (**escalabilidade**) para otimizar a velocidade das validações?
2. No caso de erros no processo, quais ações posso tomar para minimizar o impacto?
3. Como evitar a concorrência entre processos quando um tenta mover um arquivo enquanto outra tenta executar uma leitura de registros?

A necessidade de **escalabilidade** dos serviços é aparente quando se leva em consideração o aumento crescente de novas aplicações de processamento batch e a necessidade de escalar os recursos físicos para otimizar o tempo de processamento. Entretanto, incrementar os recursos físicos neste cenário apresentado se torna trabalhoso devido a dois motivos:

* Os serviços que executam o processamento Batch não foram desenvolvidos para uma arquitetura de servidores com balanceamento (**Cluster**)
* A falta de um repositório único de registros (**fila**) onde várias instâncias do mesmo serviço pudessem atuar de forma paralela para o processamento do mesmo

Desta maneira, aumentar o processamento dos serviços limita-se a uma escalabilidade vertical (memória e processador) a que horizontal (inclusão de novos servidores)

Um segundo ponto de análise é relacionado com a **tolerância a erros**. Devido a falta do repositório único e “centralizado” de registros com tolerância a falha implementada, faz-se necessário um esforço de desenvolvimento superior aos requisitos de negócio para que esse controle seja implementado. Com a possibilidade de existir uma fila de processamento de registros, a implementação deste tipo de controle ficaria a cargo do recurso e não do desenvolvedor.

No cenário apresentado, a necessidade de se executar **inúmeras cópias** do arquivo aumenta a utilização de recursos de I/O dos discos e um maior tempo de espera para o início do processamento. Em sistemas de processamento onde o tempo é relevante, este tipo de ação impacta diretamente no seu desempenho.

Além dos pontos citados acima, os seguintes aspectos são inerentes ao projeto e desenvolvimento deste tipo de sistema de processamento:

1. **Acoplamento** - O forte acoplamento entre os componentes (dados, serviço e destino) impede a separação em camadas físicas.
2. **Persistência** – Em algum momento do processo, em caso de falha, a retomada no mesmo ponto torna-se um desafio.
3. **Reprodução de erros** – Torna-se trabalho reproduzir o erro em um ambiente de teste
4. **Operação Transacional** – Em caso de uma falha de comunicação com o destino, não é possível efetuar um *rollback* da transação para posterior processamento.
5. **Comunicação** – Garantir a transmissão do dado mediante possíveis problemas nos meios de comunicação.
6. **Interoperabilidade** – Definir um padrão para o dado que facilite a comunicação com outros sistemas e diversas plataformas.

Sendo assim, quando é avaliado o aumento da disponibilidade; do aumento dos recursos; e questões relacionadas com acoplamento e persistência, os pontos citados precisam ser avaliados.

1. Recomendações
   1. Arquitetura lógica para processamento em lote

A proposta de arquitetura conceitual a seguir pretende endereçar os principais pontos de atenção levantados neste documento. Abaixo o modelo lógico atual da arquitetura:

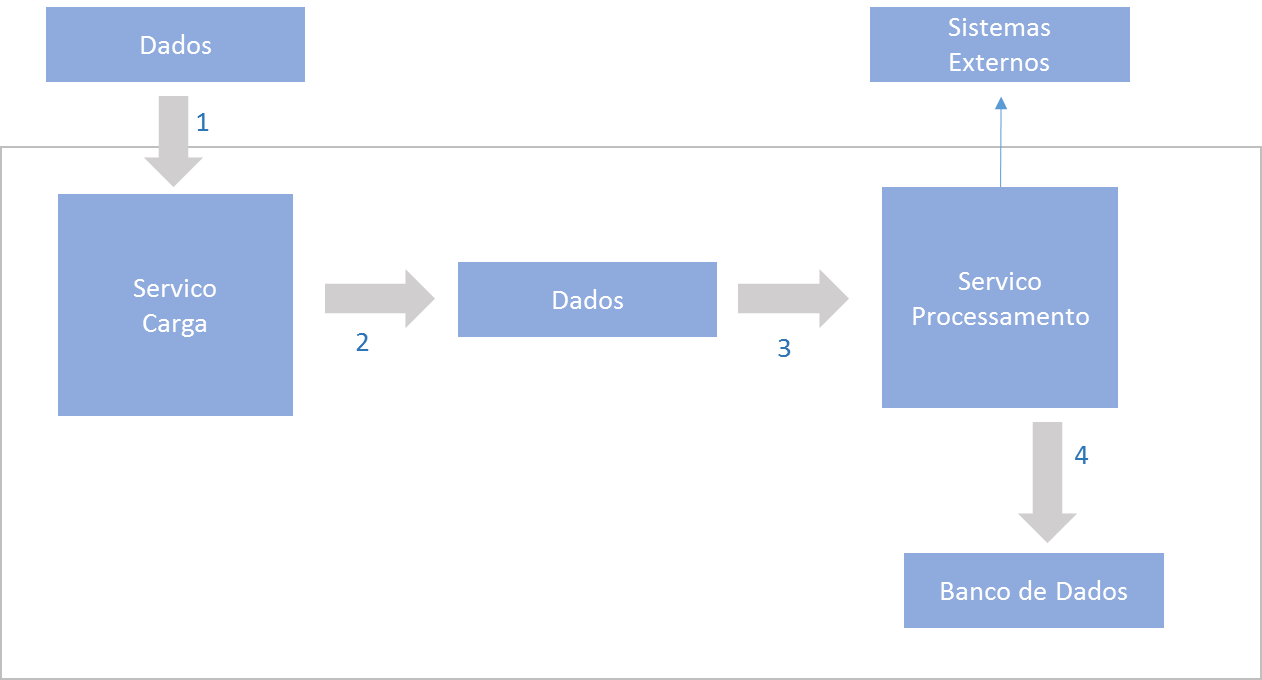


Figura 3 Atual Modelo lógico do processamento em Lote

Com base nas considerações levantadas neste documento, é proposto o seguinte modelo:

