6
O Arcabouço de Serviços

6.1
O Arcabouço

Um arcabouço (*framework*) pode ser entendido como:

> *"um conjunto de classes que constituem um design abstrato para soluções de uma família de problemas."* [18]
>
> *"um conjunto de objetos que colaboram com o objetivo de cumprir um conjunto de responsabilidades para uma aplicação ou um domínio de um subsistema."* [19]
>
> *"uma arquitetura desenvolvida com o objetivo de se obter a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização."* [20]

As principais características esperadas de um arcabouço são ilustradas pela habilidade do mesmo de ser [21]:

- Reutilizável (bem documentado e fácil de usar);
- Extensível (há possibilidade de adicionar funcionalidades);
- Seguro (evita a destruição pelo desenvolvedor);
- Eficiente (facilita e agiliza o processo de desenvolvimento);
- Completo (endereça todo o domínio pretendido);
- Define e controla o fluxo da aplicação;
- Comanda a resposta aos eventos externos.

Não se pode deixar de mencionar algumas vantagens de utilizar o arcabouço proposto como, por exemplo, sua capacidade praticamente ilimitada para processamento das simulações, inerente à escalabilidade da nuvem. O tempo da simulação, por sua vez, também é vantajoso, já que é proporcional ao investimento desejado, em razão do modelo de cobrança da nuvem.

É bom observar que o usuário final dos serviços não precisa entender as especificidades da solução, pois utiliza o serviço provido pela nuvem de forma fácil e independente do seu domínio ou tecnologia.

Vale mencionar, ainda, o fato de o arcabouço permitir a programadores, de forma facilitada, a sua implementação e extensão para atender tecnologias, domínios e negócios de seu interesse, sendo transparente a paralelização do processamento.

6.2 Prontos de Extensão

O arcabouço proposto possui os seguintes pontos de extensão (*hotspots*):

1) **Executa Simulação (*execute*):** Esse ponto de extensão permite suportar o processamento da simulação de acordo com diferentes tecnologias e domínios, de forma a obter os resultados parciais corretos.
2) **Combina Resultados (*finish*):** Esse ponto de extensão permite apoiar o processamento dos resultados da simulação de acordo com diferentes tecnologias e domínios, de forma a obter o resultado final desejado.
3) **Validação das Entradas (*test*):** Esse ponto de extensão permite testar as informações de entrada do cliente, de forma a não permitir a execução de simulações inválidas de acordo com o domínio e tecnologias.
4) **Executa Otimização (*optimization*):** O desempenho da simulação (custo x tempo) está relacionado ao número de nós envolvidos no processamento e à estratégia de divisão da simulação em tarefas. A implementação deste ponto deve ser analisada com cuidado frente ao domínio a ser atendido. Além disso, o tempo máximo de execução de uma tarefa também influencia no controle de falhas.
5) **Instala requisitos (*startup*):** De acordo com as tecnologias e domínios pode ser necessário instalar requisitos nos nós. Esse ponto de extensão permite realizar essa instalação.

Os pontos de extensão ficam isolados do restante do código do McCloud em um projeto específico (*McHotspot*). A Figura 16 apresenta a interface de implantação dos quatro primeiros pontos.

```
namespace McHotspot
{
    public interface McHotspootI
    {
        string execute(string key, double n, double index, string codein, out string message);
        string finish(string key, double n, string r, string codein, string codeout, out string message);
        string test(string key, double n, string codein, string codeout);
        void optimization(double n, string codein, string codeout,
            out double ninstancesadded, out double ntasks, out int timeoutInSeconds);
    }
}
```

Figura 16 – Interface dos Pontos de Extensão

Quando necessário aos pontos de extensão podem ser incluídas referências e outros recursos neste projeto. O arcabouço utiliza o padrão de projeto *FactoryMethod* [38] para carregar a implantação desta interface escolhida pelo cliente.

