21º Seminário de Automação e TI
Anais do Seminário de Automação & TI ISSN 2594-5335 vol. 21, num. 1 (2017)

ARQUITETURA USIMINAS PARA APOIO AO DESENVOLVIMENTO DE SISTEMAS EM AMBIENTE JAVA

David Bassanelli Sampaio [1]

Resumo
Com a crescente demanda por sistemas distribuídos e multiplataformas, houve uma larga proliferação de métodos, linguagens e tecnologias com o objetivo de atender aos mais diversos requisitos desse ambiente. Com isso, a TI de corporações que possuem desenvolvimento interno de aplicações enfrentam um grande desafio de gerir conhecimento dos profissionais mais experientes e integrar novos profissionais de forma a garantir a qualidade e agilidade na tratativa de incidentes, bem como a capacidade de evolução e criação de novas soluções. Neste trabalho, apresentaremos as diretrizes, tecnologias e processos desenvolvidos na Usiminas, consolidados em uma arquitetura. Esta arquitetura propiciou, entre outros, a estabilidade e continuidade dos sistemas existentes, a padronização da experiência final do usuário, a agilidade na construção das soluções Java, evoluindo a qualidade dos sistemas mantidos pela TI, com resultados efetivos.
Palavras-chave: Arquitetura de Desenvolvimento; Processos; Integração Contínua; Experiência de Usuário.

USIMINAS ARCHITECTURE TO SUPPORT THE DEVELOPMENT OF SYSTEMS IN JAVA ENVIRONMENT

Abstract
With the growing demand for distributed and multiplatform systems, there was a large proliferation of methods, languages and technologies in order to meet the most diverse requirements of this environment. Due to this, non-IT companies who have internal application development face a great challenge of managing knowledge of the most experienced professionals and integrating new ones in order to guarantee the quality and agility in the handling of incidents, as well as the capacity of evolution and creation of new solutions. In this paper, we will present the guidelines, processes and technologies developed by Usiminas, consolidated in an architecture. This architecture provided, among other things, the stability and continuity of existing systems, the standardization of the final user experience, the agility in the construction of Java solutions, evolving the quality of the systems maintained by IT, with effective results.
Keywords: Development Architecture; Processes; Seamless Integration; User Experience.

1. *Engenheiro de Computação pela Universidade Santa Cecília, Analista de Sistemas e Arquiteto de Soluções Java na USIMINAS, Cubatão, São Paulo, Brasil.*

1 - INTRODUÇÃO

A necessidade crescente de utilização de sistemas integrados e multiplataformas na indústria traz um grande desafio para as equipes de desenvolvimento de sistemas. Conforme novas plataformas são inseridas no contexto industrial, novas tecnologias, métodos e linguagens para desenvolvimento dos sistemas se fazem necessárias, tornando a manutenção de sistemas cada vez mais complexa. Tal complexidade reflete diretamente em fatores como a agilidade no atendimento ao cliente, a experiência do usuário, a garantia de continuidade dos sistemas e a gestão de conhecimento interno.

Para padronizar o processo de desenvolvimento de sistemas de missão crítica, a Usiminas criou a equipe de Arquitetura de Soluções com a atribuição de padronizar a forma de utilização das tecnologias já difundidas internamente e definir caminhos a serem seguidos no desenvolvimento de novas soluções, o que resultou no que ficou conhecido na Usiminas como "Arquitetura USIMINAS para desenvolvimento de sistemas em ambiente Java", que a partir de agora será referenciado apenas como "Arquitetura Java".

De acordo com Silveira:

Uma boa implementação, design ou arquitetura é aquela que permite modificações causando somente um impacto considerado justo a outras partes do sistema, e, ao mesmo tempo, diminuindo as ocorrências de tais situações, minimizando o custo de desenvolvimento e manutenção.