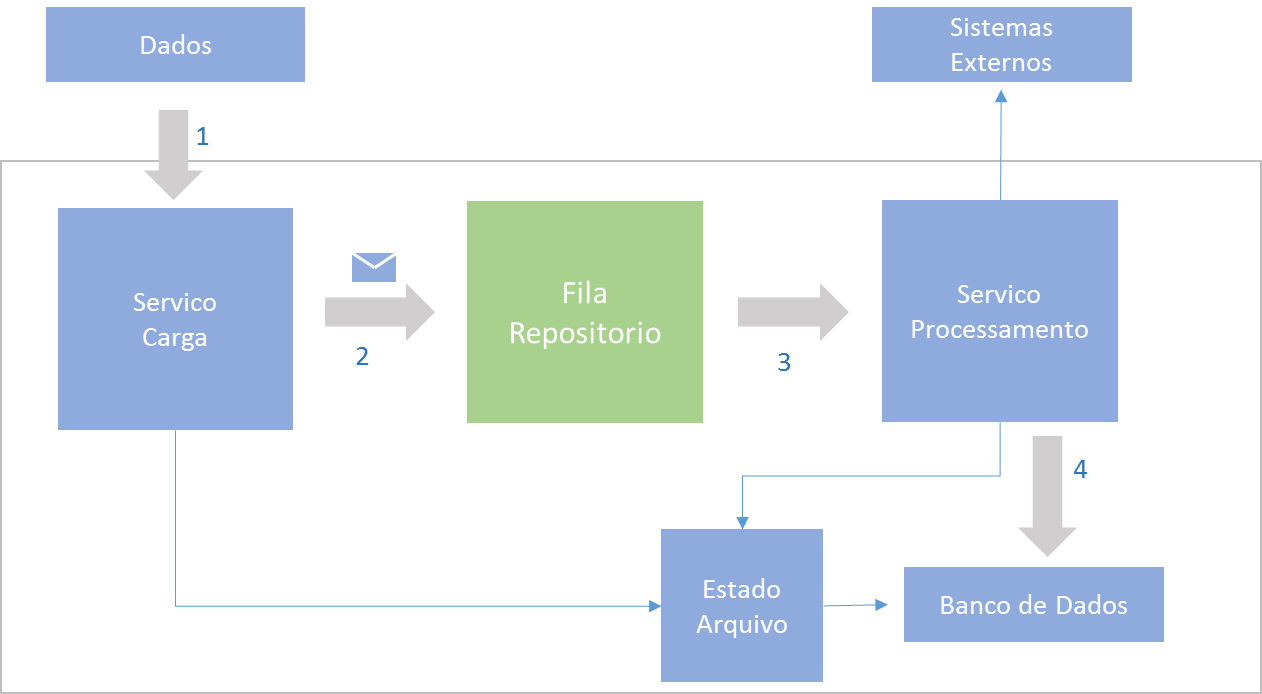


Figura 4 Proposta de modelo de arquitetura lógica

O diagrama representa a arquitetura lógica do processamento em lote de forma assíncrona. Abaixo, um resumo das principais entidades envolvidas neste desenho.

|  |  |
| --- | --- |
| Entidade | Função |
| Dados | Representa a informação a ser processada e validada pelos serviços. |
| Serviço Carga | Entidade responsável por:   * Copiar o arquivo localmente para otimizar o processamento * Executar validações relacionadas com o arquivo, tanto sua integridade física quanto pertinente ao negócio * Empacotar a informação no formato de mensagem e enviar para a tecnologia de fila |
| Fila Repositório | Entidade responsável por armazenar a informação validada no formato de mensagens. Dentre as principais funções, destacam-se:  - Implementação do processamento assíncrono  - Persistência dos dados  - Ordenação de mensagens e prioridade  - Tolerância a falhas  - Processamento transacional ou não-transacional  - Escalabilidade |
| Serviço Processamento | O serviço é hospedado em servidores de aplicação e responsável por processar as mensagens na fila. Estas mensagens podem conter o caminho físico dos arquivos, para um processamento em paralelo ou processar mensagens que contém registros em lote.    Nesta arquitetura, o serviço de processamento é hospedado em servidores de aplicação e estes são adicionados conforme a necessidade. |
| Banco de dados | Repositório final do dado processado |
| Sistemas Externos | Sistemas consultados durante o processamento para validação dos dados |
| Estado Arquivo | Entidade responsável por manter a etapa de processamento do arquivo e informações relevantes ao mesmo. |

Tabela 2 Principais entidades de arquitetura

Na visão MACRO do fluxo da informação deste modelo, a entidade **Carga** é responsável por executar a leitura e validação da fonte de **Dados.** Feita a validação, o serviço **envia** as **mensagens** para serem processadas na entidade **Fila**.

Neste modelo, cada serviço também é responsável por manter o **estado** de processamento do arquivo. Por exemplo, enquanto um serviço estiver executando validações, o arquivo em utilização fica com o estado de ocupado até que esteja livre para ser processado pelo próximo serviço. Desta forma, controlar este estado em banco de dados tem o objetivo de **reduzir a quantidade cópias de arquivos** entre diretórios, **evitar a concorrência** entre serviços e armazenar informações relevantes sobre os processamento e validação. A entidade **Estado Arquivo** é acionada sempre que for necessário.

Com as mensagens aguardando o processamento (**Queue**), a entidade **Fila** é responsável por gerenciá-las e de acordo com a tecnologia escolhida, traz os seguintes benefícios:

* **Possibilidade de gerenciar mensagens transacionais ou não-transacionais**

Entende-se como transacionais as mensagens que somente são descartadas quando a transação é finalizada. Caso contrário a mensagem é preservada para tentar ser entregue novamente.

* **Ordenação** e **prioridade** de mensagens

Mensagens podem precisar ser processadas em uma determinada ordem e com certa prioridade na fila.

* **Escalabilidade** de serviços e fila de processamento

Com a existência de uma fila, é possível ter várias instâncias de um mesmo processo para otimizar o tempo de processamento.

* “**Lock**” de mensagens, possibilitando a criação de mais para aumentar a velocidade do processamento

Proteção implementada pela fila para que uma mensagem somente seja processada por um único serviço

* Gerar **persistência** das mensagens, implementando desta forma uma tolerância a falhas.

Normalmente o servidor que hospeda a fila é encontrado em cluster e os dados são persistidos a fim de evitar perda em caso de falhas físicas.

O serviço **Processamento**pode ser representado por um ou mais serviços e é neste ponto onde a **escalabilidade horizontal** é atendida. Considerando que agora todo dado que precisa ser processado está disponível de forma isolada e em uma fila, várias instancias do serviço podem ser executadas para agilizar a operação.