A interface de cada ponto de extensão é explicado a seguir:

1) O ***execute*** recebe o identificador único da simulação (*key*); o total de realizações a ser executado, nesse caso, o total de responsabilidade da tarefa/nó (*n*); o índice da tarefa (*índex*); e o código que subsidia a execução (*codein*). Retorna em texto o resultado do processamento da tarefa, ou seja, parte da amostra. Além disso, pode retornar uma mensagem de caráter de instrumentação.
2) O ***finish*** recebe o identificador único da simulação (*key*); o total de realizações da simulação (*n*); o endereço local do arquivo com o resultado concatenado de todas as tarefas (*r*); o código que subsidiou a execução (*codein*); e o código que subsidia o processamento para obtenção do resultado a partir da amostra completa (*codeout*). Retorna em texto o resultado da simulação. Além disso, pode retornar uma mensagem de caráter de instrumentação.
3) O ***test*** recebe o identificador único da simulação (*key*); o total de realizações da simulação (*n*); o código que subsidiará a execução (*codein*); e o código que subsidiará o processamento para obtenção do resultado a partir da amostra completa (*codeout*). Retorna vazio se não ocorreu erro, ou retorna o texto explicando o erro.
4) O ***optimazation*** recebe o total de realizações da simulação (*n*); o resultado concatenado de todas as tarefas (*r*); o código que subsidiou a execução (*codein*); e o código que subsidiará o processamento para obtenção do resultado a partir da amostra completa (*codeout*). Retorna o número de instâncias a serem configuradas adicionalmente para suportar

a simulação da forma ótima (*ninstanceadded*), o número de tarefas ótimo (*ntasks*) e o tempo a ser esperado pelo processamento de uma tarefa depois de capturada da fila (*timeout*).

O *codein* e o *codeout*, acima citados, podem ser os códigos integralmente, parcialmente ou mesmo parâmetros para os mesmos.

O ponto de extensão ***startup*** não se encontra na interface, pois deve ser implementado através da escrita de instruções para ambiente Windows em linha de comando no arquivo "startup.cmd" deste mesmo projeto, onde *downloads* e instalações podem ser orquestradas. Além disso, os arquivos do contêiner com nome "*onstart*", caso existirem, são automaticamente copiados para a pasta de armazenamento de cada instância, ficando disponíveis para as implementações da interface.

Existem soluções extremamente amigáveis para gerenciamento dos armazenamentos do Azure, por exemplo, o *Azure Explorer* [22]. Outra opções específicas são *CloudBerry Explorer* [41] para blobs e *TableXplorer* [42] para tabelas. Essas ferramentas são muito úteis para o envio de arquivos para os containers, acompanhamento das mensagens que trafegam na fila e visualização das tabelas.

O entendimento da implementação dos pontos de extensão fica mais claro nas próximas seções, onde exemplos práticos são apresentados.

Cabe observar que apesar de contemplado no ponto de extensão *optimazation*, na versão atual do arcabouço, o levantamento automático de novos nós (*instancesadded*) e sua posterior liberação, não ocorre de forma automática. Necessário se faz realizar esse gerenciamento manualmente através da alteração do arquivo de configuração a ser explicado na seção 6.5. O mesmo ocorre com o tempo limite de processamento de uma tarefa, que também precisa ser configurado manualmente.

6.3 Os Serviços

Serviço é definido no dicionário como *"O desempenho de trabalho (uma função) por alguém para outro"*. *Web Service* [23] é uma solução utilizada na integração e comunicação de sistemas, permitindo integrar novas soluções a

sistemas existentes, independente da tecnologia. Isto porque, os sistemas se comunicam em formato universal (XML).

Com o uso desta tecnologia é possível desacoplar dos sistemas suas simulações e essas, por sua vez, podem ser oferecidas em ambiente de computação na nuvem como um leque de serviços para diversos domínios.

Dessa forma, essas aplicações poderão consumir estes serviços, mediante o envio de uma série dados necessários à sua execução, recebendo o resultado da simulação solicitada.

