ANEXO 7

ARQUITETURAS TECNOLÓGICAS PROCERGS

Este anexo apresenta uma visão geral das seguintes plataformas:

1. Plataforma *Microsoft* .NET;
2. Plataforma JAVA;
3. Plataforma *Android, iOS;*
4. Plataformas ECM/BPM.
5. Plataforma PHP
6. Plataforma Low-Code
7. Tecnologias para Cloud

PROCERGS – CENTRO DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO DO ESTADO DO RIO GRANDE DO SUL S/A
Praça dos Açorianos, s/nº – Centro Histórico – CEP 90.010-340 – Porto Alegre/RS – Brasil
PABX + 55 51 3210-3100 – http://www.procergs.rs.gov.br – procergs@procergs.rs.gov.br

1. Plataforma Microsoft .NET - Linguagem C#

1.1. Arquitetura e Tecnologias

A arquitetura de desenvolvimento se baseia em modelo de camadas, permitindo a utilização das mesmas regras de negócio e rotinas de acesso a dados por aplicações com diferentes finalidades, como implementação de interface com o usuário, de serviços web para integração com outras aplicações, de rotinas para processamento batch, entre outras.

As principais tecnologias de mercado utilizadas para desenvolvimento nesta arquitetura são:

- .NET 6;
- ASP.NET Core;
- Entity Framework;
- Angular/Bootstrap

O ambiente de desenvolvimento utiliza também as ferramentas:

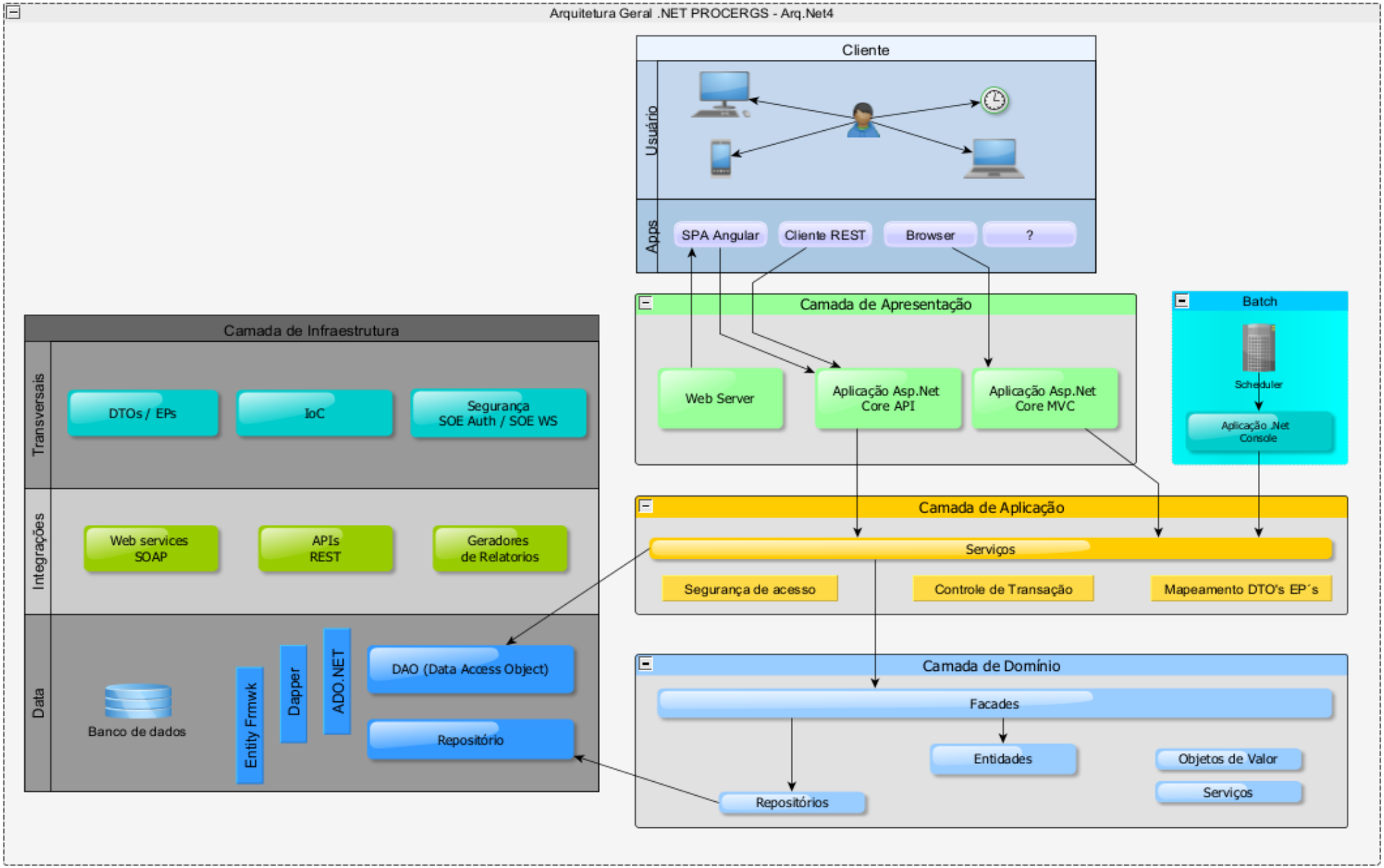
- AzureDevops (TFS)
- GIT
- Artifactory (Nuget)
- Visual Studio