Esse trabalho apresentará os diversos aspectos abordados no desenvolvimento da "Arquitetura Java" distribuído da seguinte forma: i) a seção 1 apresenta o cenário anterior ao trabalho de definição de arquitetura, abordando as dificuldades e problemas existentes; ii) a seção 2 apresenta classes e serviços criados para prover soluções padronizadas mantidas pela equipe de arquitetura; iii) a seção 3 demonstra as definições de empacotamento, modularização e distribuição de camadas; iv) a seção 4 apresenta os assuntos que foram normatizados a fim de garantir a correta utilização; v) a seção 5 aborda a padronização visual desenvolvida com o objetivo de reduzir a complexidade do desenvolvimento e garantir a experiência do usuário final; vi) a seção 6 apresenta o processo definido para o desenvolvimento das aplicações Java na Usiminas, abordando a estratégia de desenvolvimento em equipe e integração contínua da aplicação; vii) a seção 7 cita a estrutura criada para a centralização da documentação de desenvolvimento e o plano de comunicação de evoluções e correções de arquitetura.

1.1 – Cenário e problemática

A crescente fragmentação dos dispositivos que acessam e manipulam informações utilizadas na indústria trouxe a necessidade de modernização da plataforma de desenvolvimento de sistemas utilizada internamente na USIMINAS. Até então compostos majoritariamente por tecnologias cliente servidor, os sistemas possuíam características que os tornavam menos propícios para utilização de forma distribuída em múltiplas plataformas.

Com isso surge um cenário ideal para utilização de uma plataforma de desenvolvimento Web, que provêm a flexibilidade, escalabilidade e robustez necessárias para atender os requisitos impostos pelos sistemas utilizados na área industrial.

Em contrapartida, o ambiente Web possui uma alta granularidade de tecnologias que visam resolver de formas diferentes um mesmo problema, o que levou ao seguinte questionamento: "Como garantir os níveis de serviço próximo ao atual, realizando o aculturamento e a gestão do conhecimento dos profissionais de TI atuais em um ambiente tão granular e complexo?" e a resposta obtida pela USIMINAS foi através de criação de uma equipe de Arquitetura de Soluções que trabalhou com foco na definição de tecnologias e processos, desenvolvimento de facilitadores e documentação focados no ambiente Java.

2 - MATERIAIS E MÉTODOS

2.1 - Arquitetura De Software

De forma ampla, a arquitetura de software consiste na definição das tecnologias a serem utilizadas por um sistema a fim de atender a requisitos funcionais e não-funcionais, documentada na forma de modelos que podem abranger diferentes visões de um sistema.
De acordo com Pressman, Arquitetura de software é a espinha dorsal do sistema a ser construído. Afetando interfaces, estruturas de dados, desempenho e fluxo de controle de programas, maneira pela qual os testes podem ser conduzidos, a manutenção do sistema realizada e muito mais. Por todas essas razões, o projeto deve começar com as considerações arquitetônicas. Só depois de a arquitetura ter sido estabelecida devem ser considerados os elementos relativos a componentes [1].

2.2 - Camadas

Em substituição do até então utilizado modelo em duas camadas, onde tínhamos uma camada de aplicação mais pesada, centralizando regras de apresentação de negócio e uma camada de banco de dados, temos o modelo de 3 camadas, conhecido como *Model, View e Controller* (MVC).
Esse modelo propõe uma camada de apresentação (representado pelo termo *View*), responsável pela interação com usuário e executada no lado cliente, uma camada de negócio centralizando todas as regras empresariais, e que é executada em um servidor (representada pelo termo *Controller*), e uma camada de persistência, agregando todas as informações de modelo de dados e objetos do banco de dados (representado pelo termo *Model*).
Conforme Peres os objetivos da arquitetura em camadas são:

- Dividir a aplicação em módulos o mais independente possível (Modularidade);
- Reduzir o custo de manutenção da aplicação (Manutenibilidade);
- Permitir que funcionalidades sejam acrescentadas sem impactar nas já existentes (Extensibilidade);
- Permitir o reuso de classes e componentes em outros módulos da aplicação ou em outras aplicações (Reusabilidade) [2].

2.3 - Especificações / Frameworks

Apesar de uma grande quantidade de opções, o trabalho focou em aumentar a probabilidade de continuidade tecnológica, bem como compatibilidade com futuras evoluções e dessa forma foram selecionadas prioritariamente tecnologias recomendadas pela especificação oficial da linguagem, como podemos observar na figura 1.