Como mencionado, o papel WCF Role é uma interface pública baseada no *Windows Communication Foundation*, modelo unificado de programação para construir de forma rápida aplicações distribuídas, com arquitetura orientada a serviço (SOA). Esse papel permite ao McCloud disponibilizar um *Web Service* com três métodos, conforme apresentados na Figura 17.

```
[ServiceContract]
public interface IService
{
    [OperationContract]
    string Run(double n, string codein, string codeout);

    [OperationContract]
    string Result(string key);

    [OperationContract]
    string Check(string key);
}
```

Figura 17 – Protótipo dos métodos do McCloud

O **método *Run*** requer três parâmetros: o total de realizações da simulação (*n*); o código que subsidia a execução (*codein*); o código que subsidia o processamento para obtenção do resultado a partir da amostra completa (*codeout*). Esse método retorna uma chave única (*key*) que identifica a simulação solicitada. A chamada a esse método é o primeiro passo da etapa *split*.

O **método *Check*** recebe a chave única (*key*) e retorna a situação atual da simulação, que pode ser: "*processing*" (as tarefas estão sendo processadas pelos nós, nesse caso, é também indicado o percentual de tarefas já executadas até o momento); "*merging*" (a etapa de *process* finalizou e a etapa de *merge* da simulação está em andamento); "*finished* " (a simulação acabou e o resultado da simulação está disponível para download).

Finalmente, o **método *Result*** recebe a chave única (key), que identifica uma simulação e retorna o endereço para *download* do resultado em arquivo textual. Caso a simulação não tenha terminado retorna vazio.

É ideal que o volume de dados de entrada e saída nesses métodos seja insignificante frente ao volume processado (lido, calculado, criado, etc.) tendo em vista o custo e tempo de transferência envolvido.

6.4
Casos de Uso

O arcabouço, conforme Diagrama de Casos de Uso apresentado na Figura 18, possui cinco casos de uso, sendo três disponibilizados para as aplicações clientes como serviços, pelos nós que exercem o papel de comunicador (*WCF Role*), e outros dois realizados internamente pelos nós que exercem o papel de trabalhador (*Worker Role*).

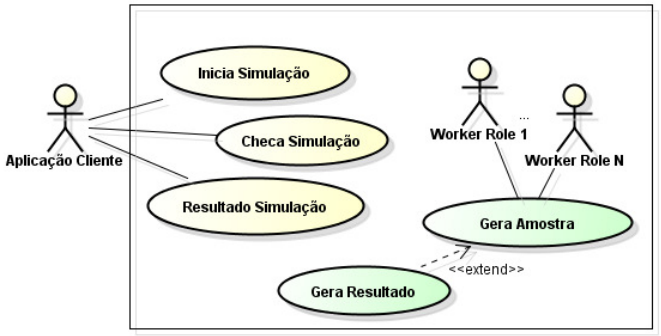

Figura 18 – Diagrama de Casos de Uso

Os casos de uso “Gera Amostra”, “Gera Resultado”, “Inicia Simulação” são ilustrados, respectivamente, através dos fluxos das etapas *process*, *merge* e *split,* no capítulo 5. Os casos de uso “Inicia Simulação”, “Checa Simulação”, “Resultado Simulação” correspondem, respectivamente, à chamada dos métodos *Run*, *Check* e *Result*, já analisados.

6.5
Configuração e Instrumentação

O arcabouço se encontra estruturado em 7 pacotes, conforme ilustrado a seguir.