O processamento das mensagens inclui validações com **Sistemas Externos**, normalmente representados por Banco de Dados e Serviços Web. Tendo o registro processado, o mesmo pode ser enviado ao repositório, representado por **Banco de Dados,** para que ocorram novas validações ou processado por outros serviços.

**Nova entidade: Estado Arquivo**

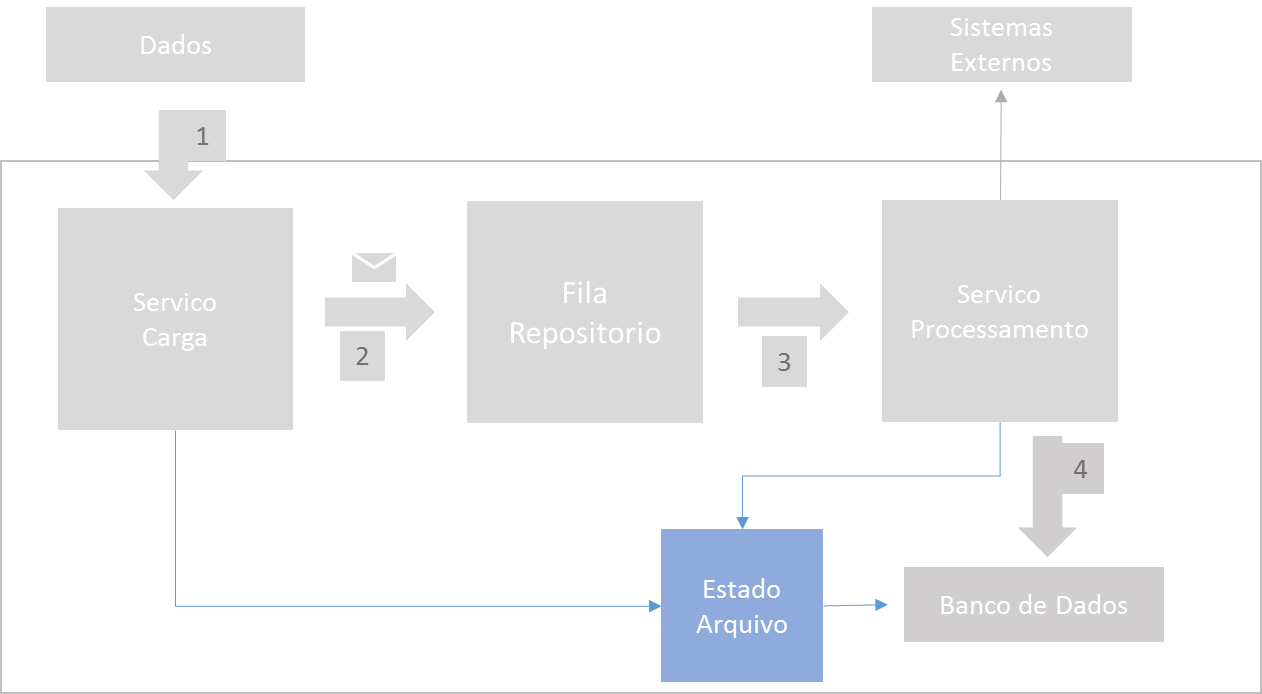
****

Figura 5 Entidade Estado Arquivo no contexto da arquitetura

**Estado Arquivo** foi acrescentado ao novo modelo de arquitetura para endereçar os pontos a seguir. A entidade é acessada pelos serviços de processamento existentes para incluir ou consultar informações referentes aos arquivos.

* **Retomada de processamento em caso de erro**

Como a entidade mantem a etapa em que se encontra o ciclo de processamento da informação, em caso de uma parada no ciclo (por exemplo, uma falha de hardware) , é possível retomar o processamento do ponto de parada.

* **Diminuir o número de cópias**

No modelo proposto, não é mais necessário executar diversas cópias do arquivo durante o ciclo já que os serviços passam a monitorar o estado e não a pasta em qual o arquivo se encontra. Com isso, pretende-se otimizar o tempo de processamento reduzindo o tempo gasto com a cópia dos arquivos.

* **Armazenar informações relevantes**

Além de armazenar a etapa de processamento, informações como tempo de processamento, quantidade de registros processamentos, tamanho do arquivo, dentre outras, podem ser salvas.

* **Manter histórico**

Alguns sistemas de processamento necessitam desta informação para saber como proceder em futuros processamentos, por exemplo, a informação que um arquivo rejeitado em uma validação anterior deve ficar armazenada, pois novos arquivos podem não processados enquanto o anterior ainda estiver rejeitado.

*Nota:*  
Existem alguns produtos de mercado que contém funcionalidades que atendem as características previstas nesta entidade. Enquanto o produto escolhido não estiver disponível para utilização na SEFAZ-SP, torna-se necessário o desenvolvimento desta entidade.

#### Cenário: Utilização do Service Broker como fila

O cenário abaixo visa ilustrar a utilização do modelo proposto para orientar desenvolvedores e arquitetos na utilização da arquitetura e o entendimento das entidades propostas.

**Requisitos**

|  |
| --- |
| Um sistema hipotético disponibiliza diversos **arquivos texto** em intervalo de tempo e quantidade irregulares em uma pasta da rede. Estes precisam ter seu layout e dados validos para poderem ser enviados para uma ou mais tabelas de banco de dados para consumo posterior.  Além disso, o **tempo de processamento** é um requisito **não-funcional** edeve ser levado em consideração. Assim, o tempo de leitura dos arquivos e o paralelismo do processamento dos mesmos é um fator relevante que a solução precisa endereçar.  As **validações** externas ocorrerem tanto em banco de dados quanto em serviços Web. Um fator importante em relação à validação é que se torna necessário saber quais arquivos foram rejeitados, pois alguns dependem do processamento de outros para executar. |

**Solução**A descrição abaixo considera o fluxo da informação entre as camadas da arquitetura para um melhor entendimento do modelo proposto.

