AURE — Módulo de Contrato Digital Off-Chain (Fase 1)

# 1️⃣ Contexto e Motivação Técnica

O módulo de **Contrato Digital Off-Chain** representa a **camada fundacional de confiança da plataforma AURE**.  
Ele é responsável por substituir processos manuais e cartoriais por fluxos digitais auditáveis, onde cada contrato tem **prova de integridade (hash criptográfico)**, **rastreabilidade ponta a ponta**, e **registro público imutável na blockchain**.

Na **Fase 1 (MVP)**, o sistema opera em modo **off-chain**, com foco em **prova jurídica e integridade digital**, sem execução financeira dentro da blockchain.  
Os contratos são:

1. Criados e assinados digitalmente pelas partes;
2. Armazenados de forma descentralizada no **IPFS (InterPlanetary File System)**;
3. E têm seu **hash (SHA-256)** registrado na **blockchain Polygon Testnet**, atuando como um **cartório digital público**.

Essa decisão técnica foi tomada por três razões principais:

* **1. Redução de complexidade e custo:** o registro de hash é suficiente para garantir autenticidade e integridade, evitando custos de gas e sobrecarga de manutenção de smart contracts complexos.
* **2. Base sólida para evolução:** a arquitetura modular e API-first permite, em fases futuras, migrar o fluxo financeiro para **smart contracts de split/escrow** sem reescrever o núcleo.
* **3. Aderência jurídica:** o uso de hash + timestamp + blockchain garante validade legal conforme **MP 2.200-2/2001 (ICP-Brasil)** e jurisprudência vigente.

# 2️⃣ Objetivo do Módulo

**Propósito:**  
Estabelecer o núcleo de confiança da plataforma AURE, permitindo a criação, assinatura e registro imutável de contratos digitais entre empresas e prestadores de serviço (PJs), com integridade técnica, validade jurídica e rastreabilidade total.

O módulo de Contrato Digital é o **primeiro bloco funcional** da AURE e serve como base para todos os demais componentes (pagamentos, split, escrow, compliance e BI).

## ****Descrição Funcional Expandida****

O **Contrato Digital da AURE** deve possibilitar:

1. **Criação de contratos digitais baseados em modelos jurídicos padronizados**, pré-aprovados pelo setor jurídico e versionados internamente.
   * Os modelos devem conter campos dinâmicos (placeholders) preenchidos via interface ou API.
   * Deve haver histórico de versões (contract\_versions) com diffs automáticos para auditoria.
   * Possibilidade de geração de contratos **multijurisdicionais** (PT-BR, EN-US, etc.).
2. **Garantia de integridade via hash SHA-256.**
   * Cada PDF gerado deve ter uma “impressão digital” única (hash) calculada localmente.
   * Qualquer alteração mínima no conteúdo gera um hash diferente.
   * O hash é armazenado no banco de dados e publicado na blockchain, garantindo imutabilidade.
3. **Armazenamento descentralizado no IPFS.**
   * O contrato finalizado é enviado a um nó IPFS (via Web3.Storage/Pinata).
   * O IPFS retorna um CID (Content Identifier), que é o endereço público e imutável do arquivo.
   * O sistema valida o upload comparando o hash local com o hash retornado pelo IPFS.
4. **Registro do hash e metadados na blockchain Polygon.**
   * A AURE publica o hash, o CID e os endereços das partes em um **smart contract Registry** na **Polygon Testnet (Amoy)**.
   * Esse registro serve como **prova pública e inviolável** da existência e autenticidade do contrato.
   * Nenhum dado pessoal é gravado on-chain (apenas hash e CID).
5. **Captura de assinaturas digitais de empresa e PJ.**
   * Cada parte deve autenticar-se com login seguro (JWT/MFA) antes de assinar.
   * A assinatura gera um hash vinculado ao IP, timestamp e ID do contrato.
   * O status do contrato evolui automaticamente: draft → sent\_to\_sign → partially\_signed → fully\_signed → registered\_onchain.
   * Fase 2 poderá integrar assinaturas avançadas ICP-Brasil/WebAuthn.
6. **Auditoria, versionamento e certificação.**
   * Cada ação (criação, assinatura, registro) gera um log imutável em audit\_logs.
   * O sistema deve emitir um **Certificado de Autenticidade** em PDF contendo:
     + hash (SHA-256), CID (IPFS), hash da transação blockchain, data e hora da assinatura e QR Code de verificação.
   * O certificado deve ser publicamente verificável via /contracts/{id}/verify.

## ****Justificativa Técnica e de Negócio****

| **Pilar** | **Benefício** | **Justificativa** |
| --- | --- | --- |
| **Integridade (SHA-256)** | Prova técnica de autenticidade | Evita falsificação e garante validade probatória em auditorias e litígios. |
| **Descentralização (IPFS)** | Redução de dependência de armazenamento centralizado | Permite prova de existência pública e reduz risco de perda de dados. |
| **Blockchain (Polygon)** | Imutabilidade e transparência | Registro imutável que serve como “carimbo digital” com validade jurídica. |
| **Assinatura Digital** | Confiança e responsabilidade legal | Garante consentimento bilateral e não repúdio (juridicamente defensável). |
| **Auditoria e Certificação** | Rastreabilidade completa | Reforça a confiança do sistema e simplifica revisões legais e fiscais. |

## ****Critérios de Sucesso (DoD — Definition of Done)****

| **Nº** | **Critério** | **Métrica de Validação** |
| --- | --- | --- |
| 1 | Contrato criado e assinado digitalmente | Fluxo fim a fim concluído em ≤5 minutos |
| 2 | Hash (SHA-256) válido e registrado na blockchain | 100% dos contratos possuem hash confirmado on-chain |
| 3 | CID IPFS acessível publicamente | ≥99,5% de disponibilidade do arquivo |
| 4 | Assinaturas registradas com trilha de auditoria | 100% das ações registradas em audit\_logs |
| 5 | Certificado PDF gerado e verificável | /verify retorna status “valid” |
| 6 | Nenhum dado pessoal armazenado on-chain | Validado por auditoria LGPD |

## ****Riscos Técnicos e Mitigações****

| **Risco** | **Impacto** | **Mitigação** |
| --- | --- | --- |
| Falha no upload IPFS | Alta | Retries com backoff + DLQ (BullMQ/SQS) |
| Hash divergente (local vs IPFS) | Alta | Verificação checksum obrigatória antes do registro |
| RPC Blockchain indisponível | Média | Circuit breaker + fallback provider (Infura/Alchemy) |
| Assinaturas inválidas ou duplicadas | Média | Idempotência + verificação de hash + bloqueio por status |
| Falha no PDF generator | Baixa | Fallback para template HTML simples + retry automático |

## ****Dependências e Interações****

* **Frontend (React/Tailwind):** formulário de criação, upload, assinatura e verificação.
* **API Core (NestJS/FastAPI):** controle de estados, geração de PDF, hash e persistência.
* **IPFS Adapter:** upload e validação de conteúdo.
* **Blockchain Adapter:** registro e leitura de provas (Ethers.js).
* **Audit Service:** logging e versionamento de contratos.
* **Notification Service:** envio de e-mails em cada transição de status.

## ****Resumo Executivo****

O **módulo de Contrato Digital da AURE** é a **coluna vertebral da plataforma**, projetado para garantir confiança técnica e jurídica em todo o ciclo de contratação.  
Ele substitui a burocracia manual por automação rastreável e auditável, formando a base sobre a qual serão construídas as camadas futuras de automação financeira, split de pagamento e inteligência de compliance.

# 3️⃣ Arquitetura de Software



**Por quê assim?**

* **BFF/Gateway** simplifica o front e concentra cross-cutting (JWT, RBAC, idempotência).
* **Services separados** (Contracts, Registry, Payments, Notify) evitam acoplamento; você escala só o que precisa.
* **Eventos assíncronos** (Kafka/SQS) dão resiliência (retries, backoff) para IPFS/blockchain/e-mail.
* **Redis** para cache, locks e idempotência.
* **PostgreSQL** central para contratos/assinaturas; **S3** para PDFs; **IPFS** para prova pública.

## 2) Componentes (Deployment View)

* **Web App (React/Tailwind)**: SPA, chama o BFF.
* **API Gateway/BFF (NestJS)**: autentica JWT, aplica RBAC, valida payloads, rate limiting, roteia para serviços.
* **Auth Service**: emissão/refresh de JWT; (futuro) OAuth2/KYC.
* **Contracts Service**: templates, merge de campos, geração de **PDF**, cálculo **SHA-256**, versionamento, persistência; expõe /contracts/\*.
* **Registry Service**: sobe **PDF para S3**, publica **CID no IPFS**, grava **hash+CID** no smart contract (Polygon Amoy), retorna txHash.
* **Payments Service** (Fase 1): integra Pix API (split off-chain), ledger interno (idempotency key).
* **NFS-e Adapter** (stub no MVP): interface única para provedores municipais (troca de driver por cidade).
* **Notify/Events**: envia e-mail (SES/SendGrid), in-app feed; consome tópicos: contract.sent, contract.signed, contract.registered.
* **Audit/Logs**: ELK (Elastic/Kibana) + export crítico para IPFS (logs de assinatura e registro).
* **Infra**: PostgreSQL (RDS), Redis (ElastiCache), S3 (PDFs e backups), Secrets Manager (chaves IPFS/Alchemy/SES), Prometheus+Grafana, OpenTelemetry.

Ambientes: **dev → staging → prod** (chaves/IPFS/chain separadas; feature flags).3) Principais fluxos (Sequence View)

### 3.1 Criar/Finalizar/Registrar Contrato

Front → BFF: POST /contracts

BFF → ContractsSvc: cria draft (DB)

Front → BFF: PUT /contracts/{id}/finalize (dados finais)

BFF → ContractsSvc: gera PDF → SHA-256 → salva S3 → grava doc\_hash (DB)

           ↳ emite EVENT contract.finalized

BFF → RegistrySvc: POST /register {doc\_hash, s3\_url}

RegistrySvc → IPFS: upload → CID

RegistrySvc → Blockchain: createAgreement(hash, CID, partes) → txHash

RegistrySvc → DB: salva CID/txHash

           ↳ emite EVENT contract.registered\_onchain

BFF → Front: status=registered + links (IPFS/Polygonscan)

### 3.2 Assinatura bilateral

Front(PJ/Empresa) → BFF: POST /contracts/{id}/sign

BFF → ContractsSvc: salva assinatura (hash assinatura, IP, UA, timestamp)

           ↳ se 1/2: status partially\_signed; EVENT contract.partially\_signed

           ↳ se 2/2: status fully\_signed;   EVENT contract.fully\_signed

### 3.3 Pagamento off-chain (gatilhado por nota aprovada)

NFS-e Adapter → EVENT invoice.approved

PaymentsSvc (consumer) → calcula split → chama Pix API (idempotencyKey)

→ salva ledger → EVENT payment.settled | payment.failed

NotifySvc envia e-mail/in-app

## 4) Esquema de dados (Physical View – MVP)

**contracts** id, company\_id, pj\_id, template\_id, service\_desc, amount\_total, currency, due\_date, split\_rules(jsonb), doc\_hash, ipfs\_cid, s3\_url, onchain\_tx, status(enum), version, created\_at, updated\_at

**contract\_signatures** id, contract\_id, signer\_id, role(enum), signature\_hash, ip, user\_agent, signed\_at

**audit\_logs** id, entity, entity\_id, action, actor\_id, payload\_hash, created\_at

**payments\_ledger (off-chain)** id, contract\_id, invoice\_id, payee\_id, amount, currency, pix\_txid, status(enum), idempotency\_key, created\_at

Índices: (doc\_hash), (company\_id, status), (contract\_id); partial index para status IN ('sent\_to\_sign','fully\_signed').