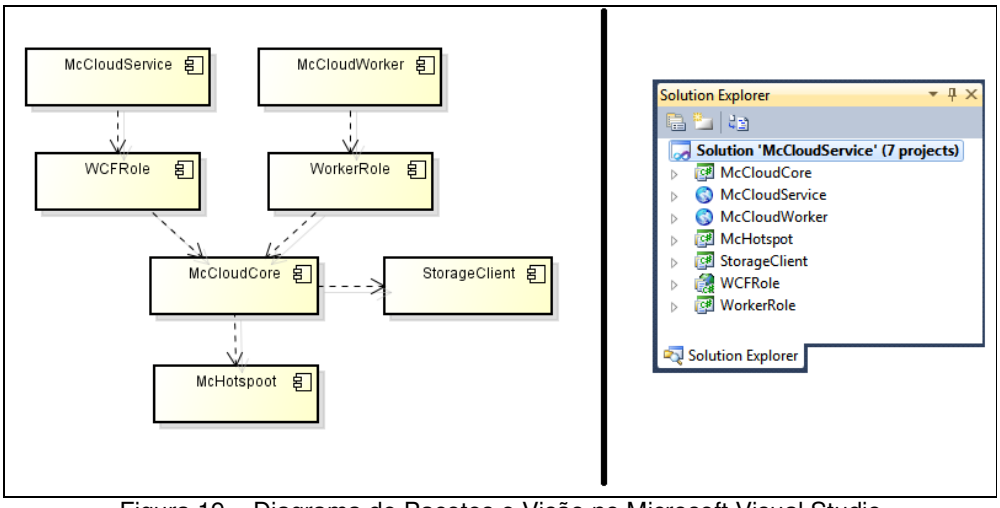

Figura 19 – Diagrama de Pacotes e Visão no Microsoft Visual Studio

O papel *WCF Role* se encontra nos pacotes (projetos) *McCloudService* e *WCFRole,* enquanto o papel *Worker Role* nos pacotes *McCloudWorker* e *WorkerRole.* O pacote *McCloudCore* concentra o núcleo de funcionalidades do arcabouço, que é consumido por ambos os papéis, já o pacote *StorageClient* fornece a este métodos de manipulação dos armazenamentos do Azure. Por último, o pacote *McHotspot* permite a implementação dos pontos de extensão do arcabouço, conforme seção 6.2.

Cada papel possui um arquivo de configuração em XML chamado *ServiceConfiguration.cscfg,* no qual é possível configurar a implementação do arcabouço, conforme exemplo abaixo.

```
<Instances count="1" />
<ConfigurationSettings>
  <Setting name="AccountName" value="..." />
  <Setting name="AccountSharedKey" value="..." />
  <Setting name="BlobStorageEndpoint" value="..." />
  <Setting name="QueueStorageEndpoint" value="..." />
  <Setting name="TableStorageEndpoint" value="..." />
  <Setting name="TimeoutDefaultInSeconds" value="7200"/>
  <Setting name="SimulationClass" value="PoC" />
  <Setting name="WorkerRoleOn"  value="true"/>
  <Setting name="Log" value="true" />
</ConfigurationSettings>
```

Figura 20 – Campos do arquivo de configuração dos papéis

As opções no arquivo ilustrado acima são: número de nós (instâncias) que exercerão o papel; dados de acesso da conta de armazenamento criada no portal do Azure (nome, chave da conta e endereço de cada serviço de armazenamento); tempo máximo padrão de processamento de cada tarefa em segundos; nome da implementação da simulação que será executada (facilita a troca no caso de existir mais de uma); e se a instrumentação está ativa (*log*). Além disso, com relação ao papel *Worker Role*, existe uma opção adicional, que identifica se os nós trabalhadores devem executar ou não as tarefas da fila, no caso negativo, ficam aguardando.

A principal vantagem de utilizarmos este arquivo é permitir que a alteração de uma destas configurações possa ser feita diretamente no portal do Azure, sem a necessidade de uma nova compilação e publicação da solução.

Diante do exposto, quando a opção *Log* estiver ativa (*true*), a execução da solução ocorrerá de forma instrumentada, ou seja, passo a passo é feito o registro das atividades da solução, armazenado na conta configurada neste mesmo arquivo, na tabela "log". Nessa tabela são salvos: código da publicação (*PartitionKey*), data e horário da gravação do registro pelo fuso horário do Azure (*Timestamp*), identificador da instância (*InstanceId*), categoria do registro [Erro, Informação ou Alerta] (*category*) e mensagem (*Message*). Essa instrumentação é muito didática e contribui para o entendimento do funcionamento do Papel *Worker Role*. Também é fundamental para depuração de problemas, que podem ocorrer como consequência de uma nova implantação dos pontos de extensão.

