ARQUITETURAS TECNOLÓGICAS PROCERGS

Este anexo apresenta uma visão geral das seguintes plataformas:

1. Plataforma *Microsoft* .NET - Linguagem C#
2. Plataforma JAVA
3. Plataformas Mobile - Android e iOS
4. Plataformas ECM/BPM
5. Plataforma PHP
6. Plataformas Low-Code
7. Plataforma Python
8. Plataforma Angular
9. Tecnologias para Cloud

1. Plataforma Microsoft .NET - Linguagem C#

1.1. Arquitetura e Tecnologias

Existem duas arquiteturas de desenvolvimento, a primeira, com API baseada em controladores, e a segunda, atualmente em desenvolvimento, com API Mínima. Ambas se baseiam em modelo de camadas.

1.1.1. Baseado em controlador – Art.NET 4

A arquitetura de desenvolvimento Arq.NET, 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 de usuário, publicação de serviços web, rotinas para processamento de batch, entre outras.

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

- .NET 9.0
- ASP NET Core MVC
- Entity Framework Core
- Swagger
- AutoMapper

1.1.2. API Mínima

Arquitetura para aplicações .NET, com foco em arquitetura limpa, modularidade e boas práticas de desenvolvimento. Fornece uma base sólida e moderna para a criação de novo projetos, utiliza API Mínima, *Domain Driven Design* (DDD), separação de responsabilidades por camadas e injeção de dependências.

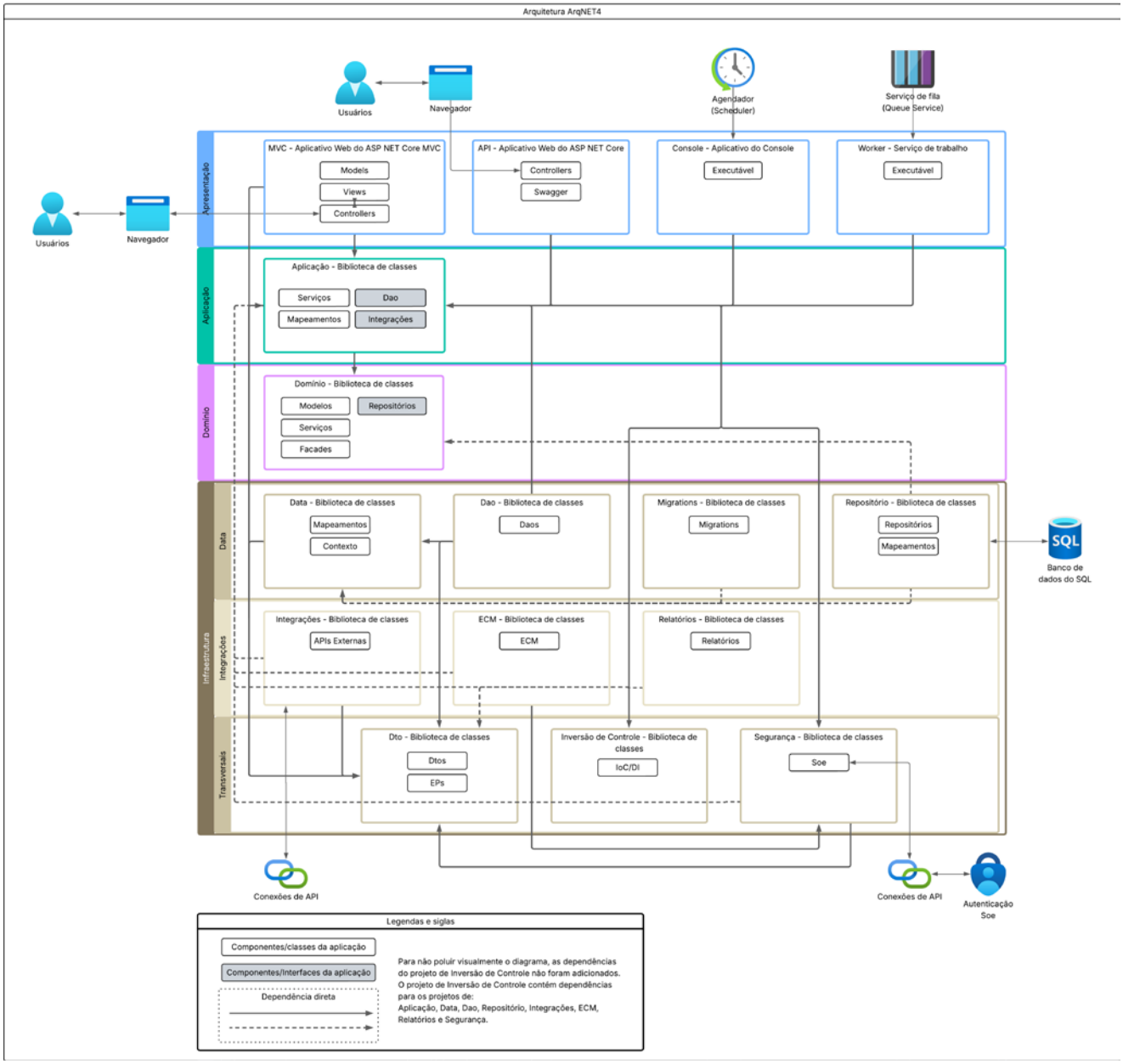
As principais tecnologias de mercado utilizadas para desenvolvimento nessa arquitetura são:

- .NET 9.0
- ASP NET Core
- Entity Framework Core
- OpenAPI
- Scalar

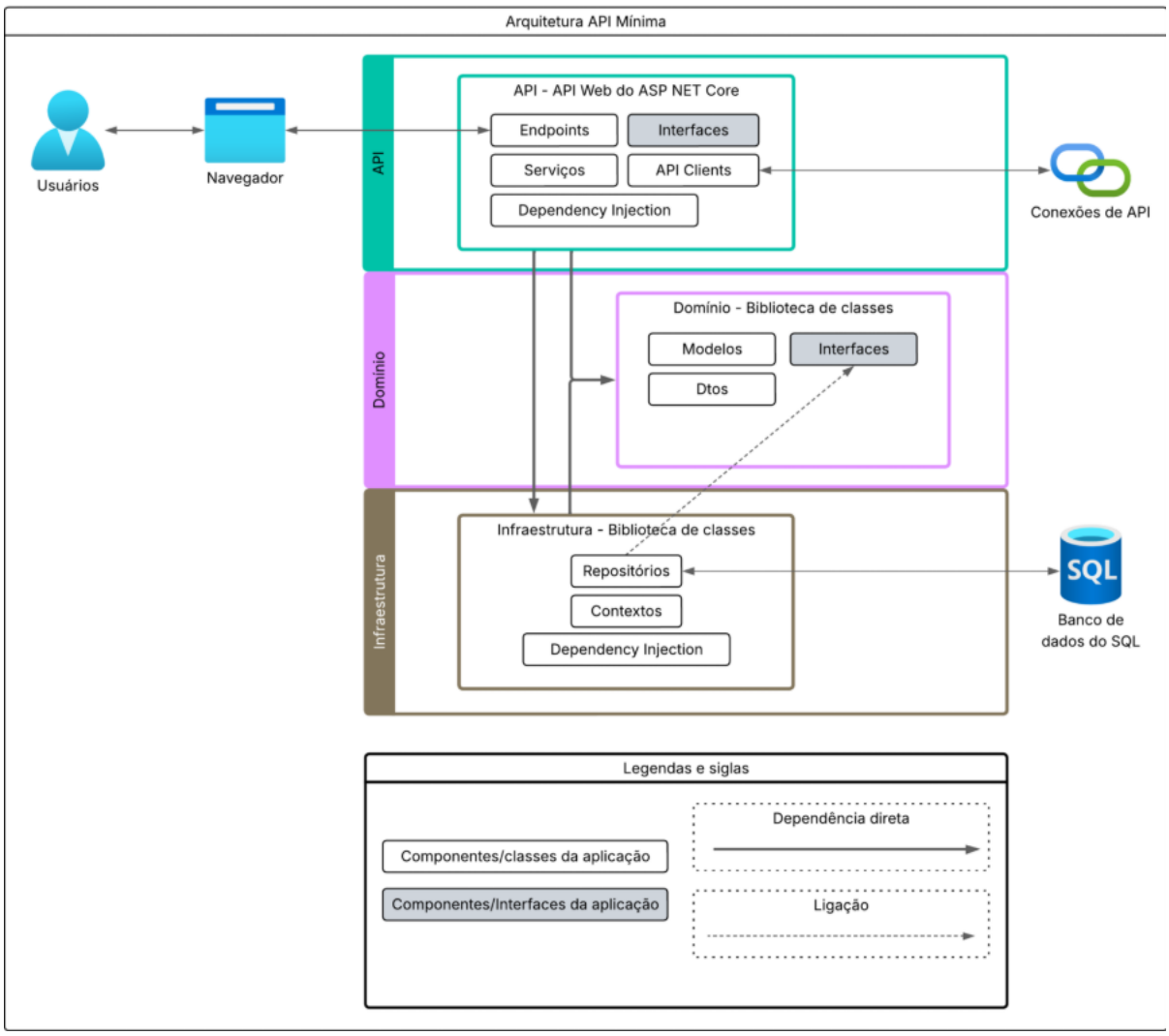
1.2. Modelagem da Plataforma

Os diagramas a seguir ilustram como as aplicações dos conceitos e uso das tecnologias citados anteriormente se relacionam para definir as arquiteturas.

1.2.1 Baseado em controlador – Arq.NET 4



1.2.2 API Mínima



1.3 Documentação

Existem implementações de referência das arquiteturas, denominadas Aplicação Modelo, que são soluções do Visual Studio, compostas por projetos de aplicações dos tipos mais comuns desenvolvidos na empresa. A Aplicação Modelo e seus arquivos README.md, juntamente com posts no Site do Desenvolvimento PROCERGS, guias e tutoriais, compõem a documentação necessária para orientar o desenvolvimento de aplicações .NET, considerando as particularidades de implementação utilizadas na PROCERGS.

1.4 Linguagens

A linguagem utilizada para desenvolvimento é o C#.

1.5 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, documento genéricos, entre outros. Estes componentes devem ser utilizados de acordo com os requisitos identificados nos projetos.