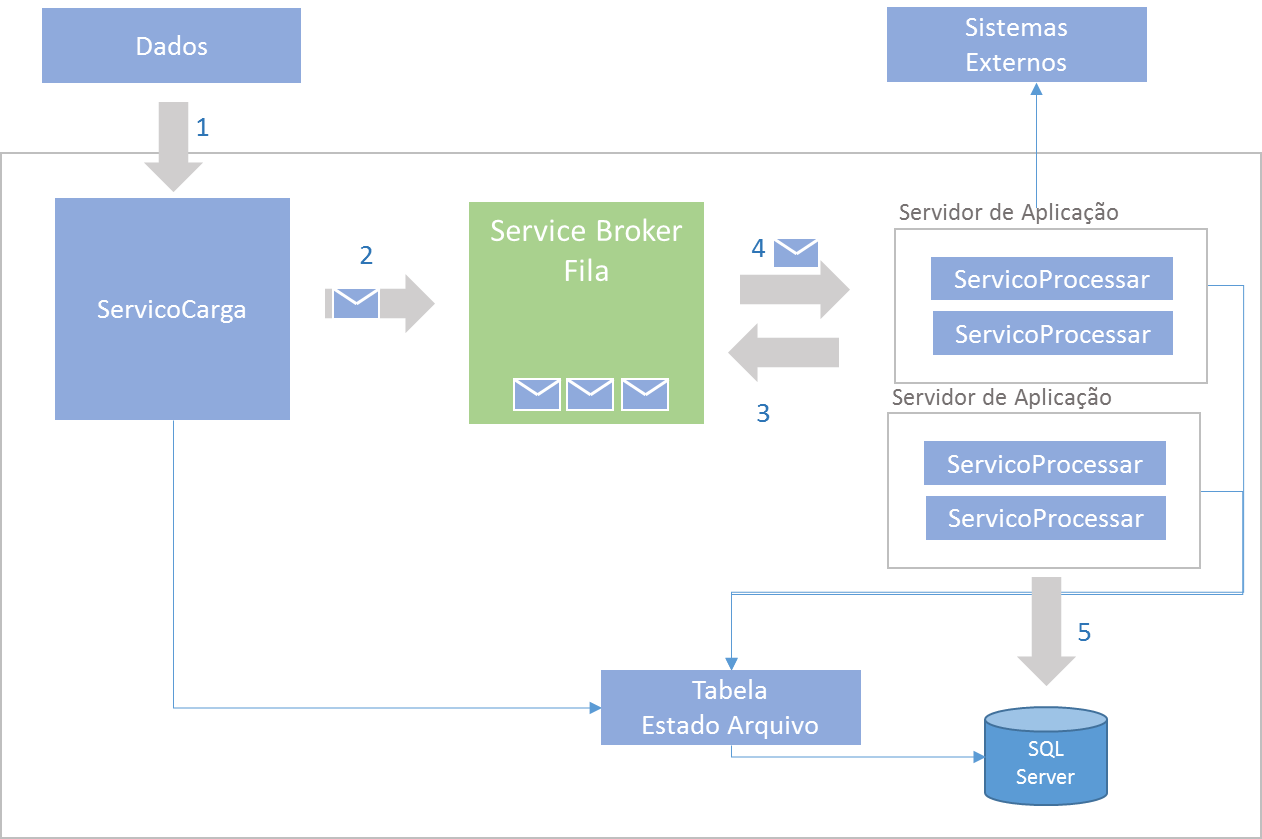


Figura 6 Visão geral do cenário

Estando o arquivo disponível na pasta da rede (**Dados**), um *Windows Services* chamado de **ServiçoCarga**, é responsável por validar o layout do arquivos (número de colunas esperadas) e se a informação é coerente (1). Neste momento, uma **tabela no banco de dados**, responsável por manter o histórico e etapa do arquivo é acionada pelo **ServicoCarga**, o qual salva informação sobre o mesmo como: *Nome do Arquivo,* *Etapa de Processamento*, *Hora Início*, dentre outras relevantes.

Sendo **validado**, sua *localização física (PATH)* e *nome* são encapsulados em forma de **mensagem** e enviada para a fila do **Service Broker** (2), no SQL Server. Mais uma vez o **Estado Arquivo** é atualizado com dados atuais e estado **“A Processar”.**

Cada tecnologia de fila possui uma característica específica para a **ordenação e prioridade das mensagens**. Assim, um ponto importante a considerar que as mensagens que necessitam de processamento em sequência podem ser geradas para atender tal função.

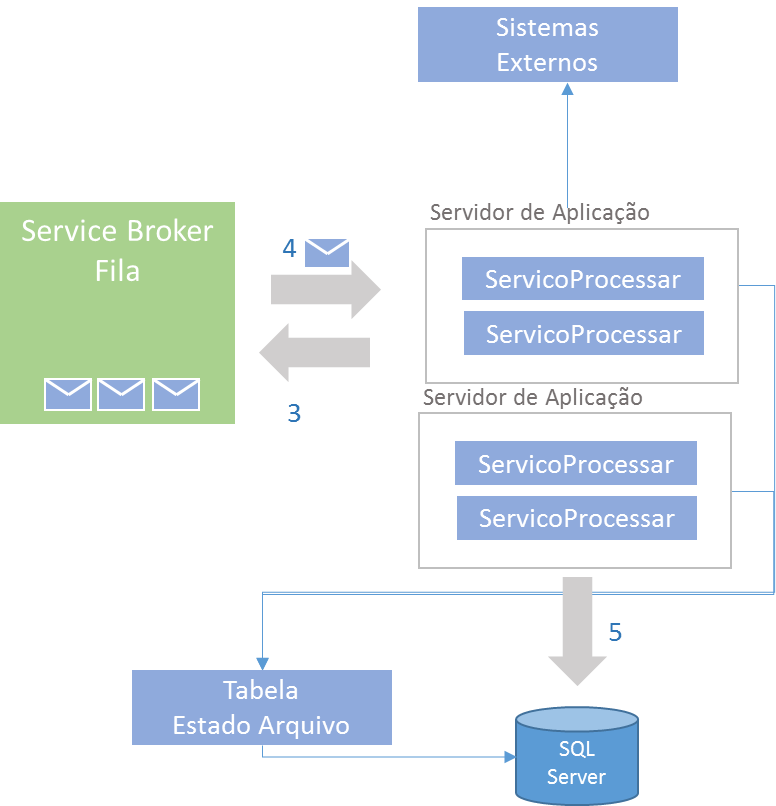


Figura 7 Detalhe do cenário - Troca de mensagens

Desta forma, os arquivos que necessitam de processamento aguardam na fila do Broker. De maneira nativa, é possível que o próprio servidor de Banco de Dados inicie o processamento das mensagens pendentes, desde que configurados para tal ação. Entretanto, devido a quantidade de mensagens para processar e a complexidade das regras de validação, a operação pode onerar o servidor de banco de dados, ocasionando em um possível problema de desempenho. Como solução, é recomendado o processamento dos arquivos em servidores de aplicação externos.