FastReport, SQL Server Reporting Services (SSRS), Cristal Report e Report Viewer.

O diagrama a seguir ilustra como a aplicação dos conceitos e uso das tecnologias citados anteriormente se relacionam para definir a arquitetura:

PROCERGS – CENTRO DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO DO ESTADO DO RIO GRANDE DO SUL S/A
Praça dos Açorianos, s/nº – Centro Histórico – CEP 90.010-340 – Porto Alegre/RS – Brasil
PABX + 55 51 3210-3100 – http://www.procergs.rs.gov.br – procergs@procergs.rs.gov.br

Padrões PROCERGS


Arquitetura Geral .NET PROCERGS - Arq.Net4
Cliente
Usuário
Apps
SPA Angular
Cliente REST
Browser
?
Camada de Apresentação
Web Server
Aplicação Asp.Net Core API
Aplicação Asp.Net Core MVC
Batch
Scheduler
Aplicação .Net Console
Camada de Aplicação
Serviços
Segurança de acesso
Controle de Transação
Mapeamento DTO's EP´s
Camada de Domínio
Facades
Entidades
Objetos de Valor
Serviços
Repositórios
Camada de Infraestrutura
Transversais
DTOs / EPs
IoC
Segurança SOE Auth / SOE WS
Integrações
Web services SOAP
APIs REST
Geradores de Relatorios
Data
Banco de dados
Entity Frmwk
Dapper
ADO.NET
DAO (Data Access Object)
Repositório

1.2. Documentação

Existe uma implementação de referência da arquitetura denominada Aplicação Modelo, que é uma solução do Visual Studio composta por projetos de aplicações dos tipos mais comuns desenvolvidos na empresa. A Aplicação Modelo, juntamente com guias e tutoriais compõem a documentação necessária para orientar o desenvolvimento de aplicações .NET, considerando as particularidades de implementação utilizadas na **PROCERGS**.