2. Plataforma Java

2.1 Arquitetura e Tecnologias

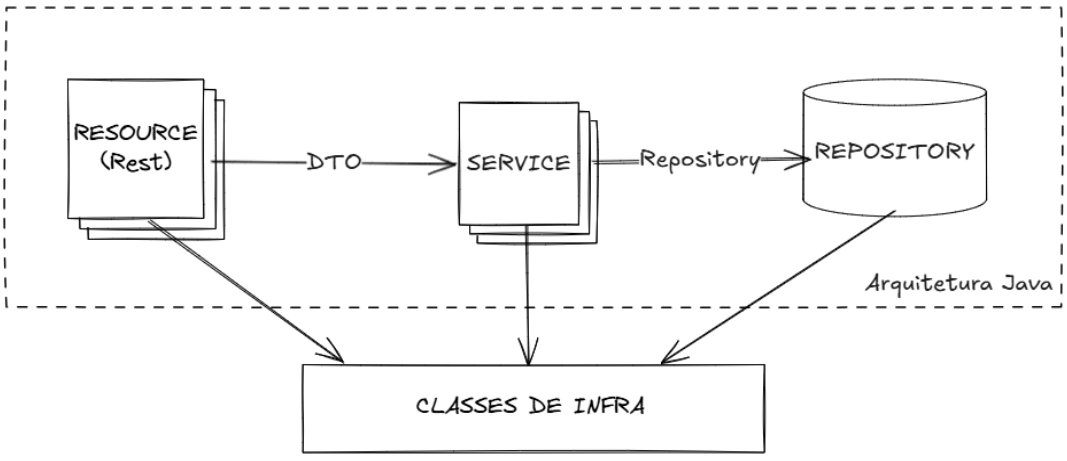
Na PROCERGS, os projetos atuais utilizam a linguagem Java para o desenvolvimento backend dos sistemas. Utilizamos o framework Java Quarkus, sempre priorizando a adoção da versão mais atual. Entre os principais componentes utilizados estão o CDI, JPA e Panache.

No ambiente de desenvolvimento, empregamos ferramentas como:

- Azure DevOps
- Kubernetes
- Maven
- VS Code
- Git

2.2 Modelagem da Plataforma

A seguir temos uma imagem do modelo estruturado para as aplicações Java.



A proposta da arquitetura da APM Quarkus se baseia no próprio framework Quarkus com variações das arquiteturas Clean Architecture, Hexagonal Architecture e SOLID. Não foi seguido estritamente por completo nenhuma destas arquiteturas, mas usado alguns conceitos neste projeto de cada uma delas. Continua sendo uma divisão por camadas e responsabilidades, mas com um foco em classes menores e com a tentativa de responsabilidades únicas em algumas classes. Além disto, foi implementado o conceito de não expor para fora do BackEnd classes que são de acesso a dados.

2.3 Documentação

Dispomos de uma Aplicação Modelo Java (APM), que serve como referência para os projetos desenvolvidos na empresa. No Site do Desenvolvimento PROCERGS, há um post detalhando a estrutura da APM. Tanto a aplicação quanto o post funcionam como um guia para orientar os desenvolvedores quanto à organização e às boas práticas adotadas nos projetos Java da Procergs.

2.4 Linguagens

A linguagem utilizada para o desenvolvimento é o Java a partir da versão 17, e como framework o uso do Quarkus.

2.5 Componentes

São utilizados CDI, JPA e Panache mas, 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. Plataformas Mobile - Android e iOS

Os apps desenvolvidos na PROCERGS seguem as *guidelines* da plataforma que está sendo utilizada. As *guidelines* são guias oficiais para desenvolvimento, design e publicação de apps e 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 desde a concepção de um app até sua distribuição na loja virtual.

3.1 Arquiteturas e Tecnologias

A arquitetura mobile adotada na PROCERGS prioriza aplicações modernas, modulares e escaláveis, com uso de boas práticas consolidadas de mercado, como Clean Architecture, MVVM e injeção de dependências.

As tecnologias recomendadas incluem:

Android:

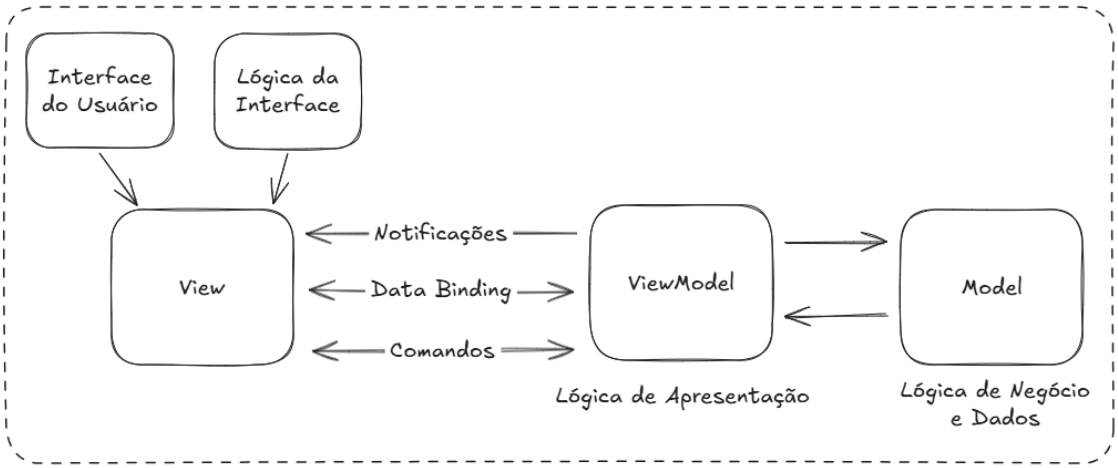
- Kotlin (preferencialmente versão 2.0).
- Jetpack Compose para UI declarativa.
- Hilt para injeção de dependências.
- ViewModel + LiveData/StateFlow.
- Retrofit + Moshi para chamadas de API.
- Capacitor Plugins (para aplicações híbridas em Angular).

iOS:

- Utilização de **Ionic Framework** com **Angular** + **Capacitor**.
- Construção de aplicações nativas para Android e iOS a partir de uma única base de código.
- Estrutura de projeto e código compartilhado com a versão Android (em projetos híbridos).
- Acesso a recursos nativos via Plugins Capacitor (câmera, geolocalização, notificações, arquivos etc.).