Figura 1. Distribuição tecnológica aplicadas nas camadas MVC definidas na arquitetura Java USIMINAS.

2.4 - Pacotes / Módulos

Na visão de empacotamento de aplicações, foram definidos padrões de projetos para atender diferentes cenários conforme demonstrado nas figuras 2 e 3.

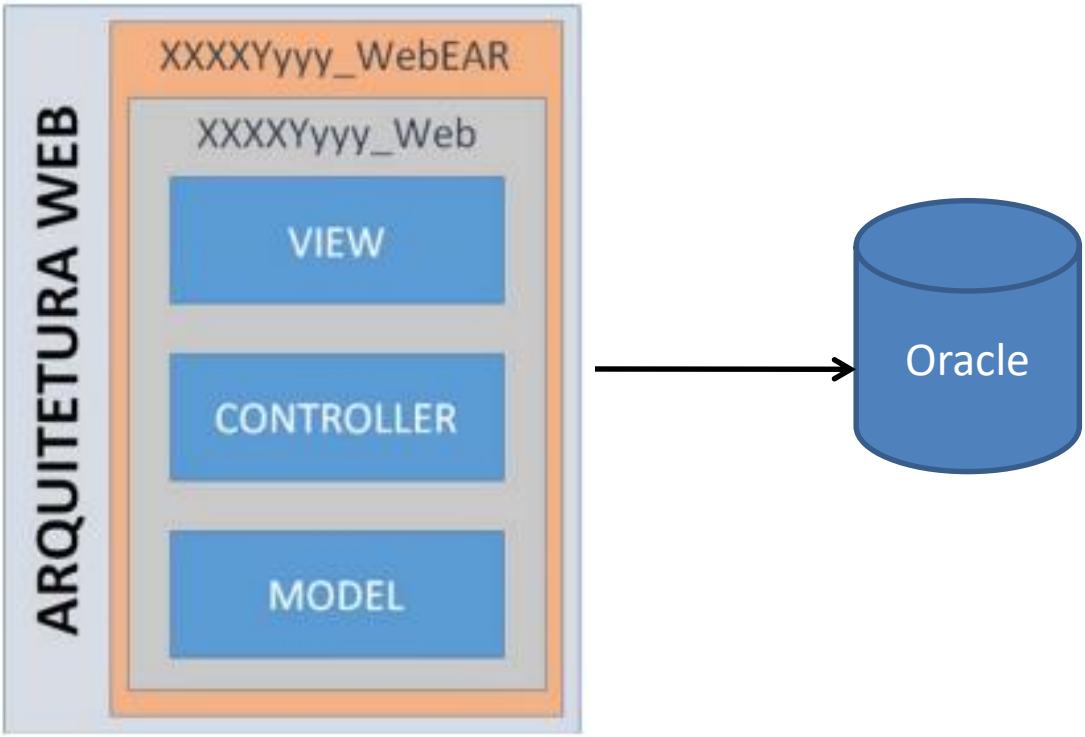

Figura 2. Representação do empacotamento de uma aplicação Web simples na arquitetura Java USIMINAS.