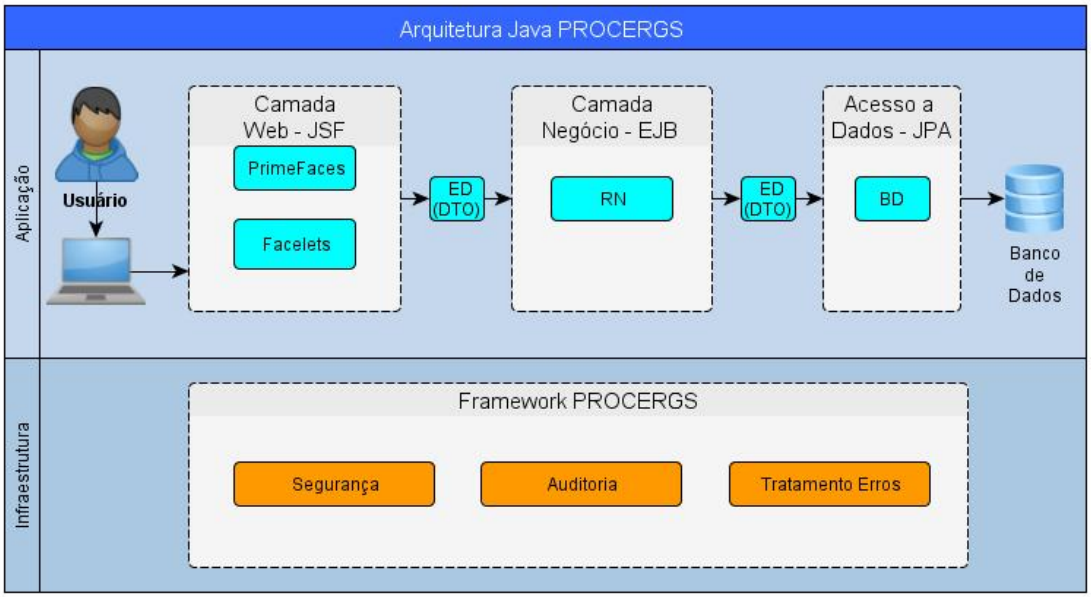
1.3. Linguagens

A linguagem utilizada para desenvolvimento é C#, podendo haver em projetos mais antigos as linguagens Visual Basic .NET.

1.4. Componentes

A **PROCERGS** possui componentes desenvolvidos internamente que facilitam a implementação de funcionalidades comuns às aplicações, relativas a diversos aspectos como segurança de acesso, certificação digital, interface com o usuário, interoperabilidade, entre outros. Estes componentes devem ser utilizados de acordo com os requisitos identificados nos projetos.

2. Plataforma JAVA



2.1. Arquitetura e Tecnologias

Plataforma JAVA EE 8, em especial os componentes:

- JSP;
- JSF 2.2;
- Primefaces 6.2;
- Bean Validation;
- CDI;
- EJB versões 3.1
- JPA versão 2.x;
- JAX-WS;

JAX-RS.

O ambiente de desenvolvimento utiliza também as ferramentas:

- AzureDevops(TFS)
- GIT
- Jenkins;
- Maven;

Eclipse ou Red Hat Jboss Developer Studio

- iReport / Jasper 5.1;

2.2. Documentação

Existe uma implementação de referência da arquitetura denominada Aplicação Modelo, que é um projeto JAVA no Eclipse composto por aplicações dos tipos mais comuns desenvolvidos na empresa. A Aplicação Modelo, juntamente com guias e tutoriais compõem a documentação necessária para orientar o desenvolvimento de aplicações JAVA, considerando as particularidades de implementação utilizadas na **PROCERGS**.

2.3. Componentes

A **PROCERGS** possui componentes desenvolvidos internamente que facilitam a implementação de funcionalidades comuns às aplicações, relativas a diversos aspectos como segurança de acesso, certificação digital, interoperabilidade, entre outros. Estes componentes devem ser utilizados de acordo com os requisitos identificados nos projetos.

3. Plataforma *Android e iOS*

As apps desenvolvidas na **PROCERGS** seguem as guidelines da plataforma que está sendo utilizada. As guidelines são guias oficiais para desenvolvimento, design e publicação de apps.

As *guidelines* são essenciais para garantir uma experiência positiva ao usuário final. Seja qual for o dispositivo ou tamanho de tela, existem orientações das empresas fornecedoras de cada plataforma para desde a concepção de uma app até sua distribuição na loja virtual.

A seguir, uma breve descrição de três destas principais etapas.

3.1. Design

Padrões de interface, usabilidade e ergonomia: devem encantar o usuário, simplificar sua vida e ainda surpreendê-lo positivamente, através de uma interface criativa e interessante.

As interfaces devem seguir as guidelines vigentes na documentação fornecida pela plataforma. Princípios de design, UI (*User Interface*) e UX (*User Experience*), padrões de escrita de código e políticas de distribuição na loja.

A produção gráfica de interface deve basear-se em ícones e tipografias compatíveis com os temas das versões para as quais estejam sendo desenvolvidos as apps.

Devem ser produzidas interfaces para as mais variadas telas de dispositivos, sejam smartphones ou tablets. Ou seja, a mesma interface produzida para uma tela pequena, bem como seus componentes visuais, deve funcionar de forma a adaptar-se perfeitamente em telas maiores sem comprometer a experiência do usuário. Essa experiência é o que deve permear todo o projeto de app.

