INGESTÃO DADOS TWITTER

RELATÓRIO TÉCNICO

ROGÉRIO BIONDI

INTRODUÇÃO

- O Objetivo deste documento é explicar de forma sumarizada o trabalho realizado, concluir a execução do trabalho solicitado pelo Banco Itaú e apresentar algumas considerações importantes.

TECNOLOGIAS UTILIZADAS[1]

- **Docker**: é uma tecnologia usada para o aprovisionamento rápido de ambientes por meio de containers. Ele será usado para fornecer o ambiente de demonstração, onde todas as ferramentas serão instaladas;
- **Cassandra**: é um banco de dados NoSQL colunar e sem masters, capaz de armazenar uma grande quantidade de informações. De acordo com o teorema de CAP, o Cassandra é uma solução AP default (Availability + Partitioning Tolerance), embora para cenários específicos possa ser configurado para atender o critério ”C” (Consistency). Os drivers para NodeJS e Python são fornecidos pela empresa DataStax, todas as versões open-source.

TECNOLOGIAS UTILIZADAS[2]

- **Python**: é uma linguagem de script de propósito geral, usada no desenvolvimento de sites, sistemas e automação de infraestrutura. Ainda é largamente utilizada para realizar operações de machine learning (aprendizado de máquina). Ela será usada no projeto com dois objetivos:
  - Acesso às APIs do Twitter;
  - Combinado com o Spark, processamento dos dados.
- **Spark**: é um framework de processamento paralelo/massivo, considerado uma evolução do map-reduce (Hadoop), pois é capaz de realizar as operações em memória e rodar sobre alguns gerenciadores de recursos, como o Mesos e YARN.

TECNOLOGIAS UTILIZADAS[3]

- **NodeJs**: é uma linguagem de script baseada inicialmente no motor Javascript do navegador Google Chrome. Com caráter assíncrono, muito eficiente no desenvolvimento de microserviços REST/JSON. Neste desenvolvimento usaremos a biblioteca Express versão 4.x. É usada também no desenvolvimento de aplicações web e a tela de demonstração será escrita usando o framework AngularJS 1.x.
- **Scala**: usado como um pré-requisito para o Spark.
- **Oracle Java**: usado como pré-requisito para o Spark e outras ferramentas. Para usar o demo é necessário aceitar a licença do software. O produto é obtido no site do fornecedor;

CRONOGRAMA PREVISTO E REALIZADO

Previsto

	17/nov	18/nov	19/nov	20/nov	21/nov	22/nov	23/nov
	Semana 1					Semana 2	
Atividade	**1**	**2**	**3**	**4**	**5**	**1**	**2**
Planejamento							
Preparação do plano de projeto							
Abertura do projeto							
Infraestrutura							
Setup Infraestrutura (Containers Docker, github etc)							
Desenvolvimento							
Modelagem base Cassandra							
Captura Tweets							
Criação API REST							
Aplicação para apresentação dos resultados							
Conclusão							
Criação do Relatório Técnico							
Entrega final							

A atividade ”Processo de Agregação Spark” não Estava prevista. Entretanto não causou impacto No cronograma final.