## 5) Integrações externas

* **IPFS**: Web3.Storage/Pinata (SDK + retries/backoff exponencial; checksum após upload).
* **Blockchain**: Polygon Amoy via **Alchemy/Infura** + Ethers.js; contrato “Registry” enxuto (somente hash/CID/eventos).
* **Pix API**: parceiro bancário (sandbox), chaves no Secrets Manager, **idempotency key** por operação.
* **Emails**: SendGrid/SES (templates versionados), dead-letter queue para falhas.

## 6) Segurança (cross-cutting)

* **Autenticação**: JWT curto (60min) + refresh (24h).
* **Autorização (RBAC)**: Empresa cria/edita/enviam; PJ assina; Admin leitura/auditoria.
* **Criptografia**: TLS 1.2+; dados sensíveis AES-256 em repouso (S3 com SSE, RDS com TDE).
* **Idempotência**: header Idempotency-Key no BFF para /finalize, /sign, /register, /payments.
* **Anti-replay**: nonce/expiração em links de assinatura.
* **Auditoria**: 100% de ações sensíveis logadas; snapshot crítico para IPFS (hash do log).
* **LGPD**: minimização de dados em e-mail; direito de export/deleção controlado.

## 7) Resiliência e performance

* **Circuit breaker** para IPFS/Blockchain/Pix (fallback e fila de reprocesso).
* **Retries**: 3 tentativas com backoff (1m/5m/30m).
* **Timeouts**: HTTP externos 5s; blockchain 15s.
* **Fila assíncrona**: registro on-chain/ e-mail nunca bloqueiam UX.
* **Cache**: Redis para listas e certificados públicos.
* **Escala**: horizontal por stateless services (BFF, Contracts, Registry, Notify).
* **SLOs**: P99 APIs < 300ms (internas), upload+registro ≤ 10s médios.

## 8) Decisões e trade-offs (ADR resumido)

* **Registry minimalista**: só hash/CID/eventos → baixo gas, menor superfície de ataque.
* **Pagamentos off-chain no MVP**: reduz complexidade jurídica/UX; mantém prova on-chain do contrato.
* **S3 + IPFS**: S3 como storage operacional; IPFS para **prova pública** (CID imutável).
* **Eventos assíncronos**: garantem UX fluida e reprocessos confiáveis.
* **BFF único**: acelera front; pode virar **API Gateway** gerenciado na fase 2.

## 9) Mapa de endpoints (BFF → Services)

**Contracts**

* POST /contracts → ContractsSvc.createDraft
* PUT /contracts/{id}/finalize → ContractsSvc.finalize (PDF, SHA-256, S3)
* POST /contracts/{id}/sign → ContractsSvc.sign (gera signature\_hash)
* POST /contracts/{id}/register → RegistrySvc.register (IPFS + chain)
* GET /contracts/{id} → ContractsSvc.getOne
* GET /contracts/{id}/certificate → ContractsSvc.certificate (hash, CID, tx)

**Public verify**

* GET /verify/{contractId}/{token} → certificado público (sem dados pessoais)

## 10) Roadmap técnico (infra/ops)

* **CI/CD**: GitHub Actions → build/test → Docker → deploy (ECS/Kubernetes).
* **IaC**: Terraform (VPC, RDS, ElastiCache, S3, IAM, ECS/EKS).
* **Monitoring**: Prometheus/Grafana; logs centralizados (ELK); alertas (Slack/Email).
* **Backups**: RDS snapshots diários; S3 versionado; chave IPFS “pinned”.
* **Runbooks**: reprocesso de contract.registered\_onchain e filas de e-mail.

## 11) Preparado para Fase 2/3 (evolução on-chain)

* **Adicionar Escrow/Split** como **novo serviço** (não altera Contracts/Registry).
* **Substituir Pix** por **stablecoin** em smart contracts (USDC/BRLx) mantendo o mesmo **ledger** e eventos.
* **KYC/KYB** e **OAuth2** no Auth Service sem impacto nos outros domínios.

# Arquitetura de Software — AURE (Fase 1: Contratos Off-Chain)

## 1) Princípios de Arquitetura

* **API-first & modular**: contratos, assinaturas, storage/IPFS, blockchain e pagamentos são serviços independentes.
* **Segurança por camadas**: JWT + RBAC, criptografia em repouso (AES-256), TLS obrigatório, segredos em cofre.
* **Imutabilidade e trilha de auditoria**: tudo que muda estado gera log com hash; contrato final tem SHA-256 + CID IPFS.
* **Resiliência**: idempotência, retries com backoff, filas assíncronas para integrações (IPFS e blockchain).
* **Observabilidade**: métricas (latência/erro), logs estruturados, tracing distribuído.

## 2) Visão Lógica (macro)

[Web/App React] ──> [BFF/Edge API (opcional)] ──> [Core API AURE]

                                         ├─> Contract Service

                                         ├─> Signature Service

                                         ├─> File/Hash Service

                                         ├─> IPFS Adapter (pinning)

                                         ├─> Blockchain Adapter (Registry)

                                         ├─> Notification Service (email/in-app)

                                         ├─> Payments Orchestrator (off-chain)

                                         └─> Audit/Log Service

          └───────────────(Auth)─────────────> Auth Service (JWT/RBAC)

[PostgreSQL]   [Object Storage]   [Queue/Worker]     [Secrets/KMS]     [Monitoring]

  contratos     pdfs/artefatos     jobs IPFS/chain     chaves/API        logs/metrics

**Observações**

* **Core API AURE** (NestJS/FastAPI) expõe endpoints REST.
* **Adapters** encapsulam chamadas a IPFS (Web3.Storage/Pinata) e blockchain (Ethers.js → Polygon Amoy).
* **Queue/Worker** processa tarefas de longa duração (upload/pin, on-chain, e-mails) fora do request/response.

## 3) Componentes e Responsabilidades

**Frontend (React/Tailwind)**

* Telas: login, contratos, assinatura, certificado, auditoria.
* Consome apenas **BFF/Edge API** (opcional) ou Core API AURE.
* Centro de notificações (in-app).

**BFF/Edge API (opcional)**

* Cache leve de leitura, agregação de endpoints p/ reduzir roundtrips.
* Normaliza payloads para o frontend.

**Auth Service**

* Login, refresh tokens, rotas públicas/privadas, **RBAC** (Empresa, PJ, Financeiro, Jurídico, Admin).

**Contract Service**

* CRUD de contratos, geração do PDF, versionamento, estados.
* Calcula **SHA-256** do PDF final.

**Signature Service**

* Registra assinaturas (hash da assinatura + IP + user-agent + timestamp).
* Regra: só vira **fully\_signed** após ambas as partes.

**File/Hash Service**

* Gera PDF (template + variáveis), calcula hash, salva artefatos em **Object Storage** (S3/GCS).

**IPFS Adapter**

* Faz **pin** do PDF final e retorna **CID**.
* Reprocessa falhas via **Queue**.

**Blockchain Adapter (Registry)**

* Publica {docHash, cid, parties, ts} no smart contract simples (registro).
* Guarda tx\_hash e emite evento de sucesso/erro.

**Payments Orchestrator (off-chain)**

* (Fase 1) Apenas estrutura e contrato de interface — cálculo de split, integração Pix para quando for ativado.
* Ledger interno de transações (mesmo ainda que sandbox).

**Notification Service**

* In-app + e-mail (SES/SendGrid). Tópicos: contract.sent, contract.signed, contract.registered\_onchain.

**Audit/Log Service**

* Persistência de logs imutáveis (evento, ator, payload\_hash, timestamp).

## 4) Fluxos principais (sequência)

### 4.1 Criação → Hash → IPFS → Registro on-chain

Frontend → Core API: POST /contracts

Core API → Contract Service: cria rascunho (status=draft)

Frontend → /contracts/{id}/finalize

Contract Service → File/Hash: gera PDF + SHA-256

File/Hash → Object Storage: guarda PDF

Core API → Queue: Job "pin-ipfs" (idContrato, fileUrl)

Worker(IPFS) → IPFS Adapter: pin(PDF) → CID

Worker → Contract Service: salva CID, status=sent\_to\_sign

Notification: EVT contract.sent\_to\_sign (PJ)

Assinaturas:

PJ: POST /contracts/{id}/sign → status=partially\_signed

Empresa: POST /contracts/{id}/sign → status=fully\_signed

Core API → Queue: Job "register-onchain" (docHash, CID, partes)

Worker(Blockchain) → Registry: createAgreement(hash, cid, parties) → tx\_hash

Worker → Contract Service: onchain\_tx, status=registered/active

Notification: EVT contract.registered\_onchain

### 4.2 Verificação de integridade

GET /contracts/{id}/verify

→ Core API baixa PDF do Object Storage (ou gateway IPFS), recalcula SHA-256 e compara com doc\_hash

→ Retorna OK/FAIL + detalhes

## 5) Modelo de Dados (essencial)

**contracts**

* id, company\_id, pj\_id, service\_desc, amount\_total, currency, due\_date
* **doc\_hash**(sha256), **ipfs\_cid**, **onchain\_tx**
* status(draft|sent\_to\_sign|partially\_signed|fully\_signed|registered|active|terminated)
* created\_at, updated\_at, **version**

**contract\_signatures**

* id, contract\_id, signer\_id, role(company|pj), signature\_hash, ip, user\_agent, signed\_at

**audit\_logs**

* id, entity, entity\_id, action, actor\_id, payload\_hash, created\_at

**notifications**

* id, user\_id, type, payload, status(new|read|archived), created\_at

*(Object Storage)*

* /contracts/{id}/v{n}/document.pdf (fonte da verdade para geração/validação do hash).

## 6) Integrações (abstraídas por adapters)

* **IPFS**: Web3.Storage/Pinata. Guardar CID no contrato.
* **Blockchain**: Ethers.js/Alchemy/Infura → **Polygon Amoy (testnet)**. Contrato **Registry** minimalista (somente provas).
* **E-mail**: SES/SendGrid. In-app obrigatório.
* **Pix/Banking**: interface prevista no Orchestrator (ativar pós-MVP).

## 7) Segurança

* **Auth**: JWT curto (1h) + refresh (24h), **RBAC** estrito.
* **TLS** em todo tráfego; CORS restrito.
* **Segredos** em **Secrets Manager/KMS**.
* **Criptografia** de dados sensíveis (AES-256 at rest).
* **Idempotency-Key** em endpoints críticos (finalize, sign, onchain).
* **Rate limiting** e **lockout** (3 tentativas de login).

## 8) Resiliência & Performance

* **Queues** (BullMQ/SQS/CloudTasks) para IPFS e Registry; retries 1m/5m/30m (máx 3).
* **Timeouts** externos: IPFS 5s, chain 15s; circuit-breaker para provedores.
* **Cache** de leituras (CID/tx\_hash) 60s (Redis) no BFF.
* **P95 API < 300ms** nas rotas síncronas (sem contar jobs).

## 9) Observabilidade

* **Logs estruturados** (JSON) com correlação (traceId).
* **Métricas**: latência/erro por endpoint, fila (enfileirados, sucesso, falha), tempo IPFS/chain.
* **Tracing**: OpenTelemetry (Core API + Workers).
* **Dashboards**: Grafana/Looker para KPIs técnicos.

