Secretaria da Fazenda do Estado do Tocantins

Centro Interamericano de Administrações Tributárias

Produto: Arquitetura Empresarial da SEFAZ-TO

Eng. Juan León Solis

Março/2016

Tabela de conteúdo

1. Introdução

Neste documento define-se a Arquitetura Empresarial para o Sistema de Informação da SEFAZ-TO, essa arquitetura está orientada a plataforma Java Enterprise Edition 7 (JEE7) a nível de desenvolvimento. Esse trabalho tem uma visão DevOps[1][2] completa.

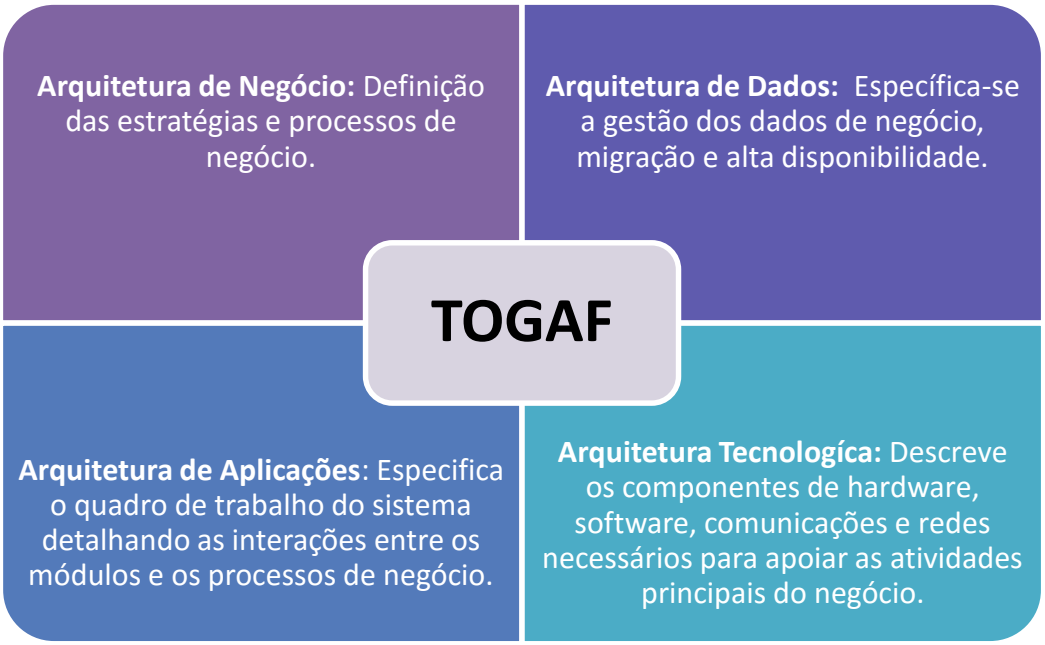
O documento é de natureza técnica e é destinado ao pessoal de TI na SEFAZ-TO. É sobre: linguagem de programação Java, plataforma JEE, em tecnologias orientadas a criação de aplicativos de negócios em ambientes Java, Banco de Dados Oracle, Networking e Segurança Informática.

Foi adicionado um glossário de termos técnicos no final do documento *[Anexo I: Glossário de Termos Técnicos]*.

1.1. O Marco de Trabalho TOGAF

The Open Group Architecture Framework (TOGAF) é um framework de arquitetura corporativa que provê uma abordagem global ao desenho, planejamento, implementação e governança de uma arquitetura corporativa.

A arquitetura é tipicamente modelada em quatro níveis ou domínios: Negócios, Aplicação, Dados e Tecnologia, são:



Arquitetura para o Novo Sistema Tributário na SEFAZ-Tocantins levará em consideração essa visão global para ter uma descrição formal da estrutura dos componentes do sistema e seus inter-relacionamentos. O modelo TOGAF usado neste documento será ajustado para os componentes principais para SEFAZ-TO.

[1] 2014 State fo DevOps Report by PuppetLabs. https://puppetlabs.com/2014-devops-report
[2] DevOps (amálgama de Desenvolvedor e Operações - Development & Operations). A definição completa está no Glosario

2. Arquitetura do Negócio

Nesta parte da arquitetura são considerados o plano estratégico do negócio e também a descrição dos processos chaves para a SEFAZ-TO na área tributária. Todo o desenvolvimento do sistema deve ser orientado para o cumprimento dos objetivos estratégicos e os processos atuais de negócios.

2.1. Plano Estratégico

A SEFAZ-TO já tem o plano estratégico, detalhado no [Produto 1.1 Gestão Estratégica].

2.2. Os Processos Chaves do Negócio Tributário

Os Processos Chave foram levantados pelos Consultores Analistas de Negócio do CIAT. De forma macro os processos são os seguintes:

Administração Tributária

- Cadastro
- Arrecadação
- Restituições
- Tributação e Benefícios Fiscais
- Processo de declarações
- Processo de gestão de ECF
- Gestão de documentos fiscais
- Gestão de Atendimento
- Trânsito
- Fiscalização estabelecimentos
- Cobrança
- Inteligência fiscal
- Fiscalização Outras Receitas (ITCD, IPVA)
- Contencioso Administrativo Tributário
- Corregedoria
- Educação Fiscal - Cultura tributaria
- Gestão da Escola Fazendária

Gestão

- Estudos Econômicos
- Processo de tomada de decisões
- Processo de planejamento estratégico
- Gestão de comunicação e administração interna
- Investigação e Pesquisa

Recursos Humanos

- Gestão de Recursos Humanos

Informática

- Desenvolvimento e manutenção de soluções informáticas
- Operação dos serviços de TI

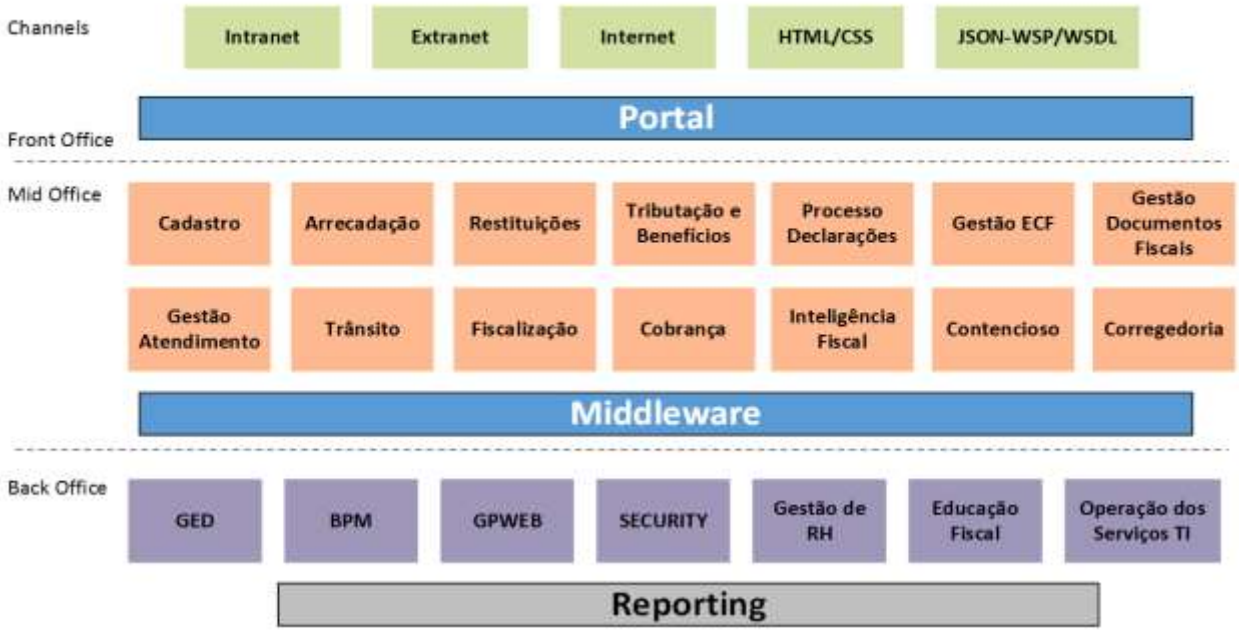
2.3. Visão da Arquitetura (Solution Concept).

A seguinte lista de premissas do projeto é definida com base na documentação existente do projeto, reuniões com os Consultores do CIAT, reuniões com o Pessoal de TI (SEFAZ-TO), também na experiência em desenvolvimento de Sistema de Informação Tributários, Aduaneiros e Planejamento de Recursos Empresariais (ERP):

- O Sistema é Web e não precisa fazer instalação alguma nos computadores dos clientes (contribuintes, contadores, funcionários, outras organizações), os usuários apenas precisam ter um browser moderno e uma boa conexão de internet. Pode ser que em alguns casos precisem uma impressora ou scanner também.

- O Sistema deverá cumprir os lineamentos estabelecidos na Política de Desenvolvimento de Aplicações da Diretoria de Informática.
- São considerados serviços transversais: gestão de segurança, monitoramento dos serviços, gestão de fluxos de trabalho, gestão de documentos e gestão de controle interno.
- Com a definição dos conceitos e serviços a serem tratados como transversais podemos construir antecipadamente os sistemas e procedimentos a respaldarem o uso destes serviços. Isto evita retrabalho posterior e permite que todas as aplicações contemplem a mesma estrutura construída com baixo esforço.
- O Sistema deve ter alta integração e rendimento com os serviços transversais.
- O Sistema deve ter alta segurança em produção e deve cumprir com as Políticas de Segurança da Diretoria de Informática.
- O Sistema deve suportar somente browsers modernos: Internet Explorer 9+, Microsoft Edge, Firefox, Chrome, Opera.
- O ambiente do Sistema é: Oracle 12c R1 como banco de dados, JBoss Wildfly como servidor de aplicações, o sistema operacional é Solaris 11.
- Os equipamentos para banco de dados, servidores de aplicações, rede, armazenamento, etc. tanto para desenvolvimento, teste, pré-produção e produção são fornecidos pela Diretoria de Informática.
- O Sistema deve ter alta disponibilidade: servidor de banco de dados, servidores de aplicações, equipamentos de segurança, rede e armazenamento em produção.
- Não vai haver um desenvolvimento para plataforma móvel.
- Os testes funcionais “alpha e beta”, serão executados no ambiente de teste pela Fábrica de Software (Continuous Integration) e os Consultores do CIAT. Os testes funcionais “release candidate” serão executados no ambiente de pré-produção pelos usuários finais.
- A visualização de informações para os testes do Sistema deve respeitar as Leis de Acesso de Informação e Privacidade do Brasil.
- Na migração dos dados do Sistema Atual (SIAT) será trabalhada com Pessoal de TI junto com os Consultores do CIAT.
- A capacitação sobre o uso do Sistema será definida pelos Consultores do CIAT e executada pela SEFAZ.

SOLUTION CONCEPT DIAGRAM



3. Arquitetura Tecnológica (TI)

Focada em alta disponibilidade, monitoramento e segurança para produção do Sistema Tributário. Não é uma alternativa para Recuperação de Desastres (Disaster Recovery).

3.1. Componentes da Arquitetura Tecnológica

A seguinte lista é uma descrição de cada componente:

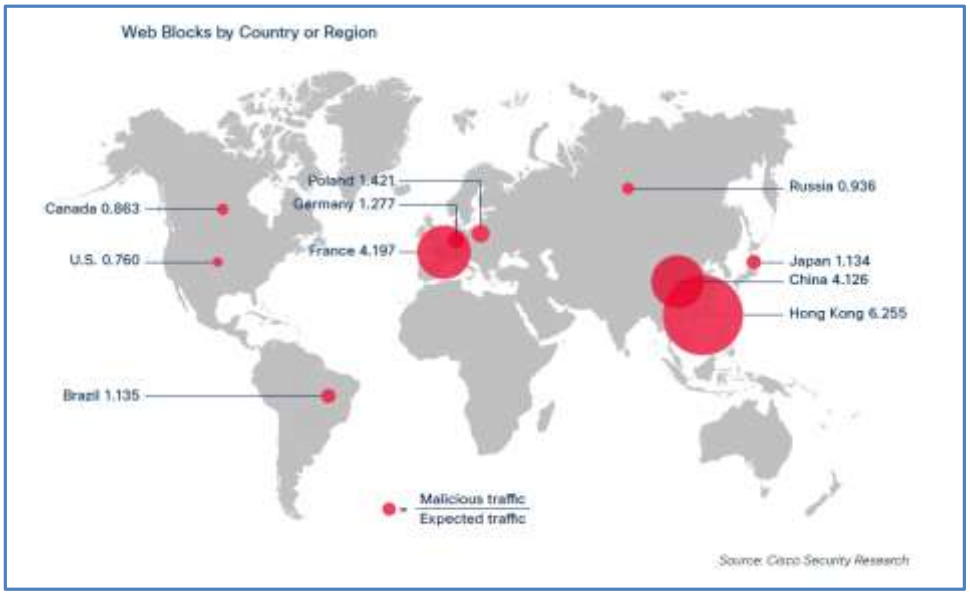
Redundância de Switch: Para garantir a alta disponibilidade para o tráfego do Centro de Dados é recomendado ter redundância em "cores switch". Além disso, estabelecer as regras de QoS (Quality of Service) para priorizar o tráfego para servidores de aplicativos Web e uma prioridade mais baixa de vídeo, correio, voz e arquivos.

Balanceador de Carga: O Balanceador de carga distribui as cargas de trabalho em vários computadores. O balanceamento de carga tem como objetivo otimizar a utilização de recursos, maximizar a produção, minimizar o tempo de resposta e evitar a sobrecarga de qualquer recurso único. O uso de vários componentes com balanceamento de carga em vez de um único componente pode aumentar a confiabilidade através de redundância. É necessária decodificação de tráfego SSL. (alguns produtos são: F5 Big-IP LTM, Cisco IOS-based router, Radware AppDirector, CoyotePoint Equalizer Appliances, Barracuda Load Balancer, Citrix NetScaler).

Certificados Secure Socket Layer (SSL): É necessário ter certificados SSL (criptografia de tráfego) para os servidores aplicativos (HTTPS). Uma vez comprado o balanceador de carga ele pode ser o decodificador de tráfego SSL para que a conexão aos servidores aplicativos seja HTTP.

Next-Generation Firewall (NGF): É uma plataforma de rede integrada que combina um firewall tradicional com outras funcionalidades de filtragem de dispositivos de rede, como um firewall de aplicativo usando inspeção profunda online de pacotes (DPI), um sistema de prevenção de intrusão (IPS) e / ou outras técnicas, como SSL e SSH interceptação, filtragem website, gerenciamento de QoS / largura de banda, inspeção de antivírus, Web SSL VPN e de integração de terceiros (ou seja, Active Directory). Para alta disponibilidade é sugerido ter 2 firewalls. De acordo ao Gartner Magic Quadrant são líderes: CheckPoint Software Technologies ou Palo Alto Networks). É preciso ativar a inspeção de antivírus. Uma característica dos NGFs e geralmente não configura-se, é o filtro geográfico. É importante para determinar a localização geográfica dos usuários que irão usar o Novo Sistema Tributário da SEFAZ-TO para poder configurar os filtros por países de acordo com o gráfico seguinte, que corresponde ao relatório Cisco 2015 Midyear Security Report[3] .

[3] Cisco 2015 Midyear Security Report. Ago 2015. http://www.cisco.com/go/msr2015



Next-Generation Intrusion Detection and Prevention System (NGIDPS): Sistema inteligente de prevenção de intrusos (ataques, hackers, segurança). São aparelhos de segurança de rede que monitoram as atividades de rede e/ou do sistema de forma a detectar procedimentos maliciosos. As principais funções dos sistemas de prevenção de intrusão são: identificar a atividade maliciosa, registrar informações sobre esta atividade, bloquear, parar e relatar estas atividades. Necessitamos 2 aparelhos, um para controle de borda e outro para controle dentro da Rede (LAN). (de acordo com Gartner Magic Quadrant são líderes: McAfee, SourceFire-Cisco e HP). Se SEFAZ-TO tem uma solução de UTM Firewall (Unified Threat Management - Gestão Unificada de Ameaças) não precisa esses equipamentos, mas deve ser considerado que se a solução UTM falhar, toda a organização estará desprotegida. Também se pode colocar o IPS em frente do firewall (boa pratica), sempre que seja determinado qual dispositivo (firewall ou IPS) pode bloquear maior quantidade de possíveis ataques, dependendo da robustez do equipamento, isto a fim de ter uma melhor proteção na primeira linha de defesa. Tal como mostrado na imagem:

Demilitarized zone (DMZ): Também conhecida como rede de perímetro, é uma subrede física ou lógica que contém e expõe serviços de fronteira externa de uma organização a uma rede maior e não confiável (Internet). A função de uma DMZ é manter todos os serviços que possuem acesso externo (HTTP, FTP, SMTP, etc.) junto em uma rede local, limitando assim o potencial de dano em caso de comprometimento de alguns destes serviços.

Honey Pot: DMZ. Conjunto de computadores cuja intenção é atrair os atacantes, fingindo ser vulnerável ou fraco para ataques. É uma ferramenta de segurança utilizada para coletar informações sobre os atacantes e suas técnicas. Honeypots podem distrair/enganar atacantes da estrutura, protegendo as máquinas mais importantes no sistema, e rapidamente alertar os administradores de um ataque, permitindo análise profunda do atacante, durante e após o ataque "honeypot".

Web Application Firewall (WAF): É a última linha de proteção para servidores de aplicativos. Um firewall de aplicativo é uma forma de firewall que controla a entrada, saída, e/ou o acesso para ou por uma aplicação ou serviço. Opera por meio do monitoramento e bloqueio de entrada, saída ou chamadas de serviço do sistema que não atendam a política configurada do firewall ou deste equipamento/software. São necessários dois appliances: um Defense Center e um SecureSphere (de acordo com Gartner Magic Quadrant: Imperva).

Application Performance Manager (APM): Sistemas de monitoramento e alertas baseados na experiência dos usuários, o sistema analisa e registra as condições normais de uso dos sistemas, baseado em cálculos matemáticos de desvios e tolerâncias definidas, alerta e toma providencias antes que os sistemas fiquem prejudicados. Essa ferramenta deve suportar o monitoramento usando transações sintéticas (simulated user interactions). Atualmente SEFAZ-TO têm Zabbix configurado pode suprir esta necessidade no caso de não ter orçamento para comprar essa ferramenta.

Log Servers: Ferramentas que executam em tempo real e pesquisa histórica, bem como relatórios e análise estatística. As operações de busca e de análise são especificadas usando uma Linguagem de Processamento de Pesquisa. O seu alcance inclui pesquisa de dados, filtragem, modificação, manipulação, inserção e exclusão. Analisa as tendências, correlaciona, identifica padrões, anomalias e exceções. Exemplo: Splunk e ELK (Elasticsearch, Logstash e Kibana) Stack.

Oracle Enterprise Manager (OEM): Ferramenta de Oracle para monitoramento do Banco de Dados Oracle, Servidores Aplicativos, Storage.

System Center: é uma plataforma de gerenciamento abrangente que ajuda a gerenciar data center, dispositivos clientes e ambientes de TI de nuvem híbrida de forma fácil e eficiente.

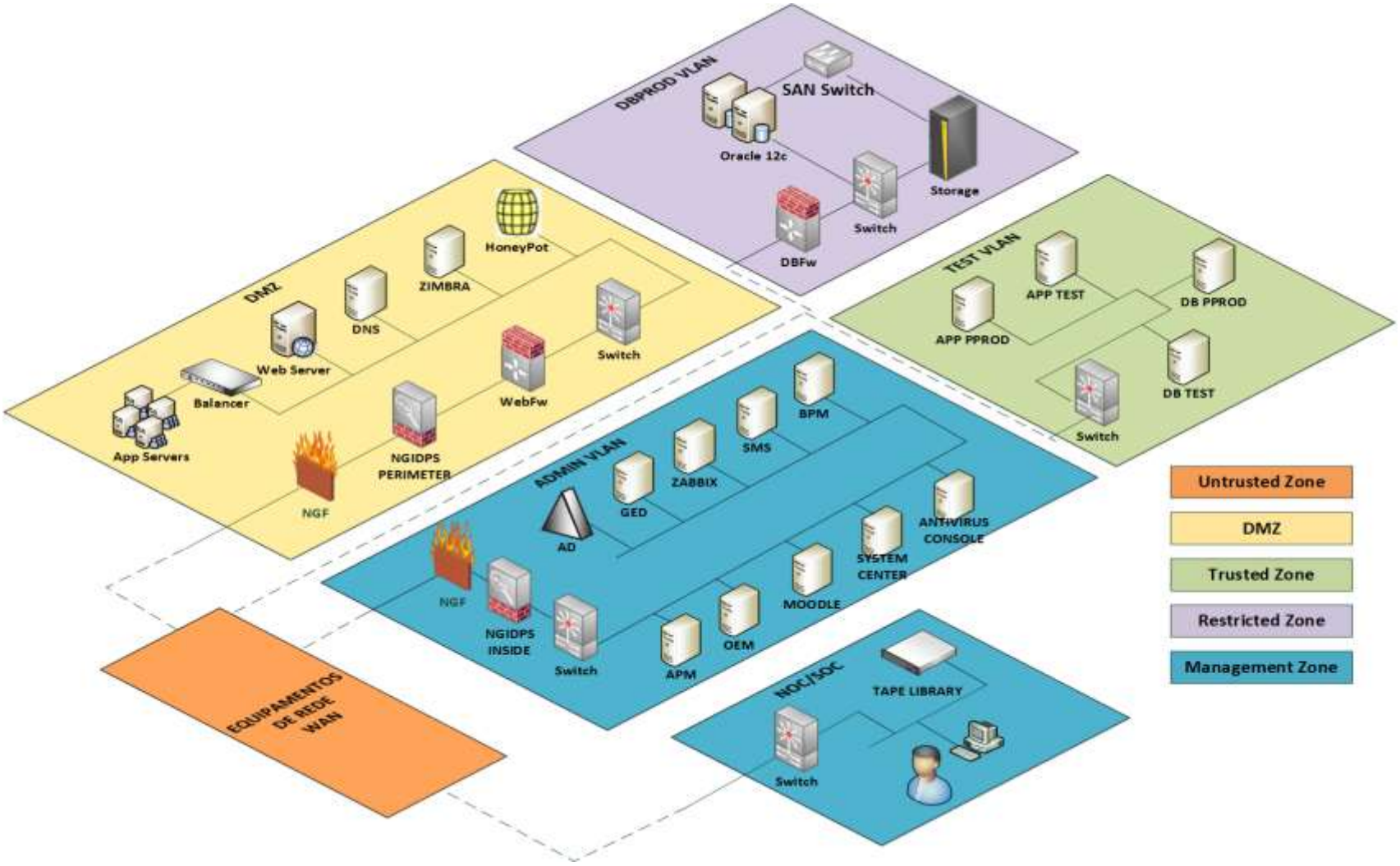
Cluster de Servidores de Aplicações: É um conjunto de servidores de aplicativos que foram instalados sobre JBoss Wildfly (Application Server) e está em cada servidor a mesma versão do Novo Sistema Tributário. Para servidores de aplicaçoes é aconselhável sempre definir pequenos servidores virtualizados (crescimento horizontal). Pode ser de 4 servidores com VMware ou outro Hypervisor que quiera usar SEFAZ-TO com suporte para Linux.