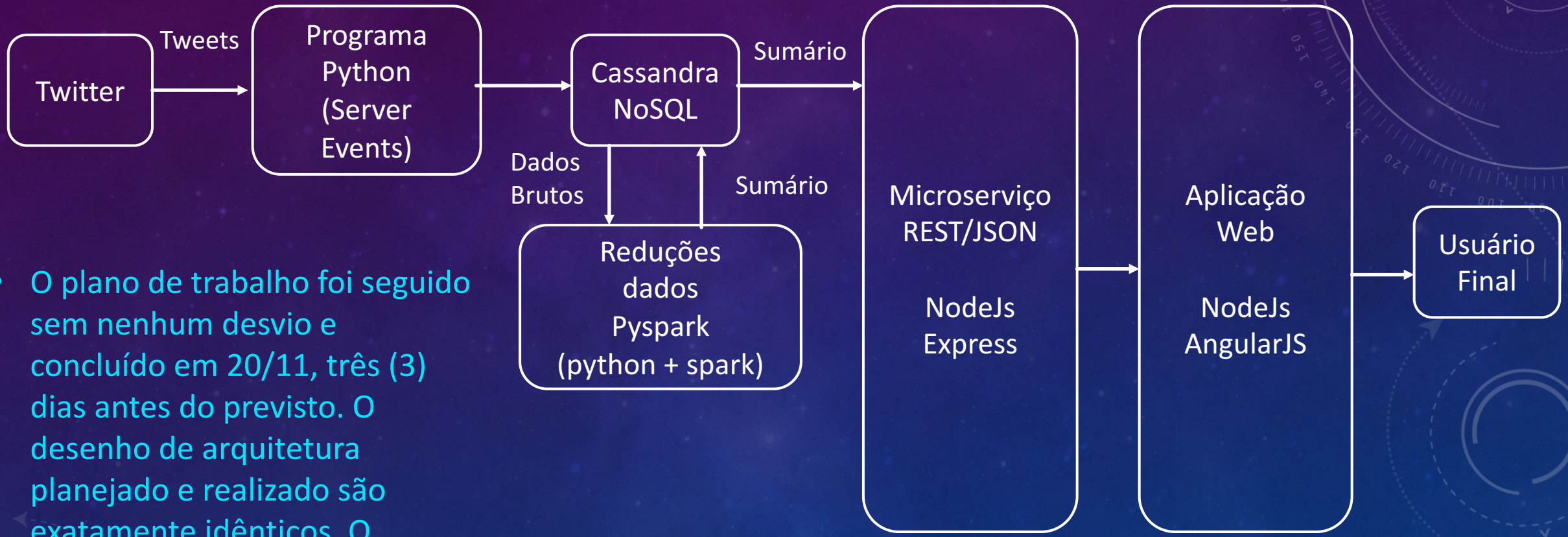
Realizado

	17/nov	18/nov	19/nov	20/nov	21/nov	22/nov	23/nov
	Semana 1					Semana 2	
Atividade	**1**	**2**	**3**	**4**	**5**	**1**	**2**
Planejamento							
Preparação do plano de projeto							
Abertura do projeto							
Infraestrutura							
Setup Infraestrutura (Containers Docker, github etc)							
Desenvolvimento							
Modelagem base Cassandra							
Captura Tweets							
Processo de agregação Spark							
Criação API REST							
Aplicação para apresentação dos resultados							
Conclusão							
Criação do Relatório Técnico							
Entrega final							

Alocação por Recurso

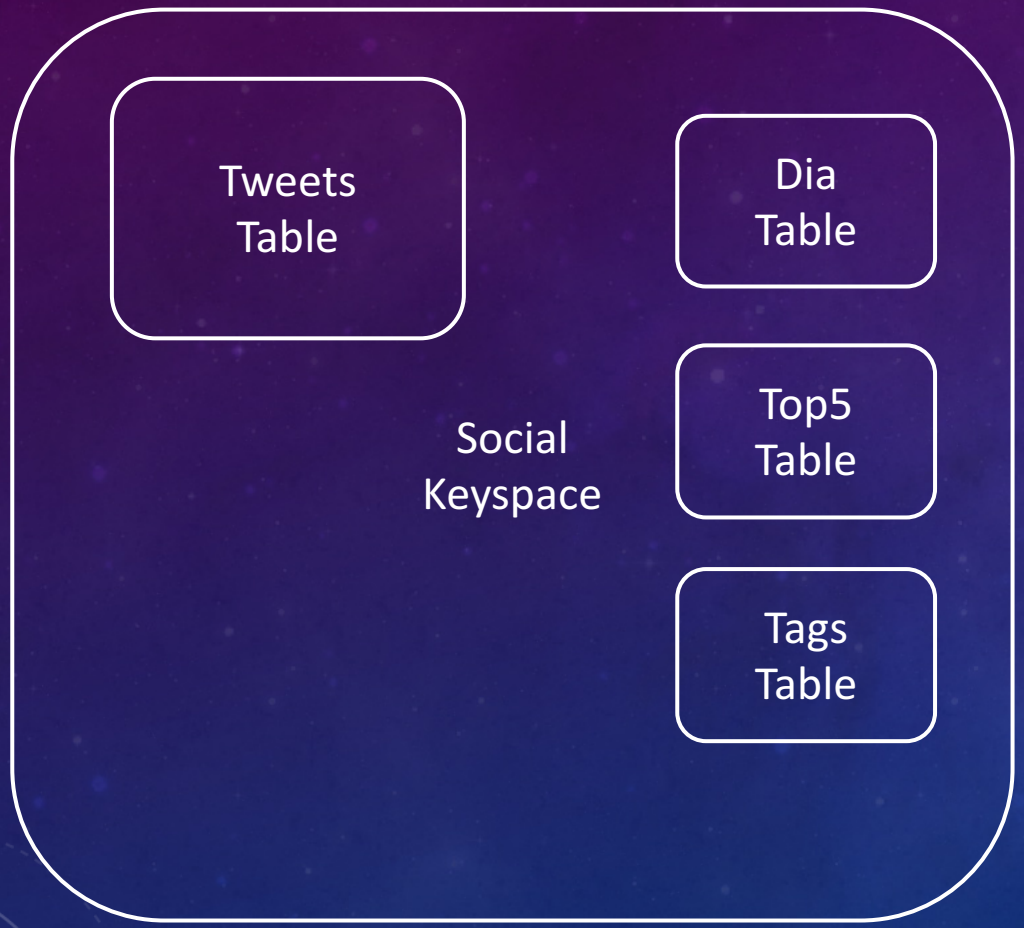
	Horas
Devops	4
Desenvolvedor Python	2
Desenvolvedor NodeJs	8
Arquiteto Software	2
TOTAL	**16**