## 10) Ambientes & CI/CD

* **Ambientes**: dev (sandbox), staging (dados de teste realistas), prod.
* **CI/CD**: testes (unit/integration), lint, SAST/Dependabot, migrations automáticas (safe), deploy blue/green.
* **Feature flags** para liberar “register on-chain” por contrato.

## 11) Stak recomendada

* **Frontend**: React + Vite + Tailwind; Zustand/Redux; Axios/React Query.
* **BFF (opcional)**: Node + Fastify.
* **Core API**: Node **NestJS** (ou Python **FastAPI**).
* **DB**: PostgreSQL 14+; **Redis** (cache/locks).
* **Queues**: BullMQ (Redis) ou SQS.
* **Storage**: S3/GCS.
* **IPFS**: Web3.Storage/Pinata SDK.
* **Blockchain**: Hardhat + Ethers.js; Polygon Amoy testnet.
* **Infra**: AWS (ECS Fargate/Lambda), CloudFront, WAF, Secrets Manager, KMS, CloudWatch.

## 12) Roadmap de evolução (resumo)

* **Fase 2**: multi-prefeituras NFS-e; **KYC/KYB**; cláusulas dinâmicas avançadas; APIs de export.
* **Fase 3**: **Escrow/Split on-chain** (stablecoin), orquestração por marcos; BI preditivo.

## 13) Sequência de endpoints usados no fluxo

1. POST /contracts → cria rascunho
2. PUT /contracts/{id}/finalize → gera PDF + SHA-256 + job IPFS
3. POST /contracts/{id}/sign (PJ) → partially\_signed
4. POST /contracts/{id}/sign (Empresa) → fully\_signed
5. POST /contracts/{id}/register → job registry on-chain
6. GET /contracts/{id}/certificate → hash, CID, tx\_hash, signers
7. GET /contracts/{id}/verify → recalcula hash e valida

## 14) Diagrama textual de implantação (deploy)

[Client]

   ↕ HTTPS (TLS)

[CloudFront/WAF]

   ↕

[Edge/BFF (opcional)]

   ↕

[Core API (ECS/Lambda)]

   ├─ PostgreSQL (RDS)

   ├─ Redis (Elasticache)

   ├─ S3 (artefatos PDF)

   ├─ Queue (SQS/BullMQ)

   ├─ Secrets (KMS/Secrets Manager)

   ├─ IPFS Gateway (Web3.Storage/Pinata)

   └─ Alchemy/Infura (Polygon Amoy RPC)

# ****4️⃣ Requisitos Funcionais e Não Funcionais (Versão Completa e Expandida)****

**Propósito:**  
Estabelecer de forma detalhada as funcionalidades obrigatórias e os critérios técnicos de qualidade que o sistema AURE deve atender para garantir **confiabilidade, segurança, escalabilidade e aderência legal** no módulo de **Contrato Digital Off-Chain**.

## ****4.1. Requisitos Funcionais (RF)****

| **Código** | **Título** | **Descrição Técnica e Objetivo de Negócio** | **Entradas / Saídas** | **Critérios de Aceite** |
| --- | --- | --- | --- | --- |
| **RF01 – Criação de Contrato** | Geração de contrato digital a partir de template jurídico padronizado. | A empresa contratante cria contratos com dados dinâmicos (serviço, PJ, prazo, valores, cláusulas). O sistema gera o PDF e o hash (SHA-256). | Entrada: dados do contrato. Saída: PDF + hash. | O contrato deve ser salvo com status = draft e hash único calculado. |
| **RF02 – Upload para IPFS** | Armazenar o PDF no IPFS e registrar o CID. | Garante descentralização e prova de integridade pública. | Entrada: PDF + hash. Saída: CID (Content Identifier). | O CID deve ser recuperável via gateway IPFS. |
| **RF03 – Registro Blockchain** | Registrar hash + CID na Polygon Testnet. | Funciona como “carimbo público” (timestamp + autenticidade). | Entrada: hash + CID. Saída: tx\_hash. | Transação confirmada ≤ 10s e registrada no BD. |
| **RF04 – Assinatura Digital (PJ e Empresa)** | Captura e validação de assinatura digital com metadados (IP, user-agent, timestamp). | Substitui assinatura física e cria trilha jurídica. | Entrada: credenciais + hash contrato. Saída: status = signed. | Ambas as partes devem ter assinatura registrada. |
| **RF05 – Certificação e Verificação de Integridade** | Recalcular hash e comparar com blockchain/IPFS. | Prova de autenticidade e integridade pública. | Entrada: hash contrato. Saída: válido/inválido. | 100% de consistência entre hashes. |
| **RF06 – Versionamento de Contratos** | Cada modificação gera nova versão vinculada ao contrato original. | Evita perda de histórico e garante rastreabilidade. | Entrada: edição contrato. Saída: versão incremental (v2, v3…). | Histórico disponível via /contracts/{id}/versions. |
| **RF07 – Controle de Acesso e Perfis** | Permitir papéis distintos (Admin, Jurídico, Financeiro, PJ). | Evitar acessos indevidos e controlar autorizações. | Entrada: JWT + role. Saída: permissões de visualização/edição. | RBAC implementado com regras validadas por role. |
| **RF08 – Logs e Auditoria Completa** | Registrar ações críticas (criação, assinatura, registro, verificação). | Garantir rastreabilidade total. | Entrada: evento + usuário. Saída: log estruturado. | 100% dos eventos persistidos em audit\_logs. |
| **RF09 – Notificações Automáticas** | Enviar alertas por e-mail/in-app sobre eventos do contrato. | Melhorar transparência e comunicação. | Entrada: evento (ex: “contrato assinado”). Saída: e-mail/notificação. | Notificação registrada em notification\_logs. |
| **RF10 – Dashboard / Relatórios** | Exibir visão consolidada de contratos (ativos, em assinatura, expirados). | Facilitar gestão e decisão do gestor. | Entrada: filtros (data, status, PJ). Saída: tabela, gráficos, exportação. | Dashboard com atualização em tempo real (≤1s). |
| **RF11 – Backup e Restauração** | Exportar/importar dados em JSON (contratos, logs, assinaturas). | Garantir continuidade e integridade dos registros. | Entrada: arquivo JSON. Saída: dados restaurados. | Backup válido e idêntico ao original. |
| **RF12 – API Pública de Consulta** | Endpoint público /contracts/verify?hash=XYZ. | Permite qualquer parte validar integridade do contrato. | Entrada: hash. Saída: resultado booleano + CID + TX. | Endpoint acessível publicamente com tempo de resposta ≤1s. |
| **RF13 – Registro de Erros e Alertas Técnicos** | Monitorar falhas (IPFS, blockchain, API). | Permitir tratamento proativo de incidentes. | Entrada: evento de erro. Saída: log + notificação DevOps. | 100% dos erros críticos notificados. |
| **RF14 – Upload de Nota Fiscal (futuro MVP+)** | Permitir upload e conferência de NF vinculada ao contrato. | Iniciar integração futura financeira. | Entrada: PDF NF + dados. Saída: status = “aguardando aprovação”. | NF vinculada ao contrato correto. |
| **RF15 – Rejeição e Reenvio de Contrato** | Caso o PJ recuse, contrato retorna ao status Em revisão. | Evitar inconsistências e ciclos travados. | Entrada: motivo da recusa. Saída: novo status + log. | Contrato editável novamente após recusa. |
| **RF16 – Certificado Digital Gerado** | Gerar um certificado de autenticidade do contrato (PDF). | Facilitar comprovação jurídica off-platform. | Entrada: contrato assinado. Saída: PDF com QR Code de verificação. | QR code funcional e verificável via /verify. |

## ****4.2. Requisitos Não Funcionais (RNF)****

| **Código** | **Categoria** | **Descrição Técnica** | **Métrica / SLA / Critério de Validação** |
| --- | --- | --- | --- |
| **RNF01 – Segurança Criptográfica** | Criptografia e privacidade | Dados sensíveis criptografados em AES-256, comunicações via HTTPS (TLS 1.3). JWT assinado (RSA-512). | 100% de comunicações seguras. |
| **RNF02 – Performance** | Tempo de resposta e latência | Tempo máximo do ciclo de criação e registro: ≤10s. Endpoints REST ≤300ms. | 95% das requisições dentro do SLA. |
| **RNF03 – Escalabilidade** | Elasticidade horizontal | Sistema deve suportar 10.000 contratos simultâneos e 1.000 uploads concorrentes. | Stress test: CPU <70%, erros <1%. |
| **RNF04 – Auditabilidade** | Logs imutáveis e rastreáveis | Todas as ações devem ser logadas com hash e timestamp. Logs exportáveis via API. | 100% das ações críticas auditadas. |
| **RNF05 – LGPD / ICP-Brasil Compliance** | Proteção de dados e validade jurídica | Consentimento explícito do PJ, opção de exclusão, e uso de chaves ICP-Brasil quando aplicável. | Auditoria LGPD ≥ 95% conforme checklist. |
| **RNF06 – Disponibilidade** | Uptime e resiliência | API e IPFS disponíveis ≥ 99,5%. | Monitoramento Grafana + CloudWatch. |
| **RNF07 – Observabilidade e Logging** | Tracing distribuído e logs estruturados | Implementar OpenTelemetry + logs JSON com traceId, actorId, contractId. | 100% de requisições rastreáveis. |
| **RNF08 – Usabilidade (UI/UX)** | Experiência e responsividade | Interface mobile-first, responsiva e intuitiva. | Score ≥85 no Google Lighthouse. |
| **RNF09 – Testabilidade** | Qualidade e automação | Cobertura mínima de testes: 90% unitário, 80% integração, 70% E2E. | Validado via pipeline CI/CD. |
| **RNF10 – Resiliência e Retentativas** | Falhas externas (IPFS, Polygon) | Fila assíncrona com retry exponencial (1m, 5m, 15m) e fallback provider. | Nenhuma perda de dados por falha de terceiros. |
| **RNF11 – Compatibilidade** | Interoperabilidade | API REST compatível com JSON e OpenAPI 3.1. | Documentação Swagger atualizada. |
| **RNF12 – Backup e Recuperação** | Continuidade operacional | Backups automáticos diários e restore point-in-time. | Recuperação ≤15min RTO, 0% perda RPO. |
| **RNF13 – Integração Contínua (CI/CD)** | Entregas seguras e rápidas | Deploy automatizado via GitHub Actions / AWS ECS. | Zero downtime em atualizações. |
| **RNF14 – Confiabilidade Blockchain** | Garantia de registro | Transações assinadas e verificáveis em Etherscan. | 100% das TX confirmadas on-chain. |

## ****4.3. Interdependências e Fluxos Críticos****

| **Dependência** | **Descrição Técnica** |
| --- | --- |
| RF01 → RF02 | Criação do contrato só é concluída após upload e retorno do CID. |
| RF02 → RF03 | Registro on-chain exige CID do IPFS. |
| RF03 → RF04 | Somente contratos registrados podem ser assinados. |
| RF04 → RF05 | A verificação depende das assinaturas e hash blockchain. |
| RF07 → RF08 | Perfis controlam permissões de log e auditoria. |
| RF09 → RF13 | Notificações devem ser disparadas em caso de falha ou erro. |

## ****4.4. Boas Práticas e Regras de Engenharia****

* **Idempotência total:** todas as rotas POST e PUT devem aceitar idempotency-key.
* **Paginação e filtros:** endpoints de listagem devem ter ?page, ?status, ?pjId.
* **Validação de schema:** JSON Schema Validation em todos os endpoints (FastAPI/NestJS DTO).
* **Retry + Circuit Breaker:** integração com IPFS e Polygon deve ser resiliente.
* **Logs estruturados:** JSON formatado com traceId, actorId, eventType, timestamp.
* **Rate limiting:** 100 req/min por IP.
* **Cache:** contratos verificados armazenados 10min (Redis TTL).