Maiores detalhes poderão ser consultados em:

Guidelines Android
http://developer.android.com/design/

Guidelines iOS
https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/Introduction/Introduction.html

3.2. Desenvolvimento

Padrões de codificação e melhores práticas para otimizar recursos nos dispositivos que estiverem executando o aplicativo.

Os códigos devem ser simples e objetivos, visando facilitar a manutenção por parte de outras equipes no futuro. Códigos muito complexos devem ser repensados, pois sempre há uma maneira de tornar as coisas mais simples, principalmente em se tratando de tecnologias móveis.

No caso de apps para o cidadão deve haver, em especial, a preocupação em produzir *software* compatível com versões do *Android* ou iOS que estejam vigentes com as versões mais utilizadas pelos consumidores no momento da concepção do projeto.

Devem ser observados padrões sugeridos por cada plataforma a fim de preservar o consumo de bateria dos dispositivos bem como sua performance.

Deverão ser previstos o desenvolvimento de soluções no modelo PWA (Progressive Web App) abrangendo frameworks para desenvolvimento como IONIC, Angular, ReactJS ou algum outro conforme diretriz tecnológica vigente.

Não devem ser adotados frameworks de terceiros a exemplo de ORM (persistência), Annotations ou quaisquer que sejam sem prévio acordo com a **PROCERGS**. A preferência se dará sempre pela produção de aplicativos de forma 100% pura dentro de cada sistema/plataforma. A ideia é evitar frameworks que agilizam o desenvolvimento em detrimento de performance. Devemos nos colocar sob o ponto de vista do usuário final em relação a sua experiência. Uma aplicação móvel desenvolvida com framework “x”,”y” ou “z” de forma produtiva mas cuja interface é lenta e consome muita bateria, não é o que buscamos.

Não incentivamos o uso de bibliotecas de terceiros que não sejam da própria plataforma. Porém, acreditamos fortemente na produção de bibliotecas próprias, devidamente documentadas e com códigos-fonte escritos de forma clara e objetiva, para fins de reuso e consequentemente, provendo mais produtividade, cujo código e documentação do mesmo sejam fornecidos juntamente com os aplicativos finais desenvolvidos.

Orientamos também a preferência sempre do uso de webservices do tipo REST Full para que a comunicação externa do aplicativo se dê da maneira mais rápida, intuitiva e eficiente possível.

Maiores detalhes poderão ser consultados em:

Guidelines Android
http://developer.android.com/develop

Guidelines iOS
https://developer.apple.com/appstore/guidelines.html

3.3. Distribuição

Para fins de publicação o desenvolvedor deve seguir uma série de diretrizes de modo a respeitar políticas de conteúdo, termos de uso da rede, spam e posicionamento na em cada uma das lojas virtuais.

Maiores detalhes poderão ser consultados em:

Guidelines Android
http://developer.android.com/distribute/

Guidelines iOS
https://developer.apple.com/appstore/guidelines.html

4. Plataformas ECM/BPM

4.1. Alfresco ECM e Bonita BPM

O desenvolvimento de aplicações usando Alfresco e Bonita na **PROCERGS** possibilita incorporar documentos e processos nas soluções geradas com estas novas tecnologias, com o armazenamento e o tratamento de Documentos no ECM e o fluxo de trabalho homem-sistema no BPM.

O ambiente ECM/BPM é composto por um conjunto de aplicações que visam disponibilizar uma solução estável e segura a gerência de documentos e de processos, aderente ao ambiente já existente na **PROCERGS**.