3.2 Modelagem da Plataforma

A modelagem das aplicações mobile na PROCERGS adota o padrão arquitetural MVVM (Model-View-ViewModel), com foco na separação de responsabilidades, escalabilidade e testabilidade do código.



View

A View é responsável por exibir a interface com o usuário e capturar interações, como cliques e entradas de dados. Ela observa o estado da ViewModel e reage automaticamente às mudanças.

Responsabilidades:

- Exibir a interface do usuário (UI).
- Reagir a alterações de estado usando data binding (vinculação de dados) com a ViewModel.
- Encaminhar eventos e comandos do usuário para a ViewModel.
- Não contém lógica de negócio, apenas lógica de apresentação visual mínima.

ViewModel

A ViewModel atua como o elo entre a View e o Model. Ela processa os dados recebidos do Model, prepara-os para exibição e expõe estados e eventos observáveis pela View.

Responsabilidades:

- Gerenciar o estado da tela e fornecer os dados prontos para exibição.
- Receber eventos da View e agir de acordo.
- Comunicar-se com o Model para buscar ou atualizar dados.
- Usar LiveData ou StateFlow para notificar a View de mudanças de estado.
- Não tem referência direta à View, garantindo baixo acoplamento.

Model

O Model representa a camada de lógica de negócio e acesso a dados da aplicação. Pode envolver regras, validações, persistência local (Room) ou chamada a serviços externos (APIs).

Responsabilidades:

- Conter a lógica de negócio central da aplicação.
- Acessar dados de fontes externas, como APIs REST ou bancos locais.
- Retornar os dados solicitados pela ViewModel.
- Atualizar o estado da aplicação quando necessário.

3.3 Linguagens

- Kotlin (principal para Android)
- Java (em projetos anteriores ao Kotlin)
- TypeScript (em projetos híbridos usando Ionic Framework com Angular + Capacitor)

3.4 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

3.5 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.

Deverá ser previsto o desenvolvimento de soluções no modelo PWA (Progressive Web App) abrangendo o framework **Ionic** com **Angular**, 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 à 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/develop/

3.6 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 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/distribute/

3.7 Documentação

A PROCERGS mantém uma **Aplicação Modelo Kotlin**, com exemplos atualizados de:

- Telas em Jetpack Compose
- Navegação e gerenciamento de estado
- Persistência local com Room
- Comunicação com APIs via Retrofit
- Aplicação de padrões como MVVM e Clean Architecture

A PROCERGS também mantém uma **Aplicação Modelo Ionic**, contemplando:

- Navegação entre páginas e uso de rotas protegidas
- Integração com plugins nativos via Capacitor
- Comunicação com APIs REST com HttpClient do Angular
- Gerenciamento de estado reativo com RxJS/NgRx
- Estrutura modular seguindo boas práticas de Angular
- Aplicação de padrões como MVVM adaptado para Angular

A documentação complementar de ambas está disponível no Site do Desenvolvimento PROCERGS.

4 Plataformas ECM/BPM

4.1 Plataforma ECM

4.1.1 Arquitetura e Tecnologias

A plataforma Alfresco ECM (Enterprise Content Management) é uma solução de gerenciamento de conteúdo corporativo de código aberto escrita em Java, projetada para ajudar organizações a capturar, armazenar, gerenciar e compartilhar documentos e outros tipos de conteúdo digital de forma segura e eficiente.

O gerenciamento de conteúdos não estruturados, como arquivos de texto, imagens, vídeos, PDFs e planilhas, é uma das principais funcionalidades do Alfresco ECM. Esses conteúdos representam a maior parte da informação gerada nas organizações

e, por não seguirem um modelo de dados fixo (como em bancos de dados relacionais), exigem ferramentas especializadas para seu controle e aproveitamento.

Como o Alfresco gerencia conteúdos não estruturados:

1. Armazenamento e Organização: Os arquivos são armazenados no repositório de conteúdo do Alfresco. São organizados em pastas e podem ter metadados personalizados conforme a regra de negócio da aplicação;

2. Indexação e Busca: Utiliza o Apache Solr para indexar o conteúdo dos arquivos pesquisáveis e seus metadados. Permite buscas por palavras-chave, filtros por tipo de documento, datas, autores, etc. Suporta busca full-text dentro de documentos pesquisáveis;

3. Controle de Versão: Cada alteração em um arquivo pode gerar uma nova versão, mantendo o histórico completo. Permite restaurar versões anteriores;

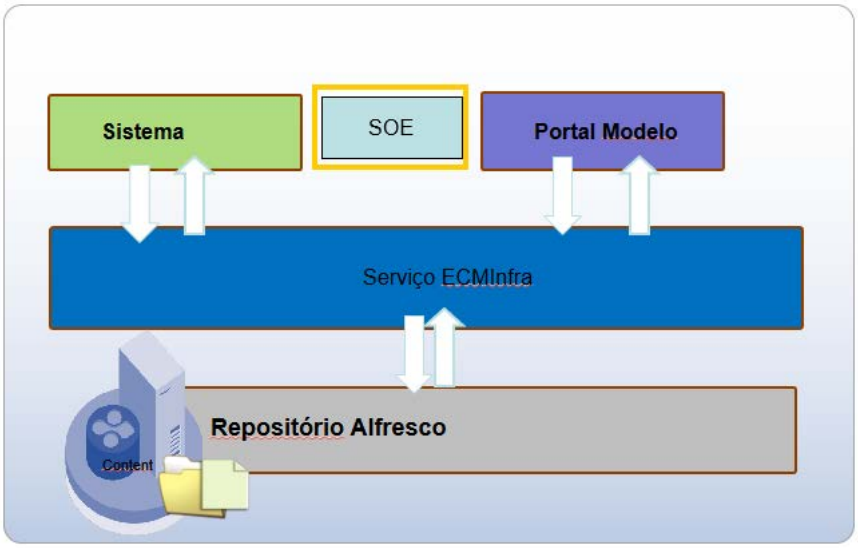
4. Controle de Acesso: Permissões granulares por usuário, grupo, pasta ou documento. Possui integração com SOEWeb/SoeAuth para autenticação e controle de permissões;