# ****5️⃣ Modelo de Dados (MER/DER + Dicionário de Dados)****

**Propósito:**  
Definir um modelo de dados **consistente, auditável e performático** para o módulo de **Contrato Digital Off-Chain** da AURE, cobrindo entidades, relacionamentos, restrições de integridade, índices, versionamento e trilha de auditoria.

## 5.1 Visão Geral (DER textual)

companies (1)───(N) contracts (1)───(N) contract\_versions

│

├────(N) contract\_signatures

│

└────(N) blockchain\_logs

contracts (1)───(N) audit\_logs users (1)───(N) audit\_logs

users (1)───(N) contract\_signatures

pjs (1)───(N) contracts

* **companies**: empresa contratante.
* **pjs**: prestadores (Pessoa Jurídica).
* **users**: contas de acesso (podem pertencer à empresa ou ao PJ).
* **contracts**: cabeçalho do contrato (estado atual, hash e CID vigentes).
* **contract\_versions**: cada versão imutável do PDF, com seu doc\_hash e ipfs\_cid.
* **contract\_signatures**: assinaturas de cada parte por versão.
* **blockchain\_logs**: registros de transações on-chain (carimbo público).
* **audit\_logs**: trilha imutável de ações (quem/quando/o quê).

## 5.2 Dicionário de Dados (tabelas principais)

### 5.2.1 contracts

| **Campo** | **Tipo** | **Nulo** | **Default** | **Descrição** |
| --- | --- | --- | --- | --- |
| id | BIGSERIAL PK | N |  | Identificador |
| company\_id | UUID FK→companies(id) | N |  | Empresa contratante |
| pj\_id | UUID FK→pjs(id) | N |  | Prestador (PJ) |
| service\_desc | TEXT | N |  | Descrição do serviço/prestação |
| amount\_total | NUMERIC(12,2) | N | 0 | Valor total do contrato |
| currency | CHAR(3) | N | 'BRL' | Moeda |
| status | contract\_status (ENUM) | N | 'draft' | Estado atual do contrato |
| current\_version\_id | BIGINT FK→contract\_versions(id) | S |  | Ponteiro para versão vigente |
| onchain\_tx | VARCHAR(100) | S |  | Hash da última TX confirmada |
| created\_at | TIMESTAMPTZ | N | now() | Criação |
| updated\_at | TIMESTAMPTZ | N | now() | Última atualização |
| deleted\_at | TIMESTAMPTZ | S |  | Soft delete (opcional) |

**Enum** contract\_status: draft, ready\_for\_sign, sent\_to\_sign, partially\_signed, fully\_signed, registered, active, terminated.

**Índices sugeridos:**

* idx\_contracts\_company\_status (company\_id, status)
* idx\_contracts\_pj\_status (pj\_id, status)
* idx\_contracts\_updated\_at (updated\_at)

### 5.2.2 contract\_versions

| **Campo** | **Tipo** | **Nulo** | **Default** | **Descrição** |
| --- | --- | --- | --- | --- |
| id | BIGSERIAL PK | N |  | Identificador |
| contract\_id | BIGINT FK→contracts(id) | N |  | Contrato pai |
| version | INT | N | 1 | Versão (1..N) |
| doc\_hash | CHAR(64) | N |  | SHA-256 do PDF |
| ipfs\_cid | VARCHAR(255) | N |  | CID (IPFS) |
| pdf\_storage\_url | TEXT | N |  | URL S3/GCS da cópia canônica |
| created\_at | TIMESTAMPTZ | N | now() | Data/hora da versão |

**Restrições:**

* UNIQUE (contract\_id, version)
* CHECK (char\_length(doc\_hash)=64)

**Índices:**

* idx\_cv\_contract (contract\_id)
* idx\_cv\_doc\_hash (doc\_hash)

### 5.2.3 contract\_signatures

| **Campo** | **Tipo** | **Nulo** | **Default** | **Descrição** |
| --- | --- | --- | --- | --- |
| id | BIGSERIAL PK | N |  | Identificador |
| contract\_id | BIGINT FK→contracts(id) | N |  | Contrato |
| version\_id | BIGINT FK→contract\_versions(id) | N |  | Versão assinada |
| signer\_id | UUID FK→users(id) | N |  | Usuário signatário |
| role | sign\_role (ENUM) | N |  | company ou pj |
| signature\_hash | CHAR(64) | N |  | Hash da assinatura eletrônica |
| ip\_address | INET | N |  | IP de origem |
| user\_agent | TEXT | N |  | Agente do cliente |
| signed\_at | TIMESTAMPTZ | N | now() | Momento da assinatura |

**Enum** sign\_role: company, pj.  
**Restrições:**

* UNIQUE (version\_id, role) (garante uma assinatura por papel por versão)
* CHECK (char\_length(signature\_hash)=64)

**Índices:**

* idx\_csig\_contract (contract\_id)
* idx\_csig\_version (version\_id)

### 5.2.4 blockchain\_logs

| **Campo** | **Tipo** | **Nulo** | **Default** | **Descrição** |
| --- | --- | --- | --- | --- |
| id | BIGSERIAL PK | N |  | Identificador |
| contract\_id | BIGINT FK→contracts(id) | N |  | Contrato |
| version\_id | BIGINT FK→contract\_versions(id) | N |  | Versão registrada |
| network | VARCHAR(32) | N | 'polygon-amoy' | Rede |
| tx\_hash | VARCHAR(100) | N |  | Hash da transação |
| registry\_method | VARCHAR(64) | N | 'register' | Método do smart contract |
| doc\_hash | CHAR(64) | N |  | Redundância para auditoria |
| ipfs\_cid | VARCHAR(255) | N |  | Redundância para auditoria |
| status | VARCHAR(16) | N | 'confirmed' | `pending |
| created\_at | TIMESTAMPTZ | N | now() | Enfileirado |
| confirmed\_at | TIMESTAMPTZ | S |  | Confirmação |

**Índices:**

* idx\_bcl\_tx (tx\_hash)
* idx\_bcl\_contract (contract\_id, version\_id)

### 5.2.5 audit\_logs

| **Campo** | **Tipo** | **Nulo** | **Default** | **Descrição** |
| --- | --- | --- | --- | --- |
| id | BIGSERIAL PK | N |  | Identificador |
| entity | VARCHAR(40) | N |  | Contract, Signature, Version, etc. |
| entity\_id | BIGINT | N |  | ID da entidade |
| action | VARCHAR(50) | N |  | create, finalize, sign, register, verify, … |
| actor\_id | UUID FK→users(id) | S |  | Usuário executor (ou system) |
| payload\_hash | CHAR(64) | N |  | Hash do conteúdo relevante da ação |
| metadata | JSONB | S |  | Dados auxiliares (ex: diffs) |
| created\_at | TIMESTAMPTZ | N | now() | Momento do evento |

**Índices:**

* idx\_audit\_entity (entity, entity\_id)
* idx\_audit\_actor (actor\_id, created\_at)
* idx\_audit\_created (created\_at)

### 5.2.6 Entidades auxiliares

* **companies** (id, name, cnpj, …)
* **pjs** (id, company\_id, corporate\_name, cnpj, …)
* **users** (id, email, role, pj\_id/company\_id, status, password\_hash, mfa\_secret?, created\_at, …)
* **notification\_logs** (id, user\_id, type, payload, status, created\_at)

## 5.3 Regras de Integridade e Negócio (DB-level)

1. **Contrato ativo exige versão**: contracts.current\_version\_id **não nulo** quando status >= sent\_to\_sign.
2. **Assinaturas por versão**: exatamente **uma** por papel (company, pj) por version\_id.
3. **Imutabilidade de versão**: registros em contract\_versions **não sofrem UPDATE** (somente INSERT).
4. **Hash bem formado**: doc\_hash e signature\_hash sempre 64 chars hex (CHECK).
5. **Sincronismo on-chain**: qualquer blockchain\_logs.status='confirmed' para uma version\_id deve atualizar contracts.onchain\_tx **e** (opcional) promover status → registered/active.
6. **Soft delete**: contracts.deleted\_at define ocultação sem remoção física (opcional).

## 5.4 Índices e Performance

* **Consultas de painel** (por status e data): índices compostos em (company\_id, status, updated\_at) aceleram filtros combinados.
* **Auditoria**: partição por **mês** em audit\_logs (opcional) para alto volume.
* **Busca por hash**: índice em contract\_versions.doc\_hash para verificação rápida.
* **TX lookup**: índice direto em blockchain\_logs.tx\_hash.

## 5.5 DDL de referência (PostgreSQL)

-- Enums

CREATE TYPE contract\_status AS ENUM

('draft','ready\_for\_sign','sent\_to\_sign','partially\_signed','fully\_signed','registered','active','terminated');

CREATE TYPE sign\_role AS ENUM ('company','pj');

-- contracts

CREATE TABLE contracts (

id BIGSERIAL PRIMARY KEY,

company\_id UUID NOT NULL REFERENCES companies(id),

pj\_id UUID NOT NULL REFERENCES pjs(id),

service\_desc TEXT NOT NULL,

amount\_total NUMERIC(12,2) NOT NULL DEFAULT 0,

currency CHAR(3) NOT NULL DEFAULT 'BRL',

status contract\_status NOT NULL DEFAULT 'draft',

current\_version\_id BIGINT REFERENCES contract\_versions(id),

onchain\_tx VARCHAR(100),

created\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

updated\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

deleted\_at TIMESTAMPTZ

);

-- contract\_versions

CREATE TABLE contract\_versions (

id BIGSERIAL PRIMARY KEY,

contract\_id BIGINT NOT NULL REFERENCES contracts(id) ON DELETE CASCADE,

version INT NOT NULL,

doc\_hash CHAR(64) NOT NULL CHECK (char\_length(doc\_hash)=64),

ipfs\_cid VARCHAR(255) NOT NULL,

pdf\_storage\_url TEXT NOT NULL,

created\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

UNIQUE (contract\_id, version)

);

-- contract\_signatures

CREATE TABLE contract\_signatures (

id BIGSERIAL PRIMARY KEY,

contract\_id BIGINT NOT NULL REFERENCES contracts(id) ON DELETE CASCADE,

version\_id BIGINT NOT NULL REFERENCES contract\_versions(id) ON DELETE CASCADE,

signer\_id UUID NOT NULL REFERENCES users(id),

role sign\_role NOT NULL,

signature\_hash CHAR(64) NOT NULL CHECK (char\_length(signature\_hash)=64),

ip\_address INET NOT NULL,

user\_agent TEXT NOT NULL,

signed\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

UNIQUE (version\_id, role)

);

-- blockchain\_logs

CREATE TABLE blockchain\_logs (

id BIGSERIAL PRIMARY KEY,

contract\_id BIGINT NOT NULL REFERENCES contracts(id) ON DELETE CASCADE,

version\_id BIGINT NOT NULL REFERENCES contract\_versions(id) ON DELETE CASCADE,

network VARCHAR(32) NOT NULL DEFAULT 'polygon-amoy',

tx\_hash VARCHAR(100) NOT NULL,

registry\_method VARCHAR(64) NOT NULL DEFAULT 'register',

doc\_hash CHAR(64) NOT NULL CHECK (char\_length(doc\_hash)=64),

ipfs\_cid VARCHAR(255) NOT NULL,

status VARCHAR(16) NOT NULL DEFAULT 'pending',

created\_at TIMESTAMPTZ NOT NULL DEFAULT now(),

confirmed\_at TIMESTAMPTZ

);

