Manual de operação de sistema - Versão 1

Índice

1. Introdução

1.1. Objetivo

Documentar os procedimentos técnicos necessários à sustentação, operação e evolução do Sistema Gestão de Clientes, contemplando sua configuração, monitoramento, manutenção, segurança e continuidade operacional.
Este manual visa apoiar as equipes técnicas responsáveis pela operação e sustentação do sistema, garantindo a padronização das rotinas e a rastreabilidade das ações em ambiente produtivo.

1.2. Visão geral do sistema

O Sistema Gestão de Clientes (GeCli) é uma solução institucional desenvolvida no âmbito do Programa Justiça 4.0, coordenado pelo Conselho Nacional de Justiça (CNJ) em parceria com o Programa das Nações Unidas para o Desenvolvimento (PNUD).
Seu objetivo é gerenciar o ciclo de vida de credenciamento e consumo de APIs e serviços digitais do ecossistema do Poder Judiciário, oferecendo uma interface única para solicitação, aprovação, autorização e monitoramento de acessos.
O sistema é composto por um frontend web responsivo e uma camada backend com APIs REST, integradas a mecanismos de autenticação e autorização via Keycloak (SSO CNJ), e a módulos de governança e observabilidade de APIs.

1.3. Público -alvo do sistema

O **Sistema Gestão de Clientes (GeCli)** é utilizado por diferentes perfis de usuários que participam do processo de **gestão, solicitação e controle de acesso** a APIs e serviços digitais do Poder Judiciário, no contexto do **Programa Justiça 4.0**.
O público-alvo abrange tanto **usuários internos do CNJ**, responsáveis pela governança e sustentação do sistema, quanto **usuários externos**, que representam instituições públicas e privadas com integração ativa ou em análise.

1. Usuários Internos – CNJ

Usuários vinculados à estrutura administrativa e técnica do Conselho Nacional de Justiça, com papéis de operação e controle:

Perfil	Função	Unidade / Atuação
Administrador Técnico	Gerencia parâmetros do sistema, configura perfis, supervisiona logs e rotinas automatizadas.	Diretoria de Tecnologia da Informação (DTI/CNJ)
Gestor de Governança e Interoperabilidade	Analisa tecnicamente solicitações de acesso, verifica aderência à política de segurança e acompanha o ciclo de credenciais.	Juízes Auxiliares da Presidência / JADTI
Administrador de Negócio	Supervisiona o fluxo de solicitações, consolida pareceres técnicos e valida informações junto ao solicitante.	Programa Justiça 4.0 (CNJ/PNUD)

Atualmente, o CNJ mantém cerca de **50 perfis internos ativos**, entre gestores técnicos, analistas e validadores administrativos.

---

2. Usuários Externos – Órgãos e Entidades Públicas

Representam órgãos do Sistema de Justiça e entes federados que utilizam o sistema para solicitar e consumir serviços interoperáveis via API.

Exemplo de Usuário	Finalidade de Acesso
Município de Cascavel (PR)	Comunicação sistêmica via MNI – Modelo Nacional de Interoperabilidade
Conselho Nacional do Ministério Público (CNMP)	Consumo de dados públicos da API Datalake
Banco do Brasil / Caixa Econômica Federal	Integração institucional e serviços de interoperabilidade
Tribunais e Ministérios Públicos	Acesso controlado a serviços do ecossistema Jus.br

Estima-se que o grupo de usuários públicos **represente aproximadamente 40% do total de credenciais ativas**.

3. Usuários Externos – Entidades Privadas e Parceiros Tecnológicos
Empresas, escritórios e associações que firmam **Acordo de Cooperação Técnica (ACT)** para acesso a dados públicos consolidados, mediante autorização formal e monitoramento do CNJ.

Exemplos de Solicitantes Ativos ou em Análise
Escavador, Forelegal, Docato, LinkLei, Serasa, Thomson Reuters, AASP, Deep Legal, Integrativa, SNAP Informática, BNDES, OAB-PR, CNIF, Enter (Talisman AI).

Essas instituições representam o maior volume de solicitações atualmente, com **mais de 180 credenciais emitidas apenas para a API do Domicílio Judicial Eletrônico (DJE)**, e dezenas de novos pedidos em tramitação para o acesso ao **Dados Judiciais.**

---

4. Escopo e Dimensão de Atendimento

Indicador	Valor Aproximado (out/2025)	Observação
Credenciais emitidas (todas as APIs)	200+	Inclui DJE, Datalake e SSO
Solicitações em tramitação	30+	Processos SEI ativos e ACTs em avaliação
Usuários internos (CNJ)	50	Técnicos e gestores de validação
Instituições externas (públicas e privadas)	450+	Entre signatárias e solicitantes
Total estimado de usuários ativos	**500+**	Expansão contínua com novos ACTs

5. Natureza do Acesso
O sistema é de **acesso restrito**, disponível apenas a usuários autenticados via **SSO (Keycloak CNJ)**.
A concessão de credenciais é condicionada à análise técnica e jurídica, assegurando conformidade com:

- **Resolução CNJ nº 574/2024** – Política de acesso a dados judiciais públicos.

- **Portaria CNJ nº 316/2023** – Diretrizes para compartilhamento e consumo de dados via APIs.

1.4. Escopo

Este manual abrange **os procedimentos técnicos necessários à sustentação, operação e continuidade do Sistema Gestão de Clientes (GeCli)**, incluindo as rotinas de configuração, segurança, monitoramento e manutenção da solução. O conteúdo foi estruturado de forma a apoiar as equipes técnicas do **CNJ, DTI e parceiros do Programa Justiça 4.0**, durante e após a transição de responsabilidades da fase de implantação (*phase-out*).

Componentes e aspectos abordados neste manual

Componente / Área	Conteúdo incluído neste manual
Arquitetura do sistema	Estrutura lógica e física dos componentes (frontend, backend, APIs e banco de dados).
Procedimentos de instalação e configuração	Etapas para instalação do ambiente, build do frontend, configuração de variáveis e provisionamento de serviços.
Procedimentos de segurança	Autenticação, autorização, papéis e políticas de acesso (Keycloak/SSO CNJ).
Rotinas operacionais	Backup, logs, monitoramento, testes de conectividade e observabilidade.
Sustentação e suporte	Fluxos de atendimento N1, N2 e N3 e responsabilidades institucionais.
Atualização e migração	Procedimentos para atualização de versões, migração de dados e publicação de novas APIs.
Descomissionamento e contingência	Diretrizes para encerramento seguro de ambientes e plano de recuperação em caso de falha.
Formulários obrigatórios	Modelos do Gestor Técnico e do Gestor Negocial (IN CNJ nº 86/2021).

Conteúdos fora do escopo deste manual

Documento / Área	Responsável / Abrangência

Manual do Usuário	Orienta o uso funcional da aplicação, fluxos de cadastro e tramitação de solicitações.
Guia de FAQ e suporte funcional	Reúne perguntas frequentes e procedimentos de autoatendimento.
Relatórios Técnicos de Migração e Testes	Documentam validações de infraestrutura, conectividade e segurança executadas durante a implantação.
Plano de Contingência e Segurança	Contém políticas e estratégias de recuperação de desastres e backup de dados.
Documentação de APIs no Portal Jus.br	Descreve endpoints, parâmetros e payloads disponíveis para integração.
Documentos Jurídicos e Acordos de Cooperação Técnica (ACT)	Tratam das condições legais de compartilhamento de dados e obrigações das partes.