Oracle Database: Para o banco de dados é recomendável que se cumpram as normas previstas no Oracle Maximum Availability Architecture (Julho/2014) e estabelecer o nível de serviço para SEFAZ-TO de acordo o modelo OMAA[4][5] *[Oracle Maximum Availability Architecture]*. A sugestão para o banco de

[4] Oracle Maximum Availability Architecture http://www.oracle.com/au/products/database/maa-096107.html
[5] High Availability Best Practices for Database Consolidation
http://www.oracle.com/technetwork/database/availability/maa-consolidation-2186395.pdf

dados é um crescimento vertical e depois em grid. Ou seja, servidores robustos com alto poder de processamento e memória em RAC.

Diagrama da Arquitetura Tecnológica



3.2. Estratégia de Backup e Recuperação

Deve-se definir a estratégia de backup e recuperação de dados para cada um dos serviços que envolvem o sistema: o que inclui banco de dados, servidores de aplicação, sistema operacional, configuração total da plataforma tecnológica, etc. Este documento não tem esse escopo.

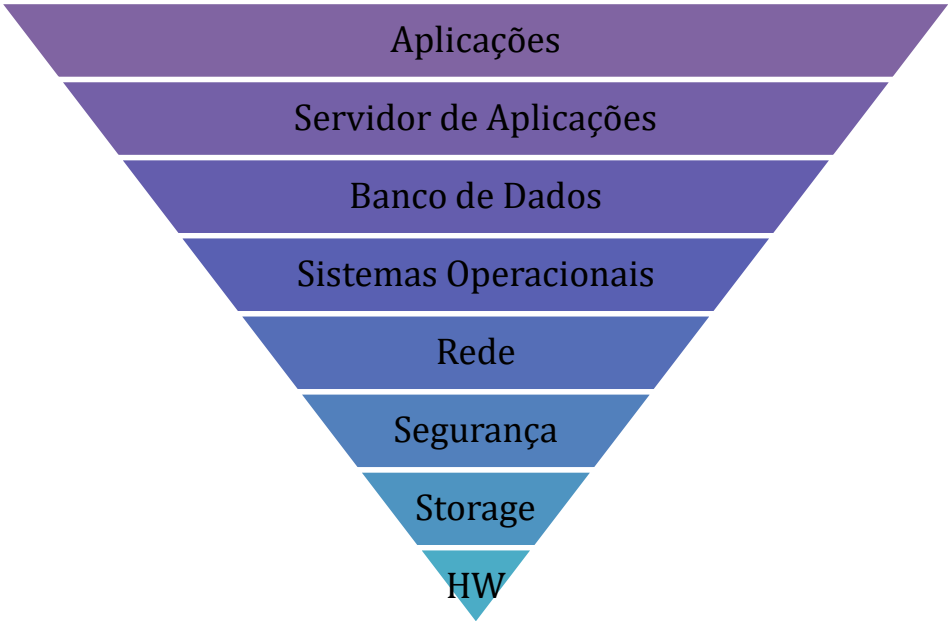
Os backups de servidores podem ser feitos com uma própria cópia de imagem com ferramentas tais como Ghost ou Symantec Backup Exec.

3.3. Desempenho do Sistema

Ter os sistemas de informação da empresa em execução com os melhores tempos de resposta e a melhor utilização dos recursos de hardware sempre foi o objetivo dos Departamentos de Informática. Para alcançar este objetivo é necessário um grande esforço que deve avançar passo a passo.

Melhorar a experiência do usuário deve ser uma prioridade no uso dos sistemas de informação, algumas características principais são: fácil de usar, acessível em todos os momentos, funcionalidade consistente, interface amigável, multi-plataforma (PC, laptop, tablet, celular), onde os objetivos de negócio e os objetivos dos usuários estão interligados. Além disso, é importante ter uma implementação de uma ferramenta APM (Application Performance Monitoring) disponível para a gestão da experiência dos usuários do Novo Sistema Tributário em NOC/SOC de SEFAZ-TO. Com essa ferramenta APM é possível identificar os possíveis gargalos no uso do sistema em tempo real.

Recomenda-se trabalhar a seguinte pirâmide de afinação onde os melhores resultados são alcançados quando são atendidos os requisitos de cima para baixo (top-down):



Esta pirâmide invertida mostra os melhores resultados de refinamento se as estratégias são aplicadas de cima para baixo, ou seja, os melhores resultados são obtidos nos níveis superiores. Mas há também uma relação inversa adicional: quanto mais se desce mais caras as soluções se tornam. Para esclarecer (por exemplo), é muito mais caro comprar um novo hardware que corrigir um aplicativo ou processo que está gerando "lentidão".

Para cada nível precisa-se de uma estratégia para sua execução (atividades):

Aplicações

- Manter as aplicações sempre executando com processos simples, eficientes, modulares, testáveis.
- Com instruções SQL altamente sintonizadas.
- Pool de conexões ao banco de dados.
- Processos de refactoring.
- Implementação de Framework ORM (Object Relational Management).
- Implementação de MVC (Model View Controller).
- Integração contínua
- Uso de Selenium (Web Interface Tester) e FireBug (Bug Finder).

Servidor de Aplicações

- Configuração para definir as informações mínimas exigidas nos logs (só erros e fatais).
- Configuração de logs em repositório único.
- Ativar o uso de Parallel Collector com o parâmetro –XX:+UseParallelGC
- Ativar compressão para: text/html, text/css, text/javascript, text/xml.
- Pooling de conexão para o banco de dados.
- Tunning de Java Virtual Machine (JVM).
- Configuração de cluster (no caso de que não vão usar balanceador de carga).
- Configuração do parâmetro max-active-sessions

Banco de Dados

- Ter instalada a última versão disponível do motor de banco de dados com todos os patches.
- Aplicar as melhores práticas para a afinação próprias de banco de dados: nível adequado de memória, parâmetros de instância, índices, armazenamento, planos de acesso atualizados para instruções SQL, etc.
- Ter uma estratégia implementada de alta disponibilidade eficaz e comprovada, conforme indicado na seção *[Oracle Maximum Availability Architecture]*.

Sistema Operacionais

- Ter atualizada a última versão dos sistemas operacionais, com todos os patches.

- Fazer um hardening ao servidor para excluir todos aqueles serviços não necessários pela aplicação, também considerar excluir as interfaces dos usuários próprias do sistema operacionais e só usar o “core server”.

Rede e Comunicação

- Sempre ter atualizado o firmware de cada dispositivo.
- Ter uma arquitetura de alta redundância como “stackable switches”.
- Routers com balanceamento de links de comunicações independentes.
- Bloquear as portas e protocolos.
- Remover os serviços que não são vitais para o uso eficiente da aplicação, tais como: protocolos e portas de videoconferência.
- Segmentar as redes (VLAN).
- Configuração dos parâmetros de desempenho do switch com base no volume de tráfego (priorização com Quality of Service).
- Nic bonding, permitir mais de uma interface de rede nos servidores.
- Avaliar o uso de aceleradores de tráfego WAN.
- Avaliar o uso de SDN (Software-Defined Networking) para uso no VNIC em Solaris.

Segurança

- Uso de dispositivos ou ferramentas que ajudem atenuar as vulnerabilidades que podem afetar o desempenho do sistema, como: Firewalls, Prevenção e Detecção de Intrusos, Web Application Firewall, Network Scanners, Encryptors, Forensics, Anti-malware, Antivirus, Rootkit Detectors, Packets sniffers. É importante o monitoramento permanente de alto consumo de tempos de resposta pelo uso destas ferramentas.
- Todo o equipamento deve ser configurado adequadamente de modo a que haja uma sobrecarga das operações do sistema ou gargalos.
- Configurar as conexões entre endereços IP dos servidores de aplicativos e banco de dados como confiável (IPTables).
- Reforçar as medidas de segurança para evitar ataques tais como: DDoS, Robôs, SQL Injection, etc.

Storage

- Ter sempre atualizado o firmware e os patches necessários.
- Usar as guias “best practices” do fabricante de Storage para Oracle.
- Configurar os parâmetros de alto rendimento nos storages externos.
- Usar RAID 5 na configuração dos servidores de aplicativos e RAID 1+0 para o banco de dados.
- No caso de ter discos de estado sólido em uma única LUN para que acessem os servidores de aplicações. Pegar o Novo Sistema Tributário em essa LUN onde todos os servidores acessem (um só deploy).

Hardware

- Ter sempre atualizados o firmware e patches necessários.
- Limpeza e manutenção preventivo cada 3 meses.
- Monitoramento constante do uso dos recursos.
- Desabilitar dispositivos não necessários.

3.3.1. Afinação em Oracle

Oracle Corporation tem 2 pacotes para usar no afinamento do banco de dados, são:

Oracle Diagnostic Pack

É parte dos produtos Oracle Database, oferece um conjunto completo de funcionalidades de monitoramento e diagnóstico automático de desempenho construído para o motor do banco de dados e Oracle Enterprise Manager. ODP fornece uma solução custo-efetiva e fácil de usar para gerenciar o desempenho do ambiente de banco de dados. Com as seguintes características:

- Automatic Workload Repository.
- Automatic Database Diagnostic Monitor (ADDM).
- Active Session History (ASH).
- Monitoramento de desempenho (banco de dados e servidor).
- Notificações de eventos: métodos, regras e programação.
- Histórico de evento e métricas (banco de dados e servidor).
- Paralizações.
- Baselines para métricas dinâmicas.
- Monitoring templates
- Monitoramento de desempenho baseado em acesso de memória.

Oracle Tuning Pack

Fornece aos administradores de banco de dados o gerenciamento de desempenho especializado do ambiente Oracle, incluindo ajuste de SQL e otimizações de armazenamento. Oracle Diagnostic Pack é um pré-requisito de Oracle Tuning Pack.

Inclui os seguintes recursos:

- SQL Access Advisor
- SQL Tuning Advisor
- Automatic SQL Tuning
- SQL Tuning Sets
- SQL Monitoring
- Reorganização de objetos

3.3.2. Afinação em Java

Embora Java é uma linguagem de programação de rápida execução para os programas mais deve sempre ter em mente que as aplicações funcionam da maneira mais ideal possível. Assim, o tempo de resposta (Response Time) dos programas é essencial.

Algumas dicas para afinação em Java são:

- Deve fazer afinação sempre de gargalos.
- Criar testes unitários dos gargalos.
- Quando um programa é analisado deve excluir os componentes como: a utilização da rede (Network utilization), ler/escrever no disco, ler/escrever no banco de dados. Ou seja, apenas analisar a execução de código.
- É sempre aconselhável desenvolver pensando em modularização e refactoring.
- Na seção *[Teste do Sistema]* tem ferramentas como JMeter para medir o consumo de CPU e memória. Medir o desempenho com Profilers.
- Para os processos é aconselhável ter um log para registrar milissegundos (início e fim), sempre medir *System.currentTimeMillis().* Exemplo: cadastro de declarações, processo de cálculo de conta corrente.
- Evitar a conversões de Strings.
- Evitar conversões de dados (casts).
- Validar sempre os dados de entrada antes do processo.
- Quando as exceções são lançadas pode gerar lentidão. É melhor ter todas as possíveis exceções consideradas.
- Evitar console output.
- Usar Caching.
- Usar Lambdas.
- Evitar configuração em xml, melhor usar anotações.

É recomendável o White Paper Java Performance, Scability, Availability and Security with Oracle Database 12c (Jun 2013) de acordo as implementações a ser feitas no banco de dados.

3.4. Considerações de Segurança do Sistema

As brechas de segurança são altamente onerosas para as organizações, como por exemplo: roubo de senhas, roubo de identificações de segurança social, roubo de cartões de débito e crédito, propriedade intelectual, uso indevido de informações confidenciais ou privilegiada, controle de dispositivos, etc.

As maiores ameaças de acordo com o relatório Cisco 2015 Midyear Security são apresentadas no *[Anexo II: Infographics of Sophisticated Attacks (First Half 2015)]* juntamente com uma vista circular de dispositivos de segurança a nível empresarial. Estes são: Angler, Rombertik, Adware Multiplug, Dridex.

É importante que SEFAZ-TO tenha um Plano de Segurança Informática e uma definição de Políticas de Segurança. Neste Plano de Segurança deve ter uma estratégia de defesa por níveis. De acordo a Gartner, pode ter diferentes níveis de defesa: Fundamentals, Advanced Technology and Lean Forward. Mas tudo depende do que quere proteger SEFAZ-TO junto com os investimentos em tecnologia do PDTI.

Na seção *[Componentes da Arquitetura Tecnológica]* detalha alguns dispositivos de segurança a nível de Infraestrutura e também o documento de Serviços Transversais fala sobre autorização e autenticação (IAM – Identity and Access Management) para o sistema. Mais aqui há outras contramedidas (para fazer em NOC/SOC) que se recomenda sua implementação para mitigar algum tipo de ataque que estiver presente no uso do Novo Sistema Tributário.

3.4.1. Hardening em Servidores

Trata-se de uma técnica que permite reduzir a área (superfície) de ataque de um possível indivíduo (hacker ou cracker) que quer pôr em risco o bom funcionamento dos servidores, algumas das ações que podem ser realizadas são as seguintes:

- Configure o BIOS do servidor para desativar dispositivos externos.
- Estabelecer senha para o BIOS.
- O particionamento de discos locais dos servidores, sempre se separa os dados em outras partições.

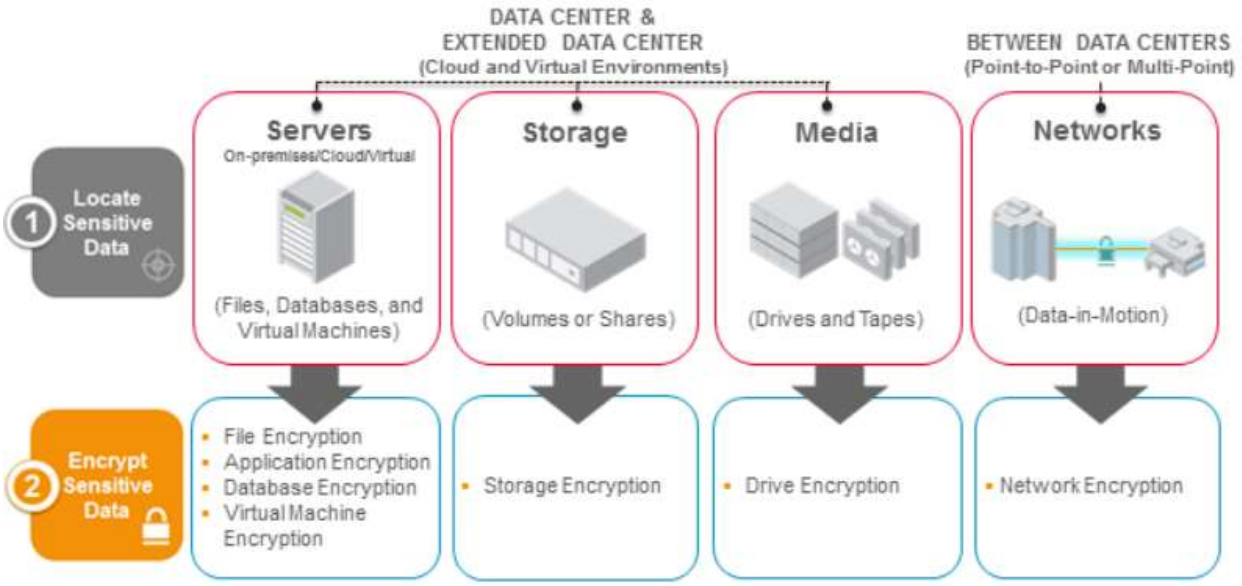
- Minimizar as características instaladas nos servidores, especialmente remover ou desinstalar aquelas que não estão sendo usadas, como por exemplo: servidores de impressão, detecção de plug and play, serviços de DHCP, etc.
- Verificar a lista de portas ativas por servidor.
- Usar ssh para os servidores, incluído os servidores Windows com (FreeSSHd).
- Manter sempre os sistemas operacionais atualizados especialmente os patches de segurança.
- Ter controle sobre as tarefas automatizadas que são executadas em servidores, se o servidor não precisa executar tarefas automatizadas (crontab) bloquear esse privilégio.
- Desativar o uso de USB.
- Para servidores Linux, configure SELinux.
- Para os servidores Windows, configure Windows Rights Management.
- Evitar o uso de pastas compartilhadas em servidores.
- Desinstalar os drivers gráficos do desktop em servidores. Por exemplo: em Linux (KDE/Gnome) e em Windows só deixar o Core Server.
- Remover as definições de IPv6.
- Estabelecer uma forte Política de Gestão de Senhas nos servidores. Bloquear contas não utilizadas.
- Expirar senhas nos servidores, não deixar senhas permanentes e mudar frequentemente.
- Habilitar a configuração IPTables (Firewall) em Linux e Firewall de Windows conforme o caso.
- Criar banners (mensagens) de advertência com referências legais sobre o uso dos equipamentos na SEFAZ-TO, tanto com sessões SSH como Windows Remote Desktop.
- Não ter dados (arquivos) em servidores de aplicações (somente configuração), toda informação deve residir no banco de dados.

3.4.2. Encriptação

Secure Socket Layer: É importante que o tráfego desde e para os servidores de produção seja criptografado por meio de certificados SSL.

Encriptação Discos: Os discos ou partições dos servidores de aplicações devem ser criptografados nos sistemas operacionais, no caso de roubo.

Ofuscação de Aplicações: Os programas que são implantados (em produção) sejam PL/SQL ou Java deve ter mecanismos onde o código fonte não é legível. Pode ser usado Oracle Wrap para PL/SQL e ProGuard para ofuscação em Java.



Encriptação no Banco de Dados: As informações utilizadas no banco de dados devem ser criptografadas, tais como: saldos de conta corrente de contribuintes, informações pessoais e empresariais, moratórias e dívidas, etc. A informação não deve ser acessível e legível para usuários não autorizados.

Oracle Advanced Security com Oracle Database 12c tem capacidade de redação de dados e criptografia pioneira da indústria, vitais para proteger informações sensíveis das aplicações. Transparent Data Encryption e Data Redaction ajudam prevenir o acesso não autorizado a informações confidenciais no nível da camada de aplicação, no sistema operacional, nos meios de backup e ainda nos exports de banco de dados. Oracle Advanced Security suporta plenamente a opção de Oracle Multitenant e está integrada com os sistemas da Oracle para desempenho “unparallel”.

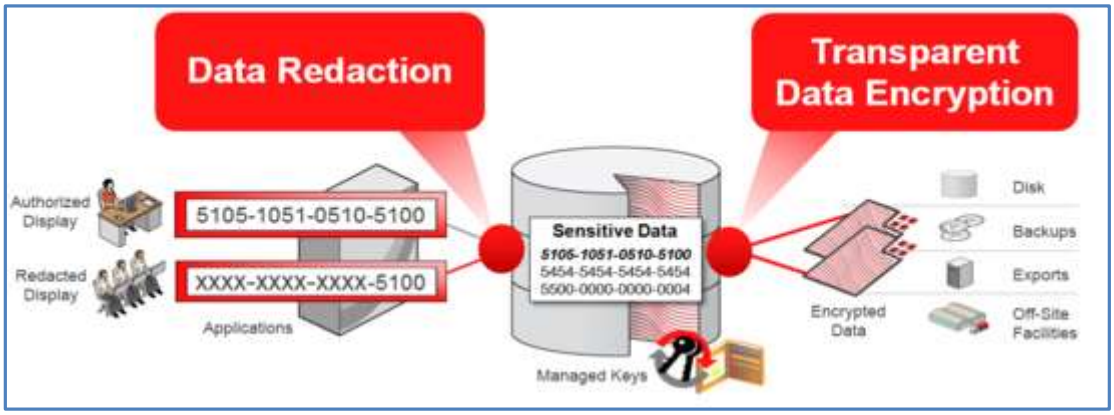
Transparent Data Encryption : assegura os dados sensíveis contra os acessos não-autorizados de fora do ambiente de banco de dados. Impede que usuários do sistema operacional privilegiados ou não-autorizados acessem diretamente a informação sensível para inspecionar o conteúdo dos arquivos no banco de dados.