5. Integrações: É acessado pelo cliente sempre através de uma aplicação Procergs.

4.1.2 Modelagem da Plataforma

Cada aplicação deve definir seu modelo de documentos (xml), contendo uma ou mais famílias de documentos (types), e seus metadados específicos. Além disso deve criar as ações no SOEWeb, conforme regra de formação, para a sincronia de seus usuários e permissões com o repositório do Alfresco.

Assim é a forma de integração das aplicações Procergs com o repositório ECM Alfresco:



4.1.3 Documentação

A documentação de como adotar o serviço ECMInfra como forma de operar o repositório de ECM Alfresco está disponível no **Site do Desenvolvimento PROCERGS.**

4.1.4 Linguagens

Não há implementação direta no repositório além da criação do xml com as famílias de documentos (types) e seus metadados específicos.

4.1.5 Componentes

Toda operação no repositório é realizada pela aplicação através do serviço ECMInfra, que é um serviço REST que oferece todas os métodos para as funcionalidades do repositório, como upload, consulta, pesquisa com metadados/full-text, versionamento, etc.

4.2 Plataforma BPM

4.2.1 Arquitetura e Tecnologias

A plataforma Bonita BPM (da Bonitasoft) é uma solução de automação de processos de negócios (BPM) que permite modelar, executar, monitorar e otimizar processos empresariais de forma visual e colaborativa. Ela é voltada para organizações que desejam digitalizar fluxos de trabalho, integrar sistemas e melhorar a eficiência operacional. Combina um motor de processos baseado na notação para mapeamento de processos BPMN 2.0, uma interface de desenvolvimento visual e uma plataforma extensível para criar aplicações de processos personalizadas. É ideal para cenários onde processos envolvem múltiplos sistemas, usuários e regras de negócio.

A arquitetura do Bonita é modular e baseada em componentes independentes, sendo seus principais elementos:

1. Bonita Studio: Ambiente de desenvolvimento visual para modelagem de processos BPMN, criação de formulários, regras de negócio e conectores. Deve ser instalado na máquina do desenvolvedor, permite simular e testar processos localmente;

2. Bonita Engine: Motor de execução dos processos, responsável por gerenciar instâncias de processos, tarefas, variáveis e regras de fluxo;

3. Bonita Portal / Bonita UI Designer: Interface web utilizada em tempo de desenvolvimento para usuários executarem tarefas, acompanharem processos e interagirem com formulários;

4. Bonita Runtime (Bonita Server): Ambiente de execução que integra o motor de processos, APIs REST, conectores e serviços de autenticação, implantado em servidor de aplicação Tomcat;

5. Banco de Dados: Armazena metadados dos processos, variáveis, usuários, histórico de execução e arquivos anexados;

6. APIs REST: Permitem integração com sistemas externos, automação de tarefas e criação de front-ends personalizados;

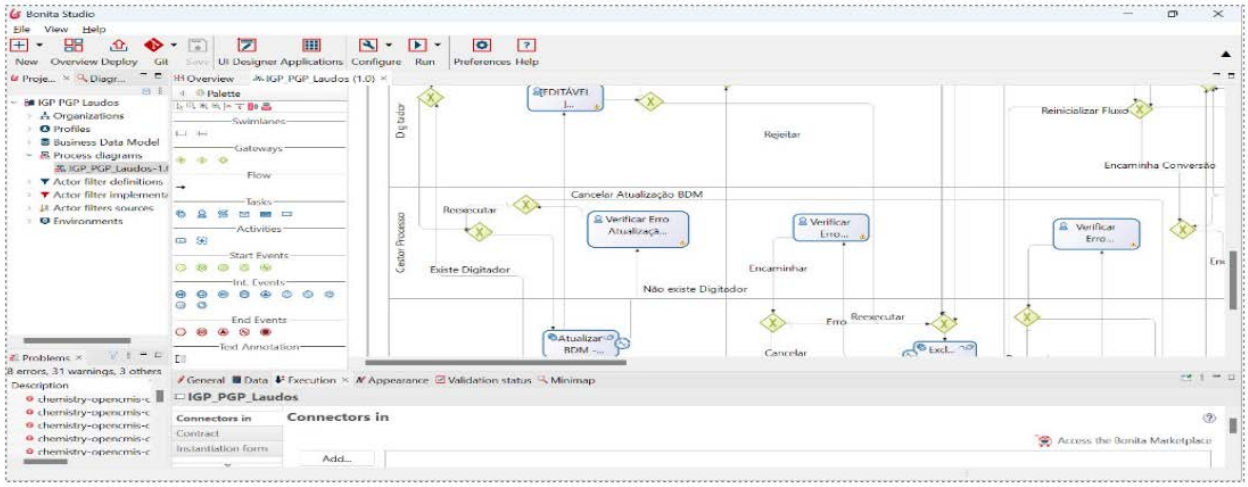
7. Conectores: Módulos reutilizáveis para integração com sistemas como bancos de dados, e-mail, web services, ERPs, ECMs (como Alfresco), etc.