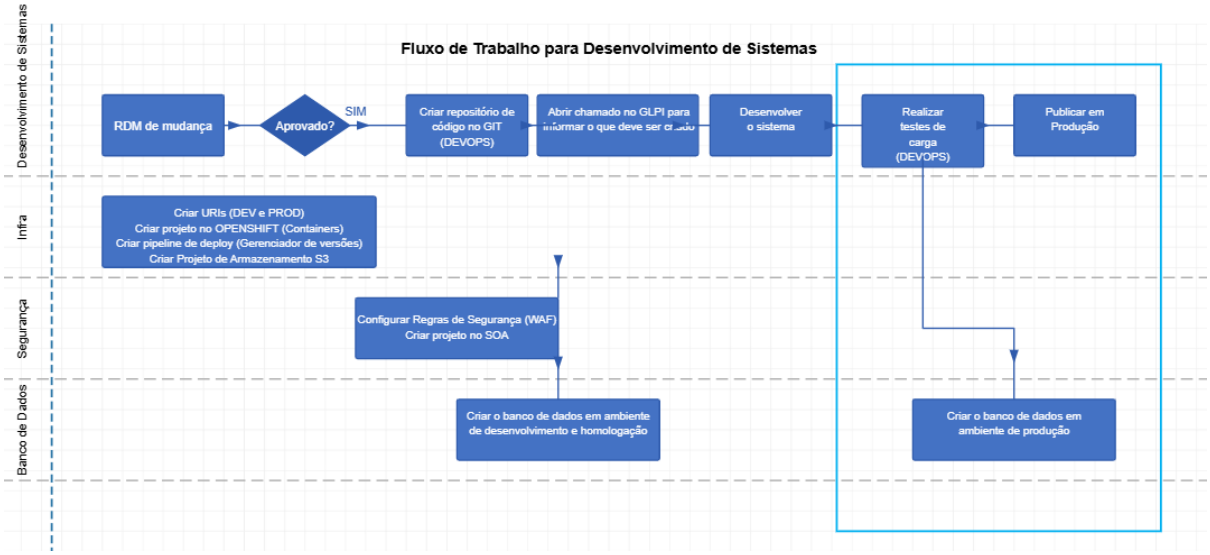
Escopo temporal e de manutenção

Este manual tem validade **enquanto durar o ciclo de sustentação da versão atual do sistema (MVP)** e deve ser **atualizado a cada entrega de versão ou modificação significativa na arquitetura, pipeline ou ambiente de produção.**

As versões futuras (v2.0 ou superiores) deverão incluir:

- Expansão do módulo de observabilidade de APIs (Fase 2);
- Integração automática com o DataLake (via pipeline de logs);
- E atualizações decorrentes de novos acordos de cooperação técnica firmados com órgãos e entidades externas.

2. Documentação técnica



2.1. Código-fonte

O código-fonte do *Sistema Gestão de Clientes* está dividido em dois repositórios oficiais do **GitLab CNJ/PDPJ**, organizados por domínio de aplicação:

Componente	Descrição	Repositório Git
Frontend (Interface Web)	Aplicação web SPA responsável pela interação com usuários e visualização dos dados. Desenvolvida em Angular e hospedada em ambiente cloud do CNJ.	https://git.cnj.jus.br/pdpj/frontends/gestao-clientes
Backend (Regras de Negócio e APIs)	Serviços REST responsáveis pelo processamento das solicitações, autenticação, autorização e integração com sistemas internos e externos. Desenvolvido em Laravel (PHP).	https://git.cnj.jus.br/pdpj/negocio/gestao-clientes

2.2. Documentação de APIs

O **Serviço de Gestão de Clientes (GeCli)** expõe uma **API RESTful pública e autenticada**, utilizada pelo frontend, por integrações internas do CNJ e por sistemas externos devidamente credenciados.

API Principal

Item	Descrição
Base URL (Produção)	https://gestao-clientes.pdpj.jus.br/
Base URL (Homologação)	https://gestao-clientes.stg.pdpj.jus.br/
Documentação técnica (Swagger UI)	https://gateway.cloud.pje.jus.br/gestao-clientes/swagger-ui/index.html
Autenticação	Bearer Token (JWT) emitido pelo Keycloak CNJ
Formato de dados	application/json
Padrão de versionamento	/api/v1/...

Principais Grupos de Endpoints

Módulo (Swagger Tag)	Objetivo	Exemplo de Endpoint
Solicitações de APIs	Controla o ciclo de solicitações de acesso (criação, análise, aprovação).	GET /solicitacoes-api
Entidade Solicitante de APIs	Gerencia os dados cadastrais e documentos da entidade solicitante.	POST /entidade-solicitante
Documentos Anexos	Faz upload/download de documentos vinculados à solicitação.	POST /documentos-anexo/upload
Catálogo de APIs	Lista, cadastra e atualiza APIs disponíveis no portal.	GET /catalogo-apis
Categorias de APIs	Classifica as APIs por tema.	POST /categorias-api
Acesso à API	Gera credenciais (clientId/secret) e controla liberações.	POST /acessos-api
Termo de Uso / Aceite	Exibe e registra aceite de termos de uso.	POST /termos-uso-usuario/{id}/aceite
Perguntas Frequentes	Gerencia as FAQs exibidas no Portal Jus.br.	GET /publico/perguntas-frequente

Motivos de Rejeição	Registra e consulta razões de indeferimento.	POST /motivos-rejeicao-api
APIs Públicas	Permite consulta pública ao catálogo e FAQs sem autenticação.	GET /publico/catalogo-apis

---

APIs Internas e de Integração

API / Serviço	Finalidade	Autenticação / Protocolo
Keycloak CNJ	Emissão e validação de tokens JWT (SSO).	OAuth2 / OpenID Connect
SEI (Sistema Eletrônico de Informações)	Recuperação do processo vinculado à solicitação.	Integração interna (API SEI Gov)
Prometheus / Grafana	Coleta de métricas e logs de disponibilidade.	HTTP (push/pull)
Portal Jus.br	Exposição pública do catálogo e termos de uso.	Catálogo de APIs

---

Formatos de Resposta e Erros

As respostas seguem o padrão do DTO ApiDataResponseDTO, com as seguintes propriedades:

```
{
 "data": {},
 "pagination": {},
 "error": {
  "status": 400,
  "message": "Requisição inválida",
```

"timestamp": "2025-11-10T10:00:00"

}

}

Erros seguem a estrutura ErrorDetailsDTO, contendo código HTTP, mensagem, timestamp e path da requisição.
Os códigos mais comuns são:

- **200 OK** – operação bem-sucedida
- **201 Created** – registro criado
- **400 Bad Request** – parâmetros inválidos
- **401 Unauthorized** – token inválido ou ausente
- **404 Not Found** – recurso inexistente
- **500 Internal Server Error** – erro inesperado no servidor

2.3. Bibliotecas utilizadas

Projeto	Biblioteca	Versão
Backend (Java)	org.springframework.cloud:spring-cloud-dependencies	${spring-cloud.version}
Backend (Java)	io.opentelemetry.instrumentation:opentelemetry-instrumentation-bom	${opentelemetry-instrumentation-bom.version}
Backend (Java)	software.amazon.awssdk:bom	${amazon-sdk.version}
Backend (Java)	org.springframework.boot:spring-boot-starter-actuator	N/A
Backend (Java)	org.springframework.boot:spring-boot-starter-amqp	N/A
Backend (Java)	org.springframework.boot:spring-boot-starter-aop	N/A
Backend (Java)	org.springframework.boot:spring-boot-starter-data-jpa	N/A
Backend (Java)	org.springframework.boot:spring-boot-starter-data-redis	N/A
Backend (Java)	org.springframework.boot:spring-boot-starter-security	N/A
Backend (Java)	org.springframework.boot:spring-boot-starter-oauth2-resource-server	N/A
Backend (Java)	org.springframework.boot:spring-boot-starter-validation	N/A

Backend (Java)	org.springframework.boot:spring-boot-starter-web	N/A
Backend (Java)	org.springframework.boot:spring-boot-starter-web-services	N/A
Backend (Java)	org.springframework.cloud:spring-cloud-starter-circuitbreaker-resilience4j	N/A
Backend (Java)	org.springframework.cloud:spring-cloud-starter-netflix-eureka-client	N/A
Backend (Java)	org.flywaydb:flyway-core	N/A
Backend (Java)	org.flywaydb:flyway-database-postgresql	N/A
Backend (Java)	org.hibernate.orm:hibernate-envers	N/A
Backend (Java)	org.hibernate.orm:hibernate-jpamodelgen	N/A
Backend (Java)	org.postgresql:postgresql	N/A
Backend (Java)	io.github.resilience4j:resilience4j-bulkhead	N/A
Backend (Java)	io.github.resilience4j:resilience4j-ratelimiter	N/A
Backend (Java)	io.github.resilience4j:resilience4j-retry	N/A
Backend (Java)	org.mapstruct:mapstruct	${mapstruct.version}
Backend (Java)	org.mapstruct:mapstruct-processor	${mapstruct.version}
Backend (Java)	org.projectlombok:lombok	${lombok.version}
Backend (Java)	com.github.ben-manes.caffeine:caffeine	N/A
Backend (Java)	org.springframework:spring-context-support	N/A
Backend (Java)	org.springdoc:springdoc-openapi-starter-webmvc-ui	${springdoc.version}
Backend (Java)	io.micrometer:micrometer-registry-prometheus	N/A
Backend (Java)	io.opentelemetry.instrumentation:opentelemetry-logback-mdc-1.0	${opentelemetry-logback-mdc-1.0.version}
Backend (Java)	net.logstash.logback:logstash-logback-encoder	${logstash-logback-encoder.version}
Backend (Java)	org.springframework.boot:spring-boot-configuration-processor	N/A

