Manual de operação de sistema - Versão 1

1. Introdução

- **Objetivo:**
  Este manual técnico tem como propósito fornecer uma visão detalhada da aplicação Sniper BC, incluindo sua arquitetura, componentes backend e frontend, fluxos de dados, integrações externas e configurações essenciais.

  O conteúdo técnico aqui descrito é voltado a desenvolvedores, arquitetos de software e DevOps responsáveis por realizar futuras manutenções corretivas, evolutivas ou integrações futuras com outros sistemas.

  O objetivo é garantir que qualquer membro da equipe técnica tenha os insumos necessários para compreender, operar e evoluir o sistema com segurança, eficiência e alinhamento com as boas práticas adotadas no projeto.

- **Visão geral do sistema:**
  O Sistema Nacional de Investigação Patrimonial e Recuperação de Ativos (Sniper) tem como objetivo permitir a investigação patrimonial e a recuperação de ativos de forma ágil e eficiente, por meio do cruzamento de dados oriundos de diferentes bases (abertas e sigilosas), consolidando as informações em uma interface visual intuitiva e rápida de usar.

  O sistema permite a identificação de vínculos e relações de interesse processual, além de possibilitar a consulta, análise e exportação de relatórios em poucos cliques. Com isso, pretende-se reduzir o tempo necessário às ações de execução e cumprimento de sentença, aprimorar a efetividade da prestação jurisdicional e fortalecer a prevenção e o combate à corrupção e à lavagem de dinheiro no âmbito do Poder Judiciário.

- **Público-alvo do sistema:**
  Magistrados, magistradas, servidores e servidoras dos tribunais integrados à PDPJ. O sistema verifica, no momento da autenticação via SSO, se a pessoa possui um perfil válido e, caso contrário, bloqueia o acesso, exibindo uma mensagem clara sobre a restrição.

- **Escopo:**
  Do ponto de vista dos requisitos não funcionais, o Sniper é uma aplicação Web desenvolvida em Python (Django) no backend e Angular 19 no frontend, implantada no ambiente Kubernetes do CNJ, com segregação entre homologação e produção. A plataforma foi projetada para garantir alto desempenho e baixo tempo de resposta. A autenticação é baseada em OpenID e utiliza tokens JWT, além de registro completo de logs e auditoria. A infraestrutura contempla mecanismos de cache para otimizar consultas às APIs externas, bem como monitoramento contínuo por meio de endpoints de health check.

  Comparando com a versão anterior do Sniper, a nova versão (Sniper BC) prevê uma mudança substancial na interface. Porém, a arquitetura do Sniper BC se mantém na mesma estrutura da da versão anterior: o usuário acessa o frontend, que faz requisições por meio de HTTPS + JSON ao backend. O backend, por sua vez, acessa dois bancos de dados (relacional e de grafos) e APIs do ecossistema PDPJ. Os coletores de bases de dados externas alimentam os dois bancos de dados com informações atualizadas, para que o backend então possa acessá-las. Assim, as mudanças esperadas para as entregas do Sniper BC compreendem a reestruturação do código do frontend (para se adequar ao novo fluxo de uso), integração com novas APIs pelo backend e criação de novos endpoints para atender às novas funcionalidades.

  Essa versão do Sniper acessa novas APIs externas e algumas delas retornam os dados de maneira assíncrona. O backend usa polling para recuperar os dados quando as requisições são finalizadas e, para controlar o acesso, é utilizado um processo worker que executa essas tarefas em segundo plano.

  O processo de deployment da aplicação se mantém o mesmo, assim como as configurações de infraestrutura (exceto pelo novo processo worker). Apenas alterações em variáveis de ambiente serão necessárias para as configurações de acesso aos novos sistemas.