4.2.2 Modelagem da Plataforma

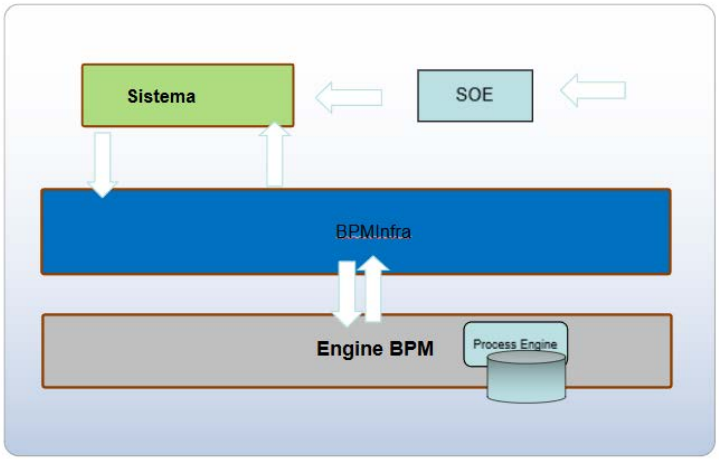
Cada aplicação deve realizar o mapeamento dos seus processos para automação no Bonita Studio, utilizando a notação BPMN 2.0. Todo o teste do roteamento do processo é feito localmente na ferramenta e depois o arquivo de execução do processo é instalado no servidor, onde é testada a integração com a aplicação. Além disso deve criar as ações no SOEWeb, conforme regra de formação, para a sincronia de seus usuários e permissões de execução das tarefas dos processos com o engine do Bonita.

O engine de processos é sempre acessado pela aplicação através do serviço BPMInfra, que contém todos os métodos das funcionalidades do engine, como: início de instância de processo, consulta lista de trabalho com as tarefas pendentes do usuário, encaminhamento de atividades, etc:

Bonita Studio - 2022.2



Assim é a forma de integração das aplicações Procergs com o engine BPM Bonita:



4.3 Documentação

A documentação de como usar o BPM na Procergs está disponível no Site do Desenvolvimento PROCERGS.

4.4 Linguagens

Deve ser utilizada a notação BPMN para modelagem dos processos.

4.5 Componentes

Toda operação no repositório é realizada pela aplicação através do serviço BPMInfra, que é um serviço REST que oferece todas os métodos para as funcionalidades do engine.

5 Plataforma PHP

O PHP é uma linguagem de programação amplamente utilizada no desenvolvimento de aplicações web do lado do servidor, com foco em processamento de requisições HTTP, manipulação de dados, acesso a banco de dados e geração dinâmica de conteúdo. Sua estrutura base é composta por scripts executados no servidor, organizados a partir de arquivos PHP, utilização de variáveis, funções, classes, controle de fluxo e integração direta com servidores web, sendo essa base comum independentemente do framework utilizado.

Na PROCERGS, alguns sistemas legados são mantidos em PHP, mas a orientação é que esta linguagem **não seja utilizada em novos projetos**. Os sistemas existentes não seguem um padrão único de arquitetura ou framework, pois foram desenvolvidos em contextos e por equipes diferentes ao longo do tempo.

5.1 Tecnologias de Desenvolvimento

As tecnologias abaixo foram utilizadas em projetos legados, sendo necessário que o desenvolvedor que for atuar em algum destes projetos as domine:

4. PHP: versão 7.2 ou superior, seguindo o padrão “stable” vigente do suporte da PROCERGS.
5. MySQL: versão 8.0.43.
    - Frameworks: Symfony, Silex, Laravel e Yii.
    - Composer como gerenciador de dependências.
    - PDO (PHP Data Objects) para acesso a banco de dados.
    - ORM (Design Pattern) para mapeamento objeto-relacional.

5.2 Orientações Gerais de Desenvolvimento:

- Implementação deve seguir o PHP Coding Standards, com os PSRs previstos em https://www.php-fig.org/psr/
- O código HTML também deverá ser validado com ferramentas tais como https://validator.w3.org.
- Sempre observar as boas práticas vigentes, validando performance e acessibilidade em casos de sites/sistemas expostos para a internet e/ou que seja uma premissa este tipo de acesso, com ferramentas como https://developers.google.com/web/tools/lighthouse

6 Plataformas Low-Code

6.1 Arquitetura e Tecnologias

Uma low-code development platform – ou plataforma de desenvolvimento com baixa programação – é um sistema online que permite a criação de outros sistemas ou aplicativos, mas com a programação básica, o que facilita e acelera sua produção.

A ferramenta Oracle APEX é uma solução de desenvolvimento low-code adotada pela PROCERGS, indicada prioritariamente para projetos de baixa complexidade e ciclo de vida curto.