No SQL Broker, o processamento externo é um recurso do **Microsoft SQL Server 2008 Feature Pack** chamado *SQL BROKER External Activator.*

Assim, quando uma mensagem é recebida pela fila, o serviço de notificação (devidamente configuração no Broker) avisa os servidores de aplicação que existem mensagens que precisam ser processadas. Os serviços iniciam o processamento das mensagens, executando assim um paralelismo no processamento dos arquivos. O controle de “Lock” nas mensagens fica a cargo do Broker.

Um ou mais serviços são acionados para processar as mensagens **(3)**. Ao acessar a mensagem, é encontrado o **caminho do arquivo**. O serviço atualiza as informações referentes ao arquivo na tabela de estado e inicia o processo.

|  |
| --- |
| /\*SCRIPT DE TESTE DO SERVICE BROKER CONVERSA ENTRE MEMBROS DO MESMO BD\*/  --Ativação do Service Broker  use MessagingDB  -- Criação dos Tipos de Mensagens  CREATE MESSAGE TYPE  [//TESTSB/OneDBSample/RequestMessage] -- Nome URI  VALIDATION = WELL\_FORMED\_XML; /\* Validação que a mensagem quando enviada/recebida pelo serviço deve obedecer para ser aceita.\*/  CREATE MESSAGE TYPE  [//TESTSB/OneDBSample/ReplyMessage]  VALIDATION = WELL\_FORMED\_XML;  GO  -- Criação do Contrato  CREATE CONTRACT [//TESTSB/OneDBSample/SampleContract]  ([//TESTSB/OneDBSample/RequestMessage]  SENT BY INITIATOR,  [//TESTSB/OneDBSample/ReplyMessage]  SENT BY TARGET  );  GO  --INITIATOR  -- Criação da Fila  CREATE QUEUE InitiatorQueueOneDB;  -- Criação do Serviço  CREATE SERVICE  [//TESTSB/OneDBSample/InitiatorService]  ON QUEUE InitiatorQueueOneDB;  GO  -- TARGET  -- Criação da Fila  CREATE QUEUE TargetQueueOneDB;  -- Criação do Serviço  CREATE SERVICE  [//TESTSB/OneDBSample/TargetService]  ON QUEUE TargetQueueOneDB  ([//TESTSB/OneDBSample/SampleContract]);  GO  --Iniciar a conversa e enviar uma RequestMessage  --Declarando as variáveis do Initiator Dialog Handle e a mensagem em si  DECLARE @InitDlgHandle UNIQUEIDENTIFIER;  DECLARE @RequestMsg NVARCHAR(100);  BEGIN TRANSACTION;  -- Inicia o diálogo  BEGIN DIALOG @InitDlgHandle  FROM SERVICE  [//TESTSB/OneDBSample/InitiatorService]  TO SERVICE  N'//TESTSB/OneDBSample/TargetService'  ON CONTRACT  [//TESTSB/OneDBSample/SampleContract]  WITH  ENCRYPTION = OFF;  -- Escreve a mensagem  SELECT @RequestMsg =  N'<RequestMsg> Mensagem de teste. Enviando ao target.</RequestMsg>';  -- Envia a mensagem  SEND ON CONVERSATION @InitDlgHandle  MESSAGE TYPE  [//TESTSB/OneDBSample/RequestMessage]  (@RequestMsg);  -- Exibe a mensagem enviada  SELECT @RequestMsg AS MensagemEnviadaInitiator;  COMMIT TRANSACTION;  GO  -- Recebe a mensagem enviada e mandar uma resposta  -- Declaração do Dialog Handler de resposta, a mensagem e a resposta a ser enviada  DECLARE @ReplyDlgHandle UNIQUEIDENTIFIER;  DECLARE @RecievedMsg NVARCHAR(100);  DECLARE @ReplyMsgName sysname;  BEGIN TRANSACTION;  WAITFOR  ( RECEIVE TOP(1)  @ReplyDlgHandle = conversation\_handle,  @RecievedMsg = message\_body,  @ReplyMsgName = message\_type\_name  FROM TargetQueueOneDB  ), TIMEOUT 1000;  SELECT @RecievedMsg AS MensagemRecebidaTarget;  IF @ReplyMsgName =  N'//TESTSB/OneDBSample/RequestMessage'  BEGIN  DECLARE @ReplyMsg NVARCHAR(100);  SELECT @ReplyMsg =  N'<ReplyMsg>Mensagem recebida.</ReplyMsg>';  SEND ON CONVERSATION @ReplyDlgHandle  MESSAGE TYPE  [//TESTSB/OneDBSample/ReplyMessage]  (@ReplyMsg);  END CONVERSATION @ReplyDlgHandle;  END  SELECT @ReplyMsg AS MensagemEnviadaTarget;  COMMIT TRANSACTION;  GO  -- Receber a resposta e finalizar a conversa  -- Declarando a resposta recebida e o Dialog Handle  DECLARE @RecvReplyMsg NVARCHAR(100);  DECLARE @RecvReplyDlgHandle UNIQUEIDENTIFIER;  BEGIN TRANSACTION;  WAITFOR  ( RECEIVE TOP(1)  @RecvReplyDlgHandle = conversation\_handle,  @RecvReplyMsg = message\_body  FROM InitiatorQueueOneDB  ), TIMEOUT 1000;  END CONVERSATION @RecvReplyDlgHandle;  SELECT @RecvReplyMsg AS MensagemRecebidaInitiator;  COMMIT TRANSACTION;  GO  -- CÓDIGO PARA DELETAR OS OBJETOS DA CONVERSA  /\*  IF EXISTS (SELECT \* FROM sys.services  WHERE name = N'//TESTSB/OneDBSmaple/TargetService')  DROP SERVICE [//TEST/OneDBSample/TargetService]  IF EXISTS (SELECT \* FROM sys.services  WHERE name = N'//TESTSB/OneDBSample/InitiatorService')  DROP SERVICE [//TESTSB/OneDBSample/InitiatorService];  IF EXISTS (SELECT \* FROM sys.services  WHERE name = N'//TESTSB/OneDBSample/TargetService')  DROP SERVICE [//TESTSB/OneDBSample/TargetService];  IF EXISTS (SELECT \* FROM sys.service\_contracts  WHERE name = N'//TESTSB/OneDBSample/SampleContract')  DROP CONTRACT [//TESTSB/OneDBSample/SampleContract];  IF EXISTS (SELECT \* FROM sys.service\_message\_types  WHERE name = N'//TESTSB/OneDBSample/RequestMessage')  DROP MESSAGE TYPE [//TESTSB/OneDBSample/RequestMessage];  IF EXISTS (SELECT \* FROM sys.service\_message\_types  WHERE name = N'//TESTSB/OneDBSample/ReplyMessage')  DROP MESSAGE TYPE [//TESTSB/OneDBSample/ReplyMessage];  IF EXISTS (SELECT \* FROM sys.service\_queues  WHERE name = N'InitiatorQueueOneDB')  DROP QUEUE InitiatorQueueOneDB;  IF EXISTS (SELECT \* FROM sys.service\_queues  WHERE name = N'TargetQueueOneDB')  DROP QUEUE TargetQueueOneDB;  GO \*/ |