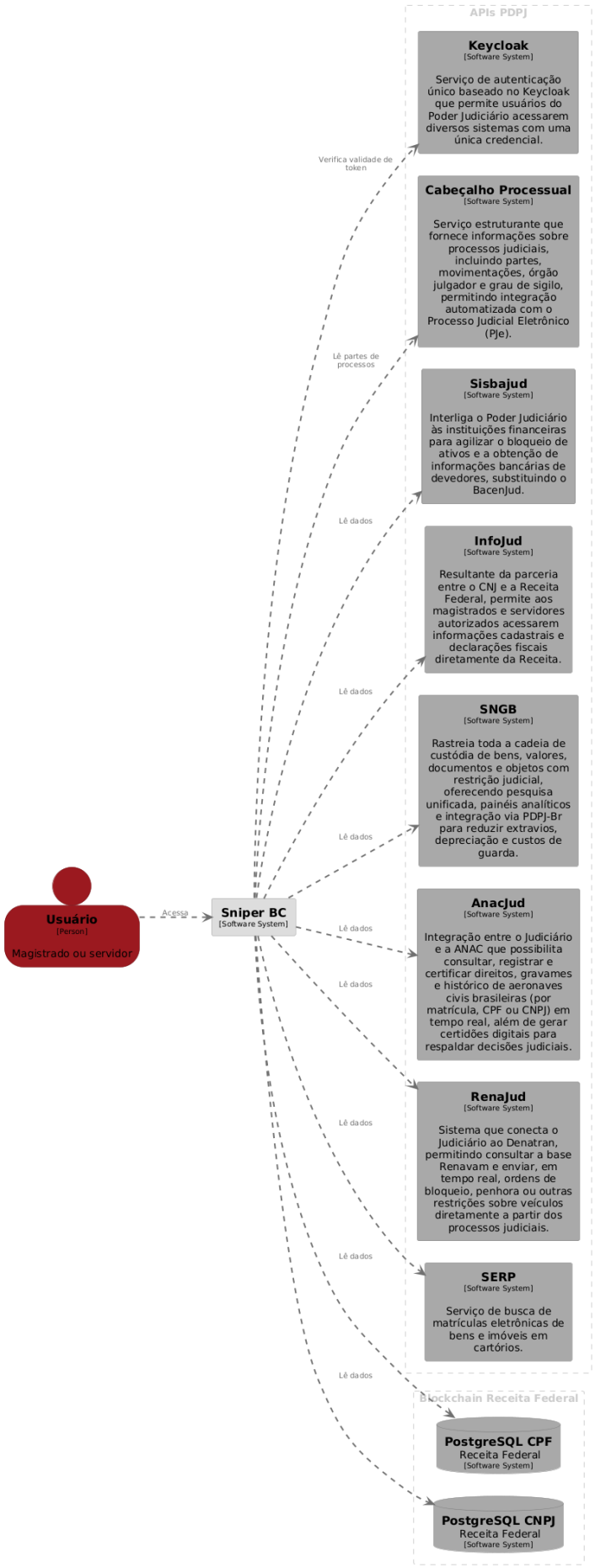
2. Documentação técnica

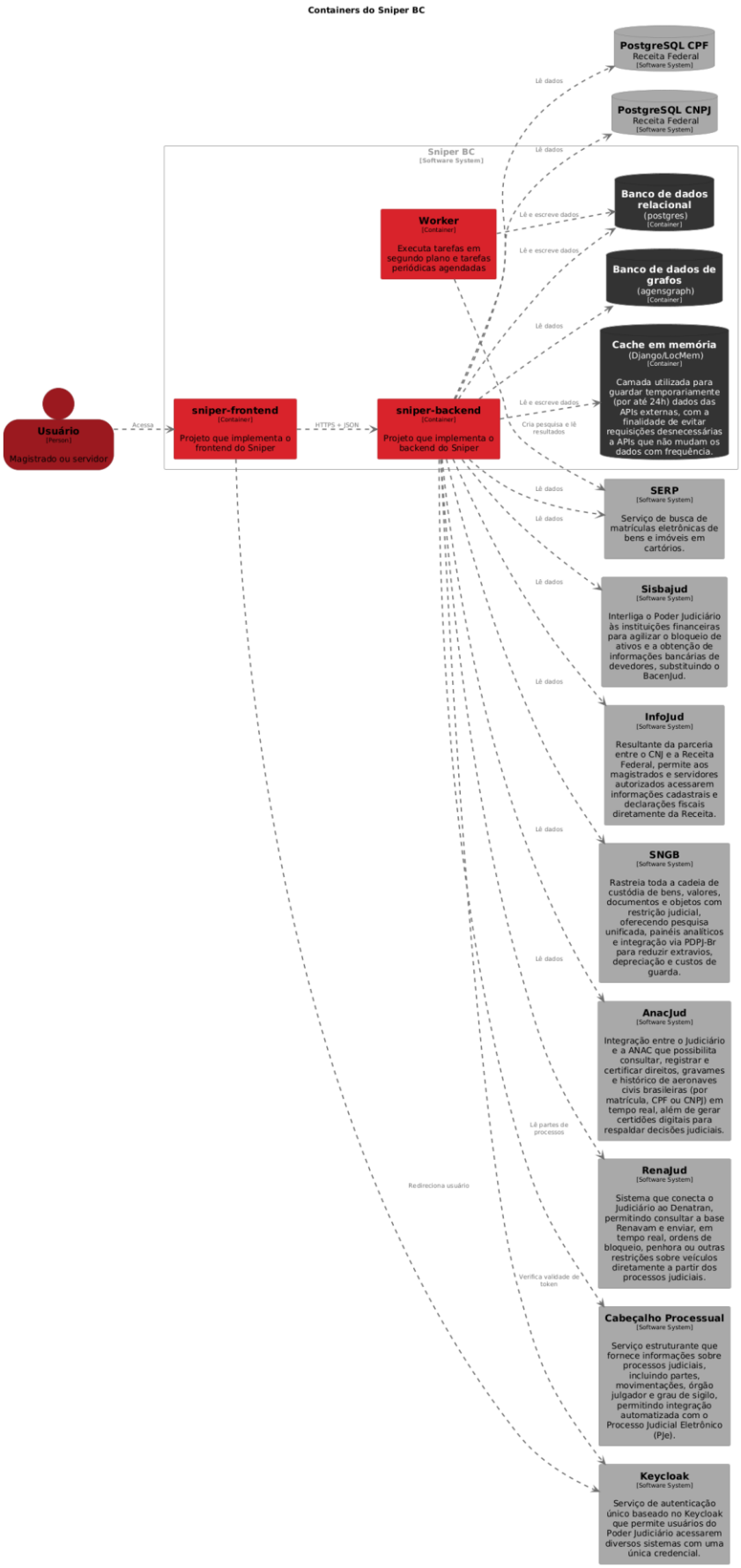
- **Arquitetura:**
  Este documento apresenta a Arquitetura do Sistema seguindo as diretrizes do modelo C4, bem como alguns detalhes importantes de implementação dessa arquitetura. O arquivo que descreve a arquitetura do sistema foi criado usando a linguagem Structurizr DSL e está disponível no repositório sniper-backend no Gitlab do CNJ (atualmente, no branch epic/sniper-bc). A partir da descrição estruturada do sistema foram geradas visualizações com 3 níveis de detalhes diferentes: contexto do sistema, container e componente.

  A arquitetura exibida nas próximas seções está dividida em 3 níveis diferentes de detalhamento:
  - Contexto do sistema, onde o Sniper BC é apresentado como uma caixa-preta interagindo com outros sistemas da PDPJ.
  - Containers do Sniper BC, onde são exibidas as peças implantáveis (deployable parts) que compõe o sistema (frontend, backend e bancos de dados) e suas interações com o exterior do sistema.
  - Componentes do backend, onde são exibidos os subsistemas implementados e como eles interagem entre si e com o mundo externo.