-- audit\_logs

CREATE TABLE audit\_logs (

id BIGSERIAL PRIMARY KEY,

entity VARCHAR(40) NOT NULL,

entity\_id BIGINT NOT NULL,

action VARCHAR(50) NOT NULL,

actor\_id UUID,

payload\_hash CHAR(64) NOT NULL CHECK (char\_length(payload\_hash)=64),

metadata JSONB,

created\_at TIMESTAMPTZ NOT NULL DEFAULT now()

);

-- índices

CREATE INDEX idx\_contracts\_company\_status ON contracts(company\_id, status, updated\_at DESC);

CREATE INDEX idx\_contracts\_pj\_status ON contracts(pj\_id, status, updated\_at DESC);

CREATE INDEX idx\_cv\_contract ON contract\_versions(contract\_id);

CREATE INDEX idx\_cv\_doc\_hash ON contract\_versions(doc\_hash);

CREATE INDEX idx\_csig\_contract ON contract\_signatures(contract\_id);

CREATE INDEX idx\_csig\_version ON contract\_signatures(version\_id);

CREATE INDEX idx\_bcl\_tx ON blockchain\_logs(tx\_hash);

CREATE INDEX idx\_bcl\_contract ON blockchain\_logs(contract\_id, version\_id);

CREATE INDEX idx\_audit\_entity ON audit\_logs(entity, entity\_id);

CREATE INDEX idx\_audit\_actor ON audit\_logs(actor\_id, created\_at);

## 5.6 Políticas de Dados e LGPD

* **Minimização**: somente atributos necessários em users, companies, pjs.
* **Pseudonimização**: mascarar CNPJ/CPF nos exports; guardar last4 para conferência.
* **Retenção**: audit\_logs particionado por mês, retenção configurável (ex.: 5 anos).
* **Direito de exclusão**: soft delete (deleted\_at) e anonimização em dados pessoais.
* **Sem dados sensíveis na blockchain**: apenas doc\_hash e ipfs\_cid (sem PII).

## 5.7 Consultas Úteis (QA/BI)

* **Última versão e integridade**

SELECT c.id, cv.version, cv.doc\_hash, cv.ipfs\_cid

FROM contracts c

JOIN contract\_versions cv ON cv.id = c.current\_version\_id

WHERE c.id = $1;

* **Assinaturas pendentes por papel**

SELECT c.id, cv.version

FROM contracts c

JOIN contract\_versions cv ON cv.id = c.current\_version\_id

LEFT JOIN contract\_signatures s1 ON s1.version\_id=cv.id AND s1.role='company'

LEFT JOIN contract\_signatures s2 ON s2.version\_id=cv.id AND s2.role='pj'

WHERE s1.id IS NULL OR s2.id IS NULL;

* **Contrato e TX confirmada**

SELECT b.tx\_hash, b.confirmed\_at

FROM blockchain\_logs b

WHERE b.contract\_id=$1 AND b.status='confirmed'

ORDER BY b.confirmed\_at DESC

LIMIT 1;

## 5.8 Boas Práticas de Modelagem

* **Imutabilidade de versão**: nunca atualizar contract\_versions; crie nova versão.
* **Chaves técnicas consistentes**: use BIGINT para relacionamentos internos de alto volume.
* **Migrações**: versionar com timestamp e **down** seguro.
* **Seed de enums**: criar contract\_status e sign\_role nas migrações iniciais.
* **Observabilidade**: sempre logar contract\_id, version\_id e traceId no app.

# ****6️⃣ Endpoints e Integrações****

**Propósito:**  
Definir, de forma executável, o contrato de APIs do módulo de Contrato Digital da AURE e suas integrações externas (IPFS, Blockchain, E-mail e Filas), garantindo padronização, rastreabilidade e previsibilidade.

## 6.1 Convenções gerais da API

* **Base URL:** https://api.aure.app/v1
* **Auth:** Authorization: Bearer <JWT> (RBAC por role).
* **Content-Type:** application/json; charset=utf-8
* **Idempotência (POST/PUT críticos):** Idempotency-Key: <uuid-v4>
* **Versionamento:** via path (/v1) e X-API-Version: 1.
* **Rate limit (default):** 100 req/min/IP (429 quando excedido).
* **Erros (padrão):**
* {
* "error": "VALIDATION\_ERROR",
* "message": "Invalid field: service\_desc",
* "traceId": "af2c5e..."
* }
* **RBAC sugerido:**
  + company\_admin, legal, finance → criação/registro.
  + pj\_user → assinar e consultar.
  + aure\_admin → leitura técnica/auditoria.

## 6.2 Endpoints REST (detalhados)

### 1) ****Criar contrato****

**POST** /contracts

**Permissões:** company\_admin ou legal  
**Idempotência:** sim  
**Descrição:** cria um contrato draft a partir de um template jurídico (server-side).

**Request (exemplo):**

{

"companyId": "4c8f2b2a-2a0e-4b6f-9b6c-2d5a4b787a10",

"pjId": "6aa89b90-7bff-4d07-bd53-9f1b2c25d9f7",

"serviceDesc": "Desenvolvimento de software - sprint 1",

"amountTotal": 25000.00,

"currency": "BRL",

"templateId": "std-service-pt-BR-v1",

"variables": {

"serviceScope": "Back-end NestJS",

"deliveryDate": "2025-11-30",

"paymentTerms": "30/60 dias",

"clauses": ["NDA", "Propriedade Intelectual"]

}

}

**Response 201:**

{

"id": 9812,

"status": "draft",

"currentVersionId": null,

"createdAt": "2025-10-19T15:32:11Z"

}

**Erros:** 400 VALIDATION\_ERROR, 401 UNAUTHORIZED, 403 FORBIDDEN.

### 2) ****Finalizar contrato (gerar PDF, hash e enviar para IPFS)****

**PUT** /contracts/{id}/finalize

**Permissões:** company\_admin ou legal  
**Idempotência:** sim  
**Descrição:** gera o PDF a partir do template/vars, calcula o **SHA-256**, grava no **Object Storage**, enfileira **upload/pinning** no IPFS e atualiza estado para sent\_to\_sign.  
**Comportamento:** operação **assíncrona** – retorna jobId para acompanhar IPFS.

**Request (opcional):**

{

"regenerate": false,

"notes": "Revisado pela área jurídica em 19/10"

}

**Response 202:**

{

"contractId": 9812,

"status": "sent\_to\_sign",

"version": 1,

"docHash": "f5f9be...9a2e",

"pdfUrl": "s3://contracts/9812/v1/document.pdf",

"jobs": {

"ipfsUploadJobId": "job\_7f1d3c"

}

}

**Erros:** 404 NOT\_FOUND, 409 CONFLICT (já finalizado), 422 UNPROCESSABLE\_ENTITY.

### 3) ****Assinar contrato (PJ/Empresa)****

**POST** /contracts/{id}/sign

**Permissões:**

* pj\_user assina em nome do PJ.
* company\_admin ou legal assina pela empresa.

**Idempotência:** sim  
**Descrição:** registra assinatura da parte (por **versão vigente**), com trilha técnica (IP, UA, timestamp). Atualiza partially\_signed/fully\_signed.

**Request:**

{

"versionId": 1,

"consent": true

}

**Response 200:**

{

"contractId": 9812,

"versionId": 1,

"status": "partially\_signed",

"signature": {

"role": "pj",

"signatureHash": "e9a3fd...11c0",

"ip": "201.55.10.23",

"userAgent": "Mozilla/5.0",

"signedAt": "2025-10-19T15:51:03Z"

}

}

**Erros:** 400 VALIDATION\_ERROR (versão inválida), 403 FORBIDDEN (papel incorreto), 409 CONFLICT (já assinou), 404 NOT\_FOUND.

### 4) ****Registrar contrato on-chain (Polygon)****

**POST** /contracts/{id}/register

**Permissões:** company\_admin ou legal  
**Idempotência:** sim  
**Descrição:** publica {docHash, cid, parties, timestamp} no **smart contract Registry** (Polygon Amoy), via **worker**.  
**Comportamento:** **assíncrono**, retorna jobId e status inicial pending.

**Request (opcional):**

{

"network": "polygon-amoy"

}

**Response 202:**

{

"contractId": 9812,

"versionId": 1,

"onchain": {

"status": "pending",

"jobId": "job\_chain\_9d2a1b"

}

}

**Webhook (on success) opcional:** POST https://app.empresa.com/webhooks/contract-registered

{

"event": "contract.registered\_onchain",

"contractId": 9812,

"versionId": 1,

"txHash": "0xa3c9...f2b",

"network": "polygon-amoy",

"confirmedAt": "2025-10-19T15:59:44Z"

}

**Erros:** 404 NOT\_FOUND, 409 CONFLICT (sem CID/hash), 503 UPSTREAM\_ERROR (RPC/IPFS indisponível).

### 5) ****Verificar integridade****

**GET** /contracts/{id}/verify

**Permissões:** público (read-only) – **não expose PII**  
**Descrição:** baixa o PDF (S3/IPFS gateway), recalcula **SHA-256** e compara com doc\_hash/on-chain.

**Response 200:**

{

"contractId": 9812,

"versionId": 1,

"valid": true,

"checks": {

"hashLocalVsDB": "match",

"hashLocalVsOnChain": "match",

"ipfsAvailability": "ok"

},

"docHash": "f5f9be...9a2e",

"ipfsCid": "bafybei...y3q",

"onchain": {

"txHash": "0xa3c9...f2b",

"network": "polygon-amoy"

}

}

**Erros:** 404 NOT\_FOUND, 424 FAILED\_DEPENDENCY (gateway IPFS caiu), 500 INTERNAL\_ERROR.

### 6) ****Certificado de autenticidade****

**GET** /contracts/{id}/certificate

**Permissões:** público (sem PII)  
**Descrição:** retorna **PDF do certificado** ou JSON com dados para renderização (hash, CID, TX, QR /verify?hash=).

**Response 200 (JSON):**

{

"contractId": 9812,

"versionId": 1,

"certificate": {

"docHash": "f5f9be...9a2e",

"ipfsCid": "bafybei...y3q",

"txHash": "0xa3c9...f2b",

"issuedAt": "2025-10-19T16:02:10Z",

"qrVerifyUrl": "https://api.aure.app/v1/contracts/9812/verify"

}

}

**Response 200 (PDF):** Content-Type: application/pdf  
**Erros:** 404 NOT\_FOUND.

## 6.3 Códigos de erro e semântica

| **HTTP** | **error** | **Quando ocorre** | **Ação do cliente** |
| --- | --- | --- | --- |
| 400 | VALIDATION\_ERROR | Campos inválidos | Corrigir payload |
| 401 | UNAUTHORIZED | JWT ausente/expirado | Reautenticar |
| 403 | FORBIDDEN | Sem permissão no RBAC | Revisar papel |
| 404 | NOT\_FOUND | Contrato/versão ausente | Verificar IDs |
| 409 | CONFLICT | Estado incompatível (ex.: já assinado) | Ajustar fluxo |
| 422 | UNPROCESSABLE\_ENTITY | Inconsistência de negócio | Corrigir regra |
| 424 | FAILED\_DEPENDENCY | IPFS/RPC indisponível | Retry após backoff |
| 429 | RATE\_LIMITED | Limite excedido | Retry com atraso |
| 503 | UPSTREAM\_ERROR | Provedor externo caiu | Retry/backoff |

Todas as respostas de erro incluem traceId.

