Arquitetura do PJe 2

Índice

1 Introdução

1.1 Escopo

Este documento tem como proposta apresentar a arquitetura PJe 2, contempla as definições das tecnologias envolvidas para implementação de uma nova versão do PJe que propõe atualizar o ambiente tecnológico, atendendo aos requisitos funcionais e os não funcionais, promovendo melhor disponibilidade, confiabilidade, escalabilidade e interoperabilidade da aplicação aos diversos usuários do PJe.

Apresenta-se também um entendimento das motivações que levaram às decisões técnicas na arquitetura proposta.

1.2 Termos e Abreviações

Esta seção apresenta o conceito de alguns termos utilizados no documento para melhor entendimento, dispostos em ordem alfabética.

Termo	Descrição
API	API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web.
REST	Representational State Transfer
Funcionalidade do PJe	Requisito funcional dependente do PJe, que não possui versionamento próprio, sendo-lhe atribuída a mesma versão do PJe. Seus dados advêm de um ou mais módulos do PJe, da base ou via serviços Web, reutilizando sua lógica de aplicação e seus dados. Sua identidade visual é a mesma do PJe, visto que é parte integrante e inseparável do mesmo. Não há artefato específico para ser implantado e nem repositório próprio.
Módulo do PJe	Requisito funcional dependente do PJe, visto que seus dados são buscados em tempo real na base de dados do PJe ou via serviços Web, mas que possui versionamento próprio. Expõe uma ou mais interfaces para que outros módulos do PJe reutilizem sua lógica de aplicação e eventualmente é responsável por um conjunto de dados do PJe. Em alguns casos, pode ser implantado de forma redundante ou contingencial para fins de escalabilidade. Sua identidade visual e suas tecnologias devem seguir obrigatoriamente o padrão de arquitetura (https://pje.csjt.jus.br/documentacao/index.php/Arquitetura_do_PJe_2) e infraestrutura do PJe. de forma que o usuário não perceba diferença entre módulo e funcionalidade. Não deve possuir recurso de autenticação próprio, pois deverá reutilizar a autenticação provida pelo PJe. Dependendo do funcionamento do módulo, pode ser recomendável a utilização de uma base replicada para não prejudicar o funcionamento do PJe.
Satélite do PJe	Requisito funcional que pode ser implantado e utilizado de forma independente do PJe. Possui dados próprios, embora eventualmente adicione, consulte ou consuma dados do PJe via módulo específico de integração, sendo vedado o acesso direto a base de dados do PJe. Seu versionamento é próprio, mas deve manter compatibilidade com a versão mais recente do PJe. Sua identidade visual e suas tecnologias devem seguir preferencialmente o padrão de arquitetura e infraestrutura do PJe. O satélite não deverá fazer uso da base de dados réplica em uso do PJe para mitigar eventuais impactos negativos no desempenho do sistema.

1.3 Definições e Restrições de componentes da arquitetura do PJe 2

Critério a ser analisado	Pode ser Funcionalidade?	Pode ser Módulo?	Pode ser Satélite?
Dependência/Acoplamento com o PJe *Definição:* grau de acoplamento com o PJe (as decisões do sistema influenciam/interferem o fluxo processual do PJe) *Forma de classificação:* Negocial *Exemplos:* ▪ Chips (Módulo) - A adição de um chip ao processo precisará de ações de servidores/magistrados para solucioná-los; ▪ Acordo pós-sentença (Funcionalidade) - A formalização de um acordo influencia diretamente no fluxo processual. *Contra-exemplos:* ▪ CTM (Módulo) - Acessa dados do PJe por meio de API (não interfere no fluxo processual) ▪ e-Prec (Módulo) - Acessa dados do PJe por meio de API (não interfere no fluxo processual) ▪ SAO (Módulo) - Acessa dados do PJe por meio de conexões JDBC (não interfere no fluxo processual)	**Sim**	**Sim**	**Não**
Execução independente do PJe *Definição:* execução independente do PJe *Forma de classificação:* Técnico *Exemplos:* ▪ Chips (Módulo) - Necessita dos dados do PJe e dos serviços para realizar as análises dos processos e eventos processuais; ▪ Acordo pós-sentença - Constitui uma etapa do fluxo processual do PJe. *Contra-exemplos:* ▪ CTM (Módulo) - Acessa dados do PJe por meio de API (não interfere no fluxo processual) ▪ e-Prec (Módulo) - Acessa dados do PJe por meio de API (não interfere no fluxo processual) ▪ SAO (Módulo) - Acessa dados do PJe por meio de conexões JDBC (não interfere no fluxo processual)	**Não**	**Não**	**Sim**
Os dados são exclusivos do componente *Definição:* dados internos do componente não são relevantes à tramitação processual. *Forma de classificação:* Técnico *Exemplos:* ▪ CTM (Módulo) - Dados refentes a zoneamento, distribuição de mandados, quantidade de oficiais de demais informações internas à central de mandados não interferem na tramitação judicial.	**Não**	**Sim**	**Sim**