ARQUITETURA FINAL



- O plano de trabalho foi seguido sem nenhum desvio e concluído em 20/11, três (3) dias antes do previsto. O desenho de arquitetura planejado e realizado são exatamente idênticos. O projeto inicial foi mantido sem nenhum imprevisto.

MODELAGEM BASE CASSANDRA



Tweets: tabela que armazena os dados de ingestão do twitter;
Dia: tabela de sumário que armazena a contagem de tweets por hora do dia;
Top5: esta tabela armazena os cinco usuários com maior número de seguidores;
Tags: esta tabela armazena o número de tweets por tag, para linguagem = pt (português)
O Cassandra não suporta ordenação com a cláusula ORDER BY, exigindo uma modelagem adequada.
O arquivo CQL que cria o modelo pode ser encontrado em:
https://github.com/rogeriobiondi/social/blob/master/cassandra/model.cql

DIAGRAMA IMPLANTAÇÃO SOLUÇÃO

Os passos detalhados para instalação, execução e teste do demo estão em:

https://github.com/rogeriobiondi/social/blob/master/README.md

OBSERVAÇÕES IMPORTANTES
INFRAESTRUTURA

- O container criado no demo não prevê nenhum tipo de otimização para reduzir o tempo de criação (build) nem de execução (run). A versão de produção deve atentar a este problema, caso o ambiente precise ser criado repetidas vezes ou não possa ficar indisponível.
- O demo contém versões standalone ou single node de todos os produtos. Uma arquitetura apropriada, em cluster, deve ser considerada, assim como a fragmentação dos produtos em diversos containers. Para mantê-los e gerenciá-los, fornecer failover ou cluster de containers, podem ser usadas soluções como o Apache Mesos + Marathon, Kubernetes, Swarm etc. Ou soluções em nuvem como o Amazon ECS.
- Foi fornecido apenas um grau de automação mínimo dos containers. Para produção podem ser usadas ferramentas como o Docker compose, Ansible (integração contínua), Puppet (configuração automática) etc

OBSERVAÇÕES IMPORTANTES
IMPLEMENTAÇÃO

- O código fornecido atinge o objetivo da maneira mais básica. Não prevê questões como parametrização. Vários parâmetros podem ser retirados dos programas (hard coded) e configurados como variáveis de ambiente ou parâmetros de inicialização dos containers. Ou ainda armazenados em um serviço como o etcd.
- Em cada ponto que julguei necessário, inclui uma marcação de TODO com as observações necessárias.
- Comentários e cabeçalhos também não foram incluídos na demo, mas podem ser especificados de acordo com os padrões do cliente. Esta nota é para indicar ciência desse aspecto importante que é a limpeza e organização do código.
- Outro ponto importante é que senhas ou dados sensíveis devem ser guardados em locais seguros, em keychain / keystore apropriados e as chaves usadas apenas na construção dos containers. Tais informações devem ser criptografadas e protegidas.