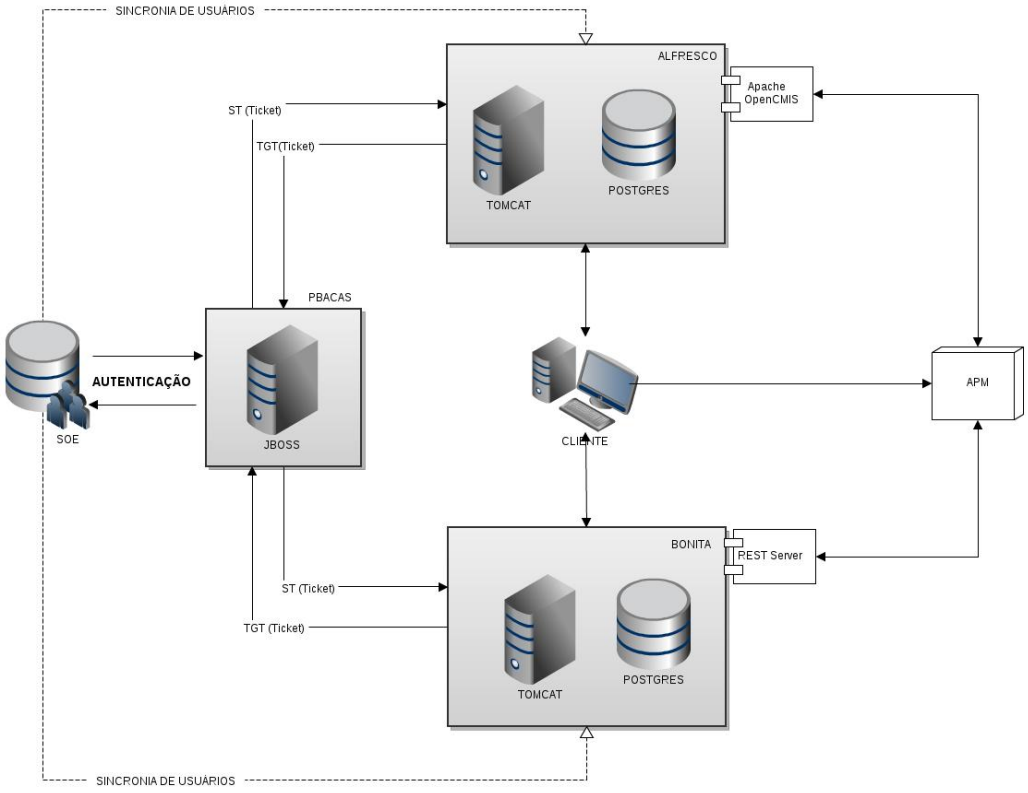
Os produtos gerados neste trabalho e que devem ser observados, no desenvolvimento de qualquer aplicação nestas tecnologias, são:

Arquitetura;

- APM - Aplicação Modelo ECM/BPM;
- Ambiente
- Documentos
- Convenções de programação **PROCERGS**.

4.2. Arquitetura

O diagrama abaixo mostra a arquitetura e a comunicação entre os componentes do Bonita e Alfresco com as aplicações (novas ou existentes) e a forma de trabalho entre as mesmas.



4.3. Descrição:

4.3.1. SOE

O SOE tem como objetivo permitir o controle de acesso às aplicações desenvolvidas na **PROCERGS**, além da administração das autorizações dadas aos usuários. Este controle de acesso e as autorizações estão baseados na criação de usuários e na definição de suas permissões em cada sistema (aplicação).

A autenticação nas plataformas pode ser feita utilizando-se o ticket de transporte do usuário logado, se a aplicação utilizar o SOEWeb, ou o access token, se a aplicação utilizar o SOEAuth, através dos componentes CMISUtil ou ECMInfraUtil (Alfresco) respectivamente, ou BPMInfra (Bonita), que consultam a base de usuários do SOE. Uma vez autenticado nas plataformas, o acesso ao Alfresco e/ou Bonita é liberado.

Existe ainda um procedimento de sincronia de usuários entre o SOE e ambas as plataformas Alfresco e Bonita. Desta maneira é possível relacionar o usuário logado na aplicação com suas permissões aos documentos e tarefas a executar nos processos.

4.3.2. APM

Aplicação Modelo que pode ser utilizada para a implementação de aplicações desenvolvidas pela **PROCERGS** e que precisem utilizar o ECM/BPM.

4.3.3. ALFRESCO

Sistema de Gestão de conteúdo empresarial, multi plataforma de Código Aberto. O Alfresco se propõe como uma alternativa para o gerenciamento de documentos, arquivos, colaboração e também conteúdos web.

O Alfresco é desenvolvido em JAVA, e tem como estratégia prover escalabilidade modular para o gerenciamento de documentação corporativa. Roda sobre um servidor de aplicação TomCat e utiliza o banco de dados PostgreSQL.

O acesso das aplicações aos dados do Alfresco é feito através do serviço OpenCMIS (Apache Chemistry). A aplicação modelo (APM) utiliza este serviço para interagir com o ambiente ECM.