Backend (Java)	br.jus.pdpj:pdpj-commons	${pdpj-commons.version}
Backend (Java)	software.amazon.awssdk:s3	N/A
Backend (Java)	software.amazon.awssdk:auth	N/A
Backend (Java)	io.opentelemetry.instrumentation:opentelemetry-aws-sdk-2.2	${opentelemetry-aws-sdk-2.2.version}
Backend (Java)	com.github.dasniko:testcontainers-keycloak	2.5.0
Backend (Java)	org.springframework.amqp:spring-rabbit-test	N/A
Backend (Java)	org.springframework.boot:spring-boot-starter-test	N/A
Backend (Java)	org.springframework.boot:spring-boot-testcontainers	N/A
Backend (Java)	org.springframework.restdocs:spring-restdocs-mockmvc	N/A
Backend (Java)	org.springframework.security:spring-security-test	N/A
Backend (Java)	org.testcontainers:junit-jupiter	N/A
Backend (Java)	org.testcontainers:postgresql	N/A
Backend (Java)	org.testcontainers:rabbitmq	N/A
Frontend (Angular)	@angular/animations	^17.3.0
Frontend (Angular)	@angular/cdk	^17.3.9
Frontend (Angular)	@angular/common	^17.3.0
Frontend (Angular)	@angular/compiler	^17.3.0
Frontend (Angular)	@angular/core	^17.3.0
Frontend (Angular)	@angular/forms	^17.3.0
Frontend (Angular)	@angular/material	^17.3.9
Frontend (Angular)	@angular/material-moment-adapter	^17.3.9
Frontend (Angular)	@angular/platform-browser	^17.3.0

Frontend (Angular)	@angular/platform-browser-dynamic	^17.3.0
Frontend (Angular)	@angular/router	^17.3.0
Frontend (Angular)	@fortawesome/fontawesome-free	^6.7.2
Frontend (Angular)	@ng-matero/extensions	^17.3.7
Frontend (Angular)	@ng-matero/extensions-moment-adapter	^17.1.2
Frontend (Angular)	@ngx-translate/core	^15.0.0
Frontend (Angular)	@ngx-translate/http-loader	^8.0.0
Frontend (Angular)	@types/jspdf	^2.0.0
Frontend (Angular)	ajv	^8.17.1
Frontend (Angular)	apexcharts	^4.0.0
Frontend (Angular)	jspdf	^3.0.1
Frontend (Angular)	keycloak-js	^26.0.5
Frontend (Angular)	moment	^2.30.0
Frontend (Angular)	ngx-mask	^19.0.7
Frontend (Angular)	ngx-permissions	^17.1.0
Frontend (Angular)	ngx-progressbar	^12.0.2
Frontend (Angular)	ngx-toastr	^18.0.0
Frontend (Angular)	rxjs	~7.8.0
Frontend (Angular)	screenfull	^6.0.2
Frontend (Angular)	tslib	^2.3.0
Frontend (Angular)	zone.js	~0.14.3
Frontend (Angular)	@angular-devkit/build-angular	^17.3.9

Frontend (Angular)	@angular/cli	^17.3.9
Frontend (Angular)	@angular/compiler-cli	^17.3.0
Frontend (Angular)	@types/jasmine	~5.1.0
Frontend (Angular)	jasmine-core	~5.1.0
Frontend (Angular)	karma	~6.4.0
Frontend (Angular)	karma-chrome-launcher	~3.2.0
Frontend (Angular)	karma-coverage	~2.2.0
Frontend (Angular)	karma-jasmine	~5.1.0
Frontend (Angular)	karma-jasmine-html-reporter	~2.1.0
Frontend (Angular)	prettier	^3.2.4
Frontend (Angular)	typescript	~5.4.2

2.4. Frameworks

O **Sistema Gestão de Clientes (GeCli)** foi desenvolvido em conformidade com os padrões técnicos da **Plataforma Digital do Poder Judiciário (PDPJ)**, utilizando frameworks de mercado amplamente adotados para garantir **modularidade, interoperabilidade e escalabilidade**.
O sistema aplica o padrão **MVC (Model-View-Controller)** no backend e **SPA (Single Page Application)** no frontend.

Link da documentação de arquitetura: https://git.cnj.jus.br/pdpj/negocio/gestao-clientes/-/blob/master/docs/arquitetura/README-02.md

Camada	***Framework / Tecnologia***	***Descrição / Finalidade***
Backend (Java / Spring)	**Java 21 / Spring Boot 3.5.4**	Framework base para desenvolvimento de microserviços. Implementa o padrão MVC e gerencia dependências, injeção de componentes e ciclo de vida da aplicação.
	Spring Cloud 2021.0.8	Suporte a aplicações distribuídas e integração com serviços da PDPJ (Discovery e Gateway).
	Spring Security + OAuth2	Camada de segurança integrada ao Keycloak para autenticação e autorização JWT.
	Spring Data JPA / Hibernate / Envers	Acesso e persistência de dados com versionamento e auditoria.
	MapStruct 1.5.5.Final	Mapeamento automático entre entidades e DTOs.
	Lombok	Reduz código repetitivo (getters, setters, builders).
	Flyway	Controle de versionamento e migração de banco de dados.
	Springdoc OpenAPI 1.8.0	Geração automática da documentação Swagger (https://gateway.cloud.pje.jus.br/gestao-clientes/swagger-ui).

	Micrometer / Prometheus / OpenTelemetry	Coleta de métricas e observabilidade.
	RabbitMQ	Mensageria assíncrona para notificações.
	Redis	Cache em memória para otimização de consultas e sessões.
Frontend (Angular)	**Angular 17 + TypeScript**	Framework SPA para construção da interface e consumo das APIs REST.
	RxJS / HttpClient	Programação reativa e controle de fluxos assíncronos.
	Bootstrap / SCSS	Layout responsivo e padrão visual do CNJ.
	ngx-translate	Internacionalização de mensagens e rótulos.
Infraestrutura / Integração	**Keycloak CNJ**	Gestão de identidade e autenticação OAuth2.
	PDPJ Discovery (Eureka)	Serviço de descoberta dos microserviços PDPJ.
	PDPJ Gateway (Zuul)	Gateway central da PDPJ para roteamento e segurança das APIs.
	Docker / Docker Compose / GKE	Contêineres e orquestração dos ambientes DSV, HML e PRD.

2.5. Casos de uso

O **GeCli** foi concebido como serviço centralizador da **governança de acesso a dados judiciais públicos** por meio de APIs expostas no **Portal Jus.br**, em conformidade com a **Resolução CNJ nº 574/2024**.
Abaixo, os principais casos de uso da aplicação.