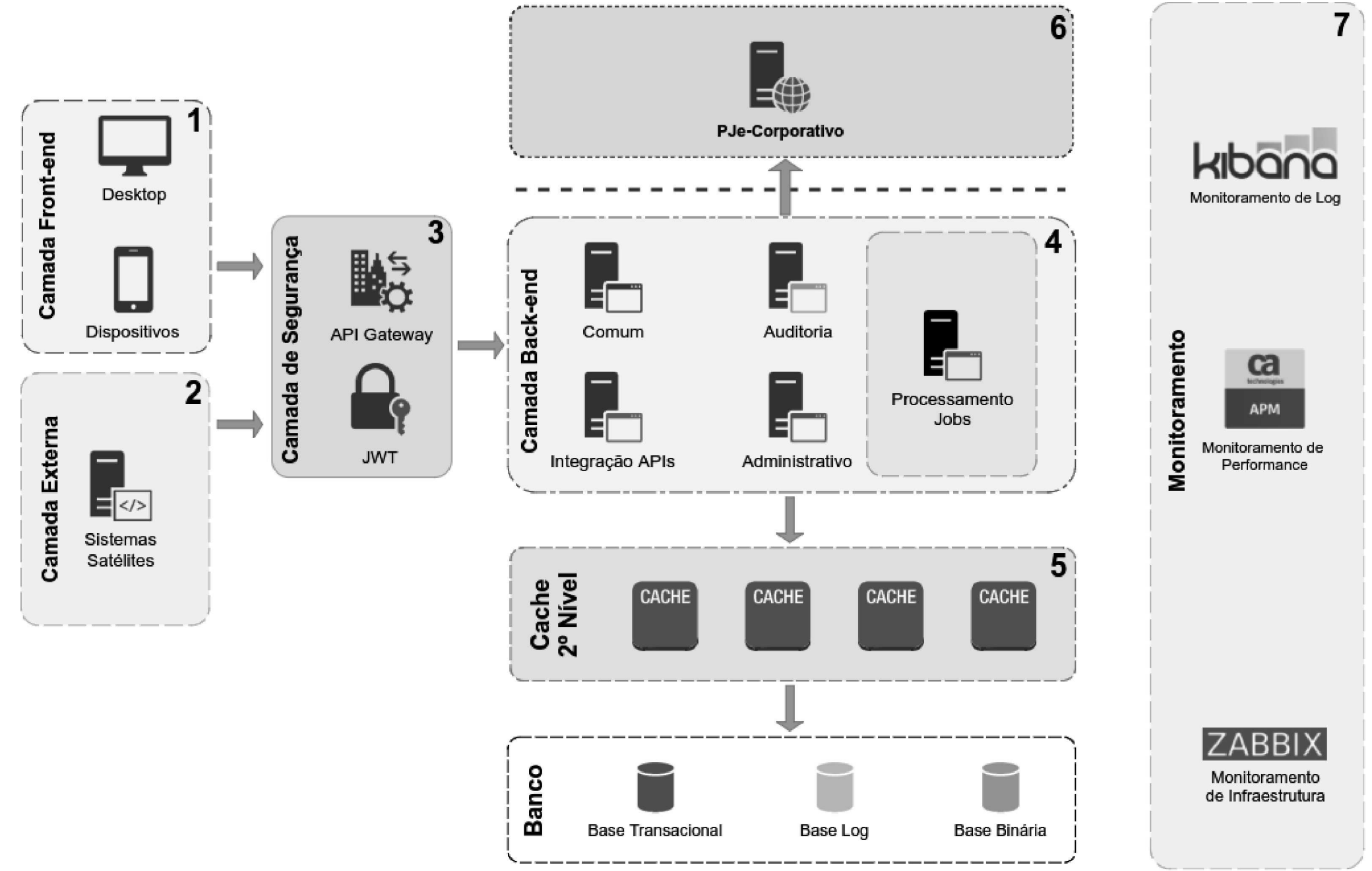
Necessidade de escalabilidade *Definição:* Pode haver necessidade de implantação de artefato específico para atender estratégias de escalabilidade da infraestrutura. *Forma de classificação:* Técnico *Exemplos:* ▪ Chips (Módulo) - Como respondem a eventos processuais para tomada de decisão (aplicar um chip ou não) podem ser necessários muitos módulos rodando em paralelo para suprimir a demanda. Entretanto, não haverá a necessidade de disponibilizar um outro PJe, mas apenas o módulo específico.	**Não**	**Sim**	**Sim**
Acesso direto à base do PJe *Definição:* acessa seus dados diretamente da base do PJe *Forma de classificação:* Técnico *Exemplos:* ▪ Acordos pós-sentença ▪ Jobs do Backend Comum (pje-comum-jobs) *Contra-Exemplos:* ▪ CTM (Módulo) - Não acessa a base do PJe diretamente, apenas via API REST; ▪ CTM (Módulo de Jobs) - acessa diretamente a base da CTM, mas dados do PJe apenas via API REST.	**Sim**	**Sim**	**Não**
Independência de grau *Definição:* precisa consultar/modificar dados de vários graus. Não possui uma base específica para determinado grau. *Forma de classificação:* Negocial *Contra-Exemplos:* ▪ CTM (Módulo) - Recebe e manipula mandados oriundos dos dois graus. Não possui uma base específica para cada grau. ▪ e-Prec (Módulo) - Recebe e manipula precatórios oriundos dos dois graus e os trata em uma base única. Não há uma base de dados para cada grau.	**Sim**	**Sim**	**Não**

2 Proposta Arquitetural

2.1 Requisitos Arquiteturais (Objetivos)

- Entregar uma arquitetura com uma separação bem definida de todas as camadas;
- Implementação da interface do usuário com total acessibilidade;
- Utilização de recursos de cache para obtenção de informações com melhor performance;

2.2 Visão Geral da Arquitetura



(1) Camada Front-end: esta camada representa as interfaces de interação com o as APIs disponibilizadas pelo Back-end.

(2) Camada Externa: representa qualquer sistema externo ao PJe que deseja consumir informações, como por exemplo os Sistemas Satélites.

(3) Camada de Segurança: camada responsável por receber todas as requisições vindas das camadas superiores, validando a autenticação de Tokens, métricas de uso, controle de requisições, versionamento das APIs, e entre outros recursos para gerenciar as APIs da aplicação.

(4) Camada Back-end: principal camada da arquitetura, que fornece o acesso as APIs e implementação das regras negociais do sistema. Nesta camada propõe-se a separação de instâncias em vários módulos negociais e processamento de tarefas assíncronas da aplicação.

(5) Cache 2º Nível: propõe o uso de cache na execução de consultas ao banco de dados, mantendo em cache informações requisitadas repetidamente, diminuindo o acesso ao banco de dados para obter informações já recuperadas em outras requisições, por meio do Infinispan do JBoss EAP 7.

(6) PJe Corporativo: módulo que provê serviços corporativos para atender ambas as instâncias do PJe(1º, 2º e 3º graus), possibilitando reaproveitar serviços como consultas a Receita Federal, validação de certificados digitais e entre outras recursos independentes de grau.

(7) Monitoramento: camada responsável por agregar os logs de todas as instâncias de servidores de aplicações, monitorando a disponibilidade e a detecção de erros ocorridos no sistema.

