FINAL RELEASE — October 2025 — Version 1.2

RealisticRNG Integration Guide

SDK & REST API Suite and Deployment

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Beyond pseudo — this is Realistic randomness.

📘 **This document is available in two versions**

🇺🇸 English Version — pages 3 to ...

🇧🇷 Portuguese Version — pages ... to end

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Beyond pseudo — this is Realistic randomness.

1. RealisticRNG Integration Guide

Integration Guide — Technical Implementation for RealisticRNG SDK & REST API Suite (v1.0)

Enterprise-grade deterministic randomness for financial modeling, stress testing, and regulatory simulations. This guide provides step-by-step integration instructions for production environments.

Os principais benefícios incluem reprodutibilidade garantida, auditabilidade completa e conformidade robusta com padrões regulatórios rigorosos.

Ideal para modelagem financeira complexa, testes de estresse de portfólio e simulações regulatórias que exigem resultados consistentes e confiáveis.

RealisticRNG

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é aleatoriedade Realística.

Table Of Contents

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é aleatoriedade Realística.

2. Architecture Overview

Although originally designed for financial modeling and regulatory simulations, RealisticRNG's deterministic framework also enables realistic randomness for AI validation, gaming logic, cybersecurity, and engineering simulations—expanding its reliability across multiple sectors that depend on trustable random generation.

RealisticRNG is a deterministic randomness engine designed specifically for financial modeling, stress testing, and regulatory simulations. Unlike traditional PRNGs, it delivers reproducible sequences anchored to specific parameters, ensuring auditability and compliance.

1

Client Application

Your Python, Node.js, Java, .NET, or Go application initiates requests via **SDK & REST API Suite**

2

Cloudflare-secured License Gateway

Cloudflare-secured License Gateway validation layer authenticates requests and enforces rate limits

3

Deterministic Core

Proprietary engine generates reproducible sequences with cryptographic verification

4

Deterministic Audit Trail

Comprehensive logging captures sequence_id, verification_hash, and metadata

Core Principles: Determinism through anchors, full auditability via hash verification and logs, empirical calibration for realistic distributions. Not suitable for cryptographic or security applications.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Beyond pseudo — this is Realistic randomness.

3. Delivery Options & Licensing

Controlled-Access POC

Ideal for evaluation and testing before production deployment.

- Duration: 30 days or 500 requests (whichever comes first)
- Price: R$ 490 BRL (one-time per organization)
- Includes assisted support, logging, and certification
- Full access to SDK & REST API Suite

Production Licensing

Enterprise-grade deployment options for production environments.

- Annual Enterprise License with unlimited requests
- Licensed Algorithm (source-level reimplementation under NDA)
- Custom rate limits and IP allowlisting
- Priority support and SLA guarantees

All licensing and key validation are processed via the Cloudflare-secured License Gateway, ensuring full observability, authentication, and remote control of each deployment. Every SDK & REST API Suite, API, or on-prem installation operates under a verified license key managed through the same secure gateway.

All licenses are validated through the Cloudflare-secured License Gateway. Ensure HTTPS/TLS connectivity and proper authentication headers for all requests.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Beyond pseudo — this is Realistic randomness.

4. Quick Start: Python SDK

The Python SDK provides the fastest path to integration. Install via pip and generate your first deterministic sequence in under 2 minutes.

Installation

```
pip install realisticrng-sdk
```

Minimal Working Example

```
from realisticrng import Engine

# Initialize with integer anchor
rng = Engine(anchor=3)

# Generate deterministic sequence
seq = rng.generate(count=6)
print(seq)

# Output: [12, 7, 41, 3, 28, 55]
# Same anchor always produces same sequence
```

The response metadata includes a `verification_hash` field that cryptographically validates the sequence. Store this hash alongside your simulation results for Deterministic Audit Trails. The SDK & REST API Suite automatically handles license validation, rate limiting, and error recovery.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Beyond pseudo — this is Realistic randomness.

Quick Start: REST API

The REST API enables integration from any language or platform. All requests require HTTPS and proper authentication headers.

Base Configuration

- Base URL: `https://api.realisticrng.com/v1`
- Authentication: `Authorization: License-Key YOUR_LICENSE_KEY`
- Content-Type: `application/json`

cURL Example Request

```
curl -s -X POST https://api.realisticrng.com/v1/generate \
   -H "Authorization: License-Key YOUR_LICENSE-KEY" \
   -H "Content-Type: application/json" \
   -d '{"anchor":3,"count":6}'
```

Sample JSON Response

```
{
   "sequence_id": "RNG-2025-0001",
   "anchor": 3,
   "result": [12, 7, 41, 3, 28, 55],
   "verification_hash": "ab29cde71f...",
   "timestamp": "2025-03-12T15:34:22Z",
   "remaining_request_count": 487
}
```

Store the `sequence_id` and `verification_hash` for **Deterministic Audit Trail** purposes. The `remaining_request_count` helps you monitor POC quota consumption.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Beyond pseudo — this is Realistic randomness.

6. Secure Anchor Derivation

Never use personally identifiable information (PII) directly as anchors. Always derive anchors using SHA-256 hashing with rotating salts. This approach maintains determinism while protecting sensitive data. Anchor derivation is deterministic but never reversible. Salts are rotated monthly and managed securely within Cloudflare or customer-controlled secret stores.