ID	Caso de Uso	Descrição / Objetivo	Atores
CU-01	**Gerenciar Solicitações de Acesso**	Permite que entidades públicas e privadas solicitem acesso às APIs de dados judiciais por meio de formulário eletrônico.	Entidade Solicitante
CU-02	**Analisar e Aprovar Solicitações**	Equipes técnicas e negociais do CNJ avaliam documentação e decidem sobre o deferimento do pedido.	Gestor Técnico / Gestor Negocial
CU-03	**Gerar Credenciais de Acesso (Client ID / Secret)**	Após aprovação, o sistema cria automaticamente as credenciais no Keycloak e as associa à API.	Sistema / Keycloak

CU-04	**Publicar Catálogo de APIs no Jus.br**	Permite exibir APIs, descrições, termos de uso e status de disponibilidade no portal público.	DTI / Portal Jus.br
CU-05	**Registrar Aceite de Termo de Uso**	O solicitante deve aceitar eletronicamente os termos antes de obter o acesso.	Entidade Solicitante
CU-06	**Gerenciar Entidades e Documentos Anexos**	Armazena dados cadastrais, comprovantes e documentos enviados.	Gestor Técnico / DTI
CU-07	**Gerenciar FAQ e Motivos de Rejeição**	Permite incluir perguntas frequentes e registrar motivos de indeferimento.	Gestor Negocial
CU-08	**Monitorar Consumo e Métricas de Acesso**	Acompanha logs, volume de chamadas e disponibilidade das APIs.	DTI / CNJ
CU-09	**Manter Configurações de Segurança e Backup**	Administra rotinas de auditoria, disaster recovery e controle de permissões.	DTI / Administrador
CU-10	**Encerrar ou Revogar Credenciais**	Permite desativar acessos de entidades e revogar tokens em casos de descredenciamento.	DTI / Gestor Técnico

2.6. Paths da aplicação e credenciais para testes de penetração

2.6.1. Contexto e finalidade

O teste de penetração (pentest) tem como objetivo avaliar a resiliência e a segurança do Sistema Gestão de Clientes (GeCli) em ambiente produtivo, identificando vulnerabilidades relacionadas a autenticação, autorização, exposição de serviços e configurações de infraestrutura.
Os testes visam assegurar que o sistema esteja em conformidade com as diretrizes de segurança da informação do Conselho Nacional de Justiça (CNJ) e com os padrões técnicos da Plataforma Digital do Poder Judiciário (PDPJ).
O processo está formalmente registrado no **protocolo SEI nº 00131/2020 – OS 2373782**, sob responsabilidade da **empresa ISH Tecnologia**, contratada pelo CNJ para execução de testes de segurança em aplicações críticas.
6.2 Escopo técnico dos testes

O escopo definido para os testes de penetração contempla:

Componente	Descrição	Ambiente
Frontend (Angular)	Aplicação web SPA acessível via HTTPS.	https://gestao-clientes.stg.pdpj.jus.br
Backend (Spring Boot)	Serviços REST protegidos por OAuth2.	https://gateway.cloud.pje.jus.br/gestao-clientes
SSO Keycloak	Autenticação e autorização via Realm CNJ.	https://sso.cnj.jus.br/auth

2.6.2. Paths e variáveis de ambiente

Os principais *paths* e variáveis de ambiente utilizados nos testes e nas rotinas de operação são:

Variável	Descrição	Valor (ambiente de produção)
API_BASE_URL	Endpoint principal da API GeCli	https://gateway.cloud.pje.jus.br/gestao-clientes
FRONTEND_URL	Interface web do sistema	https://gestao-clientes.pdpj.jus.br
KEYCLOAK_REALM	Realm de autenticação CNJ	pdpj
KEYCLOAK_CLIENT_ID	Client configurado para o frontend	gestao-clientes-frontend
KEYCLOAK_URL	Endpoint do servidor de autenticação	https://sso.stg.cloud.pje.jus.br/auth/
DATABASE_URL	Endpoint do banco PostgreSQL	Acesso restrito (VPN CNJ)

Essas variáveis são controladas por meio de arquivos .env nas instâncias Docker do sistema, com acesso limitado aos administradores técnicos do CNJ e equipe de sustentação da PDPJ.

2.6.3. Credenciais e controle de acesso

Durante a execução do pentest, foram emitidas **credenciais temporárias de acesso técnico** com escopo restrito aos endpoints de homologação, sob monitoramento da DTI/CNJ.

- **Tipo de acesso:** Bearer Token JWT (expiração 15 minutos).
- **Perfil de testes:**
- **Usuário Portal:**
  usuario.gecli
  SenhaSegura@123
- **Usuário Admin:**
  54212294087
  Cnj#4578!!

- **Ambiente:** Homologação (STG).
- **Período autorizado:** conforme OS 2373782.
- **Supervisão técnica:** Coordenação de Segurança da Informação / DTI-CNJ.
  A revogação das credenciais foi realizada ao término da execução, com registro em ata e encerramento formal do protocolo SEI.

2.7. Expectativas de acessos para o sistema em ambiente produtivo

O Sistema **Gestão de Clientes (GeCli)** é executado em ambiente **cloud gerenciado da Plataforma Digital do Poder Judiciário (PDPJ)**, dentro do ecossistema de serviços do Conselho Nacional de Justiça (CNJ).
O ambiente produtivo foi projetado para garantir **alta disponibilidade, segurança e rastreabilidade** dos acessos — tanto para usuários humanos autenticados quanto para integrações máquina-a-máquina (M2M).

2.7.1. Perfis e mecanismos de autenticação

Os acessos são regulados pelo **Keycloak CNJ (realm pdpj)**, conforme fluxos definidos no documento de arquitetura (README-02.md), com uso dos seguintes mecanismos de autenticação:

Tipo de Usuário	Finalidade	Fluxo OAuth2 Utilizado	Mecanismo de Autenticação
Usuário Interno (CNJ)	Acesso administrativo, análise e aprovação de solicitações	Authorization Code (PKCE)	Login via SSO CNJ com MFA
Usuário Externo (Órgão Público)	Solicitação de credenciais para consumo de APIs do ecossistema Jus.br	Authorization Code (PKCE)	Autenticação federada via Keycloak CNJ
Usuário Externo (Entidade Privada / ACT)	Acesso a dados públicos mediante cooperação técnica	Authorization Code (PKCE)	SSO CNJ – realm pdpj
Integração Sistema–Sistema (Backend)	Criação, revogação e auditoria de clients no Keycloak	Client Credentials	Service account gestao-clientes
Pipeline e Automação Técnica (Uso Controlado)	Testes e rotinas agendadas	Password Grant (restrito a ambiente STG)	Client gestao-clientes confidencial

Cada token emitido é do tipo **Bearer JWT**, assinado pelo servidor https://sso.cnj.jus.br/auth, contendo *claims* de controle (azp, realm_access, resource_access, clientHost, clientAddress, exp, iat, jti), conforme o exemplo auditado abaixo:

{

"iss": "https://sso.stg.cloud.pje.jus.br/auth/realms/pje",

"azp": "gestao-clientes",

"preferred_username": "service-account-gestao-clientes",

"realm_access": { "roles": ["manage-clients"] },

```
 "resource_access": {
   "realm-management": { "roles": ["manage-clients"] },
   "account": { "roles": ["view-profile"] }
 },
 "clientHost": "189.6.17.35",
 "typ": "Bearer"
}
```


2.7.2. Estrutura de autenticação e controle de tokens

O backend do GeCli mantém **autenticação federada com o Keycloak CNJ**, utilizando um **client confidential** (gestao-clientes) configurado com *service account* e escopo manage-clients.
A comunicação com a Admin REST API é autenticada via fluxo **Client Credentials**, sendo o token de serviço armazenado e controlado de forma segura conforme o **ADR-006 (Cache distribuído e lock de token)**, com as seguintes medidas:

- **Redis** utilizado como cache distribuído para o token de serviço e controle de lock;
- **TTL sincronizado** ao tempo de expiração do token (exp);
- **Lock distribuído** (SETNX) garante que apenas um pod renove o token, evitando sobrecarga no Keycloak;
- **Logs de auditoria** registram requisições de renovação e falhas 401/403;
- **Tokens de usuário final** são validados em cada requisição do frontend e nunca armazenados.

Esse modelo assegura **resiliência, consistência entre pods e conformidade com boas práticas OAuth2/OIDC**.