Tabela 3 Exemplo de código do *Service Broker* para envio e recebimento de mensagens

Durante as diversas etapas de processamento, é importante ressaltar que o estado do arquivo é atualizado sempre que um ciclo de processamento é finalizado. Quando todos os registros de um arquivo são processados, o **estado** é ajustado para **concluído** e o arquivo pode ser movido para um repositório de arquivamento.

Armazenar o estado do processamento dos arquivos não somente ajuda a controlar o fluxo da informação e as etapas de processamento quanto mantém um **histórico**. Neste cenário, possuir o histórico de um arquivo rejeitado, serve como parâmetro para uma futura validação, onde é requisito não processar se o anterior tiver sido rejeitado.

Em resumo do cenário, a utilização da fila atende dois principais pontos: Executar o processamento dos arquivos em paralelo e trazer todos os benefícios que a tecnologia da fila contém para auxiliar o processamento dos registros.

No entanto, o ponto que merece atenção neste cenário é a decisão de utilizar a fila para processar também os registros contidos nos arquivos. Alguns pontos precisam ser avaliados, por exemplo: O requisito de negócio, a quantidade de registros a serem processados em lote, a necessidade de uma resposta imediata durante alguma etapa de validação, aguardar a resposta de um sistema externo e dentre outros. Estes precisam ser analisados em conjunto para que a decisão de se utilizar o processamento assíncrono para registros seja validada.

* 1. Recomendações para Windows Services
     1. Modelo de desenvolvimento

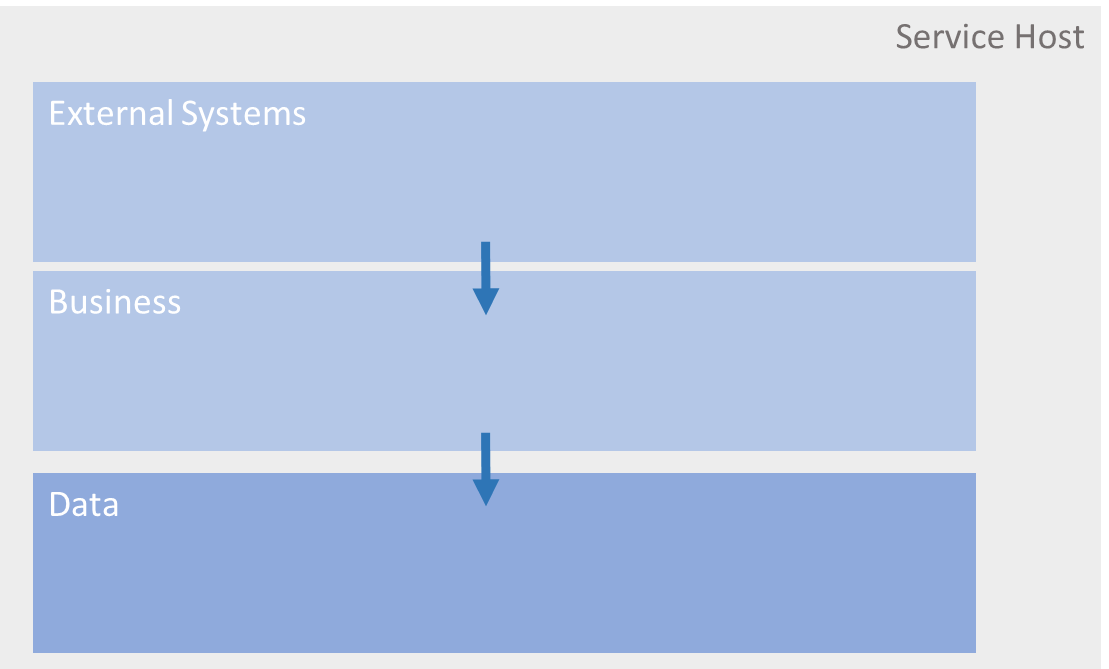


Figura 8 Padrão de desenvolvimento Windows Services

Com o objetivo de desacoplar os componentes e definir um padrão de desenvolvimento de código, é recomendado que o **desenvolvimento** de Windows Services seja baseado em camadas, seguindo as orientações encontradas no **“Guia de Desenvolvimento para plataforma .NET”.**

De uma maneira geral, os serviços desenvolvidos podem ser separados nas camadas acima. As classes relacionadas com o Windows Services estariam na camada *External Systems*. Regras de processamento e validação na camada *Business* e acesso a banco de dados e envio/leitura de mensagens na camada *Data*. Os detalhes da implementação estão descritos no guia citado.

* + 1. Definir uma conta de Windows para a execução de cada serviço

É recomendando definir uma conta Windows para que o serviço execute neste contexto. Está prática visa a rastreabilidade e segurança.