TDE também protege contra roubo, perdas ou inadequado desmantelamento de meios de armazenamento do banco de dados e seus backups. A solução é transparente para os aplicativos porque os dados são criptografados automaticamente quando se grava no armazenamento e descriptografado quando se lê no mesmo armazenamento. Os controles de acessos são reforçados no banco de dados e na camada de aplicações permanecerá em vigor. As queries SQL nunca são alteradas e não é necessária qualquer alteração no código da aplicação ou em qualquer configuração.

Data Redaction: fornece redação seletiva de dados sensíveis nos resultados das consultas antes de ser mostrado para os aplicativos clientes, para que os usuários não autorizados não possam ver estes dados sensíveis. Os dados armazenados permanecem inalterados, enquanto os dados a serem exibidos são transformados on-the-fly antes de deixarem o banco de dados. Data Redaction minimiza as mudanças

de aplicação, porque eles não alteram os dados atuais na memória interna no banco de dados, cache ou armazenamento; e preserva os tipos de dados originais e seu formato quando os dados transformados retornam para o aplicativo. Também, Data Redaction não tem nenhum impacto sobre as atividades operacionais do banco de dados tais como backup e recuperação, atualização e execução de patches, ou clusters de alta disponibilidade.

Na imagem seguinte é mostrado como funciona Transparent Data Encryption e Data Redaction.



3.4.3. Oracle Database Vault

O Oracle Database Vault controla quando, onde e por quem os dados e aplicativos podem ser acessados, protegendo sua empresa contra a ameaça mais comum à segurança: usuários internos mal-intencionados. Aplicando a separação de tarefas, mesmo entre administradores, o Oracle Database Vault serve como um poderoso controle preventivo adicional para ajudar a atender ou superar a rígida conformidade e requisitos de privacidade de hoje em dia. Ele faz isso controlando o acesso aos dados de aplicativos e do banco de dados, mesmo por superusuários e outros usuários com altos privilégios. Ele também aplica a autorização de múltiplos fatores através de regras comerciais flexíveis e rastreia quem está acessando o que e quando, através de relatórios de segurança prontos para usar. Novos requisitos de controle interno encontrados em regulamentações podem ser difíceis e caros de implementar em um ambiente com diversos aplicativos. O Oracle Database Vault permite que os controles de acesso sejam aplicados de forma transparente sob os aplicativos existentes. Os usuários podem ser impedidos de acessar determinados dados de aplicativos ou de acessar o banco de dados fora do horário normal; os requisitos de separação de tarefas podem ser aplicados por diferentes administradores do banco de dados sem exercer os privilégios mínimos que resultam em mais custos.

3.4.4. Oracle Audit Vault

O Oracle Audit Vault é uma solução de classe empresarial de gerenciamento e consolidação de auditoria que permite que as organizações simplifiquem a geração de relatórios de conformidade, detectem ameaças de forma proativa, reduzam os custos e mantenham seguros os dados de auditoria. Tendo que lidar com inúmeras exigências de regulamentações e com a crescente preocupação em relação a ameaças internas, as organizações estão utilizando os dados de auditoria do banco de dados como uma medida importante de segurança, aplicando o princípio do "confiar mas confirmar". O Oracle Audit Vault

fornece uma visualização detalhada e abrangente dos dados de auditoria extraídos do banco de dados, ajuda a garantir a integridade dessas informações e pode reduzir o custo da conformidade tornando mais fácil para os auditores e a equipe de segurança gerenciar esses dados e gerar relatórios com base neles.

3.4.5. Atualização de firmware/software

Manter atualizado o firmware dos equipamentos sejam eles: servidores, routers, switches, dispositivos de segurança, impressoras, etc. é muito importante já que favorecem na segurança, alto desempenho e melhores características.

Sempre ter atualizado todo o software utilizado como: ferramentas de gestão de equipamentos, sistemas operacionais, banco de dados, virtualização, ferramentas do Office, browsers, etc.

3.4.6. Filtros por Mac Address

Deve evitar que dispositivos desconhecidos ou não autorizados acessem a rede da SEFAZ-TO, para isso é essencial identificar o MAC Address para cada computador, servidor ou impressora para fazer um filtro de acesso desde os switches distribuídos em cada andar do edifício (área). Não usar DHCP automaticamente sem conhecer o MAC Address de cada equipamento autorizado.

3.4.7. Segmentação de Rede (VLAN)

Segmentar a rede garante que os usuários só têm acesso a determinados computadores, dando um elevado nível de segurança e também melhora o tráfego, para citar um exemplo: os computadores do departamento financeiro não podem ser acessados a partir dos computadores do departamento tributário.

3.4.8. Nmap

É um software de código aberto que nos permite analisar a rede para encontrar servidores, identificar portas abertas, serviços em execução em servidores encontrados, identifica o sistema operacional, incluindo a versão. Esta ferramenta é utilizada para testes de penetração.

3.4.9. HTTrack Website Copier

Esta ferramenta, é livre para usar, permite copiar web sites inteiros em um diretório local. Ajuda a baixar diretórios de forma recursiva, páginas HTML, imagens, documentos e arquivos de um servidor web para um computador. A cópia obtida é quase como baixar um site espelho. Esta ferramenta como outras de seu tipo (Web Crawler ou Spider) são usadas para descobrir o nível de vulnerabilidade de um web site para potenciais robôs que realizam downloads automaticamente.

Precisa fazer configurações nos servidores de aplicações para detectar web crawlers (Robot Protection).

3.4.10. Wireshark

É uma ferramenta de software livre multi-plataforma analisador de protocolo de rede que permite examinar os dados "ao vivo" desde uma rede. Os dados capturados podem ser pesquisados de forma interativa no nível de detalhe necessário do pacote. Wireshark tem características poderosas: como filtro o que nós queremos visualizar e habilitar para mostrar um fluxo reconstruído de uma sessão TCP. Suporta centenas de protocolos, deve-se tomar cuidado para ter sempre instalada a versão mais recente deste software.

3.4.11. Windows Server 2012 Security Policy

É muito importante utilizar a definição de políticas de segurança em ambientes corporativos Windows. Essas políticas (regras) permitem que os administradores configurar um ou mais computadores para proteger seus recursos computacionais (computadores e redes). Uma definição ampla e detalhes podem ser encontrados no site TechNet da Microsoft. https://technet.microsoft.com/en-us/library/hh831424.aspx

3.4.12. Antivirus and Malware

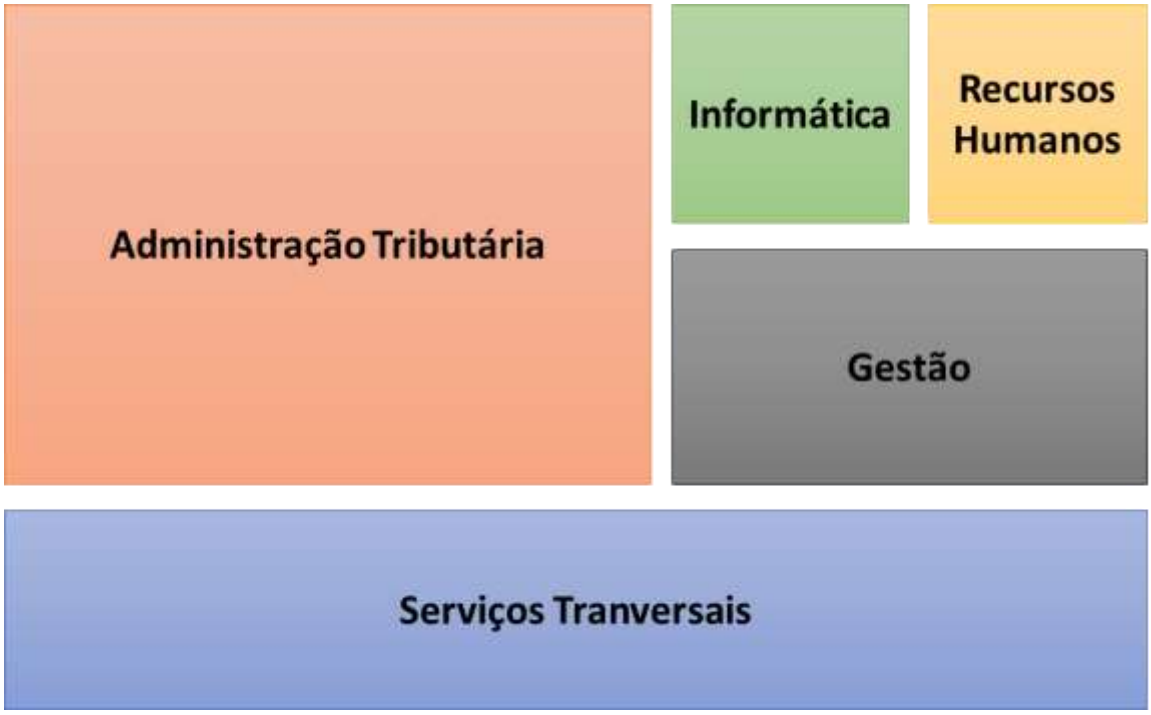
SEFAZ-TO deve ter um antivírus sempre atualizado e controlado através de um Console Centralizado. O antivírus deve incluir a detecção de malware também. O Next-Generation Firewall deve apoiar o uso de anti-vírus e malware como uma proteção adicional.

4. Arquitetura de Aplicações

Nesta seção tratamos: os componentes principais do sistema, as especificações da plataforma tecnológica Java Enterprise Edition 7, descrição da arquitetura da Aplicação Tributaria, os requisitos não-funcionais, definições de codificação, interface do usuário, teste do sistema e implantação do sistema em teste, pré-produção e produção.

4.1. Os Componentes Principais do Sistema

A seguinte imagem mostra o esquema de configuração estabelecido para o Sistema.



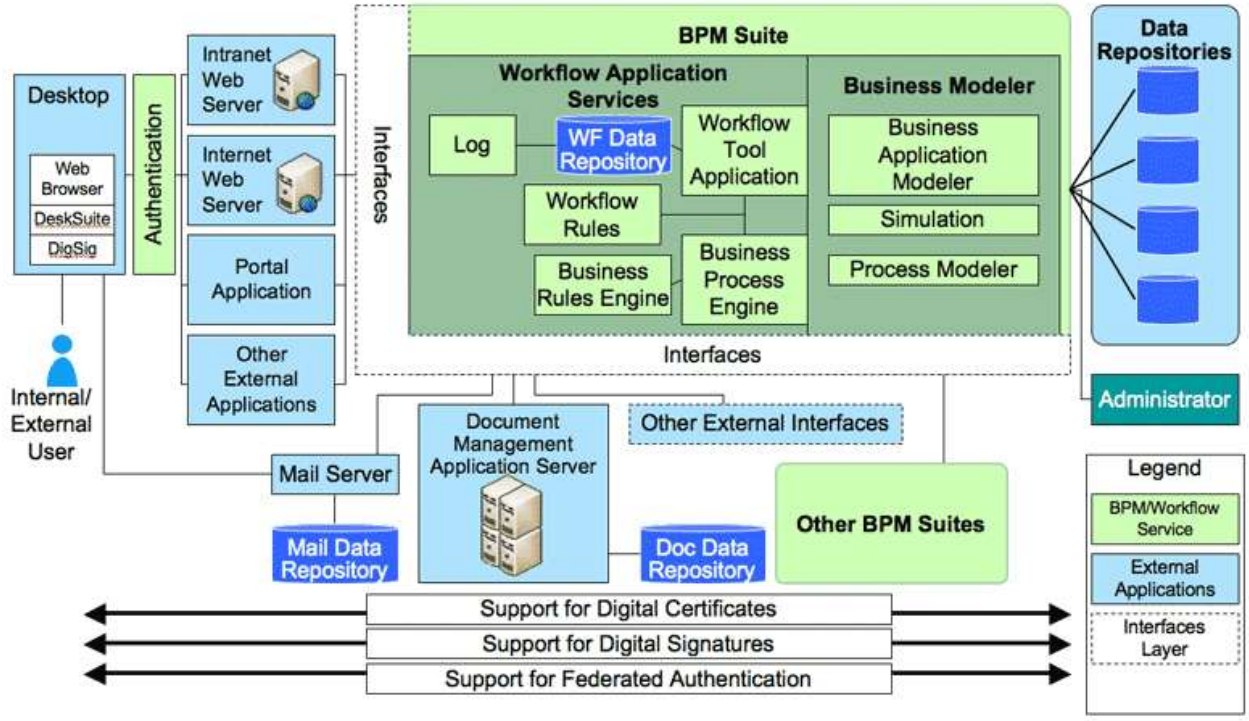
Neste esquema está mostrada a integração entre todos os componentes, incluídos os serviços transversais, que em conjunto cumprem com todas as necessidades para o novo Sistema. Para maior detalhe revisar o documento Definição dos Serviços Tecnológicos Transversais.

A seguinte lista é uma descrição de cada componente nos Serviços Transversais:

Gestão de Segurança: visa a garantir que o acesso aos recursos do sistema seja concedido, apenas aos elementos (pessoas, grupos, sistemas e equipamentos) devidamente autorizados para tal fim. Além do mais, esta gestão deve garantir que em todo o percurso trilhado pelo elemento, todas as ações fiquem devidamente registradas para posterior auditoria e controle.

Monitoramento dos Serviços: Permitem o controle de Eventos Anormais no Sistema e Suporte ao Controle de Qualidade. Esta solução gerencia exceções e erros que ocorrem no dia-a-dia do sistema. Tem uma tipificação de criticidade de exceções para medir o nível de serviço na experiência do usuário.

Gestão de Fluxos de Trabalho (BPM): Ferramenta de automação e gestão do fluxo de trabalho que implementa a grande maioria dos processos de negócio, além de prover um sistema de monitoramento de processo nativo.



Gestão de Documentos: é um componente integrado ao sistema para o gerenciamento eletrônico de documentos, com as funcionalidades de Captura de Documentos, Buscas e Pesquisas, Armazenamento e Indexação, Visualização e Impressão, Fluxo de Trabalho, Autenticação e Permissões, Assinatura Digital e Criptografia, tornando estas atividades mais ágeis, acessíveis, padronizadas, confiáveis e acima de tudo, rastreáveis.

Gestão de Comunicação Eletrônica: componente com funcionalidades para comunicação com contribuinte usando e-mail ou SMS.

Gestão de Controle Interno: componente para visualização dos indicadores operacionais de gestão institucional que permitam a identificação de pobre desempenho dos processos e verificar os níveis pré-estabelecidos de serviço.

Inteligência do Negócio: esse componente é um conjunto de ferramentas e estratégias de negócio para gerar conhecimento sobre a organização com a informação operacional de negócio, de forma rápida e fácil para os usuários finais.

Gestão de Formulários: componente que permite gerenciar todos os formulários pré-formatados e estruturados do sistema, por exemplo: Declarações de ICMS, Memo de Exportação, Auto de Infração. Esse componente permite modelar os formulários e criar as regras do negócio.

Monitor [Validador] Transacional de Formulários: É responsável pelo processamento de todos os formulários que introduziu o sistema de gestão de formulários. Esse processamento consiste em validação sintática, semântica, codificação, existências, etc. Uma vez que a execução lança as regras de execução de cada formulário, por exemplo: atualizar a conta corrente, atualizar o perfil de risco de contribuinte

Integração do Sistemas (SOA): esse componente gerencia todas as Interfaces de conexão com sistemas externos aos componentes aqui descritos. Geralmente para intercâmbio de informações com outras instituições. Por exemplo: Nota Fiscal Eletrônica com a Receita Federal, Atos Contenciosos para Procuradoria Estadual.

4.2. Matriz Sistema Organização (System Organization Matrix)

É muito importante ter um inventário do uso de aplicativos por área (departamento) na SEFAZ-TO, essa relação pode ser feita usando a Matriz Sistema/Organização. O propósito da Matriz Sistema/Organização é descrever a relação entre os sistemas (ou seja, componentes de aplicação) e unidades organizacionais dentro da empresa. Um exemplo pode ser encontrado na seguinte tabela:

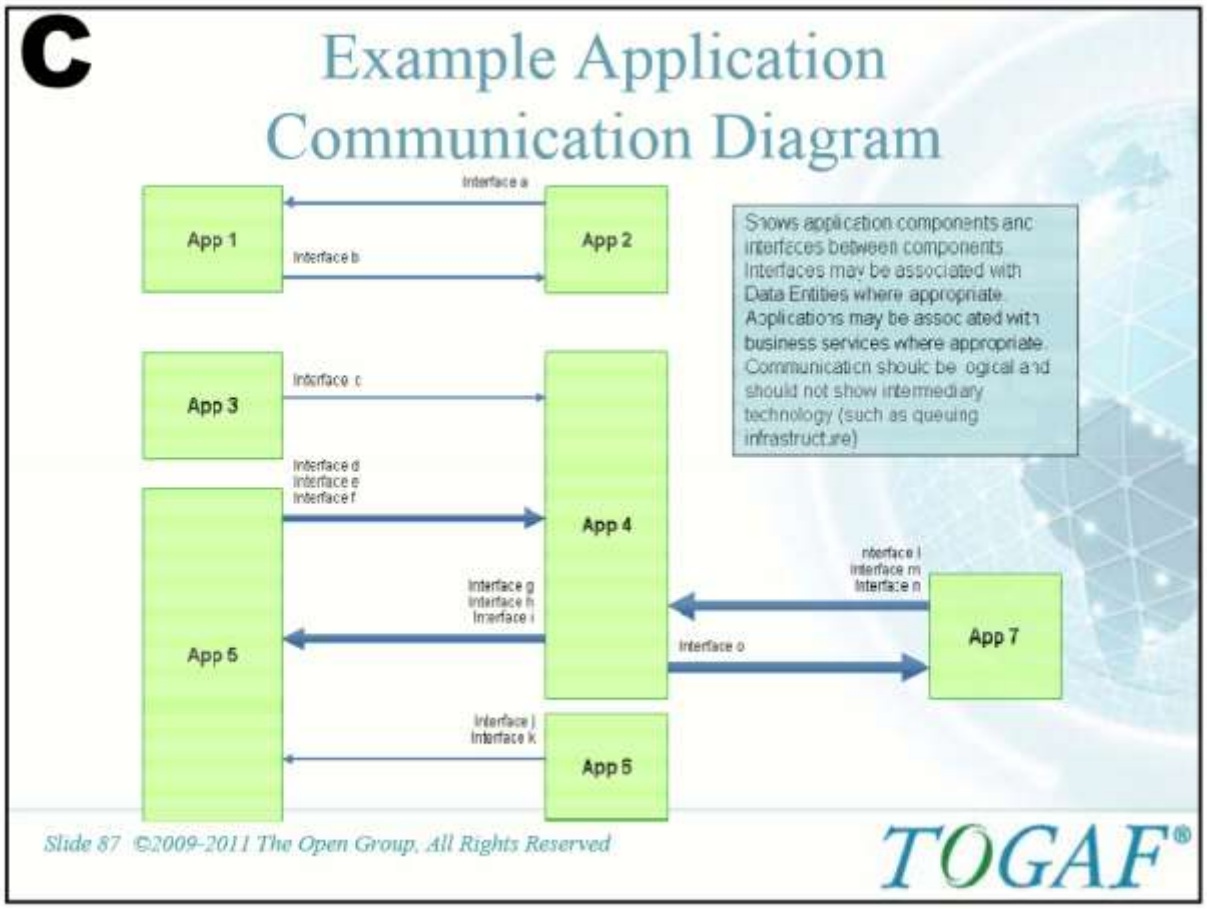
App/UO	Diretoria de Execução Financiera	Diretoria da Dívida Pública	Assesoría de Política Fiscal	Diretoria de Tributação	Diretoria de Avaliação de Resultados	Coordenadoria de Administração de Pessoal	Diretoria de Regímenes Especiais	Coordenadoria de Auditoria e Inspeção	Coordenadoria de Fiscalização
Cadastro		X		X				X	X
Arrecadação	X	X	X	X	X				

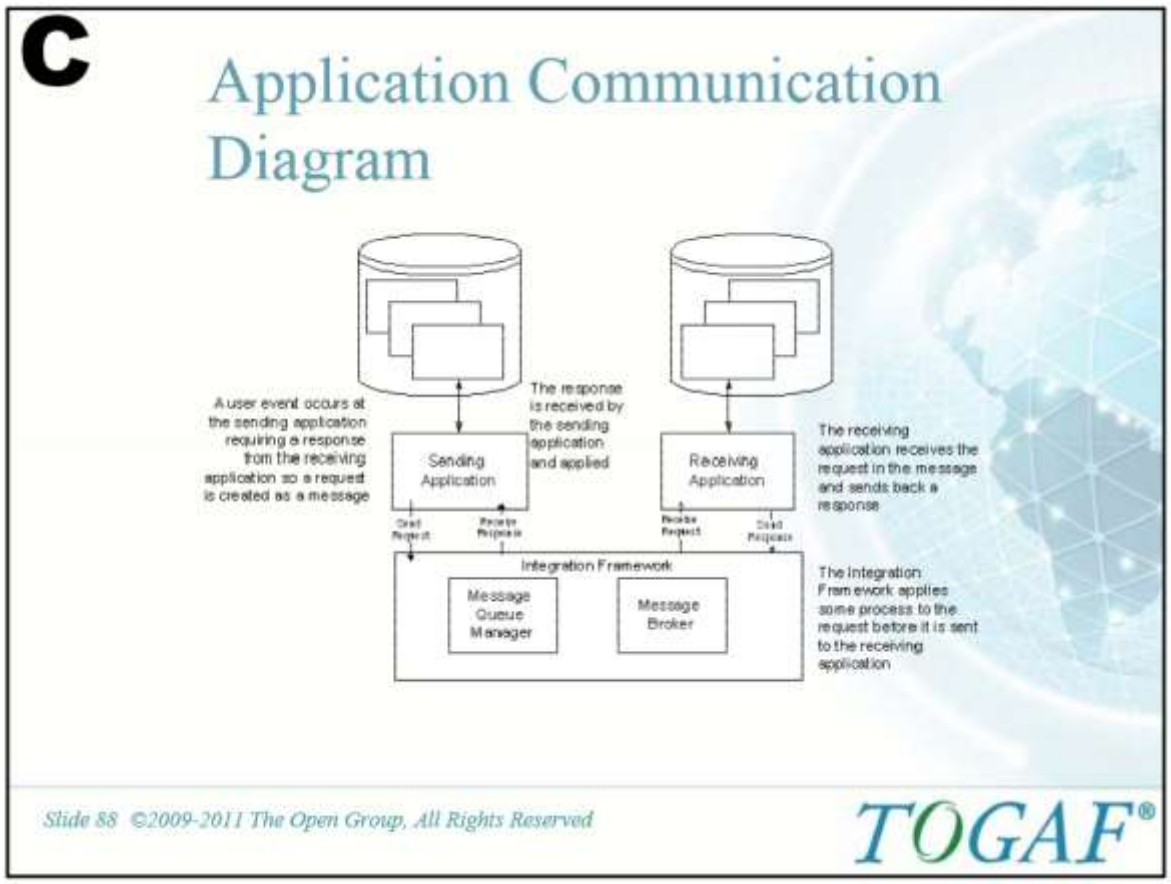
Restituições	X	X	X	X			X	X	X
Tributação	X	X	X	X	X				
Benefícios Fiscais	X		X	X					
Processo Declarações				X			X	X	X
Gestão ECF				X			X	X	X
Documentos Fiscais				X			X	X	
Gestão Atendimento					X	X			
Trânsito							X	X	X
Fiscalização				X					X
Cobrança	X	X		X					
Inteligência Fiscal								X	X
Contencioso			X		X				
Corregedoria									
GED	X	X	X	X	X	X	X	X	X
BPM				X		X	X	X	X

4.3. Diagrama de Comunicação de Aplicações (Application Communication Diagram)

A finalidade do Diagrama de Comunicação de Aplicações é descrever todos os modelos e mapeamentos relacionados com a comunicação entre as aplicações. Ele mostra os componentes dos aplicativos e as interfaces entre componentes.