Além disso, cabe observar que as principais informações de cada simulação solicitada se encontram na tabela "mcc" deste mesmo armazenamento. Nela estão disponíveis: o *codein*, *codeout* e *n* (*interactions*) fornecidos na solicitação da simulação; a chave da simulação (*key*); o número de tarefas em que a simulação foi dividida para execução (*tasks*); a duração de cada etapa em milissegundos (*durationsplit*, *durationprocess* e *durationmerge*); o status atual da simulação [1 se está na etapa *process*, 2 se está na etapa *merge* ou 3 se completou a simulação] (*status*); a data e hora de criação do registro no fuso horário do Azure (*Timestamp*); e os parâmetros retornados pela otimização (*timeout* e *instancesadded*).

6.6
Visão Geral

O fluxo geral do arcabouço é ilustrado nas figuras a seguir.

A Figura 21 apresenta as principais atividades da etapa *split*, com a interação da aplicação cliente e utilização dos armazenamentos.

A Figura 22 apresenta as principais atividades da etapa *process*, ilustrando a elasticidade no número de nós trabalhadores e a interação com os armazenamentos.

A Figura 23 apresenta as principais atividades da etapa de *merge*, e sua interação com os armazenamentos.

Por último, a Figura 24 ilustra a interação do cliente para consultar o andamento, e obter o resultado final da simulação.

O código do McCloud Service Framework, bem como especificações e informações adicionais estão publicados e disponíveis no repositório Codeplex (http://www.codeplex.com), sob a licença Ms-RL (*Microsoft Reciprocal License*), cujo acesso se faz possível também pelo endereço http://mccloud.codeplex.com.

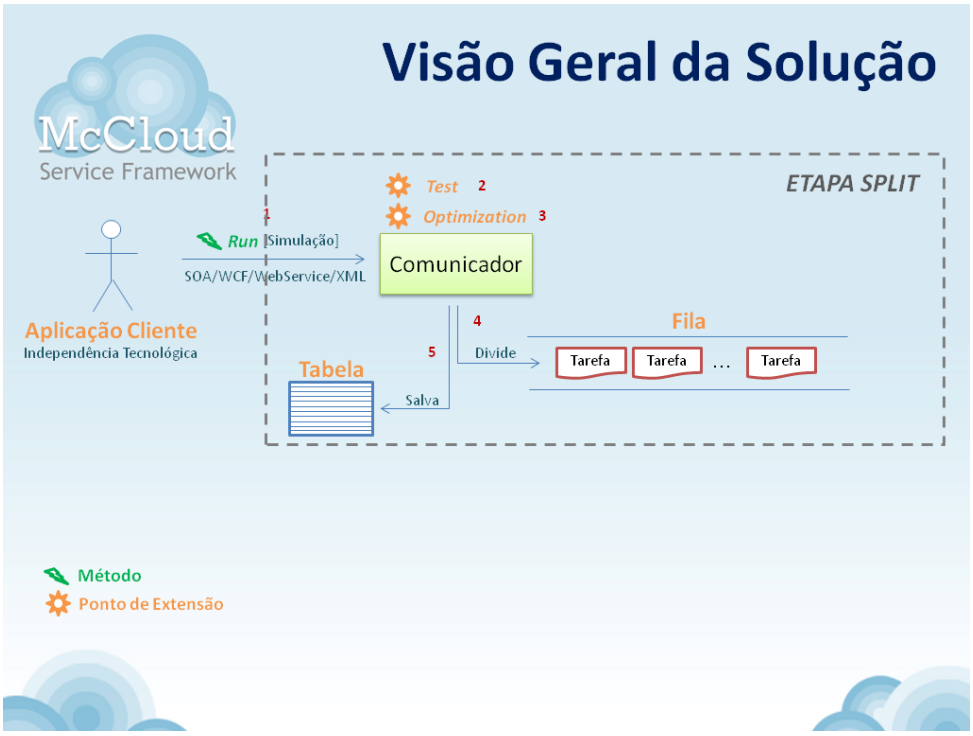

Figura 21 – Ilustração da visão geral do arcabouço (etapa split)