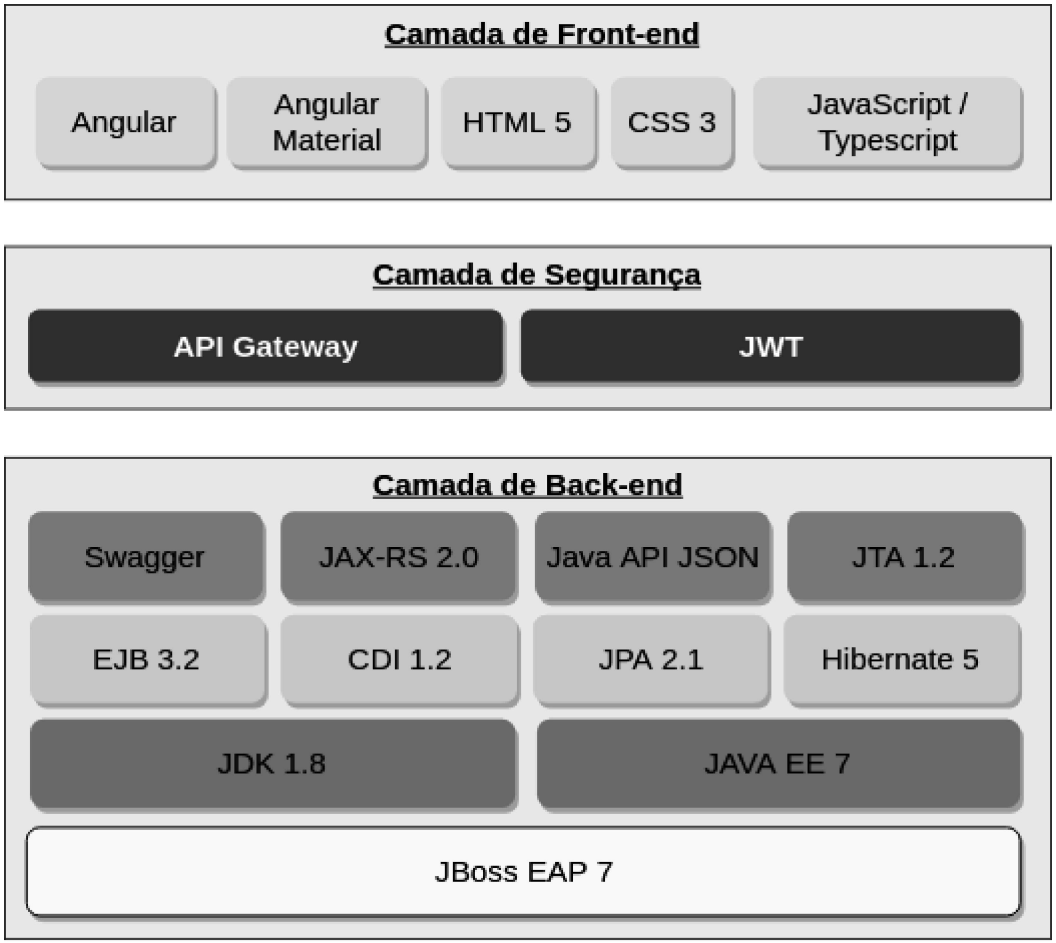
Ob(s): Além dos módulos apresentados em cada uma das camadas descritas acima há diversos outros projetos em desenvolvimento na nova arquitetura que irão resultar em componentes/nós em cada uma destas camadas caracterizadas acima. Maiores detalhes sobre os projetos podem ser visualizados na página de apresentação dos projetos. Cada um dos módulos do PJe ou sistemas satélites irá possuir um *frontend* e um *backend* próprios, que irão compor este novo ecossistema do PJe.

Como exemplos:

- PJe Consulta Pública
- PJe Central de mandados

2.3 Pilha de Tecnologias

Nesta seção são apresentadas todas as tecnologias adotadas no desenvolvimento da arquitetura com base nos requisitos do projeto:



3 Camada Front-end

- **Angular:** plataforma de desenvolvimento para construção de aplicações web (lado cliente), fornecendo recursos para implementação de SPAs (Single Page Applications) com a utilização das linguagens de marcação HTML 5, CSS3 e as linguagens de programação Javascript/Typescript.

```
Versão utilizada: 5.2.0
```

- **Angular Material:** biblioteca baseada na especificação do Material Design do Google que provê grande quantidade de componentes para utilização nos formulários do sistema, promovendo reutilização, testabilidade e acessibilidade na interface do usuário, baseados na especificação do Material Design

disponibilizado pelo Google. Aplica as especificações WCAG referente aos requisitos de acessibilidade de interfaces. Atualmente encontram-se em uma versão BETA, porém grande parte dos componentes oriundos da versão anterior 1.x já foram migrados.

```
Versão utilizada: 5.2.0
```

- **Javascript / Typescript:** linguagem utilizada para criação das interfaces do usuário junto com o framework Angular.

3.1 Decisões Arquiteturais

(1) Framework Javascript: Foram analisados os frameworks: **AngularJS, ReactJS e Angular 2** para utilização no desenvolvimento da camada de Front-end. Todos os frameworks foram analisados com base em performance, estabilidade do projeto, curva de aprendizagem, produtividade e integração com biblioteca de componentes.

De acordo com a consulta técnica solicitada a instituição **Gartner** [1] (https://git.pje.csjt.jus.br/pje2/pje-docs/blob/master/Proposta-Arquitetural/Documentacao/CSJT_WrittenResponse_Framwork%20JavaScript_23mar17.pdf), foi indicado a utilização do framework **Angular 2** para iniciação de novos projetos. Isso devido às melhorias na performance, à definição de novos recursos e ao uso crescente da comunidade que eventualmente está substituindo projetos da versão **AngularJS 1.x** para **Angular 2**. A biblioteca **ReactJS** também foi indicado pela **Gartner**, detalhado como uma biblioteca de fácil aprendizagem e dependente de outras bibliotecas. Neste sentido, é necessária a seleção de outras bibliotecas de terceiros para construir aplicações complexas. A consultoria **Gartner** também relatou que **ReactJS** é mais indicado para equipes que possuam maior expertise em Javascript e desejam montar aplicações completas usando componentes de terceiros em JavaScript. No **Angular** existe um framework completo de desenvolvimento e recursos disponíveis, com dependências mínima de terceiros, para a maioria das aplicações web no lado do cliente.

(2) Biblioteca de componentes: Foram analisados as bibliotecas **Angular Material 2, NG Bootstrap, e PrimeNG** para utilização de componentes nos formulários do sistema, tendo em vista as seguintes características: acessibilidade, variedade de componentes, suporte e integração com as versões mais recentes do Angular.

As três bibliotecas acima foram recomendadas na consulta realizada a **Gartner**. A conclusão da consultoria foi que a decisão de escolha da biblioteca deveria ser analisada com bases nos requisitos da aplicação, quantidade de componentes, customização e documentação disponível.

Acessibilidade	Em relação à acessibilidade de seus componentes o Angular Material é superior ao PrimeNG, pois existe o compromisso do fabricante em manter os requisitos de acessibilidade da WCAG. Contudo, a empresa responsável pelo PrimeNG (Primetec) disponibiliza uma forma de contratação [1] (https://www.primefaces.org/primeng/#/), por meio de aquisição de horas de desenvolvimento, para a adequação de seus componentes em acessibilidade. Segundo o site do Angular Material [2] (https://github.com/angular/material2), a biblioteca incorporará uma ferramenta de auditoria para garantir que a implementação do Angular Material esteja aderente às normas de acessibilidade. A biblioteca NG-Bootstrap é desenvolvida com base em uma das bibliotecas de HTML5 e CSS3 mais utilizadas no mercado para template de aplicações e bastante difundida na comunidade de desenvolvimento de software. Contudo, a versão implementada para o Angular encontra-se em versão Alpha e não possui acessibilidade em maioria de seus componentes.
Componentes	O conjunto de componentes do PrimeNG possui uma quantidade superior em comparação ao Angular Material e NG-Bootstrap.
Compatibilidade	O Angular Material 2 encontra-se em sua versão BETA e em processo de compatibilização com o Angular 4. O PrimeNG encontra-se em sua versão STABLE em compatibilização com o Angular 2. O NG-Bootstrap encontra-se em sua versão ALPHA em compatibilização com o Angular 2. Nenhum das três bibliotecas possuem versão estável para o Angular 4. Apesar de o Angular Material 2 estar em sua versão BETA, espera-se, por ser um subprojeto do Angular e mesmo fabricante, que a versão estável compatível com Angular 4 seja lançada nos próximos meses.
Suporte	O suporte ao Angular Material 2 e NG-Bootstrap é realizado por meio do grupo de discussão do projeto e issues no Github. O PrimeNG, além desses canais, possui uma equipe de suporte para solução de issues com um prazo reduzido, caso seja adquirida a subscrição.