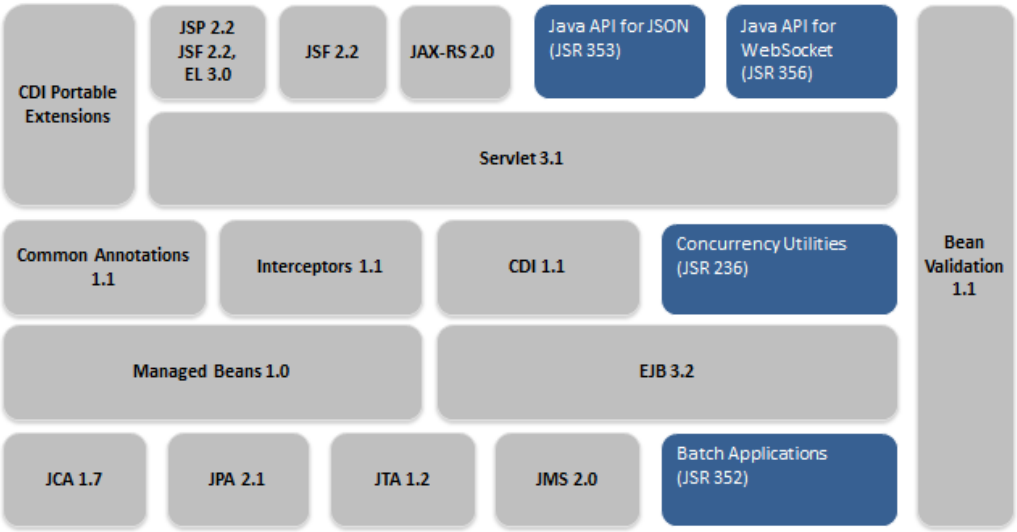
A comunicação deve ser lógica e só deve mostrar tecnologia intermédio em que é arquitetonicamente relevante. Um exemplo pode ser encontrado nas seguintes imagens:





4.4. Plataforma Tecnológica Java Enterprise Edition

O sistema será desenvolvido na plataforma Java Enterprise Edition. A seguinte tabela mostra as especificações da plataforma Java Enterprise Edition 7 (suportado por Wildfly 8.2).



Especificações	Descrição
JSR-356: Java API for Web Socket	O WebSocket é um protocolo e também uma API. As especificações tornam possível abrir uma conexão bidirecional permanente entre um cliente e um servidor, por canais full-duplex, que operam através de um único socket TCP. Isso significa que ambos os lados poderão se comunicar ao mesmo tempo, sem problemas de compartilhar portas com conteúdo HTTP existente, ou de conviver com proxy/firewall.
JSR-353: Java API for JSON Processing	JSON-P (semelhante ao JAXP) consiste de uma API de Transmissão (semelhante ao StAX) e uma API Object Model (semelhante ao DOM). JSON (JavaScript Object Notation) é um formato de intercâmbio de dados leve, e é usado em aplicações web, serviços REST e bancos de dados NoSQL.
JSR-340: Java Servlet 3.1	O principal pilar do desenvolvimento Web em Java, agora inclui servlet assíncronos. Com o processamento assíncrono de servlets, será possível fazer uma requisição para um servlet iniciar alguma tarefa em paralelo, enquanto é gerada a resposta para quem iniciou a solicitação. Suporte ao desenvolvimento de aplicações web para nuvem, usando o modelo de PaaS; melhorias de segurança, sessão e recursos; IO assíncrono baseado no NIO2, simplificação de servlets assíncronos, uso dos utilitários de concorrência do Java EE (JSR 236) e suporte a WebSockets.

JSR-344: JavaServer Faces 2.2	Integração com CDI, suporte a OSGI e a HTML5; listeners para eventos de navegação nas páginas; e componentes FileUpload e BackButton.
JSR-341: Expression Language 3.0	Possibilidade de referenciar membros e métodos estáticos; novas operações como igualdade, concatenação de Strings, sizeof e outros; integração com CDI.
JSR-245: JavaServer Pages 2.3	Framework para a construção da camada de apresentação em sistemas web desenvolvidos em Java. O objetivo é melhorar o alinhamento com JavaServer Faces (JSF) e aumentar a facilidade de desenvolvimento.
JSR-52: Standard Tag Library for JavaServer Pages (JSTL) 1.2	Encapsula como tags simples a funcionalidades principais comuns a muitos aplicativos da Web. JSTL tem suporte para tarefas comuns, estruturais como iteração e condicionais, as etiquetas para a manipulação de documentos XML, tags de internacionalização e tags de SQL. Ele também fornece uma estrutura para integrar tags personalizados existentes com tags JSTL.
JSR-352: Batch Applications for the Java Platform 1.0	Especifica um modelo de programação para aplicações em lote (batch); definindo os jobs e as suas unidades de execução; o agendamento da execução e o ciclo de vida será gerenciado pelo container.
JSR-236: Concurrency Utilities for Java EE 1.0	Fornece uma API simples, independente e clara para o uso da especificação de Concurrency Utilities (JSR 166), dentro do container Java EE.
JSR-346: Contexts and Dependency Injection for Java 1.1	Para gerenciar a construção de contextos; modo embutido para iniciar fora do container Java EE; controle declarativo sobre qual pacote ou bean deve ser verificado; injeção para métodos estáticos e envio de eventos de servlets como eventos CDI
JSR-330: Dependency Injection for Java 1.0	O objetivo é maximizar a reutilização, a capacidade de teste e manutenção do código Java, padronizando uma API de injeção de dependência extensível.
JSR-349: Bean Validation	Integração com as outras especificações do Java EE como JAX-RS e JAXB; suporte a níveis de validação, restrições em grupos e

1.1	composições com AND e OR.
JSR-345: Enterprise JavaBeans 3.2	Melhorias na arquitetura do EJB para execução em nuvem; novas anotações para facilitar a programação e aumentar o alinhamento com outras especificações da plataforma.
JSR-318: Interceptors 1.2	Interceptors são aplicáveis a Managed Beans em geral. Os Managed Beans são POJOs simples que têm o privilégio de serviços básicos pelo contêiner. Um método interceptor pode ser definido em uma classe target ou em uma classe de interceptor associado com a classe target.
JSR-322: Java EE Connector Architecture 1.7	Define uma arquitetura padrão para conexão a Enterprise Information Systems. O JCA é uma arquitetura genérica para ligar aplicações JEE a sistemas legados desenvolvidos em arquiteturas heterogéneas.
JSR-338: Java Persistence 2.1	Suporte à chamada de stored procedures e functions, geração de schemas; update e delete através da API Criteria; sincronização do contexto de persistência e injeção via CDI nos listeners.
JSR-250: Common Annotations for the Java Platform 1.2	O objetivo é desenvolver anotações para conceitos semânticos comuns nas plataformas JSE e J2EE que se aplicam a uma variedade de tecnologias individuais.
JSR-343: Java Message Service API 2.0	Mudança na forma de desenvolver, deixando mais simples a implementação; integração com CDI, melhoria no relacionamento do JMS com as outras especificações do Java EE, uma nova API obrigatória para permitir que qualquer fornecedor JMS integre-se com o container Java EE; além de algumas novas funcionalidades relacionadas a ambientes de nuvem.
JSR-907: Java Transaction API 1.2	Define uma interface de alto nível, além de anotações, e escopo para demarcar os limites da transação em uma aplicação transacional. Especifica interfaces locais que são utilizadas entre um gerenciador de transação e as partes que estão envolvidas em um sistema de transação distribuído como: a aplicação, o gerenciador de recursos e o servidor de aplicações.

JSR-919: JavaMail 1.5	É projetado para fazer adições de capacidade de correio eletrônico para simples aplicações facilmente, ao mesmo tempo, apoiar a criação de interfaces sofisticadas de usuário. Inclui classes adequadas que encapsulam funções comuns de correio e protocolos. Ele se encaixa com outros pacotes para a plataforma Java, a fim de facilitar o seu uso com outras APIs Java.
JSR-339: Java API for RESTFul Web Services 2.0	Criação e processamento facilitados de links associados aos recursos; validação com Bean Validation; uso de injeção de dependência e processamento assíncrono de requisições
JSR-109: Implementing Enterprise Web Services 1.3	Esta especificação define o modelo de programação e arquitetura de tempo de execução para a implementação da Web Services em Java.
JSR-224: Java API for XML-Based Web Services 2.2	Este projeto desenvolve e evolui a base de código para a implementação de referência da especificação Java API para XML Web Services (JAX-WS).
JSR-181: Web Services Metadata for the Java Platform	Define um formato de anotações em Java que usa uma linguagem de Metadados em Java para permitir a fácil definição de Java Web Services.
JSR-101: Java API for XML-Based RPC 1.1	Java APIs para suportar documentos XML emergentes da indústria baseada em padrões RPC.
JSR-196: Java Authentication Service Provider Interface for Containers 1.1	Busca definir uma interface padrão para que os módulos de autenticação possam ser integrados com contêineres de tal forma que estes módulos podem estabelecer as identidades de autenticação usadas por contêineres.

4.5. Tecnologia Multicamada JEE (Comparação)

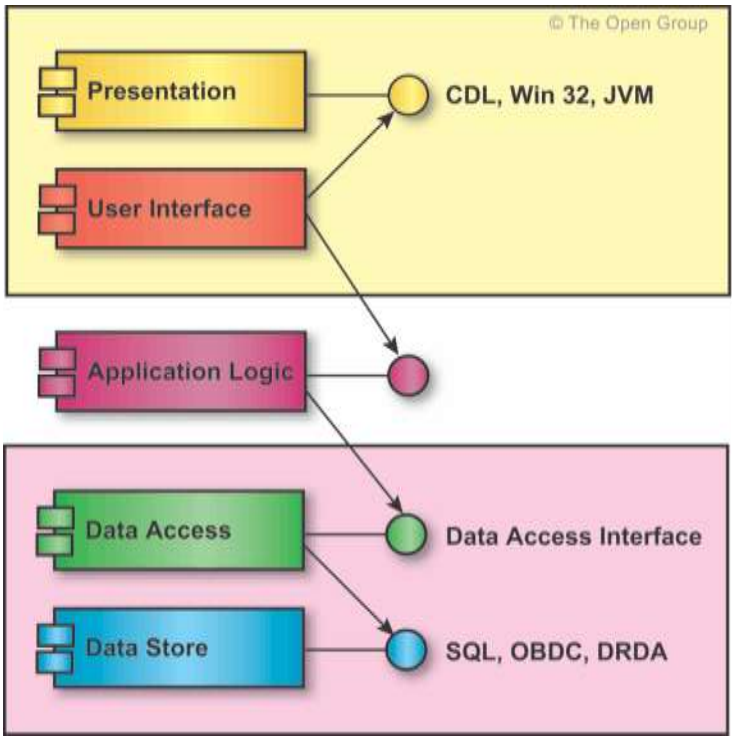
Java Enterprise Edition (Java Edição Empresarial) é uma plataforma de programação para servidores na linguagem de programação Java. A plataforma fornece uma API e um ambiente de tempo de execução

para o desenvolvimento e execução de softwares corporativos, incluindo serviços de rede e web, e outras aplicações de rede de larga escala, multicamadas, escaláveis, confiáveis e seguras.

Java EE estende a Java Platform Standard Edition (Java SE), fornecendo uma API para mapeamento objeto-relacional, arquiteturas multicamadas e distribuídas e web services. A plataforma incorpora um desenho amplamente baseado em componentes modulares rodando em um servidor de aplicação. Softwares para Java EE são primeiramente desenvolvidos na linguagem de programação Java. A plataforma enfatiza a convenção sobre configuração e anotações para configuração.

A plataforma Java EE é considerada um padrão de desenvolvimento já que o fornecedor de software nesta plataforma deve seguir determinadas regras se quiser declarar os seus produtos como compatíveis com Java EE. A plataforma contém bibliotecas desenvolvidas para o acesso a base de dados, RPC, CORBA, etc. Devido a essas características a plataforma é utilizada principalmente para o desenvolvimento de aplicações corporativas.

Na seguinte imagem mostra o desenho da arquitetura de alto nível do sistema para a implementação de Novo Sistema Tributário. A arquitetura proposta torna a aplicação mais simple e escalável.



Modelo geral de aplicações em camada Java Web

Então, vamos comparar algumas tecnologias disponíveis em Java. Para fazer isso as seguintes camadas são usadas: apresentação, lógica de negócios e persistência. A arquitetura Java EE centra-se nestas camadas que são chamadas “camadas do núcleo”, também conhecido como Middleware.

4.5.1. Camada de Apresentação

Essa é a camada onde serão produzidas e enviadas todas as requisições à lógica Java desde o usuário. Nela também se concentram o gerenciamento de sessões, salvar preferências (cookies), navegabilidade, eventos e efeitos gráficos, redirecionamento e single sign-on (SSO).

JSP, JSTL e Servlets[6]

Dentro dos desenvolvimentos rápidos sem requisitos adicionais, esta opção é a melhor. Usando programação direta em Java Server Pages, geralmente separa o código de apresentação do código Java, usando bibliotecas de tags de muitas fontes, acompanhadas de JSTL (Java Standard Tag Library) para algumas operações específicas.

Struts[7]

É um dos Frameworks mais populares, é um framework de desenvolvimento da camada controladora, numa estrutura seguindo o padrão Model2, ao contrário do Model1 referenciado na seção anterior (JSP, JSTL e Servlets). É bastante estável embora tenha sido iniciativa de código aberto, tem sido adotado por empresas de software dentro de suas ferramentas de desenvolvimento, facilitando a implementação em sistemas empresariais.

Java Server Faces[8] (JSF)

É um framework MVC em Java para a construção de interfaces de usuário baseadas em componentes para aplicações web. Possui um modelo de programação dirigido a eventos, abstraindo os detalhes da manipulação dos eventos e organização dos componentes, permitindo que o programador se concentre na lógica da aplicação. Foi formalizada como um padrão através do Java Community Process e faz parte da Java Platform Enterprise Edition.

Está especificamente orientado às telas de apresentação, desenvolvendo controles como bibliotecas de tags que tornam o desenvolvimento mais rápido, pensando em necessidades genéricas como validações de campo, tratamento de erros e apresentação de listas.

JSF 2 utiliza Facelets como seu sistema de planilhas padrão. Outras tecnologias da camada de visão, como XUL também podem ser empregadas.

Tapestry[9]

[6] http://www.oracle.com/technetwork/java/javaee/overview/index.html
[7] https://struts.apache.org/
[8] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
[9] http://tapestry.apache.org/

Tapestry é um framework Open Source para a criação de aplicativos Web em Java. Tapestry estende a funcionalidade do API Java Servlet executando-se qualquer Web Container. Está baseado em componentes de construção de páginas. É bastante robusto e permite a criação de aplicativos altamente manuteníveis.

Graças ao Java's advanced concurrency API, Tapestry lida com solicitações rápidas sem sacrificar a segurança ou a estabilidade.

Stripes[10]

É um framework de apresentação para a construção de aplicações web utilizando as mais recentes tecnologias Java. Baseado no modelo MVC (Model View Controller). O objetivo é ser um framework mais leve que Struts.

Google Widget Toolkit[11] (GWT)

O Google Web Toolkit é um toolkit de código-fonte aberto permitindo a desenvolvedores criar aplicativos com tecnologia Ajax em linguagem de programação Java. GWT suporta cliente-servidor, desenvolvimento e debugging em qualquer IDE Java. Exceto por algumas bibliotecas nativas, tudo é fonte Java que pode ser construído em qualquer plataforma com o GWT incluído Ant.

GWT enfatiza reutilização, soluções eficientes para os desafios recorrentes da Ajax, ou seja, chamadas assíncronas de procedimento remoto, gerenciamento de histórico, favoritos, Internacionalização e portabilidade entre navegadores.

Vaadin[12]

É um framework de código aberto para desenvolvimento de aplicações Web (RIA). Diferentemente de bibliotecas JavaScript e plug-ins para navegadores, Vaadin atua como uma arquitetura do lado do servidor, isto é, a maioria de sua lógica é executada no servidor. A tecnologia AJAX é utilizada no lado do cliente para assegurar uma experiência rica e interativa. No lado do cliente, Vaadin foi feito sobre e pode ser estendido com o GWT (Google Web Toolkit).

Um dos maiores recursos do Vaadin é a possibilidade de usar Java (utilizando a Java EE) como a única linguagem de programação para criar conteúdo web. O framework incorpora programação orientada a eventos e widgets, o que possibilita um modelo de programação similar à programação para GUI desktop, em contraste com o desenvolvimento web normalmente realizado com HTML e Javascript.

Vaadin utiliza o Google Web Toolkit (GWT) para renderizar a página resultante. Enquanto o GWT puro trabalha apenas no lado do cliente, ou seja, no mecanismo de Javascript do browser), o que pode levar a

[10] https://stripesframework.atlassian.net/wiki/display/STRIPES/Home
[11] http://www.gwtproject.org/
[12] https://vaadin.com/home

problemas de confiabilidade do JavaScript, Vaadin adiciona validação no lado do servidor para todas as ações. Isto significa que, se os dados do cliente estão com problemas, o servidor não os aceita.

Os componentes padrão do Vaadin podem ser estendidos com GWT widgets personalizados e com a utilização de temas CSS.

Sencha Ext JS[13]

É uma biblioteca publicada sob licença GPL utilizada para a construção de aplicações web interativos utilizando AJAX, DHTML e DOM. Este framework utiliza a arquitetura MVC como padrão de desenvolvimento, e oferece aos desenvolvedores diversos componentes de UI comuns às principais aplicações comerciais, como grid's, formulários, botões, combobox, checkbox e outros.

Recomendação para a Camada de Apresentação

A recomendação é escolher o framework mais próximo à especificação JEE. JSF tem excelentes fornecedores de suporte de ambientes JEE, foram 11 anos desde o lançamento inicial do JSF e está atualmente disponível a versão 2.2.

Existem diversas extensões de JSF como, por exemplo: Apache MyFaces, Oracle ADF, JBoss RichFaces, ICEFaces, JQuery4JSF, PrimeFaces e OpenFaces. MyFaces tem várias bibliotecas adicionais como Sandbox, Tomahawk e ExtVal.

Existem também muito bons IDEs (Integrated Development Environment) no mercado entre eles: Oracle JDeveloper, Oracle NetBeans e Eclipse. Eclipse tem um plug-in muito bom para trabalhar com JSF que é MyEclipse.

4.5.2. Camada de Lógica do Negócio

Essa camada representa o intermédio das demais na aplicação. É ela quem liga tudo que acontece na tela ao banco de dados, e, além disso, tem o papel de fazer validações, conversões, tratamento de exceções, etc. É necessário que a camada de negócio se adapte ao restante das camadas.

É nessa camada também que começa a ser pensada a divisão do aplicativo em partes menores, a modularização. Modularizar a aplicação em componentes torna o reaproveitamento de código bem mais objetivo. A partir dessa abordagem também conseguimos construir software com baixo acoplamento e razoável coesão.