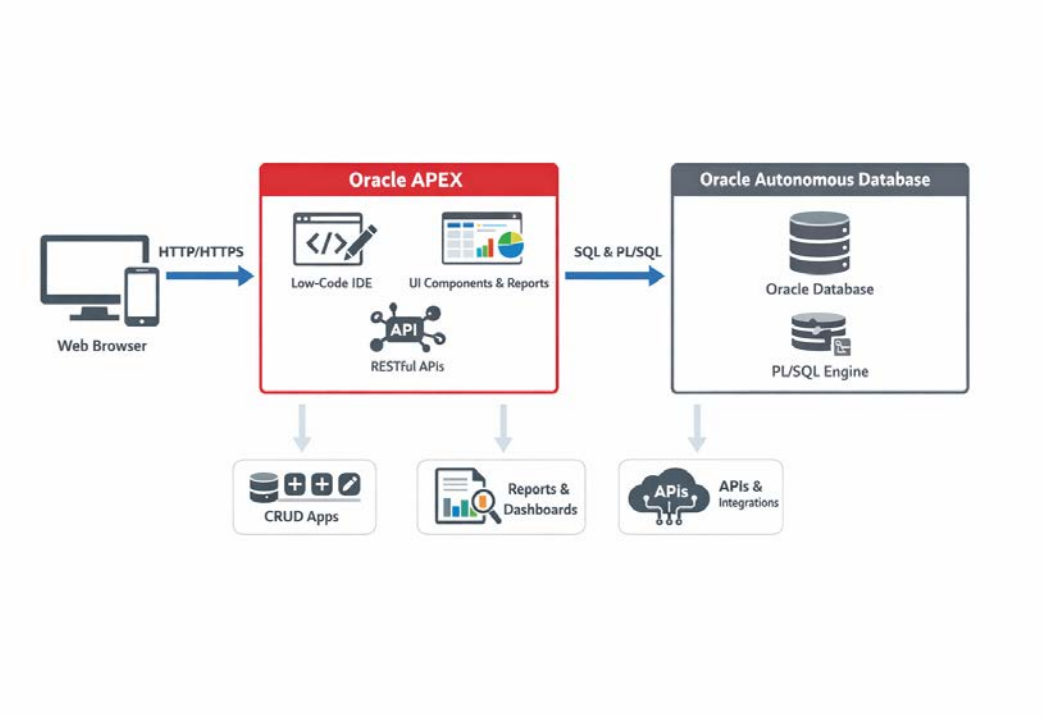
O Oracle APEX é uma plataforma simplificada de desenvolvimento web que disponibiliza um IDE acessível via navegador, não exigindo a instalação de ferramentas adicionais no ambiente do desenvolvedor. Apesar de sua abordagem low-code, é necessário conhecimento sólido em PL/SQL, uma vez que a plataforma é fortemente baseada no banco de dados Oracle e grande parte da lógica de negócio é implementada nesse contexto.

A solução oferece recursos que facilitam a criação de operações CRUD, relatórios simples, dashboards básicos e APIs REST, sendo adequada para aplicações administrativas, protótipos funcionais e sistemas com regras de negócio limitadas.

Atualmente, o Oracle APEX está habilitado exclusivamente no Oracle Autonomous Database, hospedado na nuvem pública da Oracle. Dessa forma, todo projeto que utilize essa tecnologia deve avaliar previamente os impactos de latência e conectividade nas integrações com outros sistemas, especialmente aqueles localizados em ambientes on-premises ou em nuvens distintas.

Para projetos desenvolvidos em Oracle APEX que demandem relatórios impressos ou documentos com layout complexo, recomenda-se a utilização da ferramenta JasperReports, encapsulada em uma aplicação Java, a fim de suprir limitações nativas da plataforma nesse tipo de requisito.

A figura a seguir ilustra a estrutura do Oracle APEX utilizada na Procergs:



Para o desenvolvimento de aplicações em Oracle APEX, é obrigatório que o desenvolvedor possua conhecimento sólido em PL/SQL, incluindo modelagem de dados, criação de procedures, packages, triggers e controle transacional, bem como domínio de SQL e conceitos básicos de desenvolvimento web (HTTP/HTTPS, HTML, CSS e JavaScript) para customizações da interface. Também é esperado entendimento dos recursos nativos do APEX, como segurança, autenticação, autorização, itens de página, processos e consumo/exposição de APIs REST.

Como conhecimento opcional e desejável, especialmente em projetos que demandem relatórios formatados ou documentos oficiais, recomenda-se experiência na construção de relatórios com JasperReports, incluindo design de templates (JRXML), uso de parâmetros, sub-relatórios e exportação em formatos como PDF. Adicionalmente, é desejável conhecimento em Java para encapsular o JasperReports em uma aplicação dedicada (por exemplo, um serviço REST), responsável por receber requisições do Oracle APEX, executar a geração dos relatórios e retornar os documentos gerados de forma integrada e desacoplada da aplicação principal.

7 Plataforma Python

A plataforma Python na empresa passou por uma evolução significativa ao longo dos anos. Inicialmente, sua adoção foi focada no nicho de ciência de dados, onde se tornou uma ferramenta indispensável para a análise e processamento de informações. Com o sucesso e a estabilidade demonstrada, o uso do Python foi expandido para novas áreas:

- Desenvolvimento de aplicações de back-end
- Criação de APIs e serviços
- Soluções de Inteligência Artificial
- Automação de processos
- Gestão de conteúdo baseada no CMS Plone

7.1 Arquitetura e Tecnologias

Ao trabalhar com Python, o desenvolvedor deve conhecer as seguintes ferramentas:

7.1.1 Tecnologias de Desenvolvimento

- Python: Versões 3.9+, preferencialmente 3.11 ou 3.12
- Plone: CMS para gestão de conteúdo
- ReactJS: Conhecimento para integração com front-end do Plone CMS
- FastAPI: Framework para desenvolvimento de APIs
- Uvicorn: Servidor ASGI para aplicações FastAPI

7.1.2 Conteinerização e Orquestração

- Docker: Criação de imagens para aplicações Python
- Kubernetes: Conhecimento em orquestração de containers
- OpenShift: Plataforma utilizada para gestão de containers na empresa
- Helm: Gerenciador de pacotes para Kubernetes/OpenShift