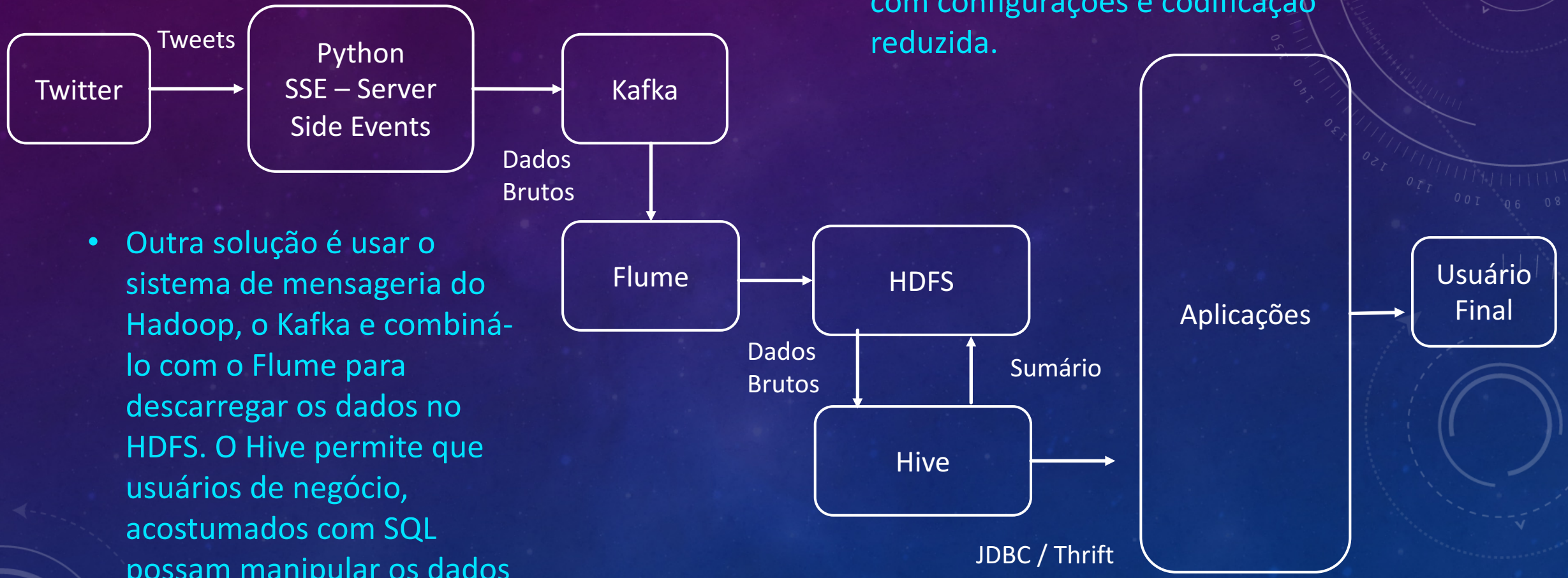
OBSERVAÇÕES IMPORTANTES
ARQUITETURA[1]

- Os sumários são mantidos em um grid de memória, de forma mais dinâmica/rápida, pois são atualizados ”à quente”, de forma incremental.

- Apesar do desenho ter sido rapidamente implementado, não é a melhor solução possível. Existem outras formas interessantes de resolver o problema, como é o caso de Streaming, que permite processar os dados conforme chegam.

OBSERVAÇÕES IMPORTANTES
ARQUITETURA^2

- O modelo map-reduce clássico é ultrapassado, mas esse desenho pode ser implementado apenas com configurações e codificação reduzida.



- Outra solução é usar o sistema de mensageria do Hadoop, o Kafka e combiná-lo com o Flume para descarregar os dados no HDFS. O Hive permite que usuários de negócio, acostumados com SQL possam manipular os dados facilmente.

OBSERVAÇÕES IMPORTANTES
ARQUITETURA[3]

- Este outro desenho é ainda mais eficiente, pois usa o stack ELK (Elasticsearch, Logstash e Kibana). É possível implementar uma solução aproximada em **1 dia**, **sem praticamente nenhuma linha de código**.
- Muitos desenvolvedores, infelizmente, sempre tentam resolver todos os problemas com desenvolvimento, nem sempre a solução que proporciona o melhor ROI para a empresa.

CONCLUSÃO

- Existem muitas formas de se resolver um mesmo problema, principalmente com as tecnologias de Big Data disponíveis, que são muitas. É importante avaliar cada caso com cuidado.
- Na mineração de dados de mídias sociais é possível expandir para muito além, como análise de sentimento (análise bayesiana, o mesmo algoritmo usado para detectar SPAM), scrapping de dados e uma série de operações que podem gerar benefícios financeiros, ajudar a entender o humor e disposição dos clientes, identificar interesses etc;
- O caso apresentado é extremamente simples de implementar, mas a solução sugerida leva a diversos problemas citados (integração contínua, segurança, parametrização...). Por outro lado fornece uma flexibilidade bem razoável de como a solução pode ser implementada.