Classes planas (Plain Old Java Object - POJO)[14]

[13] https://www.sencha.com/products/extjs/
[14] http://www.martinfowler.com/bliki/POJO.html

Definir a lógica de negócios em classes planas permite a portabilidade dessa lógica sem estar vinculado a nenhum tipo de framework ou tecnologia, é uma das melhores práticas no desenvolvimento de aplicações empresariais.

São objetos Java que seguem um desenho simplificado em contraposição aos EJBs. Este padrão é baseado na ideia de que quanto mais simples o projeto, melhor.

Enterprise Java Beans (EJB)[15]

É um componente da plataforma JEE que roda em um container de um servidor de aplicação. Seu principal objetivo consiste em fornecer um desenvolvimento rápido e simplificado de aplicações Java, com base em componentes distribuídos, transacionais, seguros e portáveis. Suporta anotações Java, que facilitam o desenvolvimento, diminuindo a quantidade de código e o uso de arquivos de configuração XML.

Um EJB normalmente contém a lógica de negócio que atua sobre os dados de negócio. Essa afirmação não era verdadeira até a introdução dos POJOs que também estão disponíveis nessa nova versão do EJB

Spring[16]

É um framework open source para a plataforma Java. Trata-se de um framework não intrusivo, baseado nos padrões de projeto inversão de controle (IoC) e injeção de dependência.

No Spring o container se encarrega de "instanciar" classes de uma aplicação Java e definir as dependências entre elas através de um arquivo de configuração em formato XML, inferências do framework, o que é chamado de auto-wiring ou ainda anotações nas classes, métodos e propriedades. Dessa forma o Spring permite o baixo acoplamento entre classes de uma aplicação orientada a objetos.

O Spring possui uma arquitetura baseada em interfaces e POJOs (Plain Old Java Objects), oferecendo aos POJOs características como mecanismos de segurança e controle de transações. Também facilita testes unitários e surge como uma alternativa à complexidade existente no uso de EJBs. Com Spring, pode-se ter um alto desempenho da aplicação.

Google Guice[17]

É um leve Dependency Injection Framework que pode ser usado por aplicações onde a relação/dependência entre Business Objects tem que ser mantida manualmente no código do aplicativo. Já que Guice suporta Java 5.0, leva o benefício da Generics e Anotações tornando o código type-safe.

Guice alivia a necessidade de “factories” e da utilização de “new” em seu código Java. Pense em Guice's @Inject como o novo “new”. Você ainda precisará escrever factories em alguns casos, mas seu código

[15] http://www.oracle.com/technetwork/java/javaee/ejb/index.html
[16] http://spring.io/
[17] https://github.com/google/guice/wiki/GettingStarted

não dependerá diretamente deles. Seu código será mais fácil de alterar, teste de unidade e reutilização em outros contextos.

Guice pretende tornar o desenvolvimento e depuração mais fácil e mais rápido. Quando os erros ocorrem, guice gera mensagens úteis.

Microservices[18]

É uma abordagem (approach) para o desenvolvimento de um aplicativo simple como um conjunto de pequenos serviços, cada um executando em seu próprio processo e comunicar-se com mecanismos leves (uma API de recursos HTTP). Estes serviços são construídos em torno de capacidades de negócios e funcionam através de mecanismos de deploy independentes totalmente automatizados. Há o mínimo possível de gerenciamento centralizado desses serviços, que podem ser escritos em diferentes linguagens de programação e utilizam diferentes tecnologias de armazenamento de dados.

Este tipo de arquitetura é de conceituação recente (2011) e já tem casos de sucesso como Netflix[19] e GloboTV[20] (no Brasil).

Recomendação para a Camada da Lógica do Negócio

Como o novo sistema tributário SEFAZ-TO será usado por milhares de usuários com muitas operações transacionais (CRUD) com pesadas validações e a necessidade de alto desempenho com tecnologia comprovada ao nível empresarial. Argumenta-se o seguinte:

Spring é uma implementação e EJB é uma especificação de JEE. As versões EJB1 e EJB2 são especificações complexas com fraco desempenho e redundância de código, historicamente rejeitadas no desenvolvimento ao nível empresarial. Com a nova especificação EJB3 (atualmente 3.2) diminuiu muito a reescrita de código pelo uso de anotações Java e melhorou muito o desempenho. Estes problemas (de complexidade de programação, alta curva de aprendizado e desempenho fraco) nas versões 1 e 2 do EJB criaram o aparecimento de Spring.

Spring é um ambiente de gestão de serviços que fornece similares características aos de um container EJB, mas com um esquema de programação consideravelmente mais simples e sem usar muitos recursos computacionais. Sendo Spring (implementação que resolve os problemas de EJB1 e EJB2) o líder no modelo de desenvolvimento na implementação do padrão de inversão de controle (IoC), é recomendado Spring em vez de CDI/EJB.

4.5.3. Camada de Persistência

[18] http://martinfowler.com/articles/microservices.html
[19] https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/
[20] http://www.infoq.com/br/presentations/de-monolito-web-para-arquitetura-de-microservices

Essa camada gerencia todos os acessos e operações com o banco de dados. A partir dessa camada temos disponíveis as funcionalidades de persistência (DML e DDL) na base de dados. Essa camada é que faz a comunicação direta com a camada de negócio.

Java Persistence API (JPA)

É uma API padrão da linguagem Java que descreve uma interface comum para frameworks de persistência de dados. A JPA define um meio de mapeamento objeto-relacional para objetos Java simples e comuns (POJOs), denominados beans de entidade.

Originou-se num projeto comum entre os desenvolvedores para se criar o padrão. Fortemente baseado nas ideias trazidas pelo Hibernate, tanto que o líder da primeira versão dessa especificação é o criador do framework.

Ou seja, é utilizado principalmente para conexão e acesso a banco de dados relacionais.

Hibernate

O Hibernate é um framework para o mapeamento objeto-relacional. Este framework facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, mediante o uso de arquivos (XML) ou anotações Java. Hibernate é um software livre de código aberto distribuído com a licença LGPL.

Tem vantagens em desenvolvimento e o desempenho do sistema, a sua configuração é baseada em arquivos XML ou anotações Java, o que permite que o processo seja flexível e de acesso a dados do sistema de gestão de transações tanto o código quanto a configuração.

Oracle TopLink

É um framework para a linguagem de programação Java desenvolvido pela Oracle que provê um mapeamento objeto-relacional O/R.

Data Acess Object (DAO)

É um padrão para persistência de dados que permite separar regras de negócio das regras de acesso a banco de dados. Numa aplicação que utilize a arquitetura MVC, todas as funcionalidades de bancos de dados, tais como obter as conexões, mapear objetos Java para tipos de dados SQL ou executar comandos SQL, devem ser feitas por classes DAO.

Recomendação na Camada de Persistência

JPA é a especificação JEE para persistência, recomenda-se usar Hibernate/JPA com Spring já que eles se integram com muita facilidade.

4.6. Descrição da Arquitetura da Aplicação Tributaria

SEFAZ-TO fez um investimento no ano passado na compra de servidores de banco de dados e servidores de aplicativos. Planejado para o crescimento do Novo Sistema Tributário, que é esperado para atender às demandas de serviços tecnológicos para os contribuintes e funcionários para os próximos anos.

Então, os servidores de aplicações destinados para a execução (atendimento) dos pedidos da Web e da lógica de negócios do sistema é recomendado uma configuração em cluster (conjunto) de vários servidores, a fim de aumentar a disponibilidade de aplicativos para os usuários.

Para o banco de dados, recomenda-se ter uma arquitetura de alta disponibilidade, tal como é descrito na seção do presente documento *[Oracle Maximum Availability Architecture]* e verificar o nível de classificação na seção *[Classificação (Níveis) de Arquitetura Oracle MAA]* que terá a arquitetura de banco de dados na SEFAZ-TO.

A comunicação entre os servidores de aplicações e os servidores de banco de dados será através de JDBC (Java Database Conectivity) para Oracle Database 12c.

Os navegadores usados pelos usuários internos (funcionários) e externos (contribuintes, contadores, etc.) vão se comunicar com os servidores de aplicações usando HTTPS. Os navegadores (browsers) devem ser modernos com as seguintes restrições em suas versões: Opera 30+, Mozilla Firefox 38+, Microsoft Internet Explorer 11+ (Microsoft Edge), Apple Safari 8+, Google Chrome 43+. São tomadas como referência as versões disponíveis em Junho/2015, e Windows XP não é mais suportado (terminou o soporte em 08 de abril de 2014) e até à data tentativa para colocar em produção o Novo Sistema Tributário (ano 2017) deve ter (SEFAZ-TO) uma nova versão de Windows mais recente nas estações de trabalho dos funcionários.

Para a camada da Web (Apresentação) será usado HTML5 e Java Server Faces. Para os serviços da lógica do negócio Enterprise Java Beans e para acesso o banco de dados JPA2 (Java Persistence API 2). A comunicação entre as camadas será feita utilizando o framework Spring.

Novo Sistema Tributário vai ser instalado como um arquivo EAR no Servidor de Aplicativos Wildfly 8.2, que terá as funções de Web Container e de EJB Container. Consulte a seção *[Implantação do Sistema em Teste, Pré-produção e Produção].*

4.6.1. Fluxo básico da Aplicação SEFAZ WebApp

A gestão de solicitações no Novo Sistema Tributário será feita de acordo com as seguintes etapas:

- Solicitar a página web de entrada (login) na aplicação Novo Sistema Tributário usando o protocolo HTTPS através do browser.
- O sistema terá filtros configurados (ServletFilter) com funções para validar o usuário, gerenciamento de sessão, criptografia das respostas e/ou validação de segurança de acesso. Esses filtros são ativados, dependendo da função do usuário (funcionário, contribuinte, etc.), uma vez que o acesso ao sistema tenha sucesso.
- Depois, todas as solicitações devem entrar através de um único componente Front Controller, no caso da tecnologia JSF, todos os pedidos devem ser atendidos pelo servlet FaceServlet, este componente é responsável por encaminhar os pedidos de acordo as regras configuradas.
- Uma vez chegado o pedido da página destino, esta vai validar os dados. Cada página JSF deve ser apoiada com um BackingBean ou ManageBean, que contém o código de controle da página. O Backing Bean deve ser registrado no arquivo de configuração no âmbito do Request.
- Nos casos em que seja necessário acessar aos serviços comuns para várias interações cada módulo deve ter um ManageBean, armazenado no âmbito da sessão.
- Tanto o BackingBean como o ManageBean acessarão a camada de serviços usando o padrão Service Locator, onde as ações do Locator serão feitas pelas classes internas do Framework Spring, que, usando seu arquivo de configuração, executará o processo de lookup das correspondentes classes que terão as interfaces de serviço ou fachadas necessárias para acessar a camada de negócio.
- Com o acesso a esses Proxies que implementam os métodos na interface exposta na camada de negócios, será feito o processamento dos pedidos.

4.6.2. Aplicação de Lógica Business App

A lógica de negócio será implementada em classes planas e expostas usando interfaces de Serviço e Fachada que permitam obter o acesso de forma simples à camada interna. Melhorando assim a apresentação dos métodos no contrato da Fachada e a implementação nas classes concretas de serviço. O controle dos processos de negócios será realizado usando o Spring Framework, assim Spring vai gerenciar: o ciclo de vida dos componentes de serviço, os limites transacionais, o acesso a camada de persistência e a injeção de serviços nas classes clientes (IoC).

Considerações adicionais a ter em conta

- Será usado o Padrão de Desenho Estrutural Fachada, as interfaces fachada vão expor os métodos de negócio (contrato) para classes de serviço com lógica específica do módulo (por exemplo: Classe de Serviço Cadastro de Contribuintes). Os métodos definidos na interface e a classe de serviço devem ter os mesmos nomes e argumentos, garantindo que o passo dos parâmetros é através de Transfer

Objects. A fachada reduz o acoplamento entre componentes clientes (de apresentação como Backing Beans), quando vai invocar mais que um serviço de negócios.

- Cada caso de uso terá um serviço de interface que será exposta através da fachada, esta interface será implementada por uma o mais classe(s) de serviço especifica POJO (Plain Old Java Object), essa(s) classe(s) tem a lógica do negócio codificada.
- Quando essa classe de serviço precisa acesso ao banco de dados, ela faz o acesso através da camada de infraestrutura de serviços de persistência fornecida pelo Framework de Hibernate, invocado e inicializado com Spring. É importante sempre usar (criar) métodos disponíveis na camada de persistência com sobrecargas diferentes, especialmente nas classes Finder de forma genérica assim eles são comuns a todos os módulos.
- Criar-se-a as entidades de domínio, que vão armazenar o estado da aplicação e serão os Transfer Objects usados para a comunicação entre as camadas.
- Os parâmetros de entrada e saída dos métodos devem ser objetos do tipo: Transfer Objects, coleções de Transfer Objects ou arrays de Transfer Objects. Nos casos em que se precisa para trabalhar com tipos de dados primitivos, deve-se usar a classe Wrapper associada (exemplo: int com Integer). Não dever usar XML o JSON para as interfaces JSF, somente em serviços SOA ou mobile (smartphones ou tablets).
- É necessário manter, se possível, a menor quantidade de objetos (entidades) na sessão de cliente Web. Não se deve deixar coleções de objetos na sessão.
- Podem-se usar utilitários adicionais, tais como: a API do Apache Commons, Quartz, Joda (com Java versão 8 e não deve ser usado), etc. Mas deve-se ter cuidado de sempre estar definido no arquivo POM (Maven) de projeto.

Aqui há um exemplo de um Diagrama de Sequência de Cadastro de Auto-Infração para ficar claro, onde:

Auditor: Ator neste diagrama de sequência. Ele está autorizado para fazer um auto infração e é designado pelo menos um caso de fiscalização, que deve registrar as conclusões dos achados obtidos.

UICadastroAutoInfracao: interface de usuário que representa a tela junto com o Backing Bean, onde os métodos para interagir entre as operações determinadas com o ator e a fachada do módulo AutoInfração.

autoInfracaoFachada: uma instância da Fachada de AutoInfração é um contrato com os métodos disponíveis para consultar, cadastrar e fazer todas as operações incluídas no módulo de AutoInfração. Aqui estão os serviços que têm relacionamento com as operações deste módulo.

contribuinteSvc: uma instância dos serviços do Contribuinte (consultar, cadastrar, alterar, etc.).

autoInfracaoSvc: uma instância dos serviços do AutoInfração (consultar, cadastrar, alterar, etc.).

contaCorrenteSvc: uma instância dos serviços da ContaCorrente (consultar, cadastrar, alterar, agregarDívida, calcularSaldo, etc.).

finder: uma instância da Classe de pesquisa do módulo de persistência para as consultas sobre o banco de dados.

crud: uma instância da Classe de criação, atualização e exclusão do módulo de persistência para os cadastros e alterações de informação no banco de dados.

alfrescoDocSvc: uma instância da Classe de Integração com o sistema GED (Alfresco), para criar os relatórios (documentos) do AutoInfração.

bmpSvc: uma instância da Classe de Integração com o sistema BPM (Bonita), para iniciar o fluxo de aprovação do AutoInfração pelo Chefe de Fiscalização.

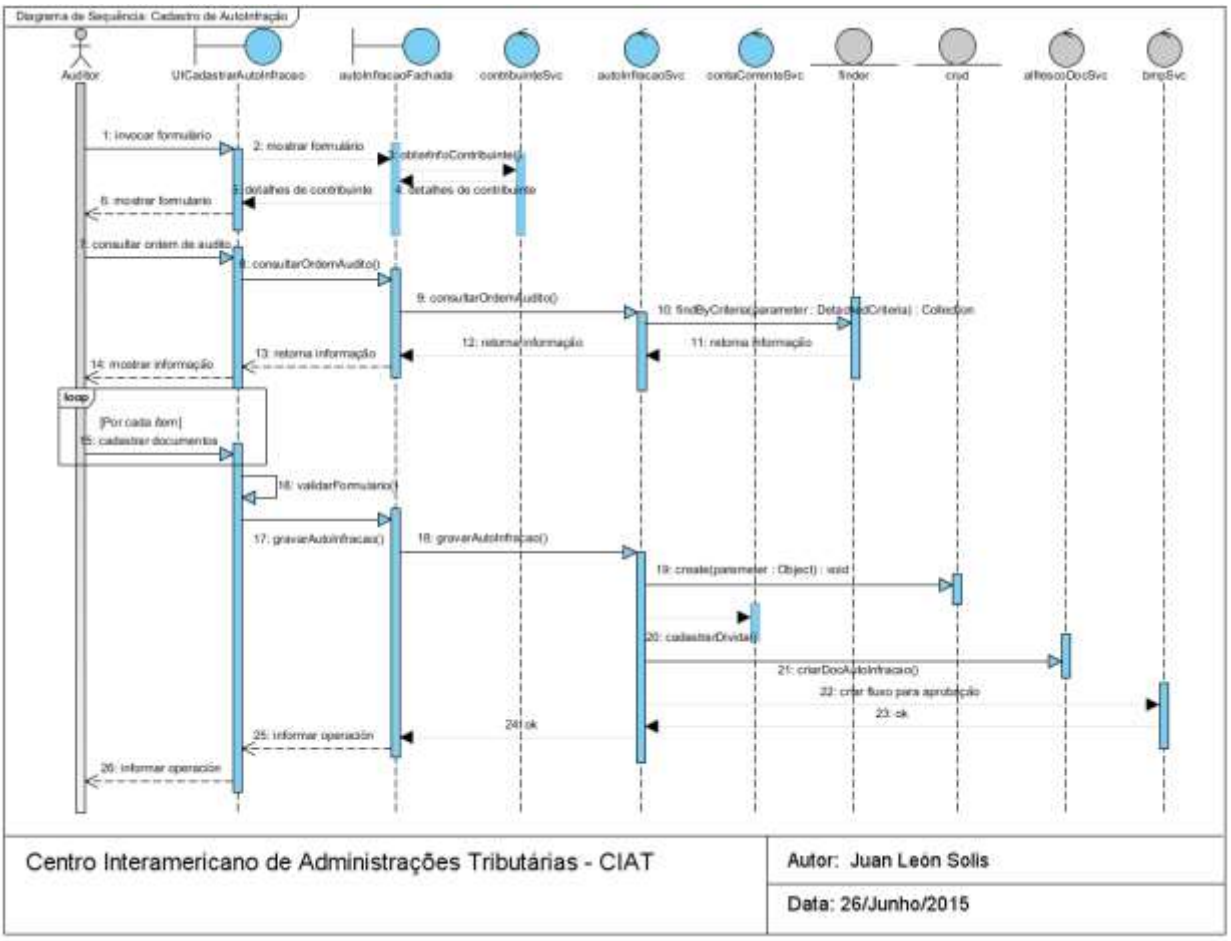


Diagrama de Secuência de Fluxo básico de um Aplicativo

4.7.Requisitos não-funcionais

As seguintes são funcionalidades que têm que ser providas pelo sistema, deve-se ter em conta cada uma delas, como parte integral do sistema.

- Facilidade de uso: os usuários deverão operar o sistema após um determinado (pouco) tempo de treinamento.
- Desempenho: obter o tempo mínimo e/ou máximo (permitido) na resposta de um processo. Exemplo: ao consultar a informação do contribuinte deve aparecer em, no máximo, 2 segundos).
- Disponibilidade: necessidade de acesso a informações a qualquer momento. O sistema estará disponível 99.8% em dias de semana de 08:00 até 18:00.
- Usabilidade: facilidade para fazer a operação (tipo de perfil de usuário definido no caso de uso, técnico, conhecedor da lógica, etc.)
- Escalabilidade: permite a capacidade do crescimento do sistema (usuários, transações).
- Eficiência: estabelecer a quantidade de requisições que o sistema deverá processar por um determinado tempo.
- Confiabilidade: ter certeza de que as operações feitas no sistema estarão cadastradas sem problema no banco de dados.
- Portabilidade: o sistema deverá rodar em qualquer plataforma (diferentes browsers).
- Flexibilidade: dar a possibilidade de requisitos e/ou regras de negócio mutantes com o tempo.
- Reusabilidade: dar a possibilidade de usar a funcionalidade em outros contextos.
- Extensibilidade: dar possibilidade de adição da funcionalidade.
- Frequência: considera-se o número de invocações do caso de uso em um período de tempo.
- Padrões: uso de programação orientada a objeto.
- Aspectos Legais: o sistema deverá atender às normas legais, tais como padrões, leis Estaduais ou Federais.
- Segurança: garantir as restrições necessárias para a execução do caso de uso, perfis de acesso, a necessidade de criptografia ou estabelecimento de canais seguros para a transmissão de informações.
- Transacionalidade: escopo transacional das operações, total ou parcial.
- Concorrência: acesso para visualizar ou modificar dados no mesmo instante de tempo.
- Documentação necessária: documentação online de ajuda dentro de cada tela para seu uso.

4.8.Definições de codificação

As seguintes são definições para considerar no momento de desenvolvimento do sistema com a missão de uma boa qualidade de código.

4.8.1. Validação de entrada de dados

O processo de validação de dados é executado em três níveis distintos que fornecem melhor confiança na qualidade dos dados ao longo do tempo no banco de dados na SEFAZ-TO.

No primeiro nível serão validações a partir do navegador, ou seja, a partir do mesmo cliente Web para entrar dados nas telas. Esta validação de dados será realizada com a biblioteca JSF para aquelas validações de tipo: numérico, alfabético, alfanumérico, dados formatados, números inteiros positivos ou negativos, controles aritméticos, lista de valores pré-carregados, datas, intervalos de datas, valores exigidos, espaços em branco, limites do texto, limites numéricos, maiúsculas, minúsculas, expressões regulares, padrões, limite o tamanho do campo a ser inserido, definição do conjunto de caracteres aceito pela aplicação, etc.