CONCLUSÃO

Foi descartado até o momento, o uso da biblioteca NG-Bootstrap pelos motivos acima. O PrimeNG, no momento atual, aparenta ser a melhor opção, pois é o único dos três que disponibiliza uma versão estável compatível com o Angular. Contudo, sua versão completa e acessível é proprietária e necessita de subscrição, o que seria um contrassenso cuja tendência na Administração Pública é tornar cada vez mais os sistemas estruturantes independentes de fabricante. Outro ponto negativo é que a eventual aquisição ou alteração de componentes é cobrada, segundo o fabricante[3] (https://www.primefaces.org/primeng/#/), por meio de horas de desenvolvimento, o que não é recomendado na legislação brasileira de licitações e contratos de TI.

No intuito de desenvolvermos um novo PJe, que seja disponibilizado para o CNJ e todos os ramos de Justiça, optamos por bibliotecas de componentes livre. Consequentemente, acreditamos que a melhor escolha para a nova arquitetura do PJe é a biblioteca Angular Material.

Mudanças para Angular 4

Nas novas versões da biblioteca de componentes Angular Material 2, o mesmo será compatível somente com o Angular 4. Por este motivo torna-se obrigatório a utilização do Angular 4 para continuar utilizando e recebendo atualização da biblioteca Angular Material.

4 Camada de Segurança

4.1 Autenticação

A autenticação para acesso as APIs do PJe 2 será realizada com o uso de tokens JWT que fornecerá informações básicas do usuário/sistema. O envio do **token** será obrigatório em cada requisição solicitada aos Back-ends da aplicação.

4.1.1 JWT (JSON Web Token)

A utilização de token no formato **JWT** permitirá a implementação de uma arquitetura *stateless* (sem estado de sessão), diminuindo o uso de memória para armazenar os dados de usuários logados na aplicação e melhorando a escalabilidade do PJe.

Tokens no formato JWT é composto por 3 conjuntos de informações, sendo o "cabeçalho", "payload" e "assinatura".

Cabeçalho: Contém as informações de qual algoritmo foi utilizado na criptografia do token.

```
?
1//header
2{
3  "alg": "HS256",
4  "typ": "JWT"
5}
```

Payload: Contém os dados do token, como nome do usuário, perfil, localização e etc.

```
?
1//payload
2{
3  "sub": "1234567890",
4  "nome": "John Doe",
5  "papel": "magistrado",
6  "login": "0000000014",
7}
```

Assinatura: Contém as informações da integridade de assinatura do Token, que é agrupamento das informações do "cabeçalho" e "payload".

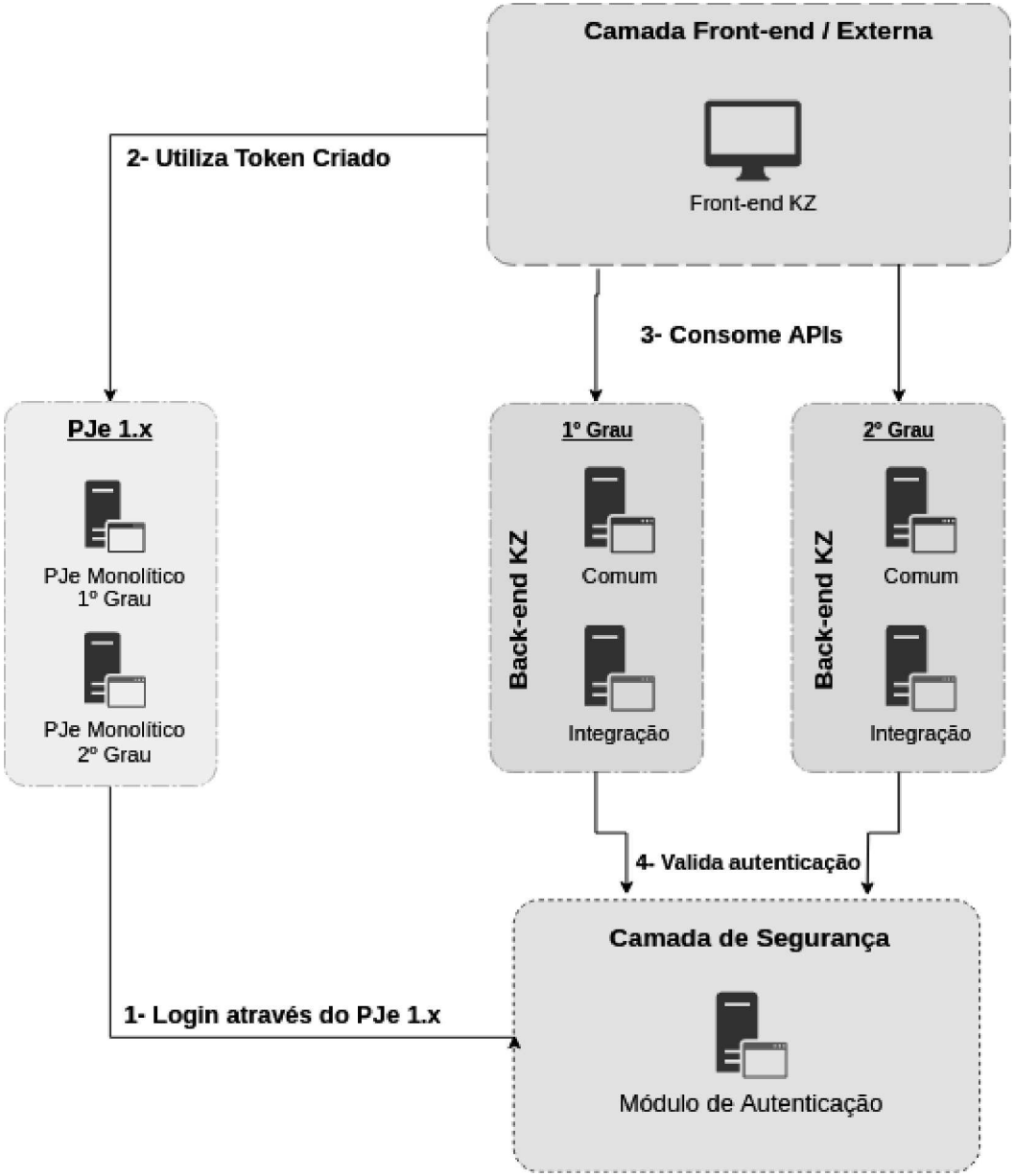
```
?
1//Signature
```

```
2
3HMACSHA256(
4  base64UrlEncode(header) + "." +
5  base64UrlEncode(payload),
6  secret_key
7)
```

4.1.2 Fluxo de Autenticação

A autenticação no PJe 2 é disponibilizada através duas formas para geração do token de segurança, sendo o Token de Serviço e Token de Usuário.