4.3.4. BONITA

O Bonita Open Solution (BOS) é uma suite de Código Aberto composta de workflow e BPM (Business Process Management). O Bonita é executado em um servidor de aplicação TomCat e utiliza o banco de dados PostgreSQL.

O acesso aos processos do Bonita e sua utilização se dá através de um serviço RESTful (BPMInfra), desenvolvido pela PROCERGS.

4.3.5. CAS/JASIG

O CAS (Serviço de Autenticação Central) é um protocolo single sign-on para a web. Sua finalidade é permitir que um usuário acesse vários aplicativos ao mesmo tempo, fornecendo suas credenciais (como o ID de usuário e senha) apenas uma vez. O nome CAS também se refere a um pacote de software que implementa este protocolo. O CAS, nomeado internamente na **PROCERGS** como PBACAS (PBA - Projeto Bonita/Alfresco) roda em um servidor de aplicação JBOSS.

A autenticação no ambiente é feita através do serviço de single sign-on PBACAS, que consulta a base de usuários do SOE. Através de um sistema de tickets, uma vez logado, o acesso ao Alfresco e/ou Bonita é liberado.

Quanto ao acesso das aplicações aos dados do Alfresco, este é feito através do protocolo OpenCMIS. Já o acesso aos processos do Bonita e sua utilização se dá através de um servidor RESTFUL. A aplicação modelo (APM) utiliza tanto o OpenCMIS quanto o REST para interagir com o ambiente ECM/BPM.

Existe ainda um procedimento de sincronia de usuários entre o SOE e ambas as soluções: Alfresco e Bonita. Desta maneira é possível relacionar o usuário logado na aplicação e os documentos e processos.

4.4. APM - Aplicação Modelo com ECM/BPM

A APM, ou Aplicação Modelo, tem como principal objetivo servir como exemplo de implementação das principais tecnologias e arquiteturas utilizadas na **PROCERGS**. Além disto, serve como laboratório para evolução das arquiteturas, modelos de implementação e ambientes utilizados e avaliação de novas tecnologias a serem adotadas na empresa.

A aplicação modelo está escrita em JAVA e .NET e para as duas linguagens existe o respectivo modelo. A diferença está no acesso ao Bonita, que no JAVA é pelo cliente do Bonita e no .Net é feito por REST.

4.4.1. ECM - Funcionalidades utilizam o serviço CMIS

- Conecta no repositório;
- Insere documentos no repositório em um determinado type e pasta e seus meta-dados;
- Lista documentos do repositório de um determinado Type;
- Consulta documento do repositório;
- Download de um documento do repositório;
- Faz checkOut e checkIn de documentos no repositório;
- Pesquisa documento através de meta-dados ou pequisa nos textos dos documentos (FULL TEXT).

4.4.2. BPM - Funcionalidades utilizam o serviço BPMInfra

- Conecta no Bonita BPM;
- Dispara um processo;
- Dá andamento no processo;
- Exibe "lista de trabalho" (lista atividades que esperam uma ação do usuário logado) genérica (de qualquer processo) e específica (de um processo específico);
- Pesquisa processos pelos seus atributos genéricos e específicos.

4.5. Documentos, Templates e Links

A **PROCERGS** possui alguns documentos que contém orientações para o Alfresco ECM e Bonita BPM, que facilitam o entendimento além de alguns templates para embasar e testar o funcionamento destas plataformas. Além disso, existem os links dos fabricantes.

4.6. Convenções de código para programas com ECM/BPM na PROCERGS

Convenções para serem utilizadas no desenvolvimento de soluções com ECM e BPM são as mesmas que já são utilizadas nas diferentes plataformas de programação (JAVA e .NET).

As principais orientações ao desenvolver com Alfresco ECM e Bonita BPM são:

- uso de webservices para integração;
- uso do padrão CMIS para o ECM;
- evitar o uso de Groovy no BPM.

5. Plataforma PHP

5.1 Fronteira entre *Site* e Sistema *Web*

A maioria dos produtos desenvolvidos pela PROCERGS chega até os usuários através da web, então surge a dúvida quanto à categoria do produto que será produzido: é um site ou um sistema? A resposta é mais complexa do que parece, tecnicamente ambos são iguais, mas levantamos algumas características que ajudam no direcionamento do projeto e na escolha das ferramentas mais apropriadas.