## 6.4 Integrações externas

### IPFS (Web3.Storage / Pinata)

* **Função:** pinning e obtenção de CID do PDF.
* **SDK:** Web3.Storage/Pinata SDK.
* **Timeout:** 5s; **Retries:** 1m, 5m, 15m; **DLQ** após 3 falhas.
* **Validação:** checksum pós-upload (hash do arquivo vs doc\_hash).

### Blockchain (Polygon Amoy via Ethers.js)

* **Função:** registrar {docHash, cid, parties, timestamp}.
* **Lib:** Ethers.js; **Provider:** Alchemy (fallback Infura).
* **Confirmations:** 1–3; **Timeout:** 15s; **Retries:** 1m, 5m, 15m.
* **Logs:** tx\_hash, block, gasUsed, status.

### E-mail (SES/SendGrid)

* **Eventos:** contract.sent\_to\_sign, contract.signed, contract.registered\_onchain.
* **Template-based:** variáveis (nome PJ, nº contrato, link).
* **Fallback:** notificação in-app quando bounce.

### Filas (BullMQ / SQS)

* **Jobs:** ipfsUpload, registerOnChain, sendEmail.
* **Idempotência:** chave única por (contractId, versionId, jobType).
* **Observabilidade:** métricas de enfileirados/sucesso/falhas (Prometheus).

## 6.5 Webhooks (opcional, para clientes enterprise)

**Assinatura de eventos:** POST /integrations/webhooks  
**Segurança:** X-Signature: HMAC-SHA256(payload)  
**Eventos suportados:**

* contract.sent\_to\_sign
* contract.partially\_signed
* contract.fully\_signed
* contract.registered\_onchain
* contract.verified

**Exemplo de payload:**

{

"event": "contract.fully\_signed",

"occurredAt": "2025-10-19T16:04:55Z",

"contractId": 9812,

"versionId": 1,

"companyId": "4c8f2b2a-2a0e-4b6f-9b6c-2d5a4b787a10",

"pjId": "6aa89b90-7bff-4d07-bd53-9f1b2c25d9f7"

}

## 6.6 OpenAPI (trecho de referência)

openapi: 3.1.0

info:

title: AURE Contracts API

version: "1.0"

servers:

- url: https://api.aure.app/v1

paths:

/contracts:

post:

operationId: createContract

security: [{ bearerAuth: [] }]

parameters:

- in: header

name: Idempotency-Key

required: true

schema: { type: string, format: uuid }

responses:

"201": { description: Created }

"400": { $ref: "#/components/responses/ValidationError" }

/contracts/{id}/finalize:

put:

operationId: finalizeContract

parameters:

- in: path

name: id

required: true

schema: { type: integer }

- in: header

name: Idempotency-Key

required: true

schema: { type: string, format: uuid }

responses:

"202": { description: Accepted }

/contracts/{id}/sign:

post:

operationId: signContract

responses:

"200": { description: OK }

/contracts/{id}/register:

post:

operationId: registerOnChain

responses:

"202": { description: Accepted }

/contracts/{id}/verify:

get:

operationId: verifyContract

responses:

"200": { description: OK }

/contracts/{id}/certificate:

get:

operationId: getCertificate

responses:

"200": { description: OK }

components:

securitySchemes:

bearerAuth:

type: http

scheme: bearer

bearerFormat: JWT

### ✅ Resumo técnico

* Endpoints padronizados com **idempotência**, **RBAC** e **erros consistentes**.
* Operações longas (IPFS/Chain) são **assíncronas**, com **jobs** e **webhooks** opcionais.
* Integrações externas com **timeouts, retries e circuit breaker**, garantindo resiliência.
* **OpenAPI 3.1** como fonte de verdade para gerar SDKs/contratos de teste de QA.

Se quiser, eu **gero o arquivo OpenAPI completo** (openapi.yaml) e um **Postman Collection** com exemplos prontos para o seu time.

.Storage), Blockchain (Ethers.js/Polygon), E-mail (SES/SendGrid), Fila (BullMQ/SQS).

# ****7️⃣ Segurança e Compliance****

**Propósito**  
Estabelecer **políticas, controles e métricas** de segurança da informação e conformidade legal que o módulo de **Contrato Digital Off-Chain** deve atender (MVP e evolução), cobrindo **autenticação/autorização, criptografia, hardening, auditoria, LGPD/ICP-Brasil**, e particularidades de **blockchain**.

## 7.1 Autenticação, Sessão e Autorização

**Requisitos (obrigatórios no MVP)**

* **JWT (RS256) de acesso:** expiração **1h**; incluir sub, role, companyId, pjId, iat, exp, jti.
* **Refresh Token:** expiração **24h**; **rotacionado** a cada uso; manter **denylist** (revogação).
* **RBAC** por perfil: company\_admin, legal, finance, pj\_user, aure\_admin.
* **MFA opcional (fase 2):** TOTP/WebAuthn para perfis sensíveis (company\_admin, aure\_admin).
* **SSO opcional (fase 2/enterprise):** OIDC/SAML (Azure AD/Okta).
* **Política de sessão:** logout forçado após inatividade de 15min (web), detecção de múltiplos logins suspeitos.
* **Idempotency-Key** em POST/PUT críticos.

**Critérios de aceite**

* Rotas sensíveis retornam **403** quando role não possui permissão.
* **Revogação** de tokens efetiva em ≤60s (lista de bloqueio em Redis).
* **Brute force** bloqueado após 5 tentativas/15min por IP/usuário.

## 7.2 Criptografia e Transporte

* **Em trânsito:** **TLS 1.3**, **HSTS** (preload), **OCSP stapling**; **Ciphers** modernas (AES-GCM/CHACHA20).
* **Em repouso:** **AES-256** para dados sensíveis em banco/objetos; **KMS** para chaves (rotação 90 dias).
* **Hash de documento:** **SHA-256** do PDF final (64 hex chars), verificado em /verify.
* **Gestão de segredos:** AWS Secrets Manager; **proibido** segredos em repositório/variáveis de build.

**Critérios de aceite**

* Nenhum endpoint responde sem **HTTPS**.
* Dumps/exports sempre **criptografados** (AES-256 + senha de transporte separada).

## 7.3 Política de Dados, LGPD e ICP-Brasil

* **Minimização:** coletar apenas o essencial (empresa/PJ/usuários).
* **Classificação de dados:** Público, Interno, Confidencial (LGPD); marcar campos PII.
* **Direitos do titular (DSR):** disponibilizar processos para **acesso, correção, exclusão** (soft-delete + anonimização).
* **Retenção:** audit\_logs por 5 anos (configurável); backups por 30 dias; documentação de retenção.
* **Consentimento:** registro explícito para PJs (data, IP, user-agent, finalidade).
* **ICP-Brasil (futuro/cliente enterprise):** permitir upgrade de assinatura para **Avançada/Qualificada** (WebAuthn/ICP) sem refatorar o fluxo.

**Critérios de aceite**

* Export de dados pessoais **anonimiza** campos sensíveis (ex.: cnpj → XX.XXX.XXX/0001-YY).
* **Nenhum dado pessoal** (PII) é gravado **on-chain** ou em logs públicos.

## 7.4 Segurança de Aplicação (OWASP)

* **Headers:** CSP (default-src 'self'), X-Frame-Options DENY, X-Content-Type-Options nosniff, Referrer-Policy strict-origin.
* **CORS:** allowlist de domínios (prod, staging), OPTIONS limitado.
* **CSRF:** não aplicável a APIs puras; para painéis web com cookies, aplicar **SameSite=strict** e token CSRF.
* **Input validation:** DTO/schema validation (AJV/Zod/Class-Validator) com mensagens padronizadas.
* **Uploads:** aceitar **apenas PDF**; validar **MIME/assinatura mágica**; antivírus/clamav opcional; **tamanho máximo** 10MB.
* **Dependências:** SCA (Dependabot/Snyk), **SAST** (CodeQL), **DAST** (OWASP ZAP).
* **SBOM:** gerar (CycloneDX) e versionar por release.

**Critérios de aceite**

* Pipeline CI bloqueia merge em caso de **vuln alta/crítica**.
* 0 findings críticos em DAST no ambiente de staging.

## 7.5 Proteção de Infra, DoS e Rate Limiting

* **WAF/CDN:** CloudFront + AWS WAF (bloqueio de padrões maliciosos, geo-fencing opcional).
* **Rate limiting:** 100 req/min/IP default; buckets menores em rotas sensíveis (login: 10/min).
* **DDoS:** proteção L3/L4 (AWS Shield Standard).
* **Segregação de redes:** VPC privada para bancos/filas; SG inbound mínimo necessário.
* **Backups:** RDS snapshots diários + PITR; **criptografados**.

**Critérios de aceite**

* Teste de carga a 2× do pico esperado sem indisponibilidade.
* RTO ≤ 15min; RPO ≤ 5min.

## 7.6 Auditoria, Logs e Observabilidade

* **Logs estruturados (JSON):** incluir traceId, actorId, contractId, event, status.
* **Eventos auditáveis:** create, finalize, sign, register, verify, login, role\_change.
* **Imutabilidade lógica:** impedir edição de audit\_logs; opcionalmente armazenar **hash do payload**.
* **Tracing:** OpenTelemetry em API e Workers; **Prometheus/Grafana** para métricas.
* **Alertas:** latência API > 300ms p95; erro 5xx > 1%; falhas em jobs (IPFS/Chain) > 2%.

**Critérios de aceite**

* 100% das ações críticas possuem entrada em **audit\_logs**.
* Dashboards prontos (latência, taxa de erro, jobs, fila).

## 7.7 Processo Seguro (SSDLC) e Vulnerabilidades

* **Branches protegidas**, revisão obrigatória por par (inclui Sec/Arch para áreas sensíveis).
* **Pre-merge:** SAST, SCA; **post-deploy:** DAST em staging.
* **Pentest** externo: 1× antes de GA e 1×/ano.
* **Gestão de vulnerabilidades:** SLA de correção — Crítica 24h, Alta 72h, Média 14d, Baixa 30d.
* **Change Management:** RFC para alterações em **Auth/RBAC/Cripto** com aprovação dupla.

## 7.8 Especificidades de Blockchain (Privacidade e Risco)

* **Dado on-chain:** somente **docHash (SHA-256)** e **ipfsCid** (sem PII).
* **Carteiras/assinatura:** chave do **Registry** mantida em **KMS/HSM** (ou serviço MPC), com **rate limit** de TX.
* **RPC providers:** **Alchemy** (primário), **Infura** (fallback); **circuit breaker** após 3 falhas/5 min.
* **Reorgs e confirmações:** considerar **1–3 confirmações**; status pending → confirmed.
* **Monitoramento de TX:** reconciliar blockchain\_logs vs. chain a cada 15min; reprocessar falhas.

**Critérios de aceite**

* 0 PII em transações/eventos.
* 100% das TX possuem tx\_hash e confirmed\_at quando sucesso.