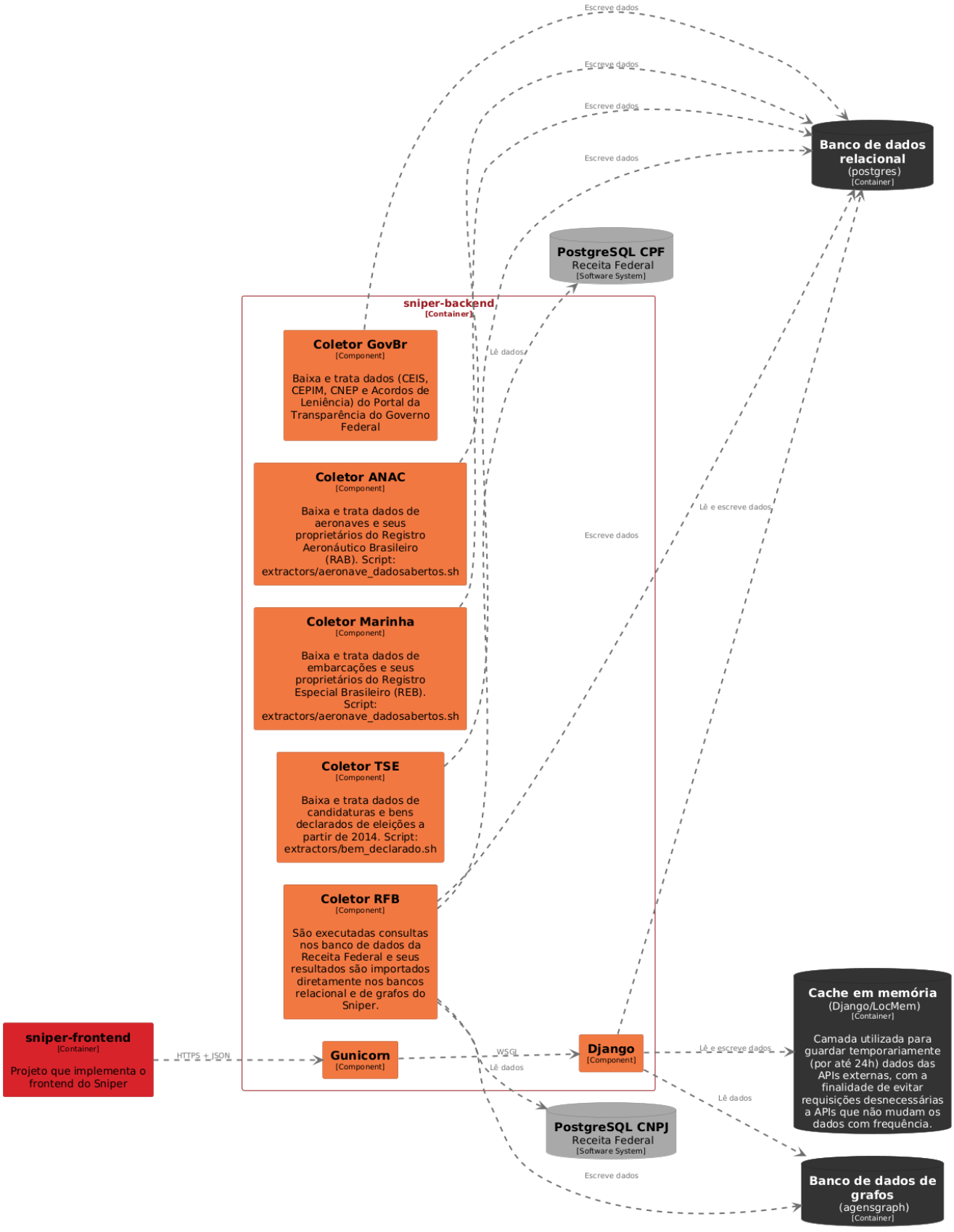
  As imagens dos diagramas gerados foram reduzidas para que caibam nesse documento e estão disponíveis em alta resolução no repositório sniper-backend na pasta docs/imgs/.

Contexto do Sistema




Containers do Sniper BC
PostgreSQL CPF
Receita Federal
[Software System]
PostgreSQL CNPJ
Receita Federal
[Software System]
Sniper BC
[Software System]
Worker
[Container]
Executa tarefas em segundo plano e tarefas periódicas agendadas
Banco de dados relacional
(postgres)
[Container]
Banco de dados de grafos
(agensgraph)
[Container]
Cache em memória
(Django/LocMem)
[Container]
Camada utilizada para guardar temporariamente (por até 24h) dados das APIs externas, com a finalidade de evitar requisições desnecessárias a APIs que não mudam os dados com frequência.
Usuário
[Person]
Magistrado ou servidor
sniper-frontend
[Container]
Projeto que implementa o frontend do Sniper
sniper-backend
[Container]
Projeto que implementa o backend do Sniper
Acessa
HTTPS + JSON
Lê dados
Lê e escreve dados
Cria pesquisa e lê resultados
SERP
[Software System]
Serviço de busca de matrículas eletrônicas de bens e imóveis em cartórios.
Sisbajud
[Software System]
Interliga o Poder Judiciário às instituições financeiras para agilizar o bloqueio de ativos e a obtenção de informações bancárias de devedores, substituindo o BacenJud.
InfoJud
[Software System]
Resultante da parceria entre o CNJ e a Receita Federal, permite aos magistrados e servidores autorizados acessarem informações cadastrais e declarações fiscais diretamente da Receita.
SNGB
[Software System]
Rastreia toda a cadeia de custódia de bens, valores, documentos e objetos com restrição judicial, oferecendo pesquisa unificada, painéis analíticos e integração via PDPJ-Br para reduzir extravios, depreciação e custos de guarda.
AnacJud
[Software System]
Integração entre o Judiciário e a ANAC que possibilita consultar, registrar e certificar direitos, gravames e histórico de aeronaves civis brasileiras (por matrícula, CPF ou CNPJ) em tempo real, além de gerar certidões digitais para respaldar decisões judiciais.
Lê partes de processos
RenaJud
[Software System]
Sistema que conecta o Judiciário ao Denatran, permitindo consultar a base Renavam e enviar, em tempo real, ordens de bloqueio, penhora ou outras restrições sobre veículos diretamente a partir dos processos judiciais.
Redireciona usuário
Verifica validade de token
Cabeçalho Processual
[Software System]
Serviço estruturante que fornece informações sobre processos judiciais, incluindo partes, movimentações, órgão julgador e grau de sigilo, permitindo integração automatizada com o Processo Judicial Eletrônico (PJe).
Keycloak
[Software System]
Serviço de autenticação único baseado no Keycloak que permite usuários do Poder Judiciário acessarem diversos sistemas com uma única credencial.

Componentes do Backend



Código-fonte:

- **Repositório de código:**
  Por conta de uma decisão negocial, a versão mais recente do projeto só entrará em produção após a entrega do Produto 8, como o presente documento detalha a entrega do Produto 5, adotamos o uso do branch **epic/sniper-bc** como sendo o que possui o código entregável.

  Backend: https://git.cnj.jus.br/pdpj/negocio/sniper-backend
  Frontend: https://git.cnj.jus.br/pdpj/frontends/sniper-frontend

- **Documentação de APIs:**
  A nova versão da API (/v2) implementa o OpenAPI/Swagger e está disponível em ambiente de homologação por meio do endereço:
  https://sniper-api.stg.pdpj.jus.br/v2/docs/

- **Tecnologias:**
  Frontend: Angular 19 e biblioteca de componentes Angular-Material, com interface inspirada no UIKit.
  Backend: Python com Django e procrastinate para a gestão de tarefas em segundo plano.
  Bancos de dados: postgres (AWS RDS) e AgensGraph (self-hosted na conta AWS do CNJ).
  Deployment: no cluster Kubernetes, por meio do CI/CD do Gitlab do CNJ.