Em caso de criação de novos validadores (Validators), estes devem ser incluídos no Framework Web de modo que estejam disponíveis para todo o projeto. Além disso, é aplicável conversores de JSF (se necessário) para que os dados passem para a próxima camada com os tipos de dados e valores corretos.

Também pode ser aplicado neste nível de validação: controle de tipo atributos html (maxsize) ou funções JavaScript (evento onblur()).

No segundo nível, criar-se novas classes que fazem as validações sobre a camada de negócios. Estas classes invocam-se antes de fazer qualquer operação, cálculo ou processo não importa se vêm da camada de apresentação (cliente) e já foram validados. Porque a camada de lógica de negócios pode ser chamada a partir de outros clientes. Por exemplo: outro módulo ou um serviço disponível na Camada de Integração. Pode usar também o Framework Commons de Apache.

No terceiro nível, é a criação de restrições (constraints) no banco de dados a fim de preservar a integridade dos dados de acordo como foi definido no modelo relacional, este tipo de validações podem ser: check constraints, not null, unique, primary key, foreign key. Por exemplo: valor de pagamento de ICMS maior do que zero.

Tipo	**Exemplo**
CHECK	ALTER TABLE SEFAZ_PAGAMENTO.RECOLHA_PAGOS ADD (CONSTRAINT CHK_VALOR_01 CHECK (VALOR_PAGAMENTO>0));
CHECK	ALTER TABLE SEFAZ_DECLARACAO.SIMPLE_NACIONAL ADD (CONSTRAINT CHK_ESTADO_EXCLUIDO CHECK (ESTADO_EXCLUIDO IN ('S','N')));
NOT NULL	ALTER TABLE SEFAZ_PAGAMENTO.RECOLHA_PAGOS MODIFY (VALOR_PAGAMENTO NOT NULL);

UNIQUE	ALTER TABLE SEFAZ_CADASTRO.CONTRIBUINTE ADD CONSTRAINT UK_ESTADO_CONTRIBUINTE UNIQUE (IDENTIFICACAO_CONTRIBUINTE, ESTADO_CONTRIBUINTE)
PRIMARY KEY	ALTER TABLE SEFAZ_CATALOGO.TIPO_PESSOA ADD CONSTRAINT PK_TIPO_PESSOA PRIMARY KEY (CODIGO_TIPO_PESSOA);
FOREIGN KEY	ALTER TABLE SEFAZ_CADASTRO.CONTRIBUINTE ADD CONSTRAINT FK_TIPO_PESSOA FOREIGN KEY (CODIGO_TIPO_PESSOA) REFERENCES SEFAZ_CATALOGO.TIPO_PESSOA(CODIGO_TIPO_PESSOA);

4.8.2. Paginação

Para os casos de uso que depois de executar uma consulta cujos resultados são mais do que 20 registros, deve-se usar paginação para otimizar a memória do servidor de aplicativos. O processo de armazenamento de lista de objetos exibidos em grid de JSF será realizado por Request (HTTPServletRequest) deve evitar usar o objeto Session (HTTPSession). Os controles próprios de JSF (dataTable) vai iterar nos resultados (lista de elementos).

Em geral, o componente do serviço da aplicação invocado (por exemplo: SimplesNacionalSvc) deve consultar o banco de dados, usando ferramentas existentes no módulo de framework do sistema na camada de dados, que através de Hibernate/JPA deve devolver o conjunto exato de registros para mostrar na tela ao usuário, esses dados serão ordenados conforme a necessidade.

Como regra, o ordenamento terá lugar no banco de dados, evitando assim alto consumo de processamento ao nível de servidor de aplicações. Apenas em casos em que não seja possível fazer o ordenamento dos resultados no banco de dados serão utilizadas classes de API de Framework local para este trabalho.

4.8.3. Exportação de dados a arquivos planos

Certamente, em muitos casos de uso, haverá resultados de consultas que podem ser exportados para arquivos simples, são sugeridos os formatos csv (comma-separated values), pdf e excel. Deve ser considerada a política de proteção de dados para a informação que pode ser gravada fora do sistema. A exportação de informações pode ser feita de uma forma genérica para todas as consultas com o componente JSF DataTable.

4.8.4. Cache

O cache é uma das técnicas mais importantes para melhorar o desempenho de aplicativos da Web. O cache pode ser obtido de muitas camadas dentro da arquitetura de aplicação. É mais vantajoso quando uma camada de arquitetura pode evitar chamadas para a camada seguinte. A facilidade do ManageBean de JSF fez o cache mais simples para a camada de apresentação. Alterando o âmbito do ManageBean, os dados que eles contêm podem ser armazenados em diferentes áreas.

O aplicativo usará um cache de dois níveis. O primeiro nível de cache existe na camada de lógica de negócios. Onde no nível de Spring permanecerá a referência dos objetos que funcionam como Proxy para as invocações dos componentes do negócio, além disso, em alguns casos o sistema contará com algumas classes que implementam o padrão Value List Handler ou Cache Service. Portanto, o cache de primeiro nível é um cache de leitura/gravação no âmbito de aplicação.

No segundo nível, o cache terá (em um pool de instâncias) aqueles objetos que são compartilhados por diferentes componentes, tanto na apresentação quanto na de negócios, parâmetros de acesso, listas gerais, mensagens e serviços compartilhados parte de Framework Local.

4.8.5. Gestão de Exceções

Uma exceção representa uma situação que normalmente não ocorre e representa algo de estranho ou inesperado no sistema. Todas as exceções devem ser adequadamente devolvidas ao componente que chamou. Para o tratamento de exceções no Novo Sistema Tributário serão considerados dois tipos de exceções: de negócio e do sistema.

As **exceções do sistema** são aqueles erros não controlados, eles são diferentes aos *bugs* no código fonte. Eles podem ser: falhas de conexão no banco de dados, erros gerados pela invocação de componentes sem a informação certa, etc. Para isso, ter em conta o "stack trace" (a pilha de chamadas Java), onde se uma exceção não for tratada adequadamente, essa exceção será enviada pelo objeto que fez a chamada.

As exceções do sistema deverão ser identificadas como *Exception*, garantindo que não há pontos de fuga. A exceção identificada deverá encapsular-se em um objeto tipo *Exception*, a exceção genérica do Novo Sistema Tributário chamada *SystemException* e essa exceção deverá ser enviada para as camadas da frente, até o componente Web que iniciou o chamado. As exceções de tipo *RuntimeException* serão *SystemException.*

As **exceções de negócio** são todos os erros controlados que ocorrem por faltas ou violações de regras de negócio codificadas nos componentes. Por exemplo: superação dos limites máximos de transações, resultados errados de pesquisa, validações insatisfatórias de dados e outras regras de negócio especificadas no sistema.

Cada vez que ocorre um erro controlado, este deve ser identificado dentro de uma classe genérica chamada *BRException (Business Rule Exception*), e isso vai ser enviado para a camada da frente. *BRException* terá uma especialização chamada *BRFException* (*Business Rule Fatal Exception*), que encapsularão o erro para falhas que impeçam que o caso de uso tenha alguma possibilidade de concluir com sucesso.

As exceções *SystemException* o *BRException* vão ter opções para armazenamento de mensagens de erro específicos para apresentação ao usuário, ou para carga de um código de erro específico. Dependendo do código de erro, o gerenciador de exceções tomará as decisões sobre as medidas a fazer.

Todas as exceções chegam ao componente que faz a invocação, que deve tomar a decisão de mostrar a mensagem diretamente na página, quando o caso de uso é recuperável por um objeto com comportamento semelhante ao objeto *Messages de JSF* (que é obrigatório em todas as páginas), ou enviar a um componente da camada de apresentação. Esse componente deve mostrar uma página de erro ao usuário final com o código e uma mensagem pré-formatada. Esta funcionalidade será introduzida nos componentes de apresentação pelo *Framework* do sistema.

4.8.6. Gestão de Logs

Para monitorar os erros de código e depuração, não é permitido utilizar os métodos System.out.print() ou <ExceptionClass>.printStackTrace(). Não é permitido manter código no repositório de controle de versão (git ou svn) com tais métodos.

Para a monitoração dos erros será estabelecido um arquivo de log por cada aplicação, será usado o API Log4J (ou Logback ou SLF4J).

Quando uma exceção é identificada, deve ser cadastrada no Log como ERROR o como FATAL de acordo com a situação.

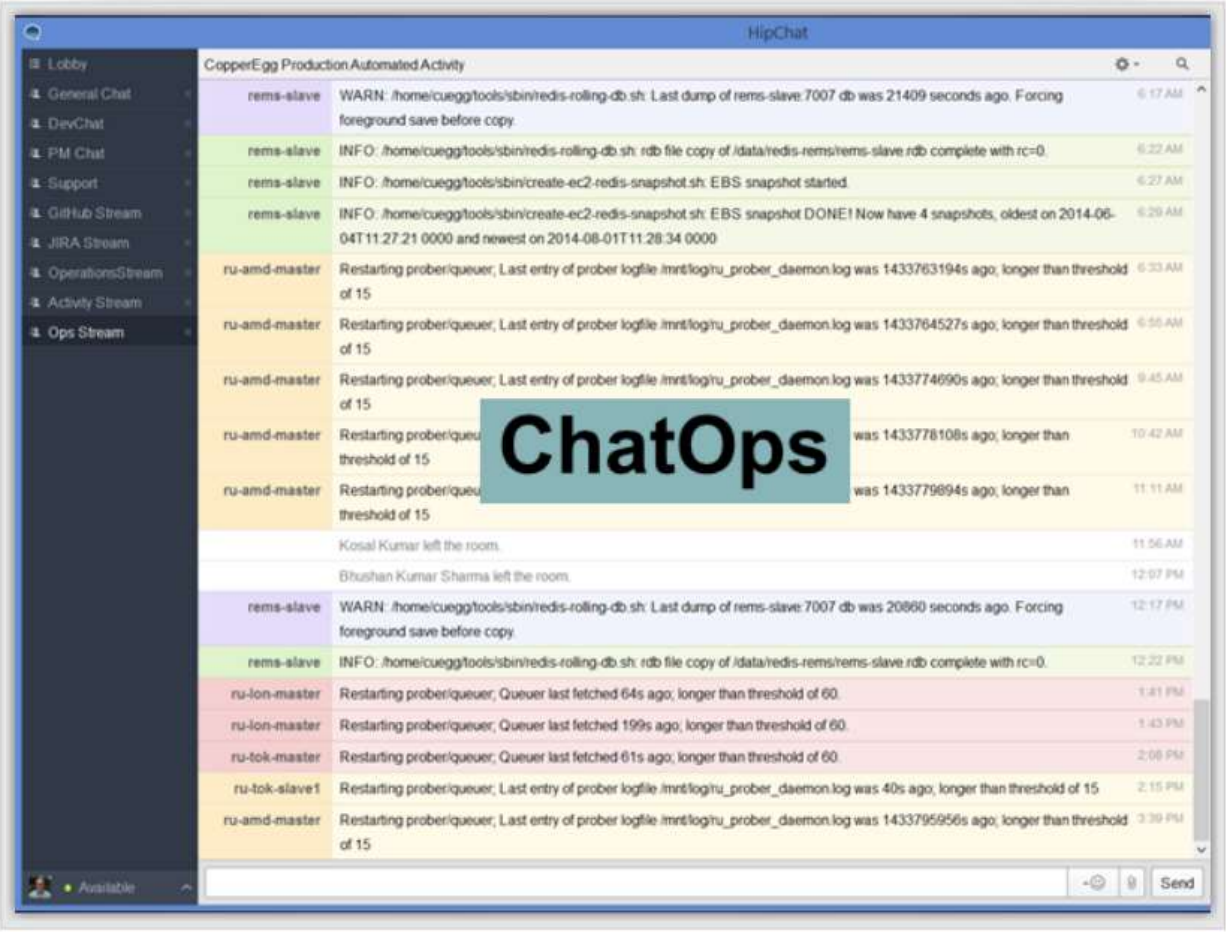
Configuração de Log

O sistema terá a capacidade de ativar e desativar determinados logs, enquanto outros não são afetados. Isto é feito com categorização das mensagens de logs. Serão utilizados os níveis fornecidos por Log4J:

NÍVEL	DESCRIPÇÃO
INFO	Usado para mensagens de operações feitas no sistema.
WARN	As mensagens de alerta sobre eventos que se deseja manter cadastrados, mas não afetam o bom funcionamento do sistema
DEBUG	Usado para escrever mensagens de depuração, este registro não deve ser ativado quando o aplicativo está em produção.
ERROR	Usado para mensagens de erro da aplicação, esses eventos afetam ao programa, mas permitem continuar o funcionamento.
FATAL	Usado para mensagens críticas no sistema.

As informações de log e monitoramento de processos vão permanecer em arquivos simples. Os arquivos gerados serão armazenados no banco de dados periodicamente.

Um exemplo é mostrado na seguinte imagem:



4.8.7. Gestão de Auditoria

Qualquer operação transacional, isto é, que altere os registros no banco de dados devem ser registrados no sistema. Este será um log de transações que também será gerenciado por Log4j em paralelo ao log de erros. Neste log será armazenada a informação da operação que foi realizada. Este processo é similar aos casos de uso em que é necessário um registro histórico de transações.

O processo de monitoramento da mudança de informações chave no sistema será limitado aos itens de parâmetro vital para o desempenho do sistema e serão incluídas as funcionalidades próprias de banco de dados. Como é determinado no documento de Serviços Transversais.

4.8.8. Camada de Integração de dados

A camada de integração do sistema com os dados será realizada usando Hibernate, o acesso aos componentes será feito contando com o Framework Spring. Serão utilizadas as opções de integração com mapeamento O/R em Hibernate. Spring fornece uma gestão segura e eficiente de sessões Hibernate e acesso às fontes de dados JDBC no contexto de aplicação, colaborando em processos de testes.

Hibernate é um framework de mapeamento O/R Open Source que evita a necessidade de usar a API JDBC diretamente no código. Hibernate suporta a maioria dos sistemas de banco de dados SQL. Para consultas complexas é usado Hibernate Query Language como ponte entre os mundos objeto e relacional.

Hibernate é menos invasivo que outros frameworks do mapeamento O/R. É usado Reflection e a geração de bytecode em tempo de execução, e geração do SQL ocorre no início da aplicação. Isto permite desenvolver objetos persistentes em conformidade com a linguagem comum Java: incluindo associação, herança, polimorfismo, composição e estrutura das coleções de Java.

4.8.9. Camada de Integração de Serviços SOA (Service Oriented Architecture)

Nesta camada devem ser consideradas todas as solicitações de dados incluindo acesso de dados de outras fontes (instituições), entrega de informação, volume, latência, transformação e qualidade de dados. Deve-se ter em conta consumir e publicar dados confiáveis no momento certo sem impacto para o sistema. Considerar o uso de SOA para intercâmbio de dados entre as instituições na fase inicial de desenvolvimento do projeto, depois, uma vez estabilizado o Novo Sistema Tributário em produção, pode-se considerar a possibilidade de estender os serviços para os contribuintes (aplicativos móveis).

Cada serviço deve ser responsável por seu próprio domínio e deve-se justificar claramente o motivo de sua criação (caso de uso).

A integração com os serviços transversais deve ser feita usando o API de cada serviço, evitar o uso de Web Services, já que não se deve incluir invocações assíncronas em aplicativos transacionais, no caso de chamadas assíncronas usar Java Message Service (JMS).

4.8.10. Servidor de Relatórios (Jasper Reports)[21]

Para a construção de relatórios pode-se usar Jasper Reports como servidor. É uma biblioteca de relatórios baseada em Java que usa qualquer fonte de dados para gerar documentos “pixel-perfect” que podem ser visualizados, impressos ou exportados em uma variedade de formatos, incluindo HTML, PDF, Excel, OpenOffice e Word.

A versão community tem as seguintes características: Report Designer, Report Viewing, Server Repository, Report Scheduling, Mobile BI.

[21] http://www.jaspersoft.com/features

4.9. Interface do Usuário

A interface do usuário vai estar em outro documento separado.

4.10. Arquitetura Móvel

Um framework muito usado nos últimos anos para desenvolvimento de aplicações web de uma só página é AngularJS (mantido pelo Google). Esse framework tem muito sucesso para aplicações moveis.

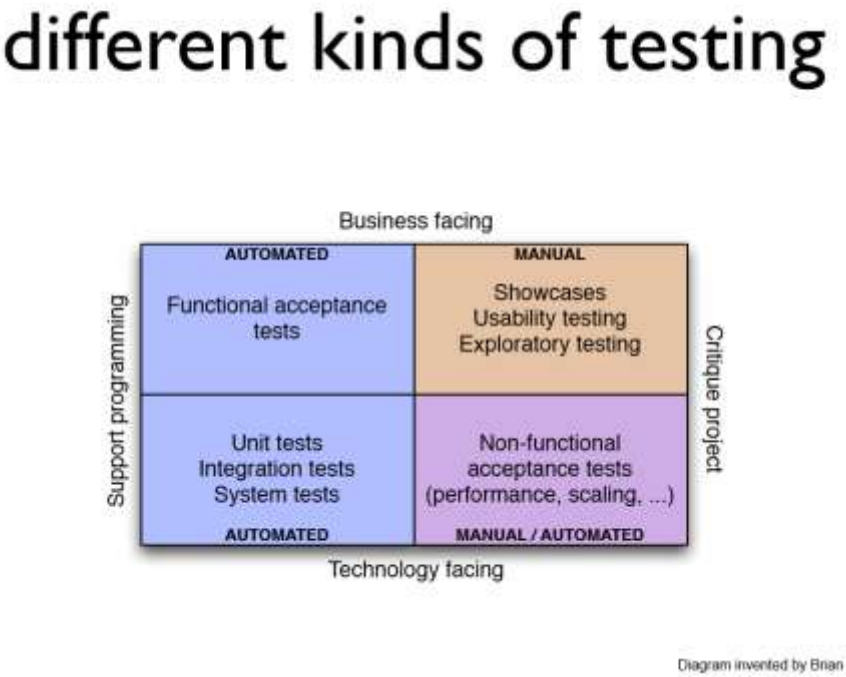
HTML é muito bom para declarar documentos estáticos, mas vacila quando tentamos usá-lo para declarar visualizações dinâmicas em aplicações web. AngularJS permite estender o vocabulário HTML para a sua aplicação. O ambiente resultante é extraordinariamente expressivo, legível e rápido para se desenvolver.

Pode-se usar recursos JSON para “data binding” na gestão de dados na camada cliente com AngularJS. E pode usar um RESTful API com Java no servidor de aplicações.

4.11. Teste do Sistema

Os testes do Novo Sistema Tributário permitem melhorar o funcionamento correto dos aplicativos. É importante o uso de ferramentas de teste para garantir melhor qualidade de software.

O processo de teste deve ser considerado a partir do início de desenvolvimento do Novo Sistema Tributário, e como uma boa prática (obrigatória) é necessário criar os testes unitários de todas as tarefas críticas dentro do sistema. É sugerido usar uma técnica Test Driven Development (TDD).



Sempre se deve validar os casos de teste, qual o objetivo e sua frequência. Os tipos de teste podem ser: unitário, de integração, de sistema, de aceitação, validação, segurança, recuperação, desempenho, estresse.

Aqui está uma lista de algumas ferramentas que podem ser usadas:

Ferramenta para Testes Unitários

JUnit: umas das ferramentas mais difundidas de testes unitários para aplicações Java, fornece uma série de classes que permitem chamar outras classes (que são as que devem ser testadas).

Ferramentas para Testes Funcionais Automatizados

Selenium: suíte de ferramentas para testes funcionais em aplicações WEB.

Badboy: grava todas as ações em uma página web como um macro e efetuar asserções por texto.

Jenkins: ferramenta de integração contínua, automatização de testes usando rotinas. Funciona como um servidor de integração contínua onde é possível programar ”jobs” para diversos projetos.

Ferramentas de Testes de Performance (desempenho/estresse)

JMeter: Originalmente foi desenvolvida para ser uma ferramenta para teste de Performance em aplicações WEB, mas que permite inúmeros tipos de testes.

Ferramentas para Teste de Integração

SoapUI: permite rapidamente criar e executar de forma automatizada testes de regressão funcional, conformidade e testes de carga.

Ferramentas de Gestão de Teste (Bugtracker)

MantisBT: Ferramenta de gestão de incidentes, prática e bastante robusta.

Bugzilla: Um completo servidor para ajudar o gerenciamento do desenvolvimento de software. Possibilita que indivíduos ou grupos de desenvolvedores e testers mantenham o controle de bugs em seus produtos.

Ferramentas para Teste de Segurança

Wireshark: é um programa que analisa o tráfego de rede e organiza por protocolos. Permite a avaliação de vulnerabilidades e ter uma imagem clara do que está acontecendo na rede.

NMap: é uma ferramenta que permite a exploração de rede e auditoria de segurança.

Burp Suite: é uma plataforma integrada para atacar aplicações web. Contém uma variedade de ferramentas com inúmeras interfaces concebidas para facilitar e acelerar o processo de atacar uma aplicação.

HP Fortify Software Security Center: Um conjunto de soluções integradas de segurança de software para automatizar os principais processos de desenvolvimento e implantação de aplicações seguras. Ajuda a resolver vulnerabilidades de software em todo o ciclo completo de vida de desenvolvimento de segurança (SDLC- Security Development Life Cycle).

4.12. Implantação do Sistema em Teste, Pré-produção e Produção

É recomendado criar ambientes de desenvolvimento, teste, pré-produção e produção, onde os ambientes de pré-produção e produção têm maior controle e segurança.

Os consultores do CIAT devem executar os testes no ambiente de teste, enquanto que os funcionários SEFAZ-TO devem testar no ambiente de pré-produção com dados mascarados.