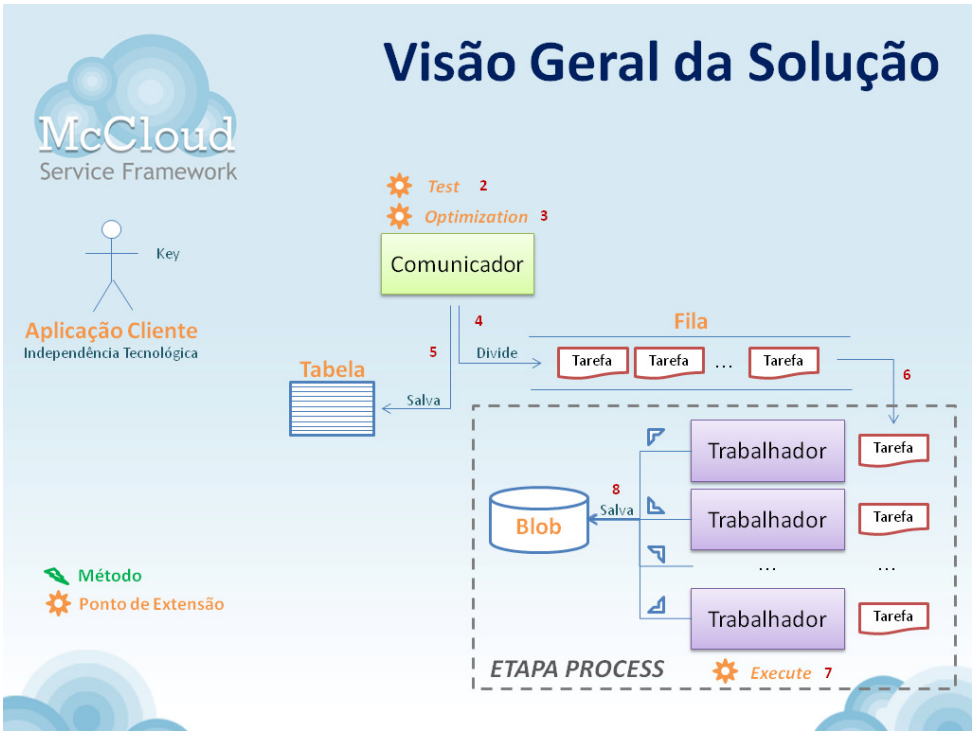

Figura 22 – Ilustração da visão geral do arcabouço (etapa process)