Detalhes de Implementação:

- **Segurança:**
  O sistema foi implementado na infraestrutura Kubernetes do CNJ e, por isso, estratégias de mitigação de ataques e rate limiting não foram implementadas diretamente pelo Sniper, pois ficam a cargo da equipe que mantém o ingress. Também não é de responsabilidade do Sniper a implementação de SSL entre o backend e o navegador do usuário, dado que o ingress também faz a gerência desse aspecto.

  A autenticação no Sniper é feita utilizando OpenID por meio de integração com o SSO da PDPJ, implementado pelo Keycloak. Ao efetuar a autenticação no SSO, o backend do Sniper verifica se o token JWT é válido e se o usuário possui o perfil

adequado para acesso ao sistema (nem todos que possuem conta no SSO da PDPJ podem ter acesso ao Sniper).

- **Cache:**
  Dado que os requisitos de cache são pequenos, optou-se por armazenar os dados em um cache em memória, utilizando a implementação de cache LRU em memória do Django (LocMemCache). Essa solução simplifica a configuração, diminuindo a necessidade de infraestrutura extra. Para cada processo em execução do Django (backend do Sniper), um dicionário Python é alocado para armazenar o cache. A implementação atual suporta um número máximo de 131.072 (128k) entradas, que expiram (TTL) em 24 horas. As configurações dessa implementação podem ser alteradas no arquivo project/settings.py.

  São armazenados em cache:
  - Valores referentes a variáveis categóricas disponíveis no banco de dados e/ou em APIs externas (mapeamentos de "id" para "nome").
  - Resultado de algumas consultas SQL recorrentes, de dados que não irão mudar em 24 horas.
  - Resultado do acesso a algumas APIs externas (por usuário que demandou o acesso) que não mudam com frequência.

- **Integração com APIs Externas:**
  A integração com APIs externas é feita por meio de REST (HTTPS + JSON). A forma de autenticação em cada API se dá por uma de duas formas, dependendo de como foi disponibilizada:
  - Acesso utilizando o token JWT do próprio usuário (gerado pelo SSO da PDPJ): Sisbajud, InfoJud, AnacJud, RenaJud, SNGB e SERP.
  - Acesso via token de sistema, criado especificamente para o Sniper: SSO (Keycloak) e Cabeçalho Processual.

  Para evitar o acesso desnecessário às APIs, alguns endpoints foram selecionados para cache -- o cache é sempre feito no escopo do usuário que está acessando o dado, para garantir o mesmo permissionamento na API de origem.

  Em todos os acessos às APIs externas é implementada uma política de retry e timeout e, caso a informação não possa ser acessada no momento, o usuário é notificado por meio do frontend de que um erro aconteceu.

- **Gestão de Configurações:**
  As configurações do backend estão definidas no arquivo project/settings.py, onde parte considerável delas é lida de variáveis de ambiente, já outras são definidas diretamente no código. As variáveis de ambiente são configuradas de duas formas:
  - Dados não sensíveis são configurados por meio de configmaps do Kubernetes, nos arquivos kubernetes/eks-hml-01/configmaps.yml (homologação) e kubernetes/eks-prd-01/configmaps.yml (produção).
  - Dados sensíveis são configurados por meio de secrets do Kubernetes, adicionados diretamente por meio do Rancher (a equipe da Pythonic Café não possui acesso) e lidos pelos arquivos kubernetes/eks-hml-01/deployment-django.yml (homologação) e kubernetes/eks-prd-01/deployment-django.yml (produção).

- **Observabilidade:**
  O backend do Sniper pode ser monitorado de três formas diferentes: por meio de logs do servidor HTTP, por meio de um endpoint específico para verificar a saúde do sistema e por emails de erro enviados à equipe de operação.

  O servidor HTTP que executa o Django foi configurado para enviar os logs ao stdout do processo principal do servidor HTTP (gunicorn), que é por fim coletado pelo Kubernetes. Porém, a equipe de desenvolvimento da Pythonic Café não possui acesso a esses logs, dado que o acesso ao Rancher (que gerencia os clusters Kubernetes) não foi concedido, e a integração do Gitlab para logs do Kubernetes não está configurada. A equipe do DTI/CNJ foi notificada e nos comunicaram que estão trabalhando para que a integração volte a funcionar. As equipes de operações do DTI e de DevOps do PNUD possuem acesso ao Rancher e, por consequência, aos logs, e poderão ser acionadas caso necessário.

  A pedido do DTI, foi implementado o endpoint /healthz/, que retorna métricas da saúde do sistema. O backend responderá se consegue se conectar e executar consultas nas bases relacional e de grafos, e se consegue acessar cada uma das APIs externas usadas no Sniper. A equipe do DTI configurou o sistema Zabbix para acessar esse endpoint de tempos em tempos e, por fim, ter métricas da saúde do sistema periodicamente. O acesso ao health check de homologação pode ser feito por meio do seguinte endereço: https://sniper-api.stg.pdpj.jus.br/healthz/.

  O Django foi configurado para que um email seja enviado à equipe de operações quando acontecerem exceções no ambiente de produção. Esses erros são enviados para o endereço operacao-sniper@cnj.jus.br, que encaminha para a

equipe do DTI e da Pythonic Café. Como sugestão de melhoria, gostaríamos de implementar o monitoramento de erros via Sentry ou ferramenta similar, mas essa questão ainda está em análise pela equipe do CNJ, dado que requer infraestrutura mais robusta para instalação, configuração e posterior operação, e está fora do escopo do Sniper BC.

Testes:

- **Introdução:**
  Este relatório tem como propósito apresentar, de forma clara e objetiva, os casos de teste mapeados para os requisitos da entrega ao CNJ, bem como os resultados obtidos durante a execução dos testes no Projeto Sniper BC. O documento visa fornecer uma visão consolidada da qualidade do software testado e apoiar a tomada de decisão quanto à sua aprovação.

- **Escopo:**
  Foram executados testes referentes às histórias de usuário descritas nos Produtos 1 e 3, que contemplam o novo fluxo de uso do sistema e acesso a novas bases de dados. Foram executados dois tipos de testes:
  - **Automatizados**, que são compostos por testes de unidade e testes mais abrangentes, executados no contexto do repositório sniper-backend. Esses foram executados em ambiente de desenvolvimento e de homologação e estão configurados no Gitlab CI para serem executados automaticamente durante o processo de deployment, que só é concluído se 100% desses testes forem executados com sucesso.
  - **Manuais**, onde usamos o ambiente de desenvolvimento local e também o de homologação do Sniper. Para esses, geramos dados fictícios nos bancos de dados, de forma a conseguirmos avaliar todas as funcionalidades.