O contexto de segurança de um Windows Services é modificado manipulando a propriedade [*Account*](http://msdn.microsoft.com/en-us/library/system.serviceprocess.serviceprocessinstaller.account(v=vs.100).aspx)*.*É possível adicionar 4 tipos de contas:

|  |  |
| --- | --- |
| Conta | Descrição |
| User | Resulta na abertura de uma caixa de diálogo solicitando as credenciais do usuário no momento que o serviço é instalado e executado. Este é o padrão. |
| LocalService | Executa em no contexto de uma conta sem privilégios e se apresenta como anônimo para qualquer servidor remoto  *Obs: Disponível somente no Windows XP e família Windows Server 2003* |
| LocalSystem | Executa no contexto de uma conta com privilégios e se apresenta com as credenciais do servidor para qualquer servidor remoto |
| NetworkService | Executa no contexto de uma conta sem privilégios e se apresenta com as credenciais do servidor para qualquer servidor remoto  *Obs: Disponível somente no Windows XP e família Windows Server 2003* |

Tabela 4 Contextos de segurança de um Windows Services

* + 1. Certificar que os serviços que acessam pastas no servidor possuam somente a permissão necessária.

As contas de serviço utilizadas para acessar recursos locais do servidor, como File System, devem possuir somente as permissões necessárias para executar a ação, por exemplo, leitura e escrita.

Caso a conta de serviço necessite de maiores permissões no servido, não atribua à conta permissões administrativas.

* + 1. Adicionar o termo “Exception” ao final do nome da classe.

Quando existe a necessidade de se criar classes para gerenciar exceções customizadas, é uma *prática de mercado* adicionar o sufixo ***Exception*** ao final de seu nome. Esta prática visa facilitar a identificação das classes de exceções customizadas no projeto.

* + 1. Desenhar o serviço sem interface com o usuário

Pelo seu papel desempenhado, este tipo de serviço é desenvolvido para executar em segundo plano no Sistema Operacional e não necessita interface com o usuário. O serviço não possui características para ser interativo e é desenhado para executar sem intervenção de usuário.

* + 1. Remover todas as referências do método Assert do serviço antes de publicar em produção

Este método mostra uma caixa de diálogo que contém a linha de código que falhou e dá a opção para executar a depuração (*debbuging)* no mesmo.

* + 1. Implementar pelo menos os métodos OnStart e OnStop

Para que um serviço atenda os mínimos dos requisitos de funcionamento, o desenvolver deve implementar, no mínimo, ações nos métodos *OnStart* e *OnStop.*

A tabela abaixo indica quais comportamentos do serviço podem ser sobrescritos:

|  |  |
| --- | --- |
| Método | Indicação |
| [OnStart](http://msdn.microsoft.com/en-us/library/system.serviceprocess.servicebase.onstart(v=vs.100).aspx) | Indicado para executar ações quando o serviço iniciar. |
| [OnPause](http://msdn.microsoft.com/en-us/library/system.serviceprocess.servicebase.onpause(v=vs.100).aspx) | Indicado para executar ações quando o serviço por pausado. |
| [OnStop](http://msdn.microsoft.com/en-us/library/system.serviceprocess.servicebase.onstop(v=vs.100).aspx) | Indicado para executar ações quando o serviço parar de executar. |
| [OnContinue](http://msdn.microsoft.com/en-us/library/system.serviceprocess.servicebase.oncontinue(v=vs.100).aspx) | Indicado para executar ações quando o serviço voltar a funcionar, após ter sido pausado. |
| [OnShutdown](http://msdn.microsoft.com/en-us/library/system.serviceprocess.servicebase.onshutdown(v=vs.100).aspx) | Indicado para informar ao serviço o que ele deve fazer caso o sistema operacional seja desligado. |
| [OnCustomCommand](http://msdn.microsoft.com/en-us/library/system.serviceprocess.servicebase.oncustomcommand(v=vs.100).aspx) | Indicado para executar ações quando um serviço receber um comando customizado.  ***Observação****: O serviço só pode receber valores compreendidos entre 128 e 256 (integer) Valores menores de 128 são reservados.* |
| [OnPowerEvent](http://msdn.microsoft.com/en-us/library/system.serviceprocess.servicebase.onpowerevent(v=vs.100).aspx) | Indicado para informar ao serviço como ele deve agir quando um evento, relacionado com energia. |

Tabela 5 Comportamentos de um Serviço

* + 1. Não armazene informações sensíveis (confidencias) nos logs.

Antes de selecionar um dado para ser salvo no LOG, tenha certeza que a informação não é sensível, ou seja, sem requisitos de confidencialidade.

* + 1. Não utilizar SQL Jobs para processamento em Lote

É comum encontrar o uso de Jobs criado no SQL Server para um processamento complementar a aplicações de lote. Entende-se como **SQL Jobs** como uma série especificada de operações executadas sequencialmente pelo **SQL Server Agent**.

Normalmente, a utilização de **Jobs** no SQL Server é destinado a administração do banco de dados e as tarefas criadas e agendadas são de responsabilidade dos **administradores do banco de dados (DBA)**. São exemplo de tarefas administrativas são: Execução de Backups, manutenção em tabelas, índices, etc.

Desta forma, é necessário avaliar a utilização deste recurso como parte complementar a aplicações.

Recomenda-se que nas operações onde ocorram copias entre tabelas, transformação de dados, importação ou exportação, se utilize como tecnologia os pacotes do **Integration Services** (**SSIS).**  Estes pacotes são desenvolvidos pelo time de desenvolvimento e enviados aos DBA’s para sua publicação, mediante criação de processo de incidente/solicitação.

A utilização de pacotes SSIS trazem performance a operação e facilitam a administração pela equipe de Banco de Dados e é previsto a utilização da ferramenta *Control-M* para o monitoramento e agendamento dos mesmos.