O aplicativo será instalado como um único Enterprise Archive (*.ear), será dividido em dois componentes: a) um Web Archive (*.war) SEFAZ WebApp com os arquivos de componentes de apresentação e outros componentes de negócios, e b) (*.jar) para cada módulo desenvolvido.

Para o deploy dos aplicativos tem que fazer uma lista de cada um dos componentes envolvidos no Novo Sistema Tributário, um exemplo é mostrado na seguinte tabela:

Logical Application Component	Physical Technology Component	Server Address	IP Address
Cadastro de Contribuintes - CCI	Web server – node 1	websvr001@sefaz.to.br	10.9.7.XXX
	Web server – node 2	websvr002@sefaz.to.br	10.9.7.XXX
	App server – node 1	appsvr001@sefaz.to.br	10.9.7.XXX
	App server – node 2	appsvr002@sefaz.to.br	10.9.7.XXX
	Database server (production)	dbsvr001@sefaz.to.br	10.9.8.XXX

	Database server (staging)	dbsvr002@sefaz.to.br	10.9.8.XXX
Load balancer and dispatcher	Dispatcher server	dptdvc001@sefaz.to.br	200.252.237.XXX

5. Arquitetura de Dados

Hoje, o Banco de Dados na SEFAZ-TO é Oracle Database 11gr2. É importante sempre ter a última versão instalada. Então, a sugestão é usar Oracle 12cr1 com todos os patches para o Novo Sistema Tributário.

5.1. Governança de Dados em Oracle

É uma especificação de direitos de decisão e um framework de responsabilização para incentivar o comportamento desejável na avaliação, criação, armazenamento, uso, arquivamento e eliminação de dados e informações. Ela inclui os processos, roles, padrão, normas e métricas que garantam a utilização eficaz e eficiente dos dados e informações para permitir atingir os objetivos de uma organização.

Os ´principais objetivos da Governança de Dados são:

- Definir, aprovar e comunicar as estratégias de dados, políticas, padrões, procedimentos, arquitetura e métricas.
- Acompanhar e fazer cumprir a conformidade com políticas de dados, padrões, arquitetura e procedimentos.
- Patrocinar, controlar e supervisionar a entrega de projetos e serviços de gerenciamento de dados.
- Gerir e resolver os problemas relacionados com dados.
- Compreender e promover o valor dos dados como ativos.

Para uma completa definição de boas práticas de Governança de dados em Oracle, confira no link a seguir: http://www.oracle.com/technetwork/articles/entarch/oea-best-practices-data-gov-400760.pdf

5.2. Ferramentas Oracle para Migração e Integração de Dados

Muitas vezes precisa-se migrar dados entre diferentes bancos de dados, incluindo em ambientes homogêneos (o mesmo motor de banco de dados e mesmo sistema operacional) ou em ambientes heterogêneos (diferentes motores de bancos de dados e/ou diferentes sistemas operacionais). Para distintos usos como: migração de plataforma, o intercâmbio de informações com outros sistemas, cruzamento de dados, análise de informação, criação de novas aplicações ou construção de DataWareHouse.

Oracle tem várias ferramentas para essas tarefas, detalhados a seguir

5.2.1. Oracle Data Integrator

É uma plataforma de integração de dados completa que cobre todas as necessidades de integração: carga de volume elevado de dados, carga de dados em lotes de alto desempenho, dados por eventos, processos de integração contínua de dados, serviços de dados através de SOA. Oracle Data Integrator (ODI) 12c é a última versão que fornece Oracle para sua estratégia de integração de dados proporcionando produtividade superior de desenvolvimento e melhorando a experiência de usuário em

redesenhar interfaces declarativas de usuário em base à fluxos e também profunda integração com Golden Gate.

ODI 12c está construído com uma arquitetura de alto desempenho e flexível com paralelismo adicionado quando são executados os processos de integração de dados. Inclui interoperabilidade com Oracle Datawarehouse Builder (OWB) para uma migração rápida e simples. Adicionalmente, ODI pode ser monitorado com uma solução simples desde Oracle Enterprise Manager 12c.

5.2.2. Oracle GoldenGate

É um pacote de software completo para integração de dados em tempo real e replicação em ambiente heterogêneo de tecnologia de informação. O produto permite soluções de alta disponibilidade, para integração de dados em tempo real, roteamento e entrega de transações de banco de dados entre sistemas heterogêneos, replicação de dados, transformações e verificação entre sistemas empresariais operacionais e analíticos.

Oracle GoldenGate (OGG) 12c fornece desempenho extremo com configuração e gestão simplificada, a forte integração com Oracle Database, suporta ambiente na nuvem, heterogeneidade estendida e segurança reforçada.

Adicional a plataforma essencial de OGG para movimento de dados em tempo real, Oracle fornece um pacote de gestão de OGG chamado Oracle GoldenGate Veridata, que é uma ferramenta para monitoramento e gestão visualmente de implantações OGG. Esta ferramenta permite comparações de alto volume e alta velocidade entre dois bancos de dados em uso.

5.2.3. Oracle Enterprise Data Quality

Fornece um ambiente de gestão completo para qualidade de dados usado para entender, melhorar, proteger e governar a qualidade dos dados. O software permite as melhores práticas de Master Data Management, Integração de Dados, Inteligência de Negócio e iniciativas de migração de dados.

Desde que existem aplicações e bancos de dados, tem havido problemas de qualidade dos dados. Infelizmente, todos esses problemas não são iguais nem as soluções para estes são as mesmas. Algumas das maiores diferenças são devido a tipos de dados, ou domínios, ou os mesmos dados em questão. Os domínios de dados mais comuns em qualidade são clientes e produtos. Buscando uma similaridade com SEFAZ-TO seria contribuinte e impostos (obrigações e declarações).

Rápido de implementar e fácil de usar, os produtos de Oracle Enterprise Data Quality traz a possibilidade de melhorar a qualidade dos dados em qualquer iniciativa de gerenciamento de dados. Os produtos de Oracle Enterprise Data Quality cobrem:

- Perfilhamento, Auditoria e Painéis de Controle
- Análise de Frases e Padronização
- União e Fusão
- Gestão de Casos

- Verificação de Endereço

5.2.4. Oracle Master Data Management

Master Data é a informação crítica de negócio para apoiar as operações transacionais e analíticas da empresa. Master Data Management (MDM) é uma combinação de aplicações e tecnologias que consolida, limpa e aumenta esses dados mestres corporativos, e sincroniza-lo com todos os aplicativos, processos de negócios e ferramentas analíticas. Isso resulta em melhorias significativas na eficiência operacional, geração de relatórios e baseada em fatos tomada de decisão.

5.3. Oracle Maximum Availability Architecture

Há muitas soluções de Alta Disponibilidade no mercado com opções como replicação, cópias de imagens do sistema operacional, replicação de storage, snapshots de servidores dedicados/virtuais, cloud computing, etc. Muitos com alta complexidade administrativa, custos elevados e ao final com tempos altos de recuperação.

Com uma solução de Alta Disponibilidade (HA - High Availability) pretende-se, em resumo:

- Minimizar o risco de falha de hardware e software.
- Minimizar o tempo de inatividade planejado (atualizações).
- Reduzir ou evitar tempos de inatividade não planejados nas aplicações.
- Habilitar recuperação rápida contra falhas.
- Proteção de perda de dados (incluindo o nível transacional)

A implementação bem-sucedida de Alta Disponibilidade depende dos níveis de serviço, que SEFAZ-TO pode desejar implementar e que sejam facilmente adaptáveis (customizáveis) com cenários reais de falhas potenciais. Considerando-se que quanto mais você quer para aumentar a disponibilidade de 100%, quanto maior for o investimento económico que tem de ser realizado em hardware, software e recursos humanos.

5.3.1. Benefícios de uma arquitetura de Alta Disponibilidade

Disponibilidade de dados: Garantir o acesso aos dados por parte dos contribuintes e funcionários para evitar interrupções de negócios. Especialmente nos dias de vencimento e de apresentação de declarações e pagamentos. Oferecendo um serviço de alta qualidade para os contribuintes e agentes arrecadadores (bancos).

Proteção de dados: Evitar perda de dados que compromete a viabilidade institucional, já que a informação dos contribuintes é essencial para o cálculo das suas obrigações fiscais e determinação do imposto a liquidar.

Desempenho: fornece tempos de resposta adequados para todas as operações realizadas no Novo Sistema Tributário para ambos, funcionários e contribuintes, assim como o intercâmbio de informações com outras instituições públicas e privadas.

Custo: Reduzir os custos de implantação, gestão e suporte para reter os recursos institucionais, tendo em conta tanto os recursos de hardware, software e profissionais encarregados das operações.

Risco: atender constantemente os níveis de serviço exigidos pelos contribuintes no Novo Sistema Tributário que funciona 24/7, assim como a evolução constante das aplicações e novos serviços para ser implementados no futuro.

5.3.2. O que é Oracle Maximum Availability Architecture (MAA)?

Oracle Corporation é uma empresa que vem inovando mais seu motor de banco de dados a cada versão/release que lança ao mercado, a última versão tem novas funcionalidades para a satisfação dos clientes mais exigentes. Sendo a SEFAZ-TO um cliente de Oracle, pode implementar essas funcionalidades disponíveis de acordo com os produtos que já tem licenciados e com suporte renovado 2015-2016.

A seguir detalham-se as novas características que dispõe a versão Oracle Database 12c release 1 (12.1.0.2.0) em quanto à Alta Disponibilidade:

Real Application Cluster (RAC)

Desde seu lançamento na versão Oracle Database 10g, tem sido um dos estandartes de Oracle quanto a Alta Disponibilidade. Oracle RAC permite que múltiplos servidores (mínimo 2) independentes, mas interconectados executem o software do gerenciador do banco de dados de maneira simultânea enquanto acessa a um banco de dados individual, isto é um grupo de servidores (cluster) acessando a um só banco de dados (localizado num armazenamento compartilhado) recebendo conexões (sessões) concorrentemente.

Este é um método de proteção para os bancos de dados Oracle em caso de falha de um dos servidores, mas também de balanceamento de carga já que distribui as sessões entre os servidores (nodos) disponíveis.

Esta tecnologia permite escalabilidade sob demanda, se precisamos mais poder computacional para as operações do banco de dados simplesmente agregamos outro servidor a nosso cluster (uma nova instância).

Oracle Database 12cr1 suporta a aplicação de patches (atualizações ou patches críticos) para os nodos de um sistema Oracle RAC de modo gradual, mantendo o banco de dados disponível em todo o processo de atualização. Para executar a atualização sem interrupção, uma das instâncias está imobilizada e parcheando-se, enquanto as outras instâncias no grupo de servidores continuam brindando o serviço. Este processo repete-se até que todas as instâncias estejam parcheadas. O método de atualização gradual pode-se utilizar para atualizações de Patch Set Updates (PSU), Critical Patch Updates (CPU), one-

off database e os patches de diagnóstico usando OPatch, atualizações do sistema operacional e as atualizações de hardware.

Data Guard

Não existe ferramenta de software física ou lógica no mercado que permita replicar o banco de dados Oracle como Data Guard. A tecnologia Data Guard permite ter uma cópia (standby) do banco de dados primário, podendo em caso de falhas na base primária que o banco de dados standby receba todas as petições em seu lugar, isso é feito através de um failover (redução permanente da base primária sem recuperação) ou switchover (substituição temporária do banco de dados secundário pelo primário, para recuperar depois o banco de dados primário).

Entre as novas funcionalidades de Data Guard na versão Oracle Database 12c, temos:

Fast Sync: sincronização de redo através da memória, não precisa escrever no disco. Isto é, todas as transações têm o acknowledge em memória, economizando tempo de I/O do redo. Assim reduzindo o tempo de sincronização. Esse mecanismo permite uma base em standby reconhecer o banco de dados primário tão rápido como recebe o arquivo redo em memória, sem esperar por entrada/saída do disco para um arquivo de log redo de standby.

Far Sync: oferece proteção de zero perda de dados para um banco de dados ao manter um banco de dados em standby sincronizado localizado a qualquer distância no planeta a partir da localização principal com mínimo custo ou complexidade.

Automatic Block Repair: a perda de dados a nível de bloco usualmente pode ser o resultado de erros intermitentes nos dispositivos de entrada/saída, assim como as corrupções de memória que são gravadas no disco. Quando Oracle Database lê um bloco e detecta que está corrupto, marca o bloco como corrupto e adverte o erro para o aplicativo. Nenhuma leitura posterior do bloco será bem-sucedida até que o bloco seja recuperado manualmente, a menos que Active Data Guard é utilizado. Com Active Data Guard, a recuperação de blocos de dados é executada automaticamente e de forma transparente.

Atualizações com Active Data Guard: substitui dezenas de passos necessários para uma atualização de banco de dados com 3 pacotes PL/SQL que automatizam muito o processo. Minimizando o tempo planejado de inatividade dos aplicativos e risco para implantar totalmente validando todas as alterações em uma réplica de produção antes de mover aos usuários para a nova versão.

Reportes ad-hoc (read only): adicional permite-nos acessar no modo de leitura as informações no banco de dados e pode-se executar relatórios removendo a carga de trabalho para o banco de dados primário.

Recovery Manager (RMAN)

Esta é a ferramenta estrela de Oracle para obter backups, restauração e recuperação de banco de dados que vão desde a recuperação completa do banco de dados, a recuperação de tablespaces/datafiles e na versão 12cr1 recupera tabelas e até blocos de dados corruptos.

Para garantir que se pode-se fazer uma recuperação do banco de dados através dos logs de transações gerados pelo banco de dados, deve-se ativar a geração de logs de arquivo (registros históricos). Esta ação deve ser realizada fora do horário de trabalho de SEFAZ-TO com os usuários do sistema desligados.

Nesta arquitetura da SEFAZ-TO onde se tem alguns bancos de dados (desenvolvimento, testes, pré-produção, produção) é aconselhável usar Recovery Manager com um catálogo de banco de dados externos. Um catálogo é simplesmente um repositório (um banco de dados) onde são registradas todas as atividades de RMAN (backups históricos, os detalhes dos backups, datas/horários, detalhes de recuperação, relatórios, etc.). Assim podemos ter disponível o histórico de backups e recuperações de cada banco de dados Oracle na SEFAZ-TO.

A seguir listam-se as novas características desta ferramenta.

- Na arquitetura multitenant (veremos mais adiante) os comandos de backup/recuperação do banco de dados agora também respaldam/recuperam o contêiner multitenant do banco de dados (CDB), incluindo todas as pluggable databases (PDB).
- Os comandos de RMAN também podem aplicar a PDB individuais.
- RMAN respalda/recupera através de diferentes plataformas para eficiência da migração de banco de dados e tablespaces.
- Pode recuperar a versão mais recente ou a mais antiga das tabelas individuais.
- As tabelas podem ser recuperadas no mesmo tablespace ou em outro diferente.
- Podem-se executar backups multi-seções de copias tipo imagem e backups incrementais.
- Rápida sincronização de um banco de dados standby com um banco de dados primário.
- Suporte direto para instruções SQL por linha de comandos RMAN (não precisa a palavra SQL).

Flex Automatic Storage Management (ASM)

Ideal para instalações em Real Application Clusters com alta demanda transacional, pois aumenta a disponibilidade do banco de dados (instância), facilitando a consolidação do cluster de banco de dados, permitindo a comutação de armazenamento entre-nodos e reduzindo o consumo de recursos relacionados com ASM até 60%.

Esta tecnologia usa “nodos hub” de RAC para comunicar-se diretamente com a instância de ASM (e entre eles). Permitindo ligar “nodos leaf” diretamente ao “nodo hub”, estes nodos leaf (servidores de aplicativos) não se ligam ao ASM, mas todas as operações são sincronizadas através dos nodos hub diretamente, como se mostra na seguinte imagem:

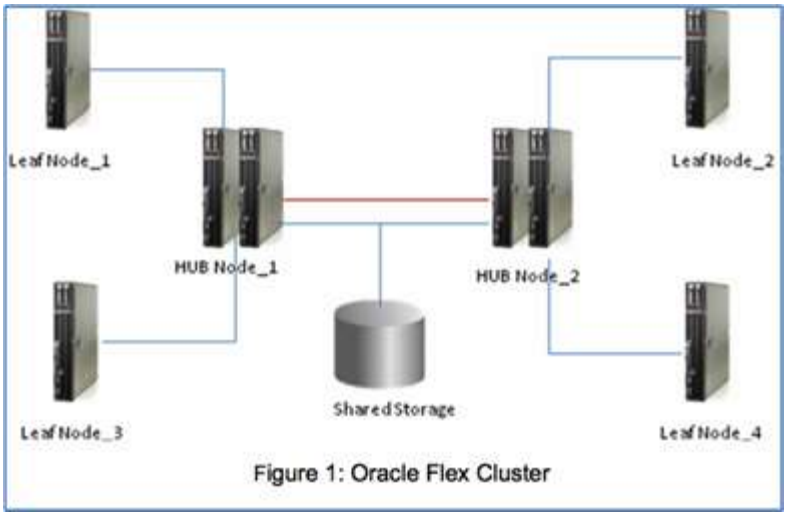


Figure 1: Oracle Flex Cluster

ASM Disk Scrubbing

Esta é uma nova característica de Oracle Database 12c que verifica corrupções lógicas de dados e repara automaticamente em grupos de discos normais e grupos de discos de alta redundância. Esta funcionalidade foi concebida para que não tenha impacto nas operações do dia-a-dia em sistemas de produção. O processo de scrubbing (depuração) repara corrupções lógicas usando os discos mirror (espelhos) do RAID de ASM.

Esta característica complementa a revisão de saúde que executa Recovery Manager (RMAN) durante o backup e recuperação.

Data Recovery Advisor (DRA)

Está é uma característica de Oracle Database 11g (melhorada em Oracle Database 12c) que permite receber e executar recomendação sobre cenários de falhas através de Recovery Manager. Deve-se instalar este complemento adicional. Suas funcionalidades são: validar o banco de dados, lista as falhas, aconselhar sobre as possíveis soluções e recuperar automaticamente a partir das recomendações.

Esta é uma ferramenta ideal que se complementa com Recovery Manager, também podem ser feitas reparações com Oracle Enterprise Manager.

Flashback Technology

Erros humanos ocorrem geralmente quando se administra bancos de dados. A tecnologia de Oracle Database Flashback é única e com um conjunto de várias soluções de recuperação de dados que permitem reverter erros humanos corrigindo os efeitos do erro.

Com Flashback o tempo necessário para recuperação de um erro depende do trabalho feito desde que o erro ocorreu até o momento de tentar recuperar os dados perdidos. O tempo de recuperação não depende do tamanho do banco de dados. Flashback é fácil de usar e suporta diferentes níveis de recuperação que inclui linhas, operações, tabelas e até todo o banco de dados inteiro.

Tecnologia	Descrição
Flashback Query	Permite consultar qualquer dado em algum momento do tempo no passado. Para ver e reconstruir dados mudados inadvertidamente.
Flashback Versions Query	Permite obter as diferentes versões de uma linha de dados numa tabela num intervalo de tempo.
Flashback Transaction Query	Permite consultar as alterações feitas a uma transação específica. Também produz as sentenças para reverter a transação (undo).
Flashback Transaction	Permite consultar todas as mudanças feitas inclusive sobre transações com dados maus e modificados posteriormente, se usa PL/SQL.
Flashback Database	Permite restaurar um banco de dados inteiro em um ponto anterior no tempo. É rápido porque somente restaura os blocos alterados.
Flashback Table	Permite recuperar as tabelas afetadas a um momento específico.
Flashback Drop	Recupera facilmente tabelas que foram dropeadas, junto com os índices, constraints e os triggers da tabela.

Multitenant

Essa funcionalidade de Oracle Database Enterprise Edition, permite a consolidação simplificada dos bancos de dados, essa consolidação não requer alterações nos aplicativos.

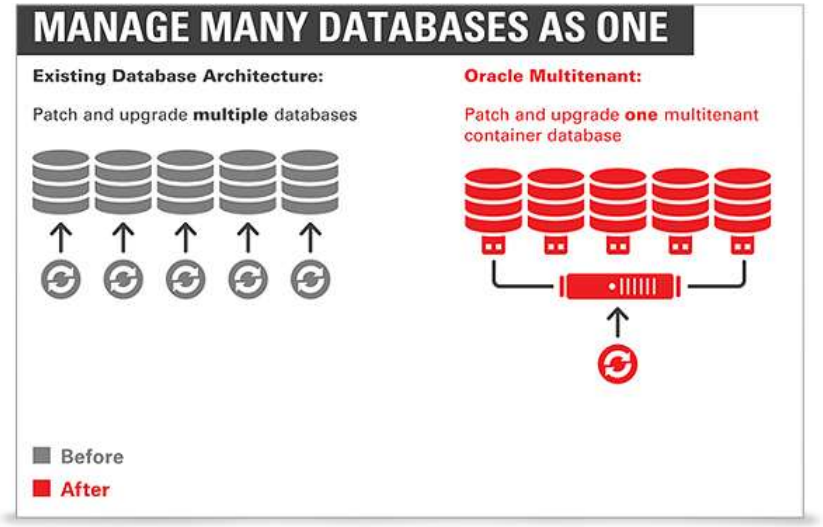
Nesta arquitetura, um banco de dados “contêiner“ multitenant (Container Database - CBD) pode acomodar muitos bancos de dados, esses bancos de dados se chamaram “pluggable database”. Assim que um administrador de banco de dados (DBA) só trabalha com o banco de dados contêiner multitenant, mais os aplicativos estão ligados somente aos bancos de dados “pluggable”.