- **Nota Técnica:**
  Para assegurar a eficácia dos testes, é fundamental que os sistemas externos acessados em ambiente de homologação possuam dados compatíveis e sincronizados com os dados existentes no próprio Sniper de homologação. Isso garante que os testes dos fluxos e integrações ocorram de forma correta e completa.

  Por exemplo, ao iniciar uma investigação no Sniper, o usuário informa um número de processo. O backend consulta a API do Cabeçalho Processual para recuperar as partes do processo, que, por sua vez, são cruzadas com dados cadastrais da

Receita Federal, armazenados localmente no banco do Sniper. Caso uma parte existente na API do Cabeçalho Processual não conste na base de dados do Sniper, o teste não poderá ser executado com sucesso. Da mesma forma, é necessário que dados sobre essas partes existam nos outros sistemas que o Sniper acessa para consultar dados de bens, como Sisbajud, AnacJud, Renajud, SNGB, dentre outros, caso contrário a pessoa que estará executando os testes não conseguirá visualizar os dados para essas partes.

Para mitigar este problema, recomendamos à equipe do CNJ que os ambientes de homologação dos sistemas da PDPJ compartilhem um conjunto comum de dados. É necessário que as equipes definam em conjunto uma lista de números de processos, CPFs, CNPJs e demais identificadores que deverão estar presentes em todos os bancos de homologação, permitindo assim a execução de testes completos e realistas.

Atualmente, as equipes responsáveis por APIs como AnacJud, Renajud, SNGB, Sisbajud e outras já disponibilizam listas de CPFs e CNPJs presentes em seus ambientes de homologação. Entretanto, nem sempre esses dados estão refletidos como partes em processos disponíveis na API do Cabeçalho Processual de homologação. Diante dessa limitação, foi implementado um mecanismo no Sniper que permite testar a integração com os demais sistemas mesmo na ausência desses dados no Cabeçalho Processual. Sempre que o número de processo informado for 0000000-00.0000.0.00.0000, o sistema retorna uma lista de partes fictícias, representando os CPFs e CNPJs disponíveis nas outras APIs de homologação. Dessa forma, os testes de integração podem ser realizados normalmente (exceto a integração específica com o Cabeçalho Processual para esse número de processo).

O único número de processo que "burla" a integração com o Cabeçalho Processual é o descrito acima, de forma que para testar a integração do Sniper com o Cabeçalho Processual - já implementada em versão anterior do sistema -, basta utilizar qualquer número de processo disponível no ambiente de homologação do Cabeçalho Processual, como 0011871-18.2023.8.26.0100.

Nota: os dados de bens declarados (TSE), embarcações (Tribunal Marítimo) e sanções (CGU) são armazenados diretamente no banco de dados do Sniper e a própria equipe que desenvolve o sistema adicionou dados fictícios no ambiente de homologação para testes.

- **Testes Manuais:**
  Os testes manuais de aceitação foram conduzidos no ambiente de homologação (https://sniper.stg.pdpj.jus.br/), onde o Sniper se conecta às APIs externas também em ambiente de homologação. Durante o desenvolvimento, testes exploratórios adicionais também foram feitos em ambiente de desenvolvimento local com o serviço fakeapi (simulador das APIs externas).

  Para cada **história de usuário (HU)** definida nos Produtos 1 e 3, percorremos o fluxo completo no sistema, validando os critérios de aceitação e capturando screenshots como comprovação da validação. Visando facilitar a comunicação sobre esse documento, os casos de teste usam a nomenclatura "CT-XXX" e testam funcionalidades específicas, muitas vezes compreendendo uma ou mais histórias de usuário. Recomendamos a leitura desse documento em conjunto com o documento de Levantamento de Requisitos, que contém as histórias de usuário.

- **Testes Automatizados:**
  Para a execução dos testes automatizados, uma **base de dados de testes** é criada; todas as migrações da aplicação são executadas e dados fictícios são inseridos, garantindo que o ambiente de testes simule cenários reais de uso do sistema. Os testes são executados em uma base de dados real (sem mocks), com exatamente o mesmo schema que existirá nos ambientes de homologação e produção. Ao final da execução, a base é destruída.

  O projeto conta com um total de **344 testes automatizados**, que cobrem **82% das linhas de código**, conforme demonstrado nos artefatos gerados pelo Gitlab CI após a execução dos testes. Os artefatos gerados são:
  - - coverage.json: dados estruturados sobre cada teste executado.
  - - test-coverage/: pasta contendo um relatório em HTML das linhas cobertas por arquivo, sendo possível ver o código-fonte, dentre outros detalhes - inicie a navegação pelo arquivo index.html.

Configuração:

- **Descrição da Instalação:**
  A instalação do sistema nos ambientes de homologação e produção se dá por meio de uma pipeline no Gitlab CI, com as seguintes regras:
  - Os commits do branch develop são encaminhados ao ambiente de homologação.
  - Os commits do branch main são encaminhados ao ambiente de produção.

Depois de testados em ambiente de homologação e aprovados pela equipe do CNJ, executamos um merge do conteúdo de develop em main, de maneira que o branch que estava instalado em homologação seja promovido para produção. Porém, o Sniper BC entrará em produção apenas a partir do Produto 8. Para evitar possíveis confusões, estamos publicando também em homologação os commits do branch epic/sniper-bc (que receberá as implementações de todos os produtos até o 8). Quando, finalmente, o Produto 8 for aprovado e implementado, o branch epic/sniper-bc será integrado ao develop e, depois de todos os testes feitos em homologação, faremos a integração ao main para que o sistema entre em produção. Essa distinção evita que códigos sejam encaminhados ao ambiente de produção antes da data de lançamento (mesmo que já testados) e, ao mesmo tempo, não deixa de seguir as recomendações do git-flow (de que os commits em develop já estão preparados para serem lançados em produção, aguardando apenas o merge no main).

Antes do processo de instalação são feitas diversas verificações, implementadas no .gitlab-ci.yml. O processo de instalação só é iniciado se as seguintes verificações forem executadas com sucesso:

- Teste das configurações do Kubernetes para o ambiente/cluster de destino do commit (eks-hml-01 ou eks-prd-01);
- Construção do container principal da aplicação em modo de testes (instala pacotes extras, como o test runner);
- Execução de todos os testes automatizados;
- Construção do container principal da aplicação (agora com o modo de testes desativado).