2.7.3. Acessos esperados e dimensionamento

O dimensionamento segue as metas de operação contínua do CNJ:

Indicador	**Valor Estimado**	**Observação**
Usuários internos simultâneos	80–100	DTI, DISI e gestores do Programa Justiça 4.0
Usuários externos simultâneos	400–500	Órgãos, MPs, TJs e entidades ACT
Chamadas diárias às APIs	50.000+	Predominância das integrações DJE e Datalake
Tempo médio de resposta	≤ 500 ms	Medido via Micrometer/Prometheus
Disponibilidade esperada	≥ 99,5%	SLA PDPJ
Capacidade de picos	até 1.000 conexões simultâneas	Controladas via Gateway Zuul com *rate limiting*
Renovação de tokens	a cada 15 minutos	Tokens curtos para reforço de segurança

2.7.4. Auditoria e rastreabilidade de acessos

Cada chamada autenticada ao backend gera um log no formato JSON estruturado, contendo:

- timestamp, usuario, endpoint, método, statusCode, IPOrigem, clientId e realm.
  Esses registros são enviados para o **serviço central de auditoria PDPJ**, integrando-se ao **Stackdriver** e **Grafana Loki**.

Os logs de renovação de tokens e operações administrativas no Keycloak são capturados em **nível INFO e WARN**, com rastreamento de exceções via **OpenTelemetry**.

2.8. Manual de operação do sistema

2.8.1. Operação do Backend (Spring Boot)

O **Backend do Sistema Gestão de Clientes (GeCli)** é responsável pela camada de negócio e integração entre a interface Angular, o Keycloak (SSO CNJ), o PostgreSQL (RDS CNJ) e serviços internos da Plataforma Digital do Poder Judiciário (PDPJ), como o Gateway, o Discovery e o módulo Corporativo.

O serviço foi desenvolvido em **Spring Boot 3.5.4**, com arquitetura modular, autenticação via OAuth2, comunicação assíncrona (RabbitMQ) e resiliência baseada em padrões do Spring Cloud.

2.8.2. Estrutura do repositório

Repositório:
https://git.cnj.jus.br/pdpj/negocio/gestao-clientes

Branch principal: master
Linguagem: Java 21
Framework: Spring Boot
Gerenciador de build: Maven

Diretório / Arquivo	Descrição
/src/main/java	Código-fonte (controllers, services, repositories).
/src/main/resources	Configurações (application.yml, mapeamentos e logs).
/docker	Scripts de build e inicialização.
/kubernetes	Manifests de deploy para EKS.
pom.xml	Dependências e versões Maven.
.gitlab-ci.yml	Pipeline de build, testes e deploy.
Dockerfile	Empacotamento da aplicação em imagem Docker.

2.8.3. Configurações de ambiente

As configurações de ambiente estão centralizadas no arquivo application.yml e complementadas por variáveis em configmaps.yml no Kubernetes.

Parâmetros principais:

```
server:
  port: 8080
spring:
  datasource:
    url: jdbc:postgresql://postgres:5432/gestaoclientes
    username: ${POSTGRES_USER}
    password: ${POSTGRES_PASSWORD}
  jpa:
    hibernate:
      ddl-auto: none
  redis:
    host: redis
    port: 6379
keycloak:
  auth-server-url: https://sso.cnj.jus.br/auth
  realm: pdpj
  resource: gestao-clientes
```

```
  credentials:
    secret: ${KEYCLOAK_SECRET}
```

Principais variáveis externas:

Variável	Descrição
SPRING_PROFILES_ACTIVE	Perfil ativo (dev, stg, prd)
DATABASE_URL	Endpoint do PostgreSQL
REDIS_URL	Servidor de cache distribuído
KEYCLOAK_URL	Endereço do servidor de autenticação
GATEWAY_URL	Rota base via Zuul PDPJ
JWT_EXPIRATION_MINUTES	Tempo de expiração do token de sessão

2.8.4. Processo de build e deploy

O backend segue o mesmo modelo CI/CD padronizado pela PDPJ, automatizado pelo GitLab e executado em pipelines.

a) Build

```
mvn clean package -DskipTests
```

Gera o artefato gestao-clientes.jar no diretório /target.
O build é empacotado via Dockerfile:

```
FROM openjdk:21-jdk-slim

COPY target/gestao-clientes.jar /app/gestao-clientes.jar

ENTRYPOINT ["java","-jar","/app/gestao-clientes.jar"]
```

A imagem gerada é publicada no registry interno:

```
registry.cnj.jus.br/pdpj/negocio/gestao-clientes:latest
```

b) Deploy (Kubernetes / EKS)

O deploy é orquestrado por meio de **Kustomize**, utilizando os manifests localizados em:
/kubernetes/overlays/eks-hml-01/

Arquivo	Função
configmaps.yml	Variáveis de ambiente e URLs do Keycloak.
ingress.yml	Exposição HTTPS via Gateway PDPJ.
kustomization.yml	Reúne os recursos para o cluster.

Ingress padrão:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: gestao-clientes
spec:
  rules:
    - host: gateway.cloud.pje.jus.br
      http:
        paths:
          - path: /gestao-clientes
            pathType: Prefix
            backend:
              service:
                name: gestao-clientes-service
                port:
```

number: 8080

2.8.5. Fluxos e integrações principais

Integração	Descrição	Tipo
Keycloak Admin REST API	Criação, atualização e revogação de *clients* PDPJ.	OAuth2 Client Credentials
Gateway PDPJ	Roteamento e autenticação JWT.	HTTPS
RabbitMQ	Comunicação assíncrona entre microserviços PDPJ.	AMQP
Redis	Cache e lock distribuído para tokens.	TCP
PostgreSQL	Armazenamento de solicitações, perfis e logs.	JDBC

O backend utiliza interceptadores para auditar cada requisição, registrando usuario, IPOrigem, endpoint, status e clientId.

2.8.6. Monitoramento e logs

Os logs seguem o padrão estruturado JSON, exportados para o **Stackdriver** via Logstash.

Exemplo de log:

```
{
  "timestamp": "2025-10-10T14:25:43.225Z",
  "level": "INFO",
  "service": "gestao-clientes",
  "usuario": "service-account-gestao-clientes",
  "endpoint": "/api/solicitacoes",
  "status": 200
}
```

Monitoramento integrado via **Prometheus / Grafana**:

- /actuator/health → verificação de disponibilidade.
- /actuator/metrics → consumo de memória, GC, conexões.

2.8.7. Backup e restauração

Os dados de produção são armazenados em instância PostgreSQL gerenciada (RDS CNJ).

- **Backup automático diário** com retenção de 30 dias.
- **Restauração** via snapshot sob demanda (rds-gestaoclientes-snapshot).
- Logs de API armazenados em bucket S3: s3://gestaoclientes-logs.

2.8.8. Atualização e rollback

1. Acesso ao pipeline **Deploy Backend PRD** no GitLab.
2. Selecionar a versão desejada (vX.Y.Z) e executar *Deploy*.
3. Em caso de regressão, acionar o job **Rollback Backend**.
4. Validar o status dos pods no EKS:
5. kubectl get pods -n gestao-clientes
6. kubectl logs <pod>
7. Testar endpoint de saúde (/actuator/health).

2.8.9. Boas práticas operacionais

- **TLS obrigatório** em todas as rotas (https://gateway.cloud.pje.jus.br).
- **Secrets e senhas** gerenciados via Vault CNJ.
- **Perfis de execução** distintos (dev, stg, prd) com isolamento de dados.
- **Limpeza de cache Redis** semanal (FLUSHDB).
- **Verificação de dependências** trimestral (mvn versions:display-dependency-updates).
- **Testes automatizados** executados a cada *merge request*.

2.9. Testes

- Relatório de cobertura e resultados de testes unitários e de integração.
- Relatório de análise estática de código (SONAR).
- Evidências de conformidade com o padrão das aplicações PDPJ-BR e de observabilidade.
- Relatório de testes de capacidade.

2.10. Arquitetura

Toda a **documentação técnica, arquitetural e de desempenho** do **Sistema Gestão de Clientes (GeCli)** está integralmente versionada e disponível no repositório GitLab institucional do CNJ/PDPJ.
O diretório /docs/arquitetura contém a totalidade dos artefatos técnicos relacionados à **arquitetura, padrões de design, otimizações de performance e casos de uso** do sistema.

2.10.1. Localização da documentação

Repositório oficial:
https://git.cnj.jus.br/pdpj/negocio/gestao-clientes
Diretório de arquitetura (documentação completa):
https://git.cnj.jus.br/pdpj/negocio/gestao-clientes/-/tree/master/docs/arquitetura
Documento principal:
https://git.cnj.jus.br/pdpj/negocio/gestao-clientes/-/blob/master/docs/arquitetura/README-02.md

---

2.10.2. Estrutura e conteúdo técnico

O diretório de arquitetura concentra os artefatos que documentam a **evolução, decisões técnicas e otimizações estruturais** do sistema, conforme tabela a seguir:

Arquivo	Conteúdo
README-02.md	Documento principal com o **diagrama da**

	arquitetura, **casos de uso** e **fluxos de integração** entre Frontend Angular, Backend Spring Boot, Keycloak, Redis e Gateway PDPJ.
README-01.md	Estrutura inicial e histórico de arquitetura do projeto.
CHANGELOG_THREAD_SAFETY_IMPROVEMENTS.md	Registro de melhorias de segurança de threads e controle de concorrência.
FACTORY_PATTERN_COMPARISON.md	Comparativo entre implementações do padrão Factory aplicadas ao GeCli.
GLOBAL_EXCEPTION_HANDLER_LOGGING_IMPROVEMENTS.md	Documenta aprimoramentos no tratamento global de exceções e registro de logs.
HTTP_CLIENT_FACTORY_CACHE_COMPLETE.md	Detalha a implementação de cache de HttpClient para otimizar requisições.
HTTP_DEPENDENCIES_ANALYSIS.md / V2.md	Análise detalhada das dependências HTTP e impacto de cada versão no consumo de recursos.

KEYCLOAK_HTTP_INTERFACE_CACHE_OPTIMIZATION.md	Otimizações na comunicação HTTP com o Keycloak, incluindo cache de tokens e reuso de conexões.
KEYCLOAK_OPTIMIZATION_SUMMARY.md	Sumário consolidado de melhorias de desempenho e segurança nas integrações com o SSO CNJ.
RESTCLIENT_CONNECTION_POOL_ANALYSIS.md	Análise de pool de conexões HTTP e tuning do RestTemplate.
RESTCLIENT_CONNECTION_POOL_INVESTIGATION.md	Investigação de gargalos e uso de connection pooling em ambientes de alta carga.
SESSION_OPTIMIZATION_SUMMARY.md	Estratégias de otimização de sessão e cache distribuído no Redis.
THREAD_SAFE_FACTORY_PATTERN.md	Implementação do padrão Factory thread-safe utilizada nos serviços internos.

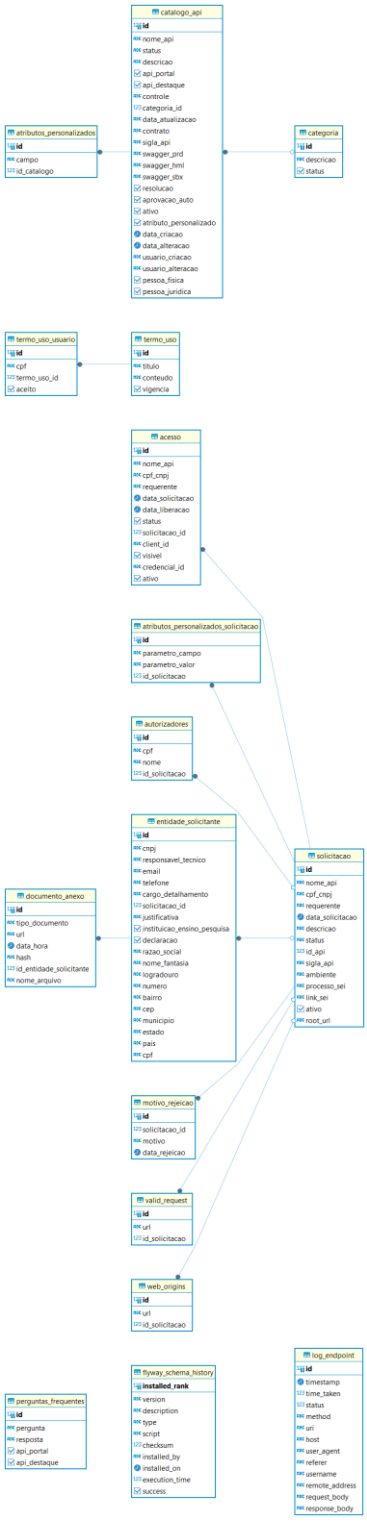
2.10.3. Casos de uso e fluxos do sistema

Os **casos de uso funcionais e técnicos** do sistema — como fluxos de solicitação, aprovação, revogação e integração com módulos PDPJ — estão descritos dentro do arquivo **README-02.md**, que também consolida:

- o **diagrama C4 da arquitetura** (Context, Container, Component e Code);
- a descrição das **camadas de aplicação** e dos componentes que compõem o sistema;
- o **modelo de autenticação e autorização** baseado em OAuth2/OpenID Connect;
- e os **pontos de integração** com sistemas externos (Domicílio Judicial Eletrônico, Corporativo e DataLake).

2.11. Banco de dados

2.11.1. Arquitetura de banco de dados.



2.11.2. Regras de negócios e triggers.

O **schema gestao_clientes**, pertencente ao banco de dados **pdpj_gestao_clientes** (PostgreSQL, instância RDS CNJ/PDPJ), **não contém triggers, funções armazenadas nem procedures**.
As **regras de negócio são integralmente implementadas na camada de aplicação**, no backend desenvolvido em **Spring Boot**, responsável por todas as validações, auditorias e garantias de integridade transacional.

2.11.2.1. Gerenciamento de versão via Flyway

A **estrutura do banco de dados** é versionada e mantida por meio do **Flyway**, ferramenta de *Database Migration* integrada ao pipeline de CI/CD do sistema. Essa ferramenta garante que a criação e evolução do schema ocorram de forma automatizada e controlada a cada publicação do backend.
A tabela **flyway_schema_history** armazena o histórico de migrações aplicadas no schema, com informações sobre versão, descrição, tipo de script e data de execução.
Exemplo de consulta:

```
SELECT version, description, type, installed_on
FROM gestao_clientes.flyway_schema_history
ORDER BY installed_rank DESC;
```

Exemplo de resultado:

Versão	Descrição	Tipo	Data de Instalação
26	Inclusão de colunas de auditoria (dt_incs, dt_altr)	SQL	2025-10-08
25	Criação de tabela entidade_solicitante	SQL	2025-09-10
24	Ajuste de constraints em solicitacao	SQL	2025-08-20

2.11.2.2. Padrão de versionamento adotado

Os scripts seguem convenção **V<versão>__<descrição>.sql**, por exemplo:
V26__ajuste_colunas_auditoria.sql
V25__criar_tabela_entidade_solicitante.sql
Cada script é executado automaticamente pelo Flyway na inicialização da aplicação e registrado em log.
Esse controle garante que todos os ambientes — **desenvolvimento, homologação e produção** — possuam a mesma estrutura de banco de dados.

2.11.2.3. Rastreamento e auditoria

A tabela **flyway_schema_history** funciona como trilha de auditoria da evolução estrutural do banco, permitindo identificar:

- A **sequência completa** de versões aplicadas;
- O **momento de aplicação** de cada alteração;
- O **responsável** (usuário do pipeline CI/CD);
- E a **compatibilidade de versão** entre backend e schema.

2.11.3. Biblioteca de Dados

tipo	nome	detalhe1	detalhe2
COLUNA	acesso.ativo	boolean	Obrigatória
COLUNA	acesso.client_id	character varying(255)	Aceita nulo
COLUNA	acesso.cpf_cnpj	character varying(100)	Aceita nulo
COLUNA	acesso.credencial_id	character varying(255)	Aceita nulo
COLUNA	acesso.data_liberacao	timestamp without time zone	Aceita nulo
COLUNA	acesso.data_solicitacao	timestamp without time zone	Aceita nulo
COLUNA	acesso.id	bigint	Obrigatória
COLUNA	acesso.nome_api	character varying(200)	Obrigatória
COLUNA	acesso.requerente	character varying(1000)	Aceita nulo
COLUNA	acesso.solicitacao_id	bigint	Obrigatória
COLUNA	acesso.status	boolean	Obrigatória

COLUNA	acesso.visivel	boolean	Obrigatória
COLUNA	atributos_personalizados.campo	character varying(255)	Aceita nulo
COLUNA	atributos_personalizados.id	integer	Obrigatória
COLUNA	atributos_personalizados.id_catalogo	bigint	Obrigatória
COLUNA	atributos_personalizados_solicitacao.id	integer	Obrigatória
COLUNA	atributos_personalizados_solicitacao.id_solicitacao	bigint	Obrigatória
COLUNA	atributos_personalizados_solicitacao.parametro_campo	character varying(255)	Obrigatória
COLUNA	atributos_personalizados_solicitacao.parametro_valor	character varying(255)	Aceita nulo
COLUNA	autorizadores.cpf	character varying(14)	Aceita nulo
COLUNA	autorizadores.id	integer	Obrigatória
COLUNA	autorizadores.id_solicitacao	bigint	Aceita nulo
COLUNA	autorizadores.nome	character varying(255)	Aceita nulo
COLUNA	catalogo_api.api_destaque	boolean	Aceita nulo
COLUNA	catalogo_api.api_portal	boolean	Aceita nulo
COLUNA	catalogo_api.aprovacao_auto	boolean	Aceita nulo
COLUNA	catalogo_api.ativo	boolean	Obrigatória
COLUNA	catalogo_api.atributo_personalizado	boolean	Obrigatória
COLUNA	catalogo_api.categoria_id	bigint	Aceita nulo
COLUNA	catalogo_api.contrato	text	Aceita nulo
COLUNA	catalogo_api.controle	character varying(20)	Aceita nulo
COLUNA	catalogo_api.data_alteracao	timestamp without time zone	Aceita nulo
COLUNA	catalogo_api.data_atualizacao	character varying(100)	Aceita nulo
COLUNA	catalogo_api.data_criacao	timestamp without time zone	Aceita nulo
COLUNA	catalogo_api.descricao	character varying(5000)	Aceita nulo
COLUNA	catalogo_api.id	integer	Obrigatória
COLUNA	catalogo_api.nome_api	character varying(100)	Obrigatória
COLUNA	catalogo_api.pessoa_fisica	boolean	Aceita nulo
COLUNA	catalogo_api.pessoa_juridica	boolean	Aceita nulo
COLUNA	catalogo_api.resolucao	boolean	Aceita nulo
COLUNA	catalogo_api.sigla_api	character varying(200)	Aceita nulo
COLUNA	catalogo_api.status	character varying(2)	Aceita nulo
COLUNA	catalogo_api.swagger_hml	character varying(300)	Aceita nulo

COLUNA	catalogo_api.swagger_prd	character varying(300)	Aceita nulo
COLUNA	catalogo_api.swagger_sbx	character varying(300)	Aceita nulo
COLUNA	catalogo_api.usuario_alteracao	character varying(255)	Aceita nulo
COLUNA	catalogo_api.usuario_criacao	character varying(255)	Aceita nulo
COLUNA	categoria.descricao	character varying(2000)	Aceita nulo
COLUNA	categoria.id	integer	Obrigatória
COLUNA	categoria.status	boolean	Aceita nulo
COLUNA	documento_anexo.data_hora	timestamp without time zone	Obrigatória
COLUNA	documento_anexo.hash	character varying(255)	Obrigatória
COLUNA	documento_anexo.id	bigint	Obrigatória
COLUNA	documento_anexo.id_entidade_solicitante	bigint	Obrigatória
COLUNA	documento_anexo.nome_arquivo	character varying(255)	Obrigatória
COLUNA	documento_anexo.tipo_documento	character varying(255)	Obrigatória
COLUNA	documento_anexo.url	text	Obrigatória
COLUNA	entidade_solicitante.bairro	character varying(100)	Aceita nulo
COLUNA	entidade_solicitante.cargo_detalhamento	character varying(255)	Aceita nulo
COLUNA	entidade_solicitante.cep	character varying(20)	Aceita nulo
COLUNA	entidade_solicitante.cnpj	character varying(18)	Aceita nulo
COLUNA	entidade_solicitante.cpf	character varying(20)	Aceita nulo
COLUNA	entidade_solicitante.declaracao	boolean	Aceita nulo
COLUNA	entidade_solicitante.email	character varying(255)	Aceita nulo
COLUNA	entidade_solicitante.estado	character varying(50)	Aceita nulo
COLUNA	entidade_solicitante.id	bigint	Obrigatória
COLUNA	entidade_solicitante.instituicao_ensino_pesquisa	boolean	Aceita nulo
COLUNA	entidade_solicitante.justificativa	character varying(10000)	Aceita nulo
COLUNA	entidade_solicitante.logradouro	character varying(255)	Aceita nulo
COLUNA	entidade_solicitante.municipio	character varying(100)	Aceita nulo
COLUNA	entidade_solicitante.nome_fantasia	character varying(255)	Aceita nulo

COLUNA	entidade_solicitante.numero	character varying(20)	Aceita nulo
COLUNA	entidade_solicitante.pais	character varying(50)	Aceita nulo
COLUNA	entidade_solicitante.razao_social	character varying(255)	Aceita nulo
COLUNA	entidade_solicitante.responsavel_tecnico	character varying(255)	Aceita nulo
COLUNA	entidade_solicitante.solicitacao_id	bigint	Aceita nulo
COLUNA	entidade_solicitante.telefone	character varying(20)	Aceita nulo
COLUNA	flyway_schema_history.checksum	integer	Aceita nulo
COLUNA	flyway_schema_history.description	character varying(200)	Obrigatória
COLUNA	flyway_schema_history.execution_time	integer	Obrigatória
COLUNA	flyway_schema_history.installed_by	character varying(100)	Obrigatória
COLUNA	flyway_schema_history.installed_on	timestamp without time zone	Obrigatória
COLUNA	flyway_schema_history.installed_rank	integer	Obrigatória
COLUNA	flyway_schema_history.script	character varying(1000)	Obrigatória
COLUNA	flyway_schema_history.success	boolean	Obrigatória
COLUNA	flyway_schema_history.type	character varying(20)	Obrigatória
COLUNA	flyway_schema_history.version	character varying(50)	Aceita nulo
COLUNA	log_endpoint.host	character varying(255)	Obrigatória
COLUNA	log_endpoint.id	integer	Obrigatória
COLUNA	log_endpoint.method	character varying(50)	Obrigatória
COLUNA	log_endpoint.referer	text	Aceita nulo
COLUNA	log_endpoint.remote_address	character varying(50)	Obrigatória
COLUNA	log_endpoint.request_body	text	Aceita nulo
COLUNA	log_endpoint.response_body	text	Aceita nulo
COLUNA	log_endpoint.status	integer	Obrigatória
COLUNA	log_endpoint.time_taken	bigint	Obrigatória
COLUNA	log_endpoint.timestamp	timestamp without time zone	Obrigatória
COLUNA	log_endpoint.uri	text	Obrigatória
COLUNA	log_endpoint.user_agent	text	Obrigatória
COLUNA	log_endpoint.username	character varying(255)	Aceita nulo
COLUNA	motivo_rejeicao.data_rejeicao	date	Aceita nulo
COLUNA	motivo_rejeicao.id	bigint	Obrigatória
COLUNA	motivo_rejeicao.motivo	text	Obrigatória
COLUNA	motivo_rejeicao.solicitacao_id	bigint	Obrigatória
COLUNA	perguntas_frequentes.api_destaque	boolean	Aceita nulo