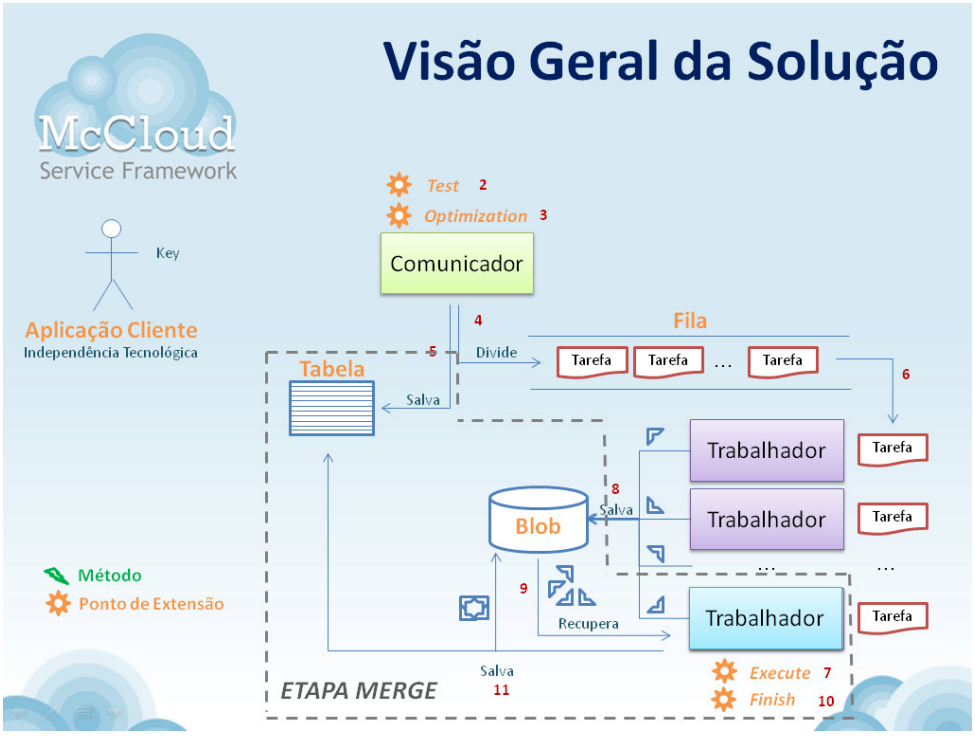

Figura 23 – Ilustração da visão geral do arcabouço (etapa merge)

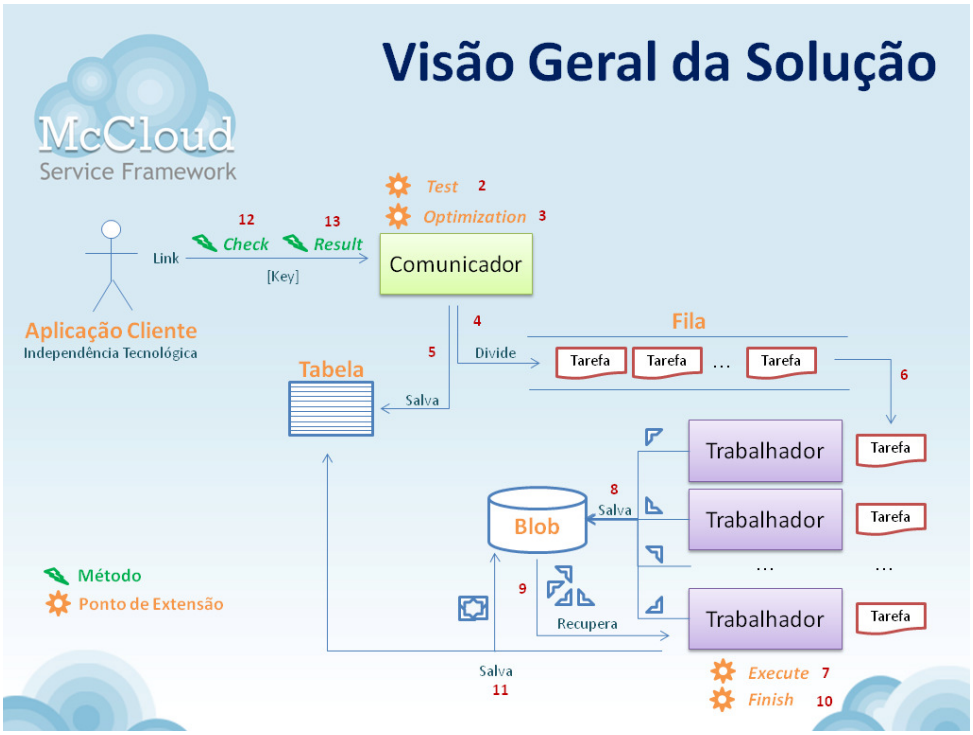

Figura 24 – Ilustração da visão geral do arcabouço (final)