- **Token de Serviço:** token utilizado para acessar serviços que não necessitam da informação do usuário de sistema que está solicitando a informação, quando é necessário validar informações de localizações e papéis do usuário para autorizar acesso a informação. Este token deve ser utilizado para autenticação entre servidores, fornecendo assim a integração de sistemas satélites para obtenção de informações de domínios da aplicação.
- **Token de Usuário:** token que identifica um determinado usuário solicitando acesso a um endpoint seguro. Este token possui, além das informações do serviço, as credenciais do usuário que deseja se autenticar no módulo de segurança e os detalhes do papel selecionado.



Fluxo básico da autenticação de acesso:

1. PJe 1.x solicita a geração de **Tokens** no módulo de segurança para que seja utilizado pelo no Front-end;
2. Front-end recupera o **Token** criado no PJe 1.x, para habilitar o acesso ou não no Front-end;
3. Front-end solicita informações aos Back-ends PJe 2 com o Token de Autenticação.

4. Back-ends PJe 2 valida o token enviado com o módulo de segurança, verificando se o mesmo está integro e não expirado.

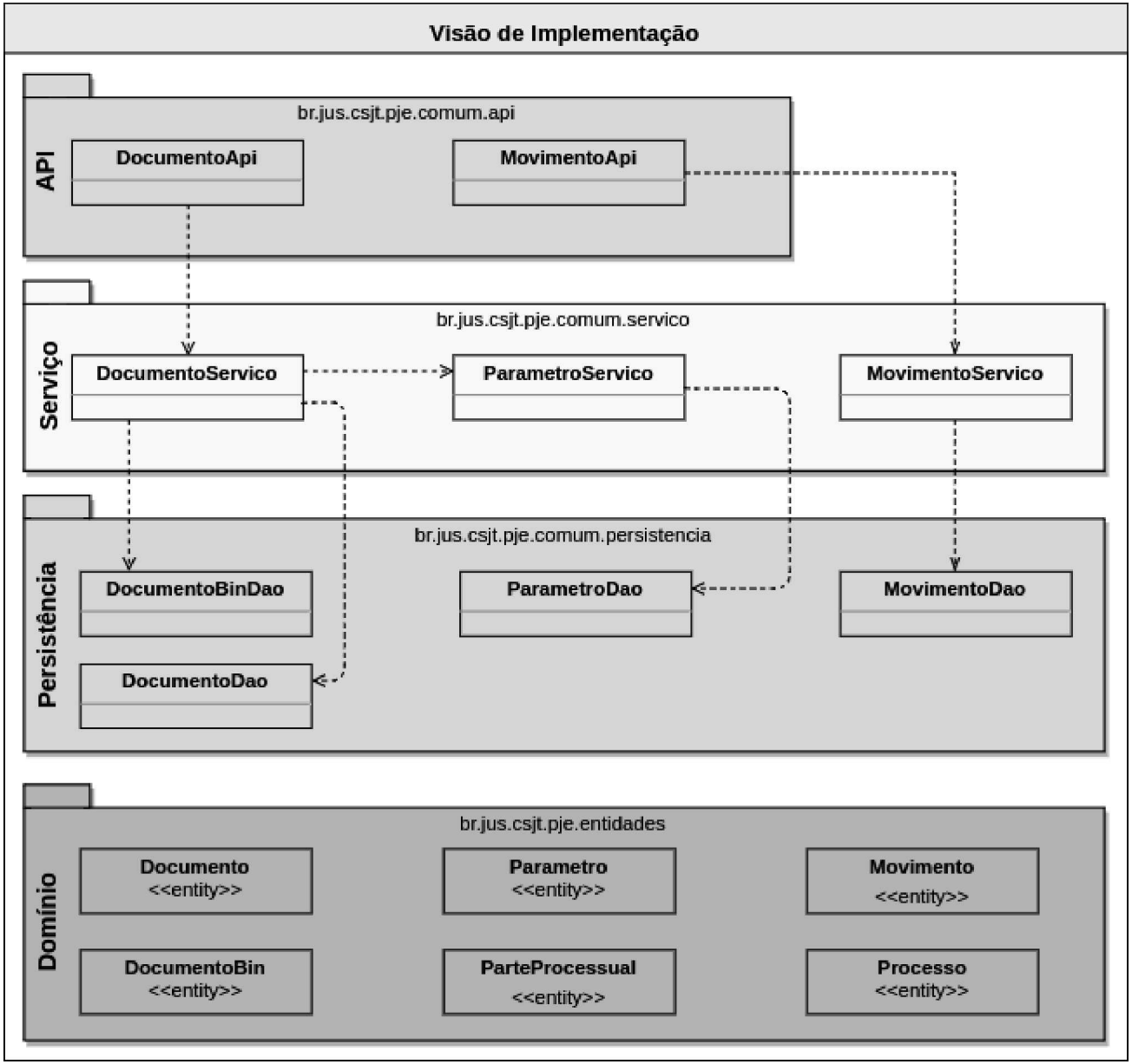
O módulo de segurança será implantado de forma a atender todos os graus de instância (1º , 2º e 3º grau), não sendo necessário haver uma instância específica para cada grau em cada TRT, o que acontece no **PJe 1.x** (monolítico).

Maiores informações técnicas da implementação: Documentação da Autenticação PJe 2 (https://git.pje.csjt.jus.br/pje2/pje-seguranca/wikis/home)