- **Impacto no Sistema:**
  Com relação ao backend, não houve quebra de compatibilidade com a API anterior (/v1). Com relação ao frontend, como o fluxo do usuário muda completamente, inclusive criando novos conceitos (como o uso do termo "investigação"), a equipe decidiu criar uma estrutura nova de URLs e as antigas deixarão de funcionar.

- **Impacto no Sistema:**
  Antes da instalação em homologação, foram executados testes manuais locais, por meio da geração de dados fictícios e da execução dos diversos componentes que são necessários à plataforma, como bancos de dados (relacional e de grafos), Keycloak etc. (esses serviços podem ser vistos no compose.yml dos repositórios).

Também foram executados centenas de testes automatizados em ambiente local. Os testes automatizados também são executados no Gitlab CI e impedem que o deployment seja feito caso exista alguma falha).

Após a instalação, foram coletados os commit IDs (SHA1) nos endereços https://sniper-api.stg.pdpj.jus.br/version (backend) e https://sniper.stg.pdpj.jus.br/version.txt (frontend). Esses endpoints não necessitam de autenticação e retornam o commit ID (SHA1) em execução no momento, comprovando que o sistema foi instalado com sucesso e está em execução.

Para essa entrega, referente ao Produto 4, os seguintes commit IDs foram verificados: d8a40dbcdd7063320bce503290f06f161c8217ce (backend) e 253fdd3bac3a232290b1540d40898ca4b3f3d299 (frontend).

Por fim, os testes manuais foram executados no ambiente de homologação. Os casos de teste (automatizados e manuais), bem como o resultado da execução dos testes manuais estão detalhados no documento "Relatório de testes", entregue em conjunto com esse.

- **Possíveis Problemas e Soluções:**
  A equipe considera que a disponibilidade de dados consistentes em ambiente de homologação é um ponto crítico para a validação da instalação não somente do Sniper, mas de todos os sistemas do ecossistema da PDPJ. Recomendamos ao CNJ que os ambientes de homologação dos sistemas da PDPJ compartilhem um conjunto comum de dados, que deverão estar presentes em todos os bancos de homologação. Dito isso, o maior desafio é a existência de dados compatíveis entre os sistemas: para demonstrar que o Sniper funciona corretamente em ambiente de homologação, as APIs externas acessadas (como Cabeçalho Processual e Sisbajud) precisam ter dados válidos e compatíveis com os dados na base de dados do Sniper. Para sanar esse problema, a equipe de desenvolvimento:
    - Solicitou a lista de números de processos e partes disponíveis no ambiente de homologação dos sistemas externos acessados pelo Sniper.
    - Executou chamadas na API do Cabeçalho Processual para listar as partes envolvidas nesses processos.
    - Gerou dados fictícios baseados nos CPFs e CNPJs encontrados nos passos anteriores.
    - Adicionou ao banco de dados do Sniper (bens declarados e sanções) os dados gerados.

- Desenvolveu código específico para o ambiente de homologação, que devolve partes fictícias se o processo 0000000-00.0000.0.00.0000 for acessado, permitindo que testes mais abrangentes sejam feitos.

Apesar do grande avanço em comparação com o Produto 2, ainda existe falta de dados para as APIs do Sisbajud e SNGB, que impossibilita o teste completo dessas integrações em ambiente de homologação. A equipe do PNUD, em conjunto com a Pythonic Café, está contatando as equipes responsáveis pelos sistemas para sanar essa questão.

No contexto dos testes automatizados, acreditamos que no backend temos uma quantidade razoável de testes, tanto unitários, quanto mais abrangentes com relação às regras de negócios, que cobrem as partes mais críticas do sistema. Porém, consideramos que o nível de testes automatizados no frontend é aquém do desejado. Para sanar esse problema, criamos mais casos de teste manuais e detalhamos melhor como devem ser feitos, com o objetivo de verificar de forma abrangente as funcionalidades implementadas pelo frontend. Os casos de teste (automatizados e manuais), bem como o resultado da execução dos testes manuais estão detalhados no documento "Relatório de testes", entregue em conjunto com esse.

Com relação à padronização de código, utilizamos code formaters/linters tanto no backend quanto no frontend, de maneira que ambos repositórios já estão de acordo com as boas práticas definidas por cada linguagem (Python, no caso do backend e TypeScript, no caso do frontend). Apesar disso, a equipe ainda não concluiu a configuração do SonarQube. A equipe de desenvolvimento da Pythonic Café considera importante a configuração do sistema e, apesar de ter tido acesso tardio à plataforma, está trabalhando para que a configuração seja concluída o mais rapidamente possível.

3. Procedimentos operacionais

- **Diferenciação entre Homologação e Produção:**
  O Sniper é composto por dois componentes principais: sniper-backend (que provê a API para o frontend e usuários) e sniper-frontend (que é acessado pelos usuários da plataforma, magistrados e servidores).

  A equipe de desenvolvimento utiliza uma estratégia de manter o máximo de sinergia entre os deployments de homologação e de produção, para que apenas as configurações, feitas em geral por variáveis de ambiente, sejam alteradas. Dessa forma, garantimos que o código executado em ambiente de homologação é o mesmo de produção e conseguimos fazer testes em homologação da maneira mais fidedigna possível ao que irá para produção. Essa estratégia evita diversos tipos de problemas que seriam descobertos apenas em ambiente de produção.

  Como instruído pela equipe de DevOps do PNUD, as configurações de implementação de ambos os projetos estão na pasta kubernetes do repositório, sendo o ambiente eks-hml-01 usado para homologação e eks-prd-01 para produção. As principais diferenças entre os dois ambientes estão nas URLs dos sistemas externos que o Sniper acessa (por exemplo, em homologação o SSO acessado é https://sso.stg.cloud.pje.jus.br/, já em produção é https://sso.cloud.pje.jus.br/).

  Conforme previsto no escopo do projeto, a entrega atual (Produto 4) contempla a adição de bases de dados (Anacjud, Renajud e SNGB). A implementação dessas funcionalidades não requer alterações na infraestrutura de execução do Sniper. Assim, comparando com o ambiente atual em produção, nenhum ajuste precisou ser feito exceto pela configuração do Gitlab CI, que foi melhorada para rodar os testes automatizados de maneira mais completa e atualizar as versões do dind (Docker in Docker) utilizadas no processo de construção do container.

- **Fluxo de Operação:**
  A utilização do sistema se dará da mesma forma da versão anterior: o usuário (magistrado ou servidor) acessará o endereço Web do Sniper, fará a autenticação via SSO PDPJ e então poderá operar o sistema normalmente.