## 7.9 Matriz RBAC (Resumo)

| **Recurso/Operação** | **company\_admin** | **legal** | **finance** | **pj\_user** | **aure\_admin** |
| --- | --- | --- | --- | --- | --- |
| Criar contrato (POST /contracts) | ✔︎ | ✔︎ | ✖︎ | ✖︎ | ✔︎ |
| Finalizar/gerar PDF (PUT /finalize) | ✔︎ | ✔︎ | ✖︎ | ✖︎ | ✔︎ |
| Assinar contrato (POST /sign) | ✔︎ | ✔︎ | ✖︎ | ✔︎ | ✔︎ |
| Registrar on-chain (POST /register) | ✔︎ | ✔︎ | ✖︎ | ✖︎ | ✔︎ |
| Verificar (GET /verify, público) | ✔︎ | ✔︎ | ✔︎ | ✔︎ | ✔︎ |
| Certificado (GET /certificate, público) | ✔︎ | ✔︎ | ✔︎ | ✔︎ | ✔︎ |
| Logs/Auditoria (consulta) | ✔︎ | ✔︎ | ✔︎ | ✖︎ | ✔︎ |

## 7.10 Métricas, SLOs e Monitoramento

| **Área** | **SLO** | **Alerta** |
| --- | --- | --- |
| **Auth** | Latência p95 < 150ms; falha < 0,5% | >0,5% 4xx auth por 5 min |
| **API** | Latência p95 < 300ms; 5xx < 1% | p95 > 300ms por 10 min |
| **Jobs** | Sucesso ≥ 98% (IPFS/Chain) | Falhas > 2% / 15 min |
| **Disponibilidade** | ≥ 99,5% | queda > 5 min |
| **Segurança** | 0 vuln críticas abertas | qualquer crítica → page SecOps |

## 7.11 Incidentes e Continuidade

* **Runbooks** para: indisponibilidade RPC, falha IPFS, picos de 5xx, vazamento de segredos, anomalias de login.
* **RTO/RPO**: 15 min / 5 min.
* **Comunicação de incidentes**: fluxo com Jurídico/Clientes (LGPD: notificação à ANPD quando cabível).

### ✅ Critérios de Aceite (Segurança & Compliance)

* Todos os endpoints protegidos por **TLS 1.3** e **JWT** (com RBAC).
* **PII** fora de blockchain e logs públicos; consentimento e retenção **LGPD** implementados.
* **Hash (SHA-256)** e **CID** verificados em /verify; **TX** confirmada registrada.
* Pipelines com **SAST/SCA/DAST** e **bloqueio** em vulnerabilidades críticas.
* Observabilidade ativa (Otel, métricas, alertas) e **audit\_logs** cobrindo 100% das ações críticas.
* Backups **criptografados** e testados; **RTO/RPO** atendidos.

# 9️⃣ Critérios de Aceite (DoD)

- Hash SHA-256 verificado  
- CID IPFS acessível  
- Transação blockchain confirmada  
- Assinaturas válidas (hash + IP + timestamp)  
- Certificado com hash, CID e TX  
- Logs auditáveis

# 🔟 Roadmap Técnico e Fases Futuras

**Propósito:**  
Definir a evolução técnica do módulo de **Contratos Digitais** da AURE, estabelecendo as fases de maturação da solução — desde o MVP (off-chain) até o modelo completo de automação contratual e financeira (on-chain com BI e compliance).

Cada fase equilibra **viabilidade técnica**, **validação de mercado** e **incremento de confiabilidade**, permitindo entregas rápidas e sustentáveis.

## ****Fase 1 — Contratos Digitais Off-chain ✅****

**Status:** Concluída / MVP funcional  
**Descrição:**  
Primeira entrega do sistema, responsável por criar a infraestrutura de confiança para contratos digitais com integridade e rastreabilidade total.

**Componentes principais:**

* Criação e assinatura digital eletrônica (empresa e PJ).
* Geração do hash (SHA-256) e armazenamento descentralizado no **IPFS**.
* Registro da prova de autenticidade na **blockchain Polygon (Amoy Testnet)**.
* Painel com status dos contratos, logs de auditoria e certificados públicos.
* Pagamentos ainda **off-chain**, via Pix/API bancária.

**Objetivo técnico:** Garantir a rastreabilidade jurídica e técnica dos contratos.  
**Critério de sucesso:** ≥ 1 contrato tokenizado e registrado on-chain, com fluxo fim a fim executado.  
**Stack principal:** NestJS, React, IPFS, Ethers.js, Polygon Amoy, PostgreSQL.

## ****Fase 2 — Integração NFS-e + Cláusulas Dinâmicas 🔄****

**Status:** Em desenvolvimento / próxima entrega  
**Descrição:**  
Expande o módulo contratual com automação fiscal e flexibilidade jurídica.

**Entregas principais:**

* Integração com APIs de **Nota Fiscal de Serviço Eletrônica (NFS-e)** — geração e conferência.
* Sistema de **cláusulas dinâmicas** (modelo parametrizável) por tipo de contrato e serviço.
* Validação automática de dados fiscais e CNPJs.
* Geração automatizada de anexos contratuais (aditivos e revisões).

**Objetivo técnico:** Automatizar o ciclo contrato → emissão → validação.  
**Critério de sucesso:** 95% das NFS-e válidas e associadas ao contrato correto.  
**Dependência:** Estrutura de contract\_versions e audit\_logs (da Fase 1).  
**Stack adicional:** Node fiscal adapters, Template Engine (Handlebars/Puppeteer).

## ****Fase 3 — Split e Escrow On-chain (Stablecoin) 🔜****

**Status:** Planejada / Blockchain Finance  
**Descrição:**  
Introduz automação financeira descentralizada (DeFi corporativa).

**Entregas principais:**

* Contratos inteligentes (**Smart Contracts**) de **Split Payment** e **Escrow** com tokens estáveis (ex.: USDC).
* Lógica condicional: liberações por marcos de entrega, datas, aprovação manual.
* Registro automático da liquidação financeira associada ao contrato on-chain.
* Módulo de reconciliação entre registros blockchain e Pix/API bancária (para dupla conferência).

**Objetivo técnico:** Reduzir inadimplência e permitir liquidação automática baseada em contrato.  
**Critério de sucesso:** Split executado com sucesso em testnet (escrow → release → proof).  
**Stack adicional:** Solidity (Smart Contracts), Hardhat, Polygon mainnet/testnet, stablecoins (USDC testnet).

## ****Fase 4 — BI Preditivo + KYC/KYB 🔜****

**Status:** Planejada / Expansão Enterprise  
**Descrição:**  
Evolui o módulo contratual para um **hub inteligente de governança e risco**, agregando compliance e inteligência de dados.

**Entregas principais:**

* **Painéis preditivos (BI)** de adimplência, risco contratual e performance de fornecedores.
* **Integração KYC/KYB** para validação automática de identidade e compliance (SERPRO, Receita, ClearSale).
* Módulo de **relatórios regulatórios** (LGPD, Bacen, Receita).
* API pública de consulta de integridade contratual para terceiros (auditores, bancos).

**Objetivo técnico:** Transformar o módulo contratual em uma infraestrutura confiável de dados e validação.  
**Critério de sucesso:** Painel ativo com métricas de risco e integrações com provedores externos.  
**Stack adicional:** Power BI/Metabase, API compliance, ML models (preditivo), Redis Streams.

## ****Resumo de Evolução Técnica****

| **Fase** | **Foco Técnico** | **Stack / Tecnologia** | **Resultado esperado** |
| --- | --- | --- | --- |
| **1** | Registro digital e prova de autenticidade | IPFS, SHA-256, Polygon Testnet | Contrato rastreável e auditável |
| **2** | Automação fiscal e cláusulas jurídicas dinâmicas | NFS-e API, Template Engine | Redução de fricção e erros fiscais |
| **3** | Split/escrow on-chain com stablecoin | Solidity, Ethers.js, USDC | Liquidação automática e segura |
| **4** | Inteligência e compliance automatizado | BI, ML, KYC/KYB APIs | Risco reduzido e confiança de mercado |

# 11️⃣ Decisões e Trade-offs (ADR)

# ****Decisões e Trade-offs (ADR)****

**Formato padrão por ADR**  
**ID · Título** — **Status** (Aceita/Provisória/Em revisão)  
**Contexto** · **Opções consideradas** · **Decisão** · **Consequências** · **Riscos & Mitigações** · **Gatilhos de revisão** · **Owner**

### ****ADR-001 · Registro Off-chain no MVP**** — ****Aceita****

**Contexto**  
Precisamos de prova pública de existência/immutabilidade de contratos, com baixo custo e baixa complexidade inicial.  
**Opções consideradas**  
(1) Execução on-chain completa (contrato inteligente com lógica de pagamento) · (2) Registro off-chain com hash on-chain · (3) Somente armazenamento centralizado.  
**Decisão**  
(2) **Registro off-chain + hash on-chain** (Polygon Testnet) para “cartório digital” (prova), sem movimentar valores.  
**Consequências**

* ✅ Custos de gás e complexidade **baixos**; entrega rápida do MVP.
* ❗ Pagamentos permanecem off-chain; automação financeira fica para Fase 2/3.  
  **Riscos & Mitigações**
* Risco: percepção de “blockchain incompleta”. Mitigar com roadmap público para split/escrow on-chain e certificado de prova on-chain.  
  **Gatilhos de revisão**
* Volume > 5k contratos/mês; demanda de clientes por **execução programável**.  
  **Owner**: Tech Lead

### ****ADR-002 · IPFS vs S3 para o PDF do contrato**** — ****Aceita****

**Contexto**  
Precisamos de armazenamento verificável e público para prova de integridade.  
**Opções**  
(1) **IPFS** (Web3.Storage/Pinata) · (2) **S3** privado com link assinado · (3) Híbrido (IPFS + cópia canônica S3).  
**Decisão**  
(3) **Híbrido**: IPFS (CID) como **fonte pública de prova** + cópia canônica em S3/GCS para latência/DR.  
**Consequências**

* ✅ Prova pública (CID) e performance/DR com S3.
* ❗ Duplica storage; exige rotina de consistência.  
  **Riscos & Mitigações**
* Risco: gateway IPFS instável → fallback provider e **retry exponencial**.  
  **Gatilhos de revisão**
* SLAs de IPFS < 99% por 2 ciclos, ou custo/latência S3 alto.  
  **Owner**: Infra Lead

### ****ADR-003 · Rede Blockchain — Polygon Testnet (Amoy)**** — ****Aceita****

**Contexto**  
Prova pública barata, compatível com EVM e bom tooling.  
**Opções**  
(1) Ethereum mainnet · (2) Polygon mainnet · (3) Polygon testnet (Amoy) · (4) L2 (Arbitrum/Optimism).  
**Decisão**  
(3) **Polygon Amoy** para MVP; **feature flag** para alternar rede.  
**Consequências**

* ✅ TX barata/rápida; ótimo para MVP e demo.
* ❗ Não é rede de produção; exige **rota de migração** a mainnet ao ir para prod.  
  **Riscos & Mitigações**
* Risco: instabilidade testnet → fallback RPC (Alchemy/Infura).  
  **Gatilhos de revisão**
* Início de pilotos pagos (prod), exigência de **compliance** on-chain.  
  **Owner**: Blockchain Lead

### ****ADR-004 · Hash + CID persistidos no BD**** — ****Aceita****

**Contexto**  
Precisamos de consulta rápida, auditoria e reconciliação.  
**Opções**  
(1) Ler sempre da chain/IPFS · (2) **Persistir em BD** (verdade operacional) + reconciliação periódica.  
**Decisão**  
(2) **Persistir no BD** (contract\_versions.doc\_hash/ipfs\_cid) e reconciliar com chain/IPFS via job.  
**Consequências**

* ✅ Baixa latência na UI; auditoria simples; export facilite.
* ❗ Risco de divergência → precisa reconciliação programada.  
  **Riscos & Mitigações**
* Tarefas de **reconcile** diárias; /verify recalcula hash no download.  
  **Gatilhos de revisão**
* Divergência > 0,5% por mês.  
  **Owner**: Backend Lead