Site	Sistema	Peso
Conta criada pelo próprio usuário ou acesso irrestrito O usuário cria e administra sua conta, e/ou geralmente acessa de forma irrestrita, sem precisar de conta.	**Conta criada por um administrador** Uma administração cria a conta do usuário e determina suas permissões de acesso.	3
O mundo acessa Qualquer pessoa que esteja conectada à internet pode acessar.	**Somente determinados usuários acessam** O ambiente é controlado. Usuários precisam fazer parte de um grupo restrito e/ou estar conectados em redes específicas.	4
Orientado para consumo de conteúdo CMS, blogs, notícias, portais.	**Orientado para execução de tarefas** Cadastros, pesquisas, listas de dados, processamentos, transações.	2
Usuário executa poucas tarefas Usuário possui um conjunto reduzido de tarefas que não fazem parte de seu trabalho no dia-a-dia.	**Usuário executa muitas tarefas** Usuário possui um conjunto grande de tarefas que fazem parte de seu trabalho diário.	2
Cada tarefa tem uma frequência baixa de execução por usuário Usuário executa as tarefas em curtos momentos de forma esporádica.	**Cada tarefa tem uma frequência alta de execução por usuário** Usuário executa as tarefas durante longos momentos de forma repetida.	1
Foco da tarefa no próprio usuário O usuário executa tarefas relacionadas a si próprio. Exemplo: Cancelar uma compra, agendar uma consulta médica, reservar um quarto de hotel.	**Foco da tarefa no objetivo do sistema** O usuário executa tarefas relacionadas ao processo automatizado pelo sistema, normalmente sem vinculação direta com o usuário em si. Exemplo: Aprovar um medicamento para o paciente, processar uma folha de pagamento.	4

Seguindo as características acima e somando-se os pesos para cada tipo, Sistema ou Site, conseguimos definir o que será desenvolvido.

Este anexo se refere somente à sistemas desenvolvidos em PHP, definido então o Symfony ou seus derivados como framework padrão.

5.2 Frameworks

5.2.1 Symfony

O Symfony é um conjunto de componentes desacoplados e reutilizáveis para a criação de sistemas web. Sua implementação é baseada nas PSRs e sua comunidade é numerosa e colaborativa. O Symfony é opensource e está sob o licenciamento MIT.
O Symfony foi construído a fim de cumprir os seguintes requisitos:

- Fácil instalação e configuração em mais plataformas (garantido para trabalhar no padrão *nix e Windows)
- Mecanismo de banco de dados independente
- Simples de usar, na maior parte dos casos e suficientemente flexível para se adaptar aos casos complexos
- Baseado na premissa de convenção sobre a configuração, o desenvolvedor precisa configurar apenas o convencional
- Compatível com a maioria das melhores práticas web e padrões de design (PSR[1])
- Código legível, documentação e fácil manutenção
- Fácil de estender, permitindo a integração com outras bibliotecas

5.2.2 Silex

Silex é um microframework PHP. É baseado no Symfony e Pimple e também inspirado pelo Sinatra.
É indicado para aplicações simples e possui as seguintes características:

- Conciso: Silex expõe uma API intuitiva e concisa que é divertida de usar.
- Extensível: o Silex possui um sistema de extensão baseado em um micro-service-contêiner Pimple que torna ainda mais fácil se amarrar em bibliotecas de terceiros.
- Testável: o Silex usa o HttpKernel da Symfony, que abstrai solicitação e resposta. Isso torna muito fácil testar aplicativos e a própria estrutura. Ele também respeita a especificação HTTP e incentiva seu uso adequado.

5.2.3 Laravel

Laravel é um Framework PHP utilizado para o desenvolvimento web, que utiliza a arquitetura MVC e tem como principal característica ajudar a desenvolver aplicações seguras e performáticas de forma rápida, com código limpo e simples, já que ele incentiva o uso de boas práticas de programação e utiliza o padrão PSR-2 como guia para estilo de escrita do código.
Para a criação de interface gráfica, o Laravel utiliza uma Engine de template chamada Blade, que traz uma gama de ferramentas que ajudam a criar interfaces bonitas e funcionais de forma rápida e evitar a duplicação de código.
Para se comunicar com um Banco de Dados o Laravel utiliza uma implementação simples do ActiveRecord chamada de Eloquent ORM, que é uma ferramenta que traz várias funcionalidades para facilitar a inserção, atualização, busca e exclusão de

[1]) PSR: São recomendações de implementação para PHP utilizando melhores práticas de desenvolvimento. http://www.php-fig.org/

registros. Com configuração simples e pequena e com pouco código podemos configurar a conexão com Banco de Dados e trabalhar com ele.