  As URLs de acesso do sistema são:
  - Homologação: https://sniper.stg.pdpj.jus.br/ (frontend) e https://sniper-api.stg.pdpj.jus.br/ (backend)

- Produção: https://sniper.pdpj.jus.br/ (frontend) e https://sniper-api.pdpj.jus.br/ (backend)

- **Testes pós Deploy:**
  A cada novo deployment, a pessoa responsável pelo commit que o engatilhou verifica se a versão esperada está sendo executada. Essa validação é feita por meio dos endereços:
  - Backend: https://sniper-api.stg.pdpj.jus.br/version (homologação) e https://sniper-api.pdpj.jus.br/version (produção).
  - Frontend: https://sniper.stg.pdpj.jus.br/version.txt (homologação) e https://sniper.pdpj.jus.br/version.txt (produção).

  Esses endpoints não necessitam de autenticação e retornam o commit ID (SHA1) em execução no momento. Caso o commit ID não seja o esperado, é necessário verificar se a pipeline do Gitlab CI foi executada com sucesso, dado que o Gitlab CI não procedirá com o deployment caso aconteça algum erro. Caso a pipeline tenha sido executada com sucesso, é necessário verificar os logs do Kubernetes por meio do Rancher (a equipe de desenvolvimento atualmente não tem acesso a esses logs, dado que o acesso ao Rancher não foi concedido e a integração com o Gitlab não funciona).

  Os testes na API são feitos tanto por meio de testes automatizados no backend, quanto por diversos casos de teste manuais, que são executados tanto em ambiente local, quanto de homologação. Recomendamos a leitura do documento "Relatório de testes", entregue em conjunto com esse, para mais detalhes sobre os casos de teste (automatizados e manuais).

4. Segurança da informação

- **Autenticação e Segurança:**
  A autenticação do usuário da API é feita por meio do SSO da PDPJ (usando OpenID), disponível nos endereços https://sso.stg.cloud.pje.jus.br/ (homologação) e https://sso.cloud.pje.jus.br/ (produção).

  Ao acessar a página inicial do sistema, o frontend verifica se o usuário já possui um token armazenado localmente. Caso não possua, o navegador é redirecionado ao SSO e, caso a autenticação seja bem sucedida, o usuário é encaminhado ao Sniper com um token. Esse token é validado pelo backend. Além da assinatura e

expiração do token JWT, o backend verifica se o usuário possui um dos perfis aceitos para acesso ao sitema (magistrado ou servidor).

Passado o fluxo de autenticação, todas as requisições enviadas do frontend ao backend adicionam o cabeçalho Authorization com o valor Bearer <token>. Do lado do backend, todas as requisições passam pelo middleware de autenticação do Django, que valida o token JWT seguindo as verificações citadas acima; isso garante que apenas acessos autorizados ao sistema sejam respondidos.

Até o momento, o backend do Sniper, acessado via token bearer do Keycloak, não faz diferenciação das funcionalidades com base no perfil de usuário, ou seja, internamente não existem funcionalidades específicas ou restrições para tipos diferentes de perfil.

Para as funcionalidades em que o Sniper precisa acessar outro sistema da PDPJ para recuperar dados, como no caso do Cabeçalho Processual, as informações do usuário final são repassadas ao sistema e o controle de acesso é feito diretamente no sistema externo; esse processo garante que, apesar do Sniper não diferenciar as funcionalidades para os usuários autenticados, os usuários só consigam acessar dados que já teriam acesso por meio de outros sistemas.

Alguns endpoints no backend não necessitam de autenticação e são configurados por meio da variáveis de ambiente KEYCLOAK_FREE_PATHS_REGEX. São eles:

- /healthz: checa a saúde do sistema (acessado pelo Zabbix do DTI/CNJ); - /ping: verifica se o sistema está ativo (acessado pelo Kubernetes).
- /version: retorna o commit ID (SHA1) da versão atual em execução (acessado pela equipe de operações para depuraçao).
- /admin: expõe a interface do Django Admin, que facilita acesso a configurações no banco de dados para depuração. A autenticação é feita internamente pelo Django, tendo os dados salvos no banco relacional na tabela auth_user; essa ferramenta pode ser utilizada para depuração, mesmo que a autenticação via Keycloak não esteja funcionando. Os usuários que têm acesso ao Keycloak não conseguem acessar o Django Admin, dado que o acesso é liberado apenas para usuários com um perfil específico no banco de dados interno do Django.

Com relação à criptografia de ponta-a-ponta por meio de HTTPS, a geração de certificados não é de responsabilidade da equipe de desenvolvimento do Sniper:

ela é mantida pela equipe de operações do DTI/CNJ, que é responsável pelo ingress do cluster Kubernetes.

- **Requisitos de Integração:**
  O acesso à API do Sniper, seja via frontend da plataforma ou via outro sistema da PDPJ, é sempre feito via token bearer, utilizando um token JWT gerado pelo SSO da PDPJ. Sistemas que necessitem acessar o Sniper devem encaminhar o token do usuário. Essa prática foi recomendada pela equipe do DTI/CNJ e é utilizada pelo Sniper para consumir dados de outros sistemas da PDPJ.

  Não foi previsto no escopo do projeto a implementação de contas/tokens de serviço, ou seja, tokens não associados a um usuário e que podem ter permissões diferentes.

- **LGPD:**
  A LAI (Lei 12.527/2011) é um pilar da transparência, garantindo acesso a dados públicos (não sigilosos) e fomentando o controle social. Já a LGPD orienta como tratar dados pessoais, buscando proteger a privacidade da cidadã e do cidadão.

  O desafio é que nem sempre os dados de diferentes órgãos seguem os mesmos padrões. Encontramos grafias diferentes para o mesmo município, formatos de data inconsistentes, ou a falta de um identificador único e confiável (chave primária), como um CPF ou CNPJ válido. Esses problemas dificultam ou até inviabilizam o cruzamento automático das informações.

  É aí que o Sniper agrega um valor imenso: ele aplica processos para **padronizar identificadores como CPFs, CNPJs e Números Únicos de Processo (NUPs),** além de tratar outras inconsistências sempre que possível. Isso permite que você, usuária ou usuário do sistema, foque na análise dos vínculos e do patrimônio, em vez de perder tempo com a "limpeza" manual dos dados ou tendo que acessar diversos sistemas diferentes para montar um "quebra-cabeças" durante a investigação patrimonial.

  Temos dois tipos de dados disponíveis no Sniper: os que são dados abertos, que toda cidadã e cidadão tenha acesso, e os que são fechados, em que apenas sistemas do Judiciário podem ter acesso.