* + 1. Ajuste a propriedade Autolog para “false” caso deseja gravar logs no Event Viewer

Caso seja necessário gravar informações sobre o serviço no Event Viewer, faz-se necessário ajustar para *False* a propriedade **Autolog.**

1. Referências

|  |  |
| --- | --- |
| Descrição | Url |
| Como extrair o log de erros do Service Broker | [How to: Retrieve Information from a Service Broker Error Message (Transact SQL)](http://technet.microsoft.com/en-us/library/ms166041(v=sql.100).aspx) |
| Depurando o MSMQ | [Troubleshooting Message Queuing](http://technet.microsoft.com/en-us/library/cc770347.aspx) |
| Construir um Windows Services com projeto de instalação | [Walkthrough: Creating a Windows Service Application in the Component Designer](http://msdn.microsoft.com/en-us/library/zt39148a(v=vs.100).aspx) |
| Depurar um serviço do Windows | [How to: Debug Windows Service Applications](http://msdn.microsoft.com/en-us/library/7a50syb3(v=vs.100).aspx) |
| Sobre o uso do Debug.Assert | [Debug.Assert Method](http://msdn.microsoft.com/en-us/library/system.diagnostics.debug.assert(v=vs.100).aspx) |
| Arquitetura de programação de um serviço | [Service Application Programming Architecture](http://msdn.microsoft.com/en-us/library/yzk7ksy2(v=vs.100).aspx) |
| Como gravar mensagens no Event Viewer | [How to: Log Information About Services](http://msdn.microsoft.com/en-us/library/f6567h1s(v=vs.100).aspx) |
| Sobre o contexto de segurança dos serviços | [How to: Specify the Security Context for Services](http://msdn.microsoft.com/en-us/library/0x72fzyf(v=vs.100).aspx) |
| Enumerações do contexto de segurança | [ServiceAccount Enumeration](http://msdn.microsoft.com/en-us/library/system.serviceprocess.serviceaccount(v=vs.100).aspx) |

Tabela 6 Tabela de Referências

# Apêndice A - Comparação entre MSMQ e Service Broker

**MSMQ** (Microsoft Messaging Queue) e **SQL Server Service Broker** são tecnologias Microsoft para mensageria e processamento assíncrono de dados. Ambos compartilham características em comum como tolerância a falhas, escalabilidade, dentro outras já citadas em capítulos anteriores.

Entretanto, existem pontos que precisam ser avaliados não apenas pelo time de arquitetura, mas também pela equipe de infraestrutura quando a implementação da tecnologia e o alinhamento a estratégia. Tomando por exemplo o **MSMQ**. A instalação de uma instancia de *queue* adiciona mais um procedimento a rotina de backup de ambiente se comparar com uma instancia do **Service Broker**, que por funcionar em conjunto com a instancia de banco de dados, seu backup segue as mesmas regras já definidas para o servidor.

Assim, além de comprar algumas diferenças entre as tecnologias, visam auxiliar no processo de utilização da mesma.

* **Limite de Mensagem**

As mensagens enviadas possuem um limite em relação ao seu tamanho. No **MSMQ** esse limite é de **4 Megabytes** e no **Service Broker** de **2 Gigabyte**.

* **Mensagens transacionais**

São consideradas mensagens transacionais aquelas que possuem as seguintes características: Executar todas as ações de uma transação ou em caso de erro, não executar nada e manter a ordem que as mensagens foram enviadas.  
  
Devido a este controle, as mensagens transacionais possuem um desempenho inferior se comparadas com mensagens não transacionais. Entende-se que nem todo requisito solicita a utilização de mensagens transacionais.

No **MSMQ**, já a possibilidade de envio de mensagens transacionais e não-transacionais, enquanto quem no **Service Broker**, somente as transacionais.

Em compensação, quando existe a necessidade de mensagens transacionais e conforme o volume de mensagens a serem processadas, o **Service Broker** mostra-se mais eficiente já que o contexto é gerenciado internamente pelo SQL Server. As transações no MSMQ são coordenadas pelo DTC e este realiza a transação em duas fases.

* **Custo**

O **MSMQ** é uma feature que precisa ser habilitada no sistema operacional (Windows Server). O **Service Broker** faz parte do SQL Server e necessita ser habilitado em um database

* **Comunicação entre aplicações**

Com o **MSMQ** é possível executar uma troca de mensagens, entre duas aplicações, sem a necessidade de se conectar com o banco de dados SQL Server. Em contrapartida, utilizando o **Service Broker**, sempre a aplicação necessita se conectar com o banco.

* **Escalabilidade**

Ambas tecnologias suportam a escalabilidade. A principal diferença entre elas é que no **Service Broker**, um recurso chamado *Activation* escala o *Queue Reader* (leitor da fila) automaticamente quando detecta que o processamento corrente não está sendo suficiente para processar as mensagens e automaticamente cria outros processos em paralelo de leitura. No **MSMQ**, os servidores disponíveis para o serviço sempre utilizam o processamento disponível.

* **Localização do Código**

O processamento de mensagens no **MSMQ** necessita que sejam desenvolvidas aplicações externas para a ação. O **Service Broker** dispensa tal desenvolvimento já que utiliza toda *engine* interna do SQL Server para processar e receber as mensagens devido ser desenvolvido em T-SQL.

A escolha do **Service Broker** traz benefícios em aplicações onde não existam validações com sistemas externos e todas as operações necessárias ocorram no banco de dados.

Apêndice B - Guia para monitoramento de Windows Services

Apesar das recomendações abaixo se destinam a equipe de monitoramento, as mesmas devem ser avaliadas durante o desenvolvimento de aplicações do tipo Windows Services para que sejam detectados futuros problemas.

#### Serviço Carga

* + Verificar se o serviço está funcionando (Start/Stop)
  + Verificar se a conta de serviço possui permissões apropriadas permissão a pasta do arquivo
  + Verificar se o arquivo existe
  + Verificar se o arquivo não está corrompido
  + Verificar se a cópia do arquivo já foi concluída
  + Verificar se o arquivo está com o layout apropriado. Aplicação deve validar o mesmo
  + Verificar se o servidor de File System está disponível
  + Verificar se o servidor de Fila está disponível
  + Verificar se o aplicativo está gravando dados de log
  + Verificar se os serviços externos estão disponíveis

#### Serviço Processamento

* + Verificar se o serviço está funcionando
  + Verificar se o servidor de fila está disponível
  + Verificar se existem mensagens no QUEUE a serem processadas
  + Verificar se a Fila está ativa
  + Verificar se o destino está disponível
  + Verificar as permissões na conta de serviço no destino
  + Avaliar o tempo de processamento das mensagens

#### Fila Repositório

* + Verificar a quantidade de mensagens na fila
  + Avaliar o tempo das mensagens na fila. Muito tempo pode indicar um problema em um dos serviços
  + Verificar as mensagens que apresentaram erro quanto ao processamento.
  + Verificar os logs do serviço
  + Monitorar a quantidade de mensagens em QUEUE
  + Verificar se o serviço está disponível
  + Verificar o meio de comunicação entre os servidores