Características do multitenant:

- Simplificação, consolidação e aprovisionamento de banco de dados.
- Consolidação simplificada dos bancos de dados com maior densidade de bancos por servidor.
- Eficácia em provisão (clonar e mover), patches e atualização de banco de dados.
- Arquitetura multitenant no nível de dados em vez de nível de aplicação para cumprir os requisitos da nuvem (Software-as-a-Service - SaaS).
- Complementa opções de Oracle Database como Real Application Cluster e Oracle Active Data Guard.

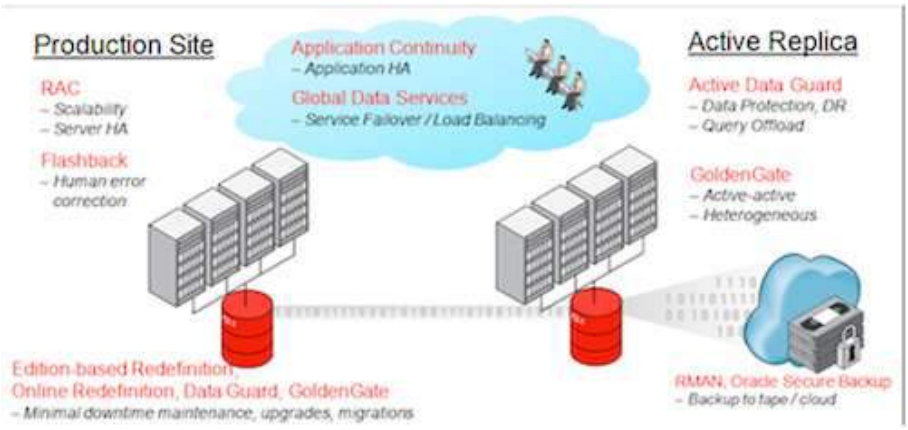
Entre as vantagens:

- Aumenta a utilização dos servidores e a escalabilidade.
- Gerencia muitos bancos de dados como se fosse um.
- Backup todos os bancos de dados como se fosse um (contêiner).
- Atende às expectativas de nível de serviço de uma gestão integrada dos recursos de carga de trabalho.
- Mantém o isolamento dos bancos de dados independentes sem alterar os aplicativos ou direitos de acesso (não afeta a segurança).



Application Continuity

Protege as aplicações das falhas nas sessões de banco de dados devido a cortes causados na instância, servidor, armazenamento, rede ou qualquer outro componente relacionado. Application Continuity rebobina os requerimentos afetados “em-execução” de modo que a falha de um nodo em RAC é mostrada para a aplicação como uma execução ligeiramente tardia. É usado para melhorar a experiência de usuário, apresentando uma comutação devido a um erro de uma forma transparente para o usuário.



Global Data Services (GDS)

É uma solução completa de gestão de carga de trabalho automatizada para os bancos de dados replicados (usando as tecnologias de replicação como Active Data Guard e Oracle Golden Gate). Expande os serviços de banco de dados ao estender-se sobre várias instâncias de banco de dados em locais próximos ou distantes. GDS estende a comutação por erros tipo RAC, gestão de serviços, balanceamento de carga entre vários bancos de dados e configuração de replicação de banco de dados. GDS está destinado para as aplicações que estão orientadas para uso de replicação.

Oracle Secure Backup (OSB)

Solução de Oracle para proteção de dados em Tape ou em armazenamento na nuvem. Centraliza o gerenciamento de cassetes de backups, oferece alto desempenho e proteção de dados heterogêneos distribuídos em ambientes Unix, Linux, Windows, NAS (Network Attached Storage). Rápido desempenho em ambientes NUMA (Non-Uniform Memory Access). Aumenta as taxas de transferência de dados sobre InfiniBand (IB) usando RDS/RDMA em vez de TCP/IP. Melhora a utilização da rede para equilibrar a carga entre as interfaces de rede. Integração com Recovery Manager.

Online Move Functionality

Movimento de dados on-line (Online Data Move) permite mover data files, enquanto os usuários estão acessando seus dados, ou seja, quando as sessões dos usuários estão on-line. Movimento de partições on-line (Online Partition Move) suporta redefinição online de multi-partições numa sessão simples. Com esta funcionalidade não é necessário ter janelas de manutenção para mover arquivos em diferentes file system.

Online Table Redefinition enhancements

Nesta versão de Oracle Database 12c melhorou a redefinição on-line de tabelas oferecendo aos DBA uma flexibilidade significativa, não apenas para modificar os atributos físicos de uma tabela, mas também desfragmentar e compactar a tabela enquanto está acessando-se o banco de dados on-line.

Com esta funcionalidade, pode-se recuperar espaço perdido na tabela desde um ponto de vista de armazenamento e desempenho.

Esta tarefa é feita com um simples comando de redefinição. Melhorado o alto desempenho (sunc_interim_table), maior flexibilidade (finish_redef_table), com uma melhor gestão de bloqueios, uma melhor disponibilidade de redefinição de partições com um só bloqueio ao nível das partições e melhor desempenho para gravar logs de alterações para partições específicas.

Hot Patching

A aplicação de patches online com o aplicativo opatch fornece a capacidade de modificar os processos em uma instância de Oracle sem ter que desligar a instância. Cada processo associado à instância verifica a existência de código parcheado num ponto de execução seguro e depois copia o código em seu espaço de processo. Com esta funcionalidade pode ter-se instalado o patch mais recente disponível do software de Oracle Database sem ter que usar as janelas de manutenção.

5.4. Classificação (Níveis) de Arquitetura Oracle MAA

Oracle fez uma classificação em relação aos níveis de alta disponibilidade que se pode obter ao utilizar as diferentes ferramentas mencionadas na seção anterior, de modo que Maximum Availability Architecture pode-se classificar para satisfação dos clientes e neste caso particular para na SEFAZ-TO como mostra-se na tabela a seguir:

HIGH AVAILABILITY AND DATA PROTECTION				
Outage class/ HA tier	Unplanned Outages (local site)	Planned Maintenance	Data Protection	Unrecoverable local outages and disaster recovery
Platinum	Zero outage for platinum ready applications	Zero application outage	Comprehensive runtime validation combined with manual checks	Zero application outage for platinum-ready applications, zero data loss
Gold	Comprehensive HA/DR	All Rolling or online	Comprehensive runtime validation combined with manual checks	Real-time failover, zero or near-zero data loss
Silver	HA with automatic failover	Some rolling, some online and some offline	Basic runtime validation combined with manual checks	Restore from backup, potential to lose data generated from last backup

Bronze	Single instance with autorestart for recoverable instance and server failures	Some online, most offline	Basic runtime validation combined with manual checks	Restore from backup, potential to lose data generated since last backup

Tendo assim que:

Platinum: está baseado em servidores em clusters e replicação. O benefício é zero perda de dados e zero interrupções.

Gold: está baseado somente em servidores em clusters. Zero ou próximo de zero perda de dados com alta disponibilidade e proteção contra desastres.

Silver: está baseado em servidores em clusters, a recuperação é usando backups manualmente. Permite a alta disponibilidade para recuperação de interrupções locais, os dados estão protegidos com o último backup.

Bronze: está baseado num único servidor, a recuperação é usando cópias de backups manualmente. Pode ter dados perdidos.

Para uma avaliação do nível de Alta Disponibilidade na SEFAZ-TO pode-se usar a ferramenta web de Oracle Assesment Tool https://maa.oracle-dashboard.com/

ORACLE

Maximum Availability Architecture (MAA) Assessment

When your databases and applications are down, business can grind to a halt. Things go from bad to worse if your data is lost in the process. And a database that can't process your transactions fast enough can be almost as bad as an outage. These scenarios increase your costs while you lose revenue and damage your customer's confidence in your business.

Take a few minutes to answer important questions about your service-level expectations, database architecture, and planned maintenance practices. You can quickly assess your organization's level of risk exposure, and learn how Oracle Maximum Availability Architecture can help protect your business and grow your ROI.

Let's get started! Begin test

The Maximum Availability Architecture covers the following high-availability areas:

- Backup and Recovery
- Database Servers
- Recovery from Human Error
- Database Storage
- Site Redundancy
- Planned Maintenance

(Answering "no" to a question in the survey results in a lower score—and a greater reason to speak to Oracle.)

Hardware and Software, Engineered to Work Together

Home | About Oracle | Terms and Conditions | Your Privacy Rights

Anexo I: Glossário de Termos Técnicos

API: (Application Programming Interface - Interface de Programação de Aplicativos) é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços. De modo geral, a API é composta por uma série de funções acessíveis somente por programação, e que permitem utilizar características do software menos evidentes ao utilizador tradicional.

APM: Application Performance Management. É o monitoramento e gerenciamento de desempenho e disponibilidade de aplicações de software. APM se esforça para detectar e diagnosticar problemas de desempenho de aplicativos para manter um nível de serviço esperado.

Applet: é um pequeno software que executa uma atividade específica, dentro (do contexto) de outro programa maior (como por exemplo uma web browser), geralmente como um Plugin.

Business Delegate: Um objeto que reside na camada de apresentação e em benefício dos outros componentes da camada de apresentação chama os métodos remotos nos objetos na camada de negócios.

Composite View: Um objeto vista está composto por outros objetos vista. Por exemplo, uma página JSP que inclui outras páginas JSP e HTML usando a diretiva include é um padrão Composite View.

Data Transfer Object: Também chamado Transfer Object ou Value Object. Um objeto serializável para a transferência de dados na rede.

Deployment: Processo de instalação e configuração de um grupo de componentes empacotados junto com um arquivo de configuração XML em um servidor de aplicações que suporta JEE.

DevOps: DevOps (amálgama de Desenvolvedor e Operações - Development & Operations) é uma metodologia de desenvolvimento de software que explora a comunicação, colaboração e integração entre desenvolvedores de software e profissionais de TI. DevOps é a reação à interdependência entre desenvolvimento de software e operações de TI.

EJB Container: Ambiente responsável de gestar os componentes de tipo EJB, baseado na configuração dos mesmos, controle de segurança e ciclo de vida de EJB.

EAR file: Um pacote que inclui uma aplicação JEE consiste em um ou vários arquivos da com extensão *.war ou EJB JAR. Tem extensão *.ear.

EJB JAR file: Um arquivo JAR que contém um módulo EJB.

Facade: O padrão Facade serve para fornecer uma interface unificada simples que faz de intermediário entre um cliente e uma interface ou grupo de interfaces complexas.

Framework: é um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação.

Java Server Pages: A tecnologia Java Server Pages (JSP) permite que designers e desenvolvedores criar sites (páginas da Web dinâmicas) de forma rápida e manutenção fácil.

JDBC: é um conjunto de classes e interfaces (API) escritas em Java que fazem o envio de instruções SQL para qualquer banco de dados relacional; api de baixo nível e base para api's de alto nível; amplia o que você pode fazer com Java; possibilita o uso de bancos de dados já instalados; para cada banco de dados há um driver JDBC.

JEE: Java Enterprise Edition.

JSE: Java Standard Edition.

NOC/SOC: Network Operation Center/Security Operation Center.

Session Façade: Usando um bean de sessão como uma fachada (facade) para encapsular a complexidade das interações entre objetos de negócios e participantes em um fluxo de trabalho. Session Façade lida com objetos de negócios e fornece um serviço de acesso uniforme aos clientes.

Value List Handler: É um objeto que gerencia a execução de consultas SQL, caching e processamento do resultado. Normalmente implementados como beans de sessão.

Value Object Assembler: Um objeto que reside na camada de negócios e cria Value Objects quando necessário.

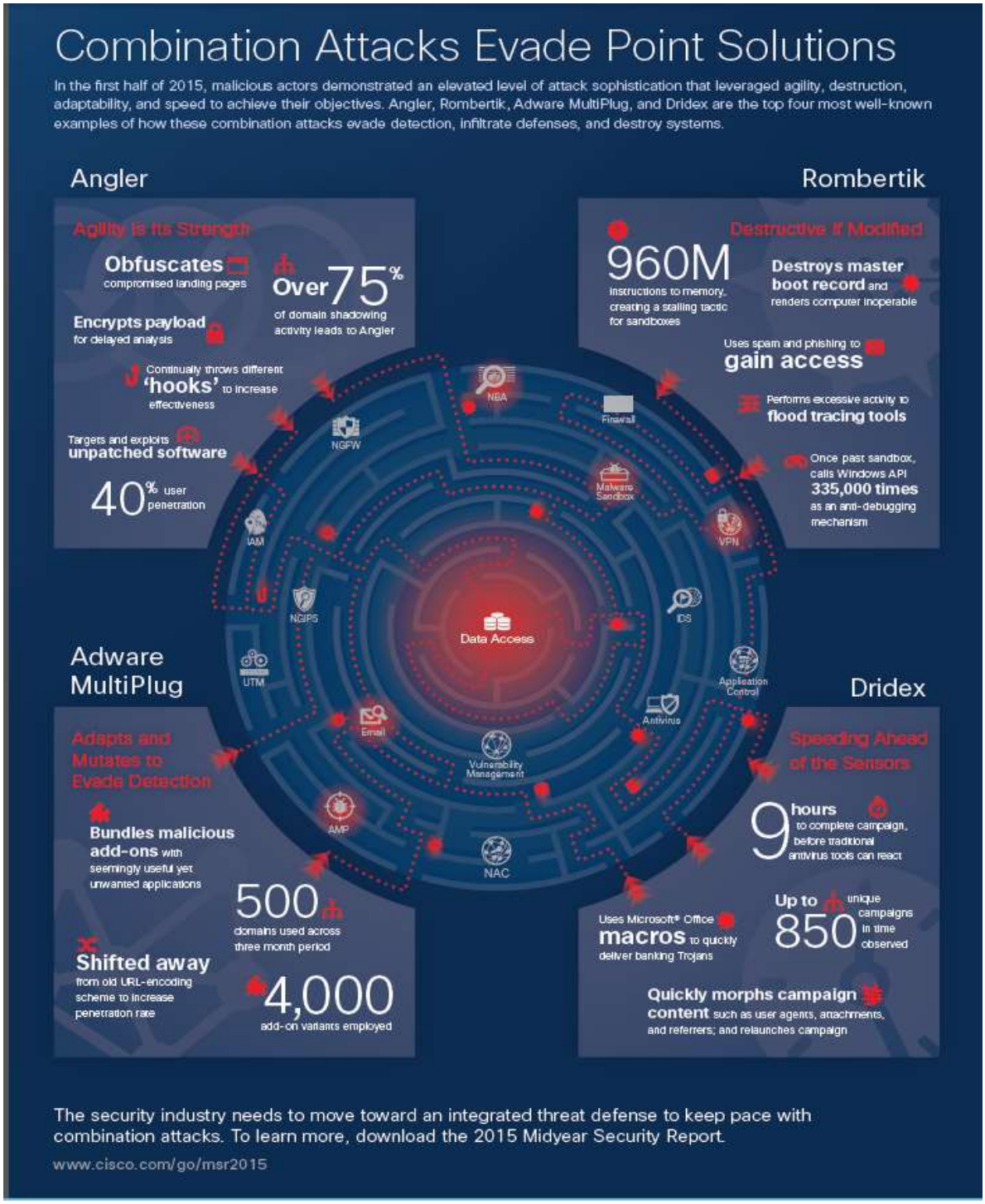
View Helper: Um objeto helper que encapsula a lógica de acesso a dados em benefício dos componentes de apresentação.

WAR file: É um arquivo JAR que contenha módulos Web, com extensão *.war

Web Container: Ambiente responsável por gerenciar componentes tipo Web. É o componente de um servidor web que interage com servlets Java. Uma recipiente web é responsável por gerenciar o ciclo de vida de servlets, mapear uma URL para um servlet particular e garantir que o requisitante da URL possua os direitos de acesso corretos.

XML (eXtensible Markup Language): Uma linguagem de marcação que permite definir os tags (markup) necessários para definir os dados e os textos em um documento XML.

Anexo II: Infographics of Sophisticated Attacks (First Half 2015)



Referencias

Design Patterns: Elements of Reusable Object-Oriented Software by Erich Gamma and Richard Helm. Nov 1994.

Refactoring: Improving the Design of Existing Code by Martin Fowler. Jul 1999.

Patterns of Enterprise Application Arquitecture by Martin Fowler. Nov. 2002.

Java Performance Tunning 2nd Edition by Jack Shirazi. Jan 2003.

UML Distilled: A brief guide to the standard object modeling language 3rd edition by Martin Fowler. Sep 2003.

The Art of Java Web Development by Neal Ford. Nov 2003.

The Object Primer: Agile Model-Driven Development with UML 2.0 3rd Edition by Scott W. Ambler. Mar 2004.

Tapestry In Action by Howard M. Lewis Ship. Mar 2004.

Code Complete Second Edition by Steve McConnell. Jun 2004.

Head First Design Patterns By Eric Freeman and Bert Bates. Oct 2004.

Beautiful Architecture by Diomidis Spinellis & Georgios Gousious. Jan 2009.

Software Performance and Scalability: A Quantative Approach by Henry H. Liu. May 2009.

Oracle RMAN 11g Backup and Recovery by Robert Freeman. Apr 2010.

Cisco Network Security Auditing by Chris Jackson. Jun 2010.

JSF 2.0 Cookbook by Anghel Leonard. Jun 2010.

JasperReports 3.6 Development Cookbook by Bilal Siddiqui. Jun 2010.

Continuous Delivery by Jez Humble and David Farley. Aug 2010.

Spring Recipes: A Problem-Solution Approach Second Edition by Gary Mak, Josh Long and Daniel Rubio. Sep 2010.

Spring Persistence with Hibernate by Paul Tepper Fisher and Brian D. Murphy. Oct 2010.

IT Architecture for Dummies by Kalani Kirk Hausman and Susan L. Cook. Nov 2010.

Oracle Golden Gate 11g Fundamentals for Oracle Student Guide by Karen Kehn. Feb 2011.

The Clean Coder: A code of conduct for Professional Programmers by Robert C. Martin. May 2011.

Just Spring by Madhusudhan Konda. Jul 2011.

Jenkins: The Definitive Guide by John Ferguson Smart. Jul 2011.

The Web Application Hacker's Handbook Second Edition by Dafydd Stuttard and Marcus Pinto. Sep 2011.

Computer Architecture: A Quantative Approach by John L. Hennesy and David A. Patterson. Fifth Edition Sept 2011.

Beginning Database Design by Clare Churcher. Jun 2012.

Beginning JSP, JSF and Tomcat by Giulio Zambon. Sep 2012.

Pro JPA 2 by Mike Keith and Merrick Schincariol. Sep 2013.

Learning Google Guice by Hussain Pithawala. Sep 2013.

Continuous Enterprise Development in Java by Andrew Lee Rubinger and Aslak Knutsen. Mar 2014.

Beginning Java 8 Languages Fundamentals by Kishori Sharan. Jun 2014.

Java 8 Recipes by Josh Juneau. Sep 2014.

Beginning Java 8 APIs, Extensions and Libraries by Kishori Sharan. Sep 2014.

Spring in Action Fourth Edition by Craig Walls. Nov 2014.

Oracle RMAN for Absolute Beginners by Darl Kuhn. Nov 2014.

Expert Oracle Database Architecture Third Edition by Thomas Kyte and Darl Kuhn. Nov 2014.

Mastering Apache Maven 3 by Prabath Siriwardena. Dec 2014.

Professional Java EE Design Patterns by Murat Yener and Alex Theedom. Jan 2015.

Building Microservices by Sam Newman. Feb 2015.

Stripes by Example by Brent Watson. Mar 2015.

Pragmatic Unit Testing in Java 8 with JUnit by Jeff Langr, Andy Hunt and Dave Thomas. Mar 2015.

Advanced Java EE Development with Wildfly by Deepak Vohra. Mar 2015.

Learning Spring Application Development by Ravi Kant Soni. Apr 2015.

Enterprise Cybersecurity by Scott E. Donaldson, Standley G. Siegel, Chris K. Williams and Abdul Aslam. May 2015.

Oracle: http://www.oracle.com

Wikipedia: http://en.wikipedia.org

DevMedia Brasil: http://www.devmedia.com.br

Camadas Java EE: http://www.devmedia.com.br/camadas-em-aplicacoes-java-ee/26714#ixzz3gSu0aZhQ

Struts: http://struts.apache.org/

Java Server Faces: http://java.sun.com/javaee/javaserverfaces/

Trapestry: http://jakarta.apache.org/tapestry/

GWT: http://www.gwtproject.org/

Enterprise Java Beans: http://www.oracle.com/technetwork/java/index-jsp-140203.html

Spring: http://www.springframework.org/

Hibernate: http://www.hibernate.org/

Google Guice: https://github.com/google/guice

Top Link: http://www.oracle.com/technetwork/middleware/toplink/overview/index.html

InfoQ: http://www.infoq.com/br/articles/websocket-java-javaee

2014 State of DevOps Report by PuppetLabs. https://puppetlabs.com/2014-devops-report

Best Practices for Oracle and HP 3Par http://h20195.www2.hp.com/V2/getpdf.aspx/4AA4-4519ENW.pdf

Oracle White Paper. Java Performance, Scalability, Availability & Security with Oracle Database 12c. Jun 2013. http://www.oracle.com/technetwork/database/application-development/12cdb-java-perf-scal-ha-security-1963442.pdf

TOGAF Version 9 Enterprise Edition. Samples, Matrices and Diagrams. Apr 2011. http://www.opengroup.org/bookstore/catalog/i093.htm

Gartner Magic Quadrant for Web Application Firewalls. http://www.gartner.com/technology/reprints.do?id=1-2JHK9Z5&ct=150715&st=sb

Cisco 2015 Midyear Security Report. Ago 2015. http://www.cisco.com/go/msr2015