5.2.4 Yii

Yii é um framework de alta performance em PHP que utiliza componentes para o desenvolvimento de grandes aplicações Web. Permite máxima reutilização de códigos na programação Web e pode acelerar significativamente o processo de desenvolvimento. O nome Yii (pronunciado `i`) representa as palavras *fácil* (easy), *eficiente* (efficient) e *extensível* (extensible).

Este framework destina-se preferencialmente ao desenvolvimento de aplicações departamentais de clientes, desenvolvidas e hospedadas em seus próprios ambientes.
Qualquer utilização diferente da referida acima deverá ser avaliada e autorizada pela equipe de tecnologia da **PROCERGS**.

5.3 Outros requisitos

5.3.1 Gerenciamento de Dependências

O uso do Composer[2] como gerenciador de dependências é necessário devido à arquitetura do framework totalmente baseado em componentes (Bundles) e também por ser padrão de mercado relacionado à tecnologia.

5.3.2 Acesso à Banco de Dados

É obrigatório o uso de PDO[3] para acesso à banco de dados e recomendável o uso de alguma implementação ORM[4], como Doctrine2 DBAL, Eloquent ORM, Aura SQL, Propel ou ZF2 Db. Não serão aceitas implementações sem “bind params”.

5.3.3 Versão do PHP

Para desenvolvimento e deploy das aplicações web e sites, a versão recomendada é a 7.2 ou superior, seguindo o padrão “stable” vigente do suporte da PROCERGS.

5.3.4 Validação de código

Toda a implementação deverá ser submetida à uma análise estática de código a fim de validar o PHP Coding Standards. Ferramentas como http://cs.sensiolabs.org/ podem ser utilizadas para esta análise.

O código HTML também será validado com ferramentas tais como https://validator.w3.org (HTML), https://developers.google.com/web/tools/lighthouse (para verificação de performance e demais boas práticas) e http://www.dasilva.org.br/

---

[2]) Composer: É uma ferramenta para gerenciamento de dependências em PHP. Ele permite que você declare as bibliotecas dependentes que seu projeto precisa e as instala para você.

[3]) PDO: (PHP Data Objects) é uma extensão que fornece uma interface padronizada para trabalhar com bancos de dados, cuja finalidade é abstrair a conexão e interações com os bancos.

[4]) ORM: É uma técnica (Design Pattern) de mapeamento objeto relacional que visa criar uma camada de mapeamento entre nosso modelo de objetos (aplicação) e nosso modelo relacional (banco de dados) de forma a abstrair o acesso ao mesmo.

(para validação de acessibilidade, em casos de sites/sistemas expostos para a internet e/ou que seja uma premissa este tipo de acesso).
PSR
Toda a implementação deverá estar aderente aos PSRs previstos em http://www.php-fig.org/.

6. Plataforma Low-Code

Arquitetura e Tecnologias

Uma low-code development platform – ou plataforma de desenvolvimento com baixa programação – são sistemas online que permitem a criação de outros sistemas ou aplicativos, mas com a programação básica, o que facilita e acelera sua produção. Pelos benchmarks e o Gartner, para o primeiro ano de uso do low-code é esperado 20% de ganho nas atividades de codificação nos projetos de software, em relação ao desempenho atual.

Demais etapas do projeto independem do low-code. Algumas empresas projetam ganhos de até 40% na codificação após o segundo ano de low-code.

Devido à característica, o perfil de profissional para esta atividade requer conhecimento na ferramenta que possui uma curva de aprendizado extremamente curta comparadamente às demais tecnologias de mercado.

A PROCERGS estima em média um ganho de produtividade de 30%.

7. Tecnologias para Cloud

Desenvolvimento de aplicações na forma de imagens Docker para execução em ambiente Kubernetes. Processos de build e deploy automatizados através do Azure Devops ou Jenkins.