5 Camada Back-end

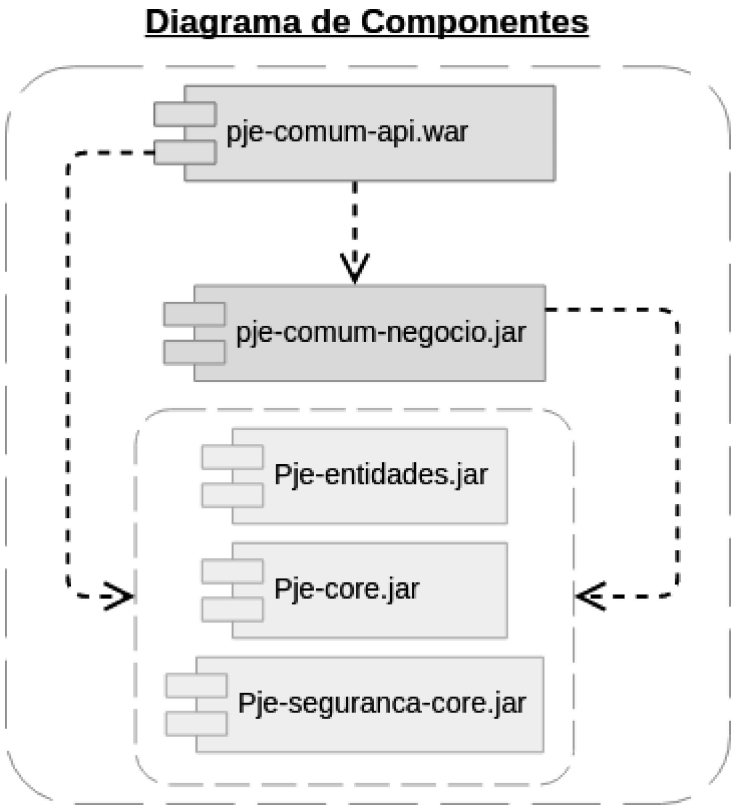
1. **Camada de API:** camada responsável por receber as requisições de acesso a aplicação, realizadas pelos usuários o outros sistemas que integram com a aplicação.
2. **Camada de Serviço:** camada responsável por encapsular toda a lógica negocial do sistema, aplicando regras e manipulação dos dados.
3. **Camada de Domínio:** camada que encapsula as informações do negócio da aplicação, realizando o mapeamento objeto-relacional das tabelas com a utilização da especificação JPA.
4. **Camada de Persistência:** camada responsável por encapsular o acesso ao banco de dados da aplicação.

5.1 Visão de Implementação



5.2 Organização dos Componentes

No diagrama abaixo é apresentado os componentes existentes na arquitetura proposta.



1. **PJe-core**: contém os artefatos da base arquitetural, que provê uma padronização de desenvolvimento da aplicação.
2. **PJe-entidades**: contém o mapeamento de entidades que referenciam as tabelas físicas do banco de dados, com a utilização da especificação JPA.
3. **PJe-seguranca-core**: contém os filtros e interceptadores para controle da autenticação e autorização da aplicação.
4. **PJe-comum-negocio**: contém a implementação das regras negociais da aplicação e o acesso ao banco de dados.
5. **PJe-comum-api**: contém as interfaces dos *endpoints* no formato REST. É a primeira camada da aplicação que recebe as requisições solicitadas pelos usuários.

6 Mecanismos Arquiteturais

6.1 Política de Criação de APIs

As APIs devem ser implementadas utilizando o conceito RESTful, que utiliza os recursos providos pelo HTTP, quanto ao tipo de operações, códigos de retorno, definição de cabeçalhos e outras informações disponíveis no protocolo.

Exemplo de APIs utilizando o conceito RESTful:

Verbo HTTP	API	Descrição
GET	/processos/{idProcesso}	Recupera os metadados de um processo contendo informações básicas.
GET	/processos/{idProcesso}/documentos/	Recupera os documentos de um processo.
GET	/processos/{idProcesso}/documentos/{idDocumento}/conteudo	Recupera o conteúdo de um documento do processo.
DELETE	/processos/{idProcesso}/documentos/{idDocumento}	Excluir um documento do processo.
POST	/processos/{idProcesso}/movimentos/	Insere uma nova movimentação ao processo.
PUT	/processos/{idProcesso}/movimentos/{idMovimento}/visibilidade	Atualiza a visibilidade de uma movimentação do processo.

Verbos HTTP:

Verbo HTTP	Quando usar ?
GET	Recuperar informações.
POST	Inserir novas informações.
PUT	Atualizar informações.
DELETE	Excluir informações

6.2 Gerenciamento de Transações

O gerenciamento de transações será realizado com os recursos disponíveis na especificação EJB 3.2, que fornece mecanismos transacionais para gerenciamento de transações distribuídas entre diferentes base de dados. Com estes recursos, o servidor de aplicações JBoss EAP 7 gerenciará as transações da aplicação, fazendo com que o desenvolvimento priorize as regras negociais da aplicação, sem haver a necessidade de realizar o controle manual de transações no banco de dados.

Por padrão todos os serviços da camada negocial estão configurados em um contexto transacional. A tabela abaixo apresenta os tipos de transações suportadas na especificação EJB.

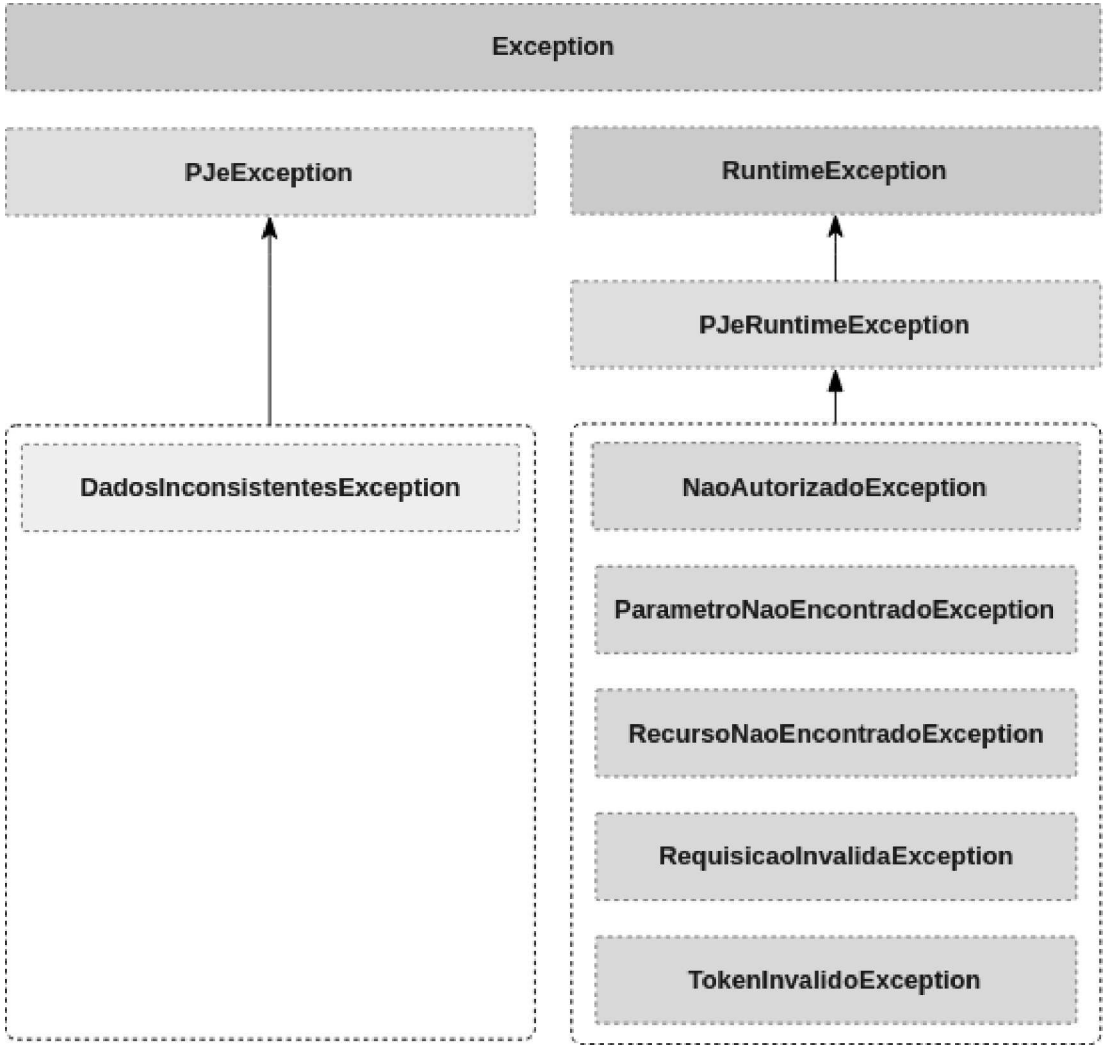
Tipo Transação	Transação existente na chamada ?	Efeito
REQUIRED	Não	Container cria uma nova transação.
	Sim	Método é incluído na transação já aberta.
REQUIRES_NEW	Não	Container cria uma nova transação.
	Sim	Container cria uma nova transação e a transação existente na chamada anterior é suspendida até o término desta.
SUPPORTS	Não	Nenhuma transação é criada.
	Sim	Método é incluído na transação já aberta.
MANDATORY	Não	Este tipo de transação determina que uma transação já deve ter sido aberta ao invocar método com este tipo. Caso não exista, é lançada uma exceção: "javax.ejb.EJBTransactionRequiredException".
	Sim	Método é incluído na transação já aberta.
NOT_SUPPORTED	Não	Nenhuma transação é criada.
	Sim	A transação existente anterior a chamada do método é suspendida até o término desta chamada.
NEVER	Não	Nenhuma transação é criada.
	Sim	Se houver uma transação já criada é lançada uma exceção: "javax.ejb.EJBException".