### ****ADR-005 · Pagamentos Off-chain (Pix/API bancária) no MVP**** — ****Aceita****

**Contexto**  
Evitar complexidade de stablecoin/escrow já no MVP.  
**Opções**  
(1) On-chain stablecoin + escrow · (2) **Off-chain** via Pix/API · (3) Mistos.  
**Decisão**  
(2) **Off-chain** agora; **on-chain** (split/escrow) nas Fases 2/3.  
**Consequências**

* ✅ Simplicidade/regulatório resolvidos; MVP mais rápido.
* ❗ Sem atomicidade contrato↔pagamento.  
  **Riscos & Mitigações**
* Ledger interno com **idempotência**, retentativas e comprovante anexado ao contrato.  
  **Gatilhos de revisão**
* Exigência de **escrow** ou SLAs contratuais.  
  **Owner**: Product/Tech Lead

### ****ADR-006 · Core API: NestJS (Node) vs FastAPI (Python)**** — ****Aceita****

**Contexto**  
Equipe tem maior familiaridade com Node; ecossistema sólido para queues, web3 e tooling.  
**Opções**  
(1) **NestJS (Node)** · (2) FastAPI (Python).  
**Decisão**  
(1) **NestJS** no MVP (DX, decorators, módulos, guards), mantendo **adapters isolados** para eventual troca.  
**Consequências**

* ✅ Velocidade de desenvolvimento alta; excelente eco com BullMQ, Ethers.js.
* ❗ Tarefas ML futuras podem preferir Python (não impacta core de contratos).  
  **Riscos & Mitigações**
* Definir **BFF/Adapters** com fronteiras REST claras.  
  **Gatilhos de revisão**
* Requisito de ciência de dados pesada dentro da Core API.  
  **Owner**: Tech Lead

### ****ADR-07 · Processamento Assíncrono (Queues) para IPFS/Blockchain**** — ****Aceita****

**Contexto**  
Upload/pinning e TX on-chain são operações lentas/instáveis.  
**Opções**  
(1) Síncrono no request · (2) **Assíncrono com filas** (BullMQ/SQS).  
**Decisão**  
(2) **Workers assíncronos** com idempotency, retries (1m/5m/15m) e DLQ.  
**Consequências**

* ✅ UI responsiva; resiliência a flutuações externas.
* ❗ Complexidade de orquestração (estados pendentes).  
  **Riscos & Mitigações**
* Status e eventos claros; **painel de jobs** para operação.  
  **Gatilhos de revisão**
* SLO de jobs < 98% por dois ciclos.  
  **Owner**: Backend Lead

### ****ADR-008 · Assinatura Digital Eletrônica Simples (hash + IP + UA + TS)**** — ****Provisória****

**Contexto**  
MVP precisa de assinatura juridicamente defensável sem HSM/ICP-Brasil.  
**Opções**  
(1) ICP-Brasil (certificado A3) · (2) **Eletrônica simples** com provas técnicas e consentimento · (3) OIDC+WebAuthn.  
**Decisão**  
(2) **Eletrônica simples** com trilha robusta (hash documento, IP, user-agent, timestamp, consentimento).  
**Consequências**

* ✅ Rápida, baixo custo; suficiente para muitos contratos B2B.
* ❗ Alguns clientes podem requerer ICP ou WebAuthn.  
  **Riscos & Mitigações**
* Prever **upgrade** para assinatura avançada (WebAuthn/ICP) por feature flag.  
  **Gatilhos de revisão**
* Exigência jurídica de cliente enterprise.  
  **Owner**: Legal/Tech Lead

### ****ADR-009 · Versionamento Imutável por**** contract\_versions — ****Aceita****

**Contexto**  
Cada modificação deve gerar uma nova “âncora” de integridade (doc\_hash + CID).  
**Opções**  
(1) Atualizar registro do contrato · (2) **Tabela de versões imutáveis** com ponteiro no contrato.  
**Decisão**  
(2) **contract\_versions imutável**; contracts.current\_version\_id aponta para versão vigente.  
**Consequências**

* ✅ Auditoria simples; verificação por versão; rollback previsível.
* ❗ Aumento no volume de storage (controlável).  
  **Riscos & Mitigações**
* Particionamento por data/contrato em alta escala.  
  **Gatilhos de revisão**
* Crescimento > 100k versões/mês.  
  **Owner**: Data/Backend Lead

### ****ADR-010 · RBAC + JWT (1h) com Refresh (24h)**** — ****Aceita****

**Contexto**  
Controle de acesso por papéis (Empresa, PJ, Financeiro, Jurídico, Admin) e sessão segura.  
**Opções**  
(1) Sessions server-side · (2) **JWT stateless** com refresh · (3) OIDC externo (Auth0/Cognito).  
**Decisão**  
(2) **JWT** assinado (RS256), expira em 1h; refresh em 24h; **guards** por role.  
**Consequências**

* ✅ Escalável e simples; fácil em microserviços.
* ❗ Revogação exige blacklist/rotate.  
  **Riscos & Mitigações**
* Rotate de chaves (KMS) e **token revocation list**.  
  **Gatilhos de revisão**
* Incidentes de segurança ou adoção de IdP corporativo.  
  **Owner**: Security Lead

### ****ADR-011 · Observabilidade: OpenTelemetry + Logs JSON + Grafana**** — ****Aceita****

**Contexto**  
Necessidade de tracing, métricas e logs estruturados para auditoria e SRE.  
**Opções**  
(1) Logs simples apenas · (2) **Otel + Prometheus/Grafana + logs JSON** · (3) APM proprietário.  
**Decisão**  
(2) **OpenTelemetry + Prometheus/Grafana + logs JSON** com traceId/actorId/contractId.  
**Consequências**

* ✅ Observabilidade padronizada e barata.
* ❗ Overhead operacional inicial.  
  **Riscos & Mitigações**
* Dashboards templates e **alertas** prontos (SLO/SLA).  
  **Gatilhos de revisão**
* Crescimento de times/serviços → avaliar APM.  
  **Owner**: SRE Lead

### ****ADR-012 · API REST com OpenAPI 3.1 + Idempotency-Key**** — ****Aceita****

**Contexto**  
Contratos e registros exigem idempotência e documentação clara.  
**Opções**  
(1) REST sem contrato formal · (2) **REST com OpenAPI + idempotência** · (3) GraphQL.  
**Decisão**  
(2) **REST com OpenAPI 3.1**, **Idempotency-Key** em POST/PUT críticos e **paginação** padrão.  
**Consequências**

* ✅ SDKs geráveis; QA automatizável; menos regressões.
* ❗ Menos flexível que GraphQL em agregações.  
  **Riscos & Mitigações**
* Queries agregadas via endpoints específicos ou BFF.  
  **Gatilhos de revisão**
* Painéis complexos exigindo **GraphQL/BFF**.  
  **Owner**: API Lead

### ****ADR-013 · LGPD: Pseudonimização e Retenção**** — ****Aceita****

**Contexto**  
Dados pessoais de usuários (empresa/PJ) sob LGPD.  
**Opções**  
(1) Guardar tudo em claro · (2) **Pseudonimizar + retenção** · (3) Tokenização completa.  
**Decisão**  
(2) **Pseudonimização**, **mascaramento** em exportações, **retenção** configurável (ex.: 5 anos para audit\_logs), e **direito de exclusão** com anonimização.  
**Consequências**

* ✅ Reduz risco legal; atende auditorias.
* ❗ Implementação + processos de DSR (Data Subject Request).  
  **Riscos & Mitigações**
* Playbooks de atendimento LGPD; **trilhas de consentimento**.  
  **Gatilhos de revisão**
* Auditoria LGPD, entrada em mercados regulados.  
  **Owner**: Legal/Data Lead

### ****ADR-014 · Certificado de Autenticidade com QR Code (/verify)**** — ****Aceita****

**Contexto**  
Prova simples para terceiros (fora da plataforma).  
**Opções**  
(1) Sem certificado · (2) **PDF certificado com QR** que consulta /verify?hash=.  
**Decisão**  
(2) **Gerar certificado** com hash, CID, TX e QR Code (verificação pública).  
**Consequências**

* ✅ Autenticidade fácil de provar; reduz suporte.
* ❗ Endpoint público precisa rate limit e cache.  
  **Riscos & Mitigações**
* **Rate limiting**, **cache 10min**, e **anonymização** de dados pessoais.  
  **Gatilhos de revisão**
* Abusos no endpoint público.  
  **Owner**: Backend Lead

### ****ADR-015 · Template de PDF: HTML-to-PDF (WeasyPrint/Puppeteer)**** — ****Aceita****

**Contexto**  
Templates jurídicos dinâmicos e versionados.  
**Opções**  
(1) PDFKit programático puro · (2) **HTML-to-PDF** (CSS, controle visual).  
**Decisão**  
(2) **HTML/CSS → PDF** (Puppeteer/WeasyPrint) com **templates versionados** em repositório (ou S3).  
**Consequências**

* ✅ Agilidade para jurídico/design; visual consistente.
* ❗ Requer sandbox/headless e tuning de fonts.  
  **Riscos & Mitigações**
* Pipeline de fontes, testes visuais (snapshot PDF).  
  **Gatilhos de revisão**
* Performance de render em alto volume.  
  **Owner**: Front/Backend Lead

# Resumo Executivo

O **módulo de Contrato Digital Off-Chain da AURE** é o **pilar central de confiança** da plataforma, responsável por digitalizar e automatizar todo o ciclo contratual entre empresas e prestadores de serviço (PJs), garantindo **segurança, validade jurídica e rastreabilidade técnica**.

Ele permite que contratos sejam **gerados, assinados e registrados** de forma 100% digital, substituindo processos manuais, cartoriais e sujeitos a falhas por um fluxo totalmente auditável e escalável.

A solução adota uma arquitetura **off-chain híbrida**, que combina:

* **Integridade criptográfica** via hash SHA-256, assegurando que o documento não foi alterado.
* **Armazenamento descentralizado** no IPFS, que funciona como um cofre público imutável.
* **Registro público de autenticidade** em blockchain (Polygon Testnet), garantindo transparência e prova de existência verificável globalmente.

Com isso, a AURE estabelece uma **infraestrutura confiável e econômica**, que:

* Elimina a dependência de intermediários jurídicos e cartoriais.
* Reduz custos operacionais e o risco de litígio.
* Fornece rastreabilidade completa de assinaturas, versões e aprovações.
* Cria a base técnica para futuras automações de **split payment**, **escrow**, e **compliance financeiro-jurídico**.

O módulo foi desenvolvido para ser **modular, API-first e escalável**, preparado para integrar-se a:

* Plataformas fiscais (NFS-e),
* Bancos e meios de pagamento (Pix/API Open Finance),
* Smart contracts financeiros (fase on-chain futura).

### ****Visão de Impacto****

| **Dimensão** | **Resultado Esperado** |
| --- | --- |
| **Negócio** | Redução de até 80% no tempo de formalização contratual e 60% no custo jurídico-operacional. |
| **Técnico** | 100% dos contratos rastreáveis, assinados e registrados com integridade comprovada. |
| **Estratégico** | Base para expansão da AURE como **infraestrutura fintech** de confiança para automação contratual e liquidação de pagamentos. |

### ****Mensagem-Chave****

“O módulo de Contrato Digital da AURE não é apenas uma funcionalidade — é o **coração da infraestrutura de confiança** que permitirá à empresa transformar o modo como o Brasil valida, executa e liquida contratos corporativos.”

Link mockup  
  
https://chatgpt.com/canvas/shared/68f54030fb6481919af973810068337a