COLUNA	perguntas_frequentes.api_portal	boolean	Aceita nulo
COLUNA	perguntas_frequentes.id	integer	Obrigatória
COLUNA	perguntas_frequentes.pergunta	character varying(800)	Obrigatória
COLUNA	perguntas_frequentes.resposta	character varying(1500)	Obrigatória
COLUNA	solicitacao.ambiente	character varying(4)	Aceita nulo
COLUNA	solicitacao.ativo	boolean	Obrigatória
COLUNA	solicitacao.cpf_cnpj	character varying(100)	Aceita nulo
COLUNA	solicitacao.data_solicitacao	timestamp without time zone	Aceita nulo
COLUNA	solicitacao.descricao	character varying(2000)	Aceita nulo
COLUNA	solicitacao.id	integer	Obrigatória
COLUNA	solicitacao.id_api	bigint	Aceita nulo
COLUNA	solicitacao.link_sei	character varying(200)	Aceita nulo
COLUNA	solicitacao.nome_api	character varying(200)	Obrigatória
COLUNA	solicitacao.processo_sei	character varying(20)	Aceita nulo
COLUNA	solicitacao.requerente	character varying(1000)	Aceita nulo
COLUNA	solicitacao.root_url	character varying(200)	Aceita nulo
COLUNA	solicitacao.sigla_api	character varying(100)	Aceita nulo
COLUNA	solicitacao.status	character varying(20)	Obrigatória
COLUNA	termo_uso.conteudo	text	Obrigatória
COLUNA	termo_uso.id	bigint	Obrigatória
COLUNA	termo_uso.titulo	character varying(200)	Obrigatória
COLUNA	termo_uso.vigencia	boolean	Obrigatória
COLUNA	termo_uso_usuario.aceito	boolean	Obrigatória
COLUNA	termo_uso_usuario.cpf	character varying(14)	Obrigatória
COLUNA	termo_uso_usuario.id	bigint	Obrigatória
COLUNA	termo_uso_usuario.termo_uso_id	bigint	Obrigatória
COLUNA	valid_request.id	bigint	Obrigatória
COLUNA	valid_request.id_solicitacao	bigint	Aceita nulo
COLUNA	valid_request.url	character varying(200)	Obrigatória
COLUNA	web_origins.id	bigint	Obrigatória
COLUNA	web_origins.id_solicitacao	bigint	Aceita nulo

COLUNA	web_origins.url	character varying(200)	Obrigatória
FOREIGN KEY	acesso.solicitacao_id	→ solicitacao.id	
FOREIGN KEY	atributos_personalizados.id_catalogo	→ catalogo_api.id	
FOREIGN KEY	atributos_personalizados_solicitacao.id_solicitacao	→ solicitacao.id	
FOREIGN KEY	autorizadores.id_solicitacao	→ solicitacao.id	
FOREIGN KEY	catalogo_api.categoria_id	→ categoria.id	
FOREIGN KEY	documento_anexo.id_entidade_solicitante	→ entidade_solicitante.id	
FOREIGN KEY	entidade_solicitante.solicitacao_id	→ solicitacao.id	
FOREIGN KEY	motivo_rejeicao.solicitacao_id	→ solicitacao.id	
FOREIGN KEY	termo_uso_usuario.termo_uso_id	→ termo_uso.id	
FOREIGN KEY	valid_request.id_solicitacao	→ solicitacao.id	
FOREIGN KEY	web_origins.id_solicitacao	→ solicitacao.id	
TABELA	acesso		
TABELA	atributos_personalizados		
TABELA	atributos_personalizados_solicitacao		
TABELA	autorizadores		
TABELA	catalogo_api		
TABELA	categoria		
TABELA	documento_anexo		
TABELA	entidade_solicitante		
TABELA	flyway_schema_history		
TABELA	log_endpoint		
TABELA	motivo_rejeicao		
TABELA	perguntas_frequentes		
TABELA	solicitacao		
TABELA	termo_uso		
TABELA	termo_uso_usuario		
TABELA	valid_request		
TABELA	web_origins		

3. Procedimentos operacionais

Início e parada do sistema: Procedimentos específicos para iniciar e parar o sistema de forma segura.

Backup: Procedimentos de backup dos dados, frequência, mídia de armazenamento e testes de recuperação.

Restauração: Procedimentos para restaurar o sistema em caso de falha.

Monitoramento: Ferramentas e procedimentos para monitorar o desempenho do sistema.

Manutenção: Rotinas de manutenção preventiva e corretiva.

4. Segurança da informação

Políticas de segurança: Políticas de acesso, senhas, backup, etc.

Medidas de segurança: Firewalls, antivírus, sistemas de detecção de intrusão, criptografia.

Planos de contingência: Planos para lidar com incidentes de segurança e recuperação de dados.

7. Glossário

Definição de termos técnicos utilizados no manual

Termo / Sigla	Definição
AWS (Amazon Web Services)	Plataforma de serviços de computação em nuvem utilizada para hospedagem do banco de dados e dos ambientes da PDPJ.
Backend	Camada de aplicação responsável pelo processamento das regras de negócio e pela comunicação com o banco de dados. No Gestão de Clientes, é desenvolvido em **Java (Spring Boot)**.
Banco de Dados PostgreSQL	Sistema gerenciador de banco de dados relacional utilizado para armazenar as informações estruturadas do sistema.
DBeaver	Ferramenta utilizada para acessar e documentar o banco de dados, gerar diagramas e extrair metadados do schema gestao_clientes.
DISI / SEGSI	Divisão de Segurança da Informação do CNJ, responsável por supervisionar e aprovar as práticas de segurança aplicadas nos sistemas sob sua gestão.
Docker	Plataforma de virtualização em contêineres utilizada para empacotar e executar os componentes do sistema localmente ou em ambiente cloud.
Flyway	Ferramenta de migração e versionamento de banco de dados, responsável por aplicar scripts incrementais e controlar a evolução estrutural do schema.
Frontend (Angular)	Camada de apresentação e interface com o usuário, desenvolvida em Angular, responsável pela interação com as APIs do backend.
Gateway PDPJ	Componente de roteamento da **Plataforma Digital do Poder Judiciário (PDPJ)** que intermedia as requisições entre o usuário e os serviços internos.
GitLab	Plataforma de versionamento e automação de código utilizada para controle de versões, pipelines e integração contínua (CI/CD).
Homologação (HMG)	Ambiente destinado à validação técnica e funcional das entregas antes da publicação em produção.

Termo / Sigla	Definição
JWT (JSON Web Token)	Token de autenticação utilizado pelo sistema para validar sessões e autorizações de acesso.
Keycloak	Solução de autenticação e autorização baseada em Single Sign-On (SSO) utilizada pelo CNJ para gerenciar identidades e permissões de acesso.
Pipeline CI/CD	Conjunto automatizado de etapas que realizam a integração, teste e implantação contínua do código-fonte.
Redis	Banco de dados em memória utilizado como cache de sessões e filas de processamento para otimização de desempenho.
Schema	Conjunto lógico de tabelas, relacionamentos e objetos do banco de dados, neste caso denominado gestao_clientes.
Spring Boot	Framework Java utilizado para construção do backend e exposição de APIs REST no sistema Gestão de Clientes.
Swagger	Interface web para documentação e teste das APIs disponíveis no backend.
Trigger	Mecanismo de banco de dados que executa comandos automaticamente em resposta a eventos (INSERT, UPDATE, DELETE). Não há triggers implementadas neste schema.

8. Apêndices

Informações adicionais e artefatos técnicos complementares

Esta seção reúne documentos e artefatos técnicos gerados durante o ciclo de desenvolvimento e operação do sistema **Gestão de Clientes**, com o objetivo de garantir rastreabilidade, suporte à manutenção e continuidade operacional.

2.11.4. Diagramas e documentação técnica

Documento	Descrição	Localização
Diagrama de Arquitetura do Sistema	Representação da estrutura geral do sistema, contendo os componentes Frontend (Angular), Backend (Spring Boot), Keycloak, Redis, Gateway PDPJ e PostgreSQL.	Diretório /docs/arquitetura/README-02.md – repositório GitLab CNJ/PDPJ
Diagrama ER – Banco de Dados	Modelo relacional das entidades do schema gestao_clientes, gerado via DBeaver.	Arquivo ERD_gestao_clientes.png
Docker Compose	Definição dos serviços locais utilizados para execução e testes integrados do sistema (PostgreSQL, Redis, Keycloak, RabbitMQ, etc.).	Arquivo docker-compose.yml
Relatório de Pentest	Documento emitido pela empresa ISH Tecnologia S.A., formalizado no protocolo SEI 00131/2020 – OS 2373782, com resultados do teste de invasão realizado no ambiente de homologação.	Anexo no processo SEI correspondente
Histórico de Migrações Flyway	Registro de todas as versões aplicadas no schema gestao_clientes.	Tabela flyway_schema_history – Banco de Dados RDS AWS