6.3 Tratamento de Exceções

Todas as exceções ocorridas na aplicação possuirão um identificador único para auxiliar na rastreabilidade do erro para a devida análise.

Apresentação de erros ao usuário: Toda mensagem de erro apresentado ao usuário deve ser completamente negocial, não sendo permitido apresentar qualquer detalhe técnico ou a pilha de erros da exceção ao usuário do sistema.

Abaixo encontra-se a hierarquia de exceções disponibilizadas na aplicação para indicar erros que podem ser tratados ou não.



Checked Exceptions são utilizadas para erros recuperáveis enquanto que **Unchecked Exceptions** são utilizadas para erros irrecuperáveis.

Exceções checadas, verificadas em tempo de compilação do PJe:

- **PjeException**: Exceção genérica do PJe.
- **DadosInconsistentesException**: Exceção que informa dados inconsistentes na base de dados.

Exceções não checadas, identificadas em tempo de execução do PJe:

- **PjeRuntimeException**: Exceção genérica do tipo RuntimeException.

- **ParametroNaoEncontradoException**: Exceção que indica a inexistência de um parâmetro obrigatório no sistema.
- **RecursoNaoEncontradoException**: Exceção que indica que recurso obrigatório solicitado não foi encontrado.
- **RequisicaoInvalidaException**: Exceção que indica que os parâmetros solicitados na requisição são inválidos, não foram preenchidos ou possuem valores incorretos.
- **NaoAutorizadoException**: Exceção que indica a não autorização de acesso a recursos protegidos pelo usuário solicitante.
- **TokenInvalidoException**: Exceção que indica que o token de segurança utilizado na requisição de informações está inválido, não permitindo a continuidade da requisição.

Todas as exceções que ocorrerem serão identificadas na camada mais superior da aplicação e retornarão o código de erro HTTP específico:

Exceção	Código Erros HTTP
PjeException	500
DadosInconsistentesException	500
PjeRuntimeException	500
ParametroNaoEncontradoException	404
RecursoNaoEncontradoException	404
RequisicaoInvalidaException	400
NaoAutorizadoException	401
TokenInvalidoException	400

6.4 Mecanismo de Acesso ao Banco de Dados

O banco de dados utilizado nesta arquitetura permanecerá sendo o Postgresql.

O acesso ao banco de dados se dará com uso do conceito ORM (Object Relational Mapper) através da especificação JPA e o provedor Hibernate, os quais provêem recursos para mapear as tabelas do banco de dados com o modelo de programação orientado a objetos. Além disso, auxiliam na execução de comandos no banco de dados abstraindo a criação de queries, aumento da produtividade, utilização de recursos de cache e a criação de queries no formato SQL ANSI para manter a compatibilidade com outros fabricantes de banco de dados, facilitando uma futura mudança, caso necessário.

O controle e abertura de conexões serão gerenciados pelo servidor de aplicações JBoss EAP 7, que será o responsável por prover as conexões para as requisições solicitadas pelos usuários do sistema.

6.5 Comparativo entre o uso de Triggers e Observers para movimentações do JBPM

O relatório abaixo tem por objetivo expor os pontos negativos e positivos que foram levantados em relação ao uso de métodos observers e triggers de banco. Este levantamento se deu durante a proposta de um novo modelo de dados para fazer a contagem de processos em agrupamentos que está sendo realizada pela equipe de desenvolvimento do Tribunal Regional do Trabalho da 12ª Região e CTPJe. Relatório Técnico Triggers e Observers

7 Formas de comunicação

São as formas permitidas para envio e recebimento de dados entre o PJe 2 e os demais sistemas que ele interage (módulos, sistemas satélites, PJe legado...). Destaca-se que esse item não trata de acesso à base de dados. Para isso, veja o item Termos e Abreviações.

7.1 Entre o PJe legado e o PJe 2

1. Para comunicação síncrona; (APIs REST ou JMS com Request/Reply)
2. Para comunicação assíncrona; (JMS - Leitura/Escrita dos 2 lados)

Obs.: PJe Legado e PJe Backend Comum podem acessar a mesma base de dados.

Obs. 2: A comunicação entre PJe 2 e PJe Legado dar-se-á apenas no sentido PJe Legado -> PJe 2. Não será permitido, sob qualquer hipótese, acesso do PJe 2 aos serviços do PJe Legado.

7.2 Entre o PJe 2 e os módulos

1. Para comunicação síncrona; (APIs REST)
2. Para comunicação assíncrona; (JMS - Leitura/Escrita dos 2 lados)

7.3 Entre o PJe 2 e os Sistemas Satélites

1. Para comunicação síncrona; (APIs REST)
2. Para comunicação assíncrona:
    1. APIs REST para requisições oriundas do Sistema Satélite;
    2. "JMS - Leitura do lado do sistema satélite", consumindo informações incluídas pelo PJe em uma Fila ou Tópico. O sistema satélite terá permissão somente de leitura. (Apenas o PJe realizará a escrita em Filas ou Tópicos).
3. Extração de dados (ETL):
    1. Para os sistemas classificados como satélites, podemos criar a nível de infraestrutura réplica da réplica, possibilitando assim que estes sistemas consultem informações mais atuais.