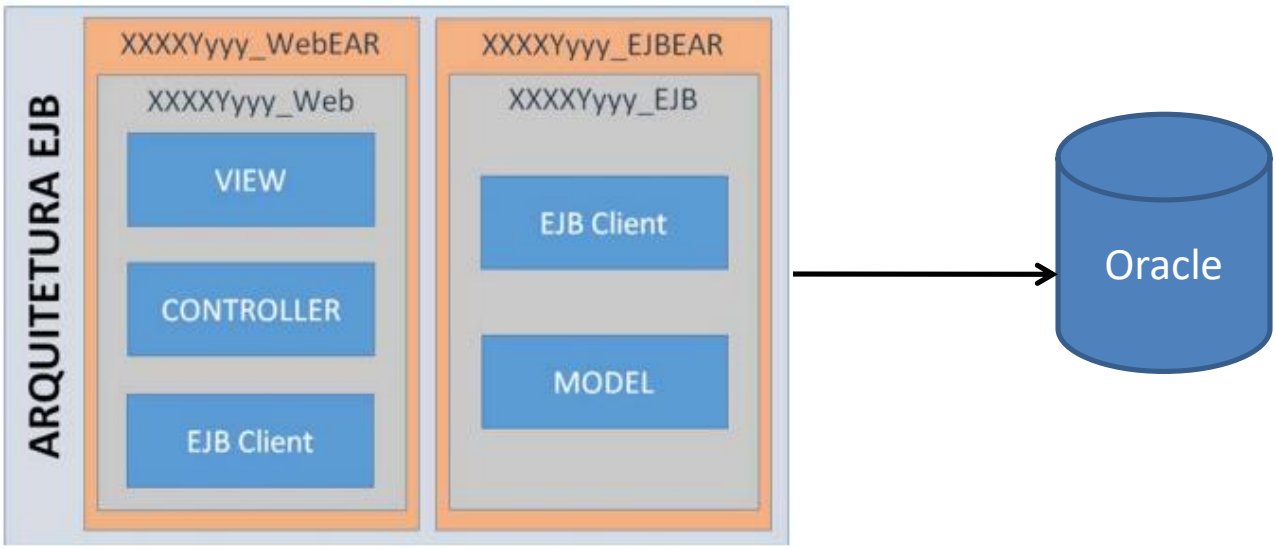

Figura 3. Representação do empacotamento de uma aplicação contendo uma camada Web e uma camada de serviços EJB na arquitetura Java USIMINAS.

3 – RESULTADOS E DISCUSSÃO

3.1 - CLASSES E SERVIÇOS

Parte da complexidade de desenvolver em um ambiente web está no fato de integrar muitas tecnologias de diversos fabricantes diferentes. Durante o desenvolvimento de sistemas, grande parte do esforço de codificação é gasto na construção de códigos conhecidos como "códigos de infraestrutura", ou seja, códigos cuja única finalidade é integrar ou configurar tecnologias, muitas vezes de forma repetitiva e mecanizada.
De acordo com Erl os princípios de design de serviços propõem que:

- Serviços são reutilizáveis;
- Serviços compartilham um contrato formal;
- Serviços possuem um baixo acoplamento;
- Serviços abstraem a lógica;
- Serviços são capazes de se comporem;
- Serviços são autônomos;
- Serviços evitam alocação de recursos por longos períodos;
- Serviços são capazes de serem descobertos [3].

Visando tirar proveito dos benefícios citados, foram desenvolvidas classes genéricas e serviços possibilitando que o desenvolvedor acesse recursos e efetue integrações com menor acoplamento com o ambiente interno da empresa, e consequentemente possibilitando que o foco seja centralizado na escrita de códigos voltados ao atendimento ao negócio, diminuindo a complexidade da linguagem e aumentando a confiabilidade, estabilidade e manutenebilidade do código gerado.
Além dos benefícios técnicos acima citados, nota-se também como benefício o aumento de produtividade dos desenvolvedores, uma vez que as classes e serviços entregam facilitadores e aceleradores de escrita de código reduzindo a complexidade e aumentando a confiabilidade na utilização e integração de recursos entre as diversas camadas dos sistemas.

Abaixo segue a relação de classes e serviços criados na USIMINAS:

- Arquétipo para projetos Java Web;
- Autenticação dos sistemas unificada com serviço de autenticação de rede;
- Cadastro e controle de Menu de aplicações centralizado;
- Classes facilitadoras de acesso ao banco de dados e controle de transações;
- Classes facilitadoras para emissão de relatórios.
- Classes facilitadoras para validação de segurança.