Exemplos de dados abertos são: candidaturas às eleições e bens declarados pelos candidatos e candidatas (publicados pelo Tribunal Superior Eleitoral) e lista de empresas e sócios ou sócias dessas empresas (publicado pela Receita Federal).

Exemplos de dados fechados são: informações disponíveis apenas em sistemas como Sisbajud, dados sensíveis cadastrais do CPF e CNPJ (fornecidos ao CNJ por meio de convênio com a Receita Federal), dentre outros.

O Sniper junta tanto bases de dados dados abertas, quanto bases privadas e concatena essas informações, de forma a facilitar todo o processo de investigação.

Vale notar que para que essa integração funcione bem e possa ser expandida, é fundamental que os órgãos que publicam dados abertos sigam **boas práticas** na gestão e publicação de seus dados: usar portais únicos, links estáveis, nomenclaturas padronizadas, formatos abertos (como CSV) e fornecer documentação clara. A adoção dessas práticas facilita a inclusão de **novas bases de dados** não somente no Sniper, mas em outros sistemas do Judiciário e também no seu uso pela sociedade civil.

7. Glossário

Sniper - Aplicação web desenvolvida pelo CNJ para acesso, análise e correlação de dados de fontes diversas, com foco em investigações e diligências.

Sniper BC - Nova versão da aplicação Sniper, com reformulação da interface frontend e integração com novas APIs externas.

Pythonic Café - Empresa contratada para desenvolver a nova versão do sistema Sniper

Django - Framework web em Python usado no backend do Sniper.

Angular 19 - Framework JavaScript usado no frontend da aplicação.

Kubernetes (K8s) - Plataforma de orquestração de containers utilizada para implantar, escalar e gerenciar o Sniper no ambiente do CNJ.

OpenID - Protocolo de autenticação usado para login seguro dos usuários na plataforma.

JWT (JSON Web Token) - Formato de token utilizado para autenticação e autorização, transmitido nas requisições HTTP do frontend para o backend.

Logs e Auditoria - Registro sistemático de eventos e atividades do sistema, utilizado para rastreabilidade e segurança.

Cache - Mecanismo de armazenamento temporário utilizado para acelerar respostas a requisições frequentes, principalmente para dados de APIs externas.

Health Check - Endpoint exposto pela aplicação para indicar se ela está funcionando corretamente. Utilizado para monitoramento automático.

HTTPS + JSON - Padrão de comunicação segura entre frontend e backend usando HTTPS para criptografia e JSON como formato de dados.

Banco de Dados Relacional - Armazena dados estruturados em tabelas (ex: PostgreSQL). Utilizado pelo backend do Sniper.

Banco de Dados de Grafos - Banco que modela dados como vértices e arestas, ideal para relações complexas entre entidades (ex: Neo4j).

Ecossistema PDPJ - Conjunto de sistemas e APIs interoperáveis do Poder Judiciário, integrados ao Sniper.

Coletor - Componente responsável por buscar e armazenar dados de fontes externas nas bases de dados do Sniper.

API - Interface de comunicação com sistemas externos. Algumas APIs usadas pelo Sniper retornam dados de forma assíncrona.

Polling - Técnica em que o backend verifica repetidamente se a resposta de uma API assíncrona está disponível.

Worker - Processo que roda em segundo plano para executar tarefas como polling de APIs e ingestão de dados.

Deployment - Processo de publicação da aplicação nos ambientes.

Variáveis de Ambiente - Parâmetros de configuração do sistema definidos externamente, utilizados para conectar o Sniper com APIs, bancos ou serviços externos.

Modelo C4 - Abordagem para documentação de arquitetura de software que utiliza 4 níveis de visualização: Contexto, Containers, Componentes e Código.

Structurizr DSL - Linguagem de domínio específico (DSL) usada para modelar arquiteturas de software com base no modelo C4.

GitLab - Plataforma de versionamento de código e DevOps usada pelo CNJ para hospedar o repositório do projeto Sniper.

Branch - Ramificação de desenvolvimento em sistemas de controle de versão (como Git), usada para organizar alterações isoladas, como no caso da epic/sniper-bc.

Container - No contexto do modelo C4, refere-se às partes executáveis/deployáveis do sistema (ex: frontend, backend, banco de dados).

Componente - Unidade lógica menor dentro de um container, como módulos, serviços ou classes no backen.

Angular-Material - Biblioteca de componentes UI para Angular baseada nas diretrizes de design Material da Google.

UIKit - Kit de design visual (ou sistema de design) usado como inspiração visual para a interface do Sniper.

Procrastinate - Biblioteca de gerenciamento de filas e tarefas assíncronas em Python com suporte a PostgreSQL.

PostgreSQL - Sistema de banco de dados relacional utilizado no backend. No Sniper, está hospedado na nuvem (AWS RDS).

AgensGraph - Banco de dados orientado a grafos com base no PostgreSQL, usado para representar relações complexas entre dados.

AWS RDS - Serviço gerenciado de banco de dados relacional da Amazon Web Services, usado para hospedar o PostgreSQL do Sniper.

CI/CD - Práticas de automação do desenvolvimento e deploy de software. No Sniper, o CI/CD do GitLab gerencia os deploys no cluster Kubernetes.

5. Apêndices

- **HU** x **CT:**
  Para cada história de usuário (**HU**), percorremos o fluxo completo no sistema, validando os critérios de aceitação. Visando facilitar a comunicação sobre esse documento, os casos de teste usam a nomenclatura "**CT-XXX**", sendo esses:
  - CT-001: Acesso com Usuário sem Perfil Válido
  - CT-002: Aceitação de Termos de Uso no Primeiro Acesso ou na Renovação de Termos
  - CT-003: Criação de Investigação Patrimonial a Partir de Processo
  - CT-004: Visualização de Detalhes da Investigação Patrimonial
  - CT-005: Visualização de Dados Cadastrais de uma Parte
  - CT-006: Acesso a Histórico de Investigações do Usuário
  - CT-007: Habilitar Buscas em Sistemas Externos
  - CT-008: Imprimir Resultado de uma Investigação
  - CT-009: Navegar no Grafo
  - CT-010: Visualizar Contas em Instituições Bancárias (Sisbajud)
  - CT-011: Visualizar Embarcações (Tribunal Marítimo)
  - CT-012: Visualizar Bens Declarados (TSE)
  - CT-013: Visualizar Sanções Diversas (CGU)
  - CT-014: Visualizar Bens Bloqueados (SNGB)
  - CT-015: Visualizar Aeronaves (AnacJud)
  - CT-016: Visualizar Veículos Automotores (Renajud)

6. Índice