Obs.: A comunicação entre sistemas satélites e o PJe 2 será feita com o uso do PJe Integração. Desta forma, qualquer solicitação do satélite para o PJe 2 será feita através deste módulo.

8 Módulo de Batch Jobs

O PJe 2, a partir da versão 2.3, provê suporte a Batch Jobs no Backend Comum, com base em:

- **EJB Timers** - provido pelo Java EE 7;
- **JSR 352 Batch Jobs**, provido pelo Java EE 7 (com nome JBeret no JBoss 7);
- biblioteca **pje-core-jobs**: abstração de diversas atividades comuns de agendamento e execução de batch jobs;
- módulo **pje-comum-jobs**: módulo contendo os jobs do backend comum.

A biblioteca **pje-core-jobs** também suporta a confecção de batch jobs similares, em módulos fora do backend comum.

Descrevemos na página Arquitetura dos Jobs do PJe 2 maiores detalhes.

9 Módulos - Boas práticas

9.1 Consumo de serviços Rest

A classe utilizada para consumir serviços Rest é a ServicoRestInvokerBuilder. Essa classe já adiciona as informações básicas de autenticação, tratamento de erros e rastreamento das requisições.

9.2 Dependência de entidades de outros módulo

As entidades do Módulo que façam referências a outra entidades externas ao do Módulo em questão, devem fazer referências somente aos identificadores das chaves primárias. As informações das outras tabelas devem ser obtidas via APIs Rest. É possível demandar a área de Administração de Dados do CSJT a criação de Views relacionadas a tabelas de schemas diferentes do módulo em questão. A criação da View terá sua viabilidade analisada pela área citada.

9.3 Acessar parâmetros do sistema

O módulo do PJe que precisar utilizar parâmetros, deve realizar essa chamada na API existente no Backend Comum: http://portal.pje.redejt/arquitetura/api-docs/?url=http://portal.pje.redejt/arquitetura/api-docs/pje-comum-api.json#/parametros/buscarParametroPeloNome

No caso para identificar qual a localização do módulo Comum, sugere-se até o momento o uso de um parâmetro no Jboss, visto que ainda não temos algum Service Discovery ou API Management. Exemplo parâmetro: (-Dpje.comum.url=....).

9.4 DTO

Se o módulo precisar obter informações de entidades de outros módulos e precisa ter um DTO para receber os dados de outro módulo, a implementação atual permite que o DTO seja duplicado em relação ao outro módulo.

9.5 Token de segurança

A passagem de token na comunicação entre APIs é realizada automaticamente, veja a implementação da classe ServicoRestInvokerPje: (br.jus.csjt.pje.core.util.ServicoRestInvokerPje.configurarComunicacaoInterna(ServicoRestInvokerBuilder, Builder))

```
?
1 /**
2  * Realiza a configuração básica em comunicações internas entre os módulos do PJe.
3  *
4  * @param builder
5  * @param request
6  */
7 private void configurarComunicacaoInterna(ServicoRestInvokerBuilder builder, Builder request) {
8     if (!builder.isComunicacaoExterna()) {
9         request.header("X-Request-ID", DadosContextoRequisicao.getInstance().getIdentificadorRequisicao());
10
11        String accessToken = builder.getAccessToken();
12        if ((accessToken == null) && (DadosContextoRequisicao.getInstance().getTokenAutenticacao() != null)) {
13            accessToken = DadosContextoRequisicao.getInstance().getTokenAutenticacao().getAccessToken();
14        }
15        if (accessToken != null) {
16            request.header(HttpHeaders.AUTHORIZATION, "Bearer " + accessToken);
17        }
18    }
19}
```

No caso de a comunicação com APIs que aceitem somente Tokens de Serviço, é possível atribuir o token através de: (br.jus.csjt.pje.core.util.ServicoRestInvokerBuilder.setAccessToken(String))

Como obter um token de serviço:

br.jus.csjt.pje.seguranca.core.util.ComunicacaoSegurancaUtil.obterTokenServico(String, String, GrauInstanciaEnum)

9.6 Dependência do módulo de segurança SJT

Os módulos negociais dependem do módulo de segurança (SJT) através da dependência com a biblioteca (pje-seguranca-core).

10 Referências

10.1 Back-end

1. JPA 2.1 (https://jcp.org/en/jsr/detail?id=338)
2. EJB 3.2 (http://download.oracle.com/otndocs/jcp/ejb-3_2-fr-spec/index.html)
3. Java Exceptions (https://docs.oracle.com/javase/tutorial/essential/exceptions/)
4. Java Runtime Exceptions (https://docs.oracle.com/javase/tutorial/essential/exceptions/runtime.html)
5. REST Wikipedia (https://pt.wikipedia.org/wiki/REST)
6. RESTful (https://becode.com.br/o-que-e-api-rest-e-restful/)
7. RESTful Tutorial (http://www.restapitutorial.com/lessons/whatisrest.html)

10.2 Segurança

1. Documentação da Autenticação PJe 2 (https://git.pje.csjt.jus.br/pje2/pje-seguranca/wikis/home)
2. JWT (https://jwt.io/)

10.3 Front-end

1. Angular Material 2 (https://github.com/angular/material2)
2. Acessibilidade - Angular Material 2 (https://material.io/guidelines/usability/accessibility.html#accessibility-writing)
3. Consulta Gartner - Frameworks Front-end (https://git.pje.csjt.jus.br/pje2/pje-docs/blob/master/Proposta-Arquitetural/Documentacao/CSJT_WrittenResponse_Framwork%20JavaScript_23mar17.pdf)
4. Definição: Angular 2 (https://git.pje.csjt.jus.br/pje2/pje-frontend/wikis/escolha-angular)
5. Definição: Angular Material 2 (https://git.pje.csjt.jus.br/pje2/pje-frontend/wikis/escolha-angular-material)
6. Análise biblioteca: MaterializeCSS (https://git.pje.csjt.jus.br/pje2/pje-docs/blob/master/Proposta-Arquitetural/Documentacao/Analise_MaterializeCSS_CTPJE4428.pdf)
7. Análise biblioteca: UI-Bootstrap (https://git.pje.csjt.jus.br/pje2/pje-docs/blob/master/Proposta-Arquitetural/Documentacao/Analise_UIBootstrap_CTPJE4439.pdf)

8. Análise biblioteca: NG-Bootstrap vs PrimeNG (https://git.pje.csjt.jus.br/pje2/pje-docs/blob/master/Proposta-Arquitetural/Documentacao/ComparacaoBootstrapVsPrimeNG_CTPJE-4948.pdf)
9. PrimeNG PRO (https://www.primefaces.org/primeng/#/)

Disponível em "https://pje.csjt.jus.br/documentacao/index.php?title=Arquitetura_do_PJe_2&oldid=13677"

- Esta página foi modificada pela última vez em 30 de outubro de 2018, às 17h31min