3.2 - NORMAS E PADRÕES

Com o objetivo de garantir a adoção por todas as equipes de desenvolvimento interno, foram criadas normas internas que estabelecem de forma clara e concisa os principais padrões e regras de utilização das tecnologias que compões a Arquitetura Java Usiminas.
As normas foram criadas considerando as melhores práticas recomendadas pelo documento de convenção de códigos da linguagem e abordam os temas abaixo:

- Arquitetura para desenvolvimento Web:
  - Ferramentas e tecnologias;
  - Bibliotecas externas homologadas;
  - Bibliotecas internas;
  - Nomenclatura de sistema;
  - Nomenclatura de módulo;
  - Nomenclatura de artefatos internos ao sistema;
  - Empacotamento;
  - Camadas.
- Tratamento de erros e validação em aplicações Web;
- Padronização visual para aplicações Web.

3.3 - UNIFICAÇÃO DO PADRÃO VISUAL

Durante o desenvolvimento interno, percebemos que o desenvolvimento da camada visual da aplicação se demonstrava como a mais custosa de todo o processo, fazendo com que a maior parte do esforço estivesse concentrada em posicionar elementos em tela, definir cores e criar regras de apresentação.
Com o objetivo de permitir que esse esforço fosse focado na criação de códigos de negócios mais robustos, houve um trabalho amplo de definição de um padrão visual para os sistemas desenvolvidos em Java que resultou na criação de templates de tela pensados para atender os principais cenários internos. Conforme podemos ver na figura 4, os templates definem regiões específicas para conter determinados tipos de informação, como o menu, área lateral para filtros, uma área para exibição de resultados bem como diversos outros padrões não presentes na imagem.
Além de padronizar o posicionamento das informações em tela, o trabalho também abordou a padronização da experiência do usuário através da definição de cores e estilos dos componentes disponíveis na arquitetura.
A padronização de uso desses componentes foi realizada através da criação de bibliotecas que contém folhas de estilos que aplicam de forma clara e transparente as definições de cor, fonte, tamanho de texto e componentes permitidos.
Por fim houve a definição dos artefatos que traduzem a conversação do usuário com o sistema, tais como as caixas de diálogo, mensagens de informação e validação e

telas de comunicação de erro ao usuário. Esses artefatos foram incorporados às bibliotecas de apoio e ao arquétipo padrão para criação de novos sistemas e são disponibilizados de forma transparente às equipes de desenvolvimento.