7.1.3 Banco de Dados e Persistência

6. Bancos de dados: PostgreSQL, MySQL, Oracle
7. ORMs: SQLAlchemy, Django ORM
8. Gestão de credenciais: Integração com sistemas de secrets do OpenShift

7.1.4 CI\CD e Versionamento

9. Git: Controle de versão
10. Azure DevOps: Pipelines de CI/CD
11. OpenShift Build: Construção automatizada de imagens
12. GitHub: Familiaridade com fluxos de trabalho baseados em GitHub

7.1.5 Padrões Arquiteturais

O desenvolvedor deverá seguir os padrões arquiteturais estabelecidos pela empresa:

13. Arquitetura em camadas: Separação clara entre domínio, infraestrutura e interfaces
14. Princípios SOLID: Especialmente para desenvolvimento de serviços e APIs
15. Clean Architecture: Para projetos de maior complexidade
16. API-First: Desenvolvimento orientado por APIs bem documentadas

7.2 Requisitos de Documentação

A documentação de uma aplicação Python deve seguir os itens constantes nas diretrizes internas da empresa para a documentação de sistemas (MDP). No caso de API's, o desenvolvedor deve utilizar alguma ferramenta como Swagger que gera a documentação dos endpoints em tempo de execução e a disponibilize em rota alternativa.

A documentação de sistemas Python para APIs deve atender aos seguintes requisitos:

- Documentação automática via Swagger/OpenAPI
- Exemplos de uso para cada endpoint

17. Documentação de modelos de dados e esquemas
    - Descrição clara de fluxos de autenticação e autorização

7.3 Qualidade de Código

O desenvolvedor deverá utilizar as seguintes ferramentas para garantir a qualidade do código:

- Black: Formatação de código
- isort: Organização de imports
- Ruff: Linting rápido
- MyPy: Verificação de tipos estáticos
- Pytest: Testes automatizados

7.4 Entregáveis Esperados

Para cada projeto, o desenvolvedor deverá entregar:

- Código fonte: Seguindo os padrões arquiteturais e de qualidade
- Documentação: Conforme especificado acima
- Arquivos de configuração: Docker, Kubernetes/Openshift, Helm
- Testes Automatizados: Unitários, integração e funcionais

- Pipelines de CI/CD: Configurados para Azure DevOps

8 Plataforma Angular

8.1 Arquitetura e Tecnologias

Na PROCERGS, a plataforma Angular é utilizada para o desenvolvimento de aplicações front-end, com foco em modularidade, reutilização de componentes e integração com APIs backend em .NET ou Java (Quarkus).

A arquitetura padrão segue princípios de separação de responsabilidades, utilizando os seguintes conceitos e tecnologias:

- Angular: versão LTS mais recente (preferencialmente Angular 19+).
- TypeScript como linguagem principal.
- RxJS para programação reativa.
- NgRx (Redux) ou Signals para gerenciamento de estado (conforme necessidade do projeto).
- Bootstrap e NGX-Bootstrap como framework principal de interface gráfica, utilizado para estilização, construção de layouts responsivos e componentes visuais reutilizáveis.
- Angular Material pode ser utilizado complementarmente para construção de layouts.
- Lazy Loading de componentes para otimização de desempenho.
- SCSS como pré-processador de estilo.
- i18n para internacionalização.
- Integração com backend REST/OpenAPI.

Além disso, recomenda-se a adoção de boas práticas como:

- Estrutura de pastas organizada por domínios (feature-based structure).
- Componentes reutilizáveis.
- Testes unitários com Jest ou Karma/Jasmine.
- Testes de integração com Cypress.

8.2 Modelagem da Plataforma

A modelagem das aplicações Angular na PROCERGS está alinhada ao padrão moderno de desenvolvimento introduzido nas versões mais recentes do framework, adotando componentes standalone como estrutura principal, com foco em simplicidade, modularidade e melhor desempenho de build.

A navegação é realizada via Angular Router, com uso de Guards para segurança e Resolvers para carregamento de dados.

8.3 Documentação

Existe uma **Aplicação Modelo Angular**, que serve como base para novos projetos. Essa aplicação inclui exemplos de:

- Estrutura modular.

- Integração com APIs REST.
- Gerenciamento de estado.
- Tematização e internacionalização.
- Boas práticas de codificação e testes.

A documentação complementar está disponível no Site do Desenvolvimento PROCERGS.

9 Tecnologias para Cloud

O ambiente PROCERGS de Cloud - Computação em Nuvem Pública, Privada ou Híbrida - segue os princípios:

- Cloud Native Apps - Aplicações desenvolvidas de forma a tirar o melhor proveito dos serviços do ambiente Cloud.
- Cloud First – Para aplicações novas ou migrações de legado, a prioridade é desenvolver para ambiente de Cloud. Outros ambientes somente se houverem requisitos que impeçam o uso da Cloud.
- A ordem de prioridade é:
    - SaaS – Software como serviço: Functions, Lambdas
    - Contêiner - Imagens Docker, Pods executando em Kubernetes, repositório Harbor
    - Máquina Virtual – Imagem de Sistema Operacional gerenciada pela PROCERGS, agentes de gerenciamento, segurança e monitoria.
- Preferência no uso de serviços PaaS – Plataforma como Serviço - para componentes da solução, tais como: Balanceador de Carga, API Manager, Armazenamento de arquivos, Bancos de Dados, Modelos de Inteligência Artificial e outros onde aplicável.
- Coleta de métricas no padrão OpenTelemetry.
- Esteira DevSecOps no serviço Azure DevOps, automatizando o GIT, pipeline de build, verify e deploy.
- Preferência na automação com práticas de IaC – Infraestrutura como código - usando Ansible Automation Platform.