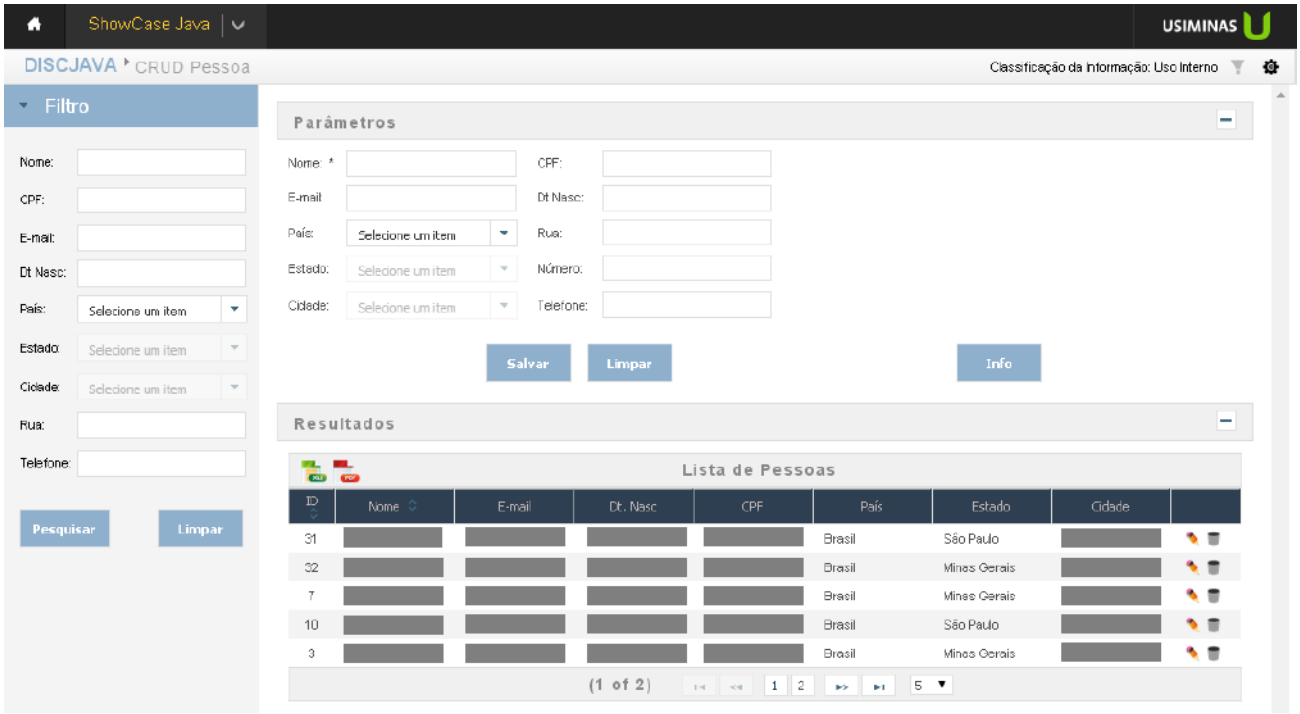

Figura 4. Padrão visual para aplicações com filtro e área de resultado.

3.4 - DOCUMENTAÇÃO E COMUNICAÇÃO

Parte importante para garantir a adoção e a continuidade de uma arquitetura é a disponibilização de uma documentação técnica onde seja possível para um profissional de desenvolvimento compreender a finalidade e a forma de utilização dos componentes e tecnologias disponíveis no ambiente e que seja de fácil acesso e confiável.

Para evitar que os desenvolvedores mantenham cópias desatualizadas de documentos e tenham acesso de forma rápida e centralizada às documentações, foi criado um site na intranet onde todas as informações relevantes de arquitetura estão centralizadas. Estão disponíveis informações como documentação de classes, tutoriais de configuração de ferramentas e ambiente, utilização de classes de arquitetura e criação e configuração de sistemas existentes, bem como informações técnicas sobre evoluções e correções de erros em classes em bibliotecas internas.

Em complemento ao site interno, foi criado um padrão de comunicação periódico contendo informações relevantes para os desenvolvedores sobre o ambiente.

4 – CONCLUSÃO

A criação da arquitetura USIMINAS trouxe uma série de ganhos qualitativos e quantitativos que entre eles podemos destacar:

- Redução da curva de aculturamento para integração de novos profissionais ao time de desenvolvimento da empresa, considerando a alta padronização do

ambiente e ampla documentação tecnológica e do processo interno de desenvolvimento;
- Redução de 8 horas para 2 horas no tempo de configuração de um novo sistema, utilizando os arquétipos e documentos de arquitetura;
- Aumento da estabilidade dos sistemas em ambiente produtivo, uma vez que os principais pontos de vazamento de memória foram encapsulados pelos facilitadores criados na arquitetura, e que padronizam a utilização e gestão de recursos de hardware e rede;
- Maior foco no desenvolvimento de códigos de negócio por parte da equipe de sistema, uma vez que os códigos para acesso e gerenciamento de recursos de infraestrutura estão encapsulados em classes e serviços de arquitetura.
- Aumento de produtividade dos desenvolvedores com a utilização dos serviços e classes genéricas que permitiram a abstração da integração entre camadas e serviços externos acessados pelos sistemas.

Além dos benefícios qualitativos percebidos pela equipe de TI, existem também os benefícios aos clientes que são percebidos através da identidade visual única que facilita a utilização e familiarização com novos sistemas e na maior agilidade por parte da TI na correção de problemas reportados.

REFERÊNCIAS

1 PRESSMAN, Roger S. Engenharia de Software-Uma Abordagem Profissional. 7. ed. ARMED, 2011.
2 PERES, Marcela Mariotti. Arquitetura em três camadas – parte 1.[página da internet][acesso em 11/06/2017]. Disponível em: http://marcelamperes.wordpress.com/2011/07/14/arquitetura-em-tres-camadas-parte-1/.
3 ERL, THOMAS. SOA: Princípios de design de serviços. Prentice hall Brasil, 2009.