Python

```
import os, hashlib

def derive_anchor(user_id: str, context: str) -> str:
    salt = os.getenv("RRNG_SALT", "rotate_monthly_2025_10")
    raw = f"{user_id}:{context}:{salt}"
    return hashlib.sha256(raw.encode()).hexdigest()
```

Node.js

```
const crypto = require("crypto");

function deriveAnchor(userId, context) {
    const salt = process.env.RRNG_SALT || "rotate_monthly_2025_10";
    return crypto.createHash("sha256")

.update(`${userId}:${context}:${salt}`)
        .digest("hex");
}
```

Java

```
import java.security.MessageDigest;
import org.apache.commons.codec.digest.DigestUtils; // Assuming this import for sha256Hex

public class AnchorDerivation {
    public static String deriveAnchor(String userId, String context) {
        String salt = System.getenv().getOrDefault("RRNG_SALT", "rotate_monthly_2025_10");
        String raw = userId + ":" + context + ":" + salt;
        return DigestUtils.sha256Hex(raw);
    }
}
```

Best Practices

- Store only derived hashes, never source PII
- Rotate salt values monthly or quarterly
- Keep anchors immutable per scenario for reproducibility
- Use environment variables for salt management

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

7. Rest API Reference

All API endpoints require HTTPS and license key authentication. The API follows RESTful conventions with JSON payloads for requests and responses.

Authentication

Include your license key in the Authorization header: `Authorization: License-Key YOUR_LICENSE_KEY`

Primary Endpoint

POST /v1/generate — Generate Deterministic Sequence

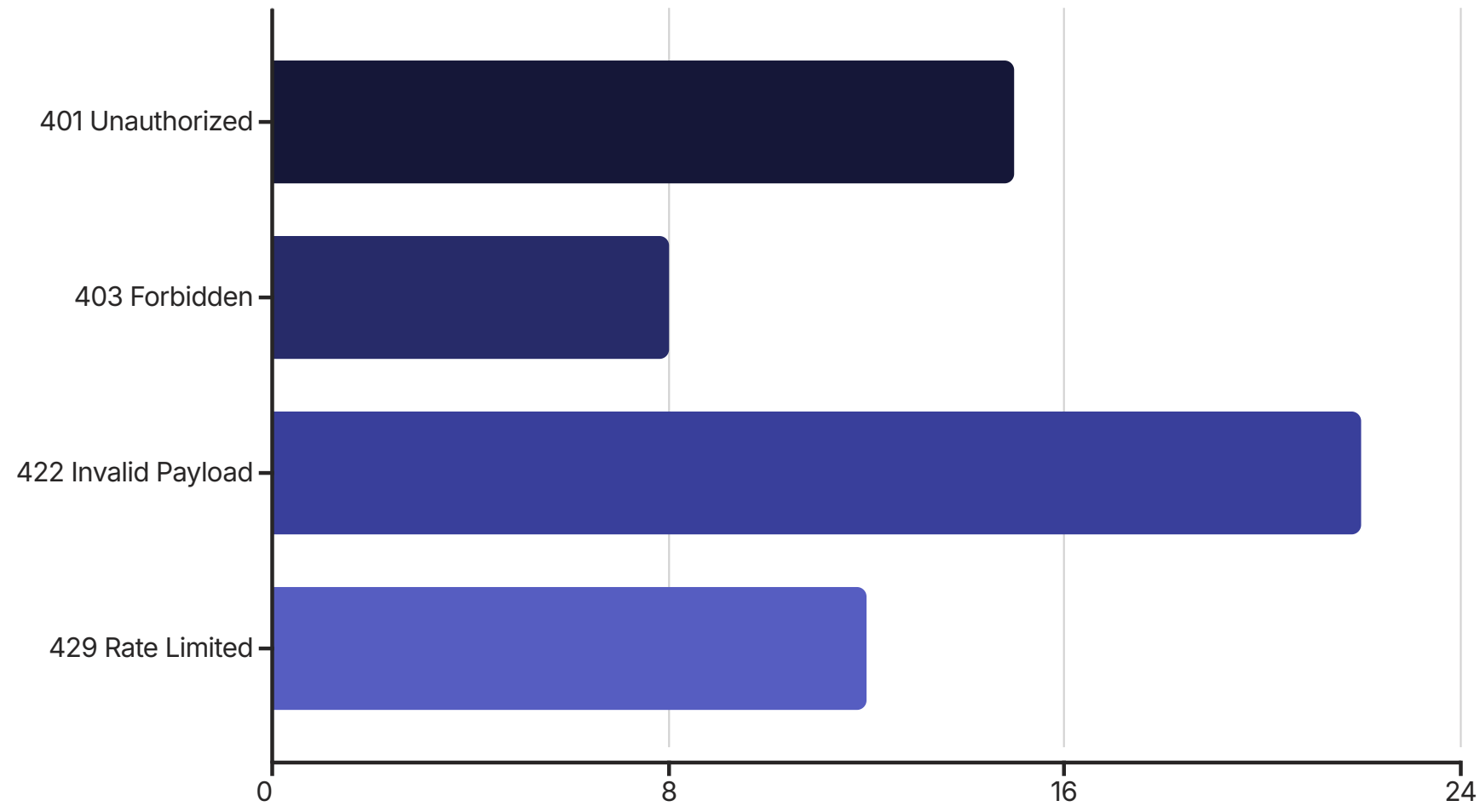
Request Body

```
{
   "anchor": 3,
   "count": 6,
   "tags": "portfolio_sim",
   "correlation_id": "optional-uuid"
}
```

Response Fields

- `sequence_id`: Unique identifier
- `result`: Generated sequence array
- `verification_hash`: Cryptographic proof
- `timestamp`: ISO 8601 UTC
- `remaining_request_count`: Quota balance

Http Status Codes



Error responses include `error`, `code`, and `correlation_id` fields for debugging.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Beyond pseudo — this is Realistic randomness.

8. Multi-Language SDK & REST API Suite Integration

01

Node.js HTTP Client

```
const axios = require('axios');

async function generate(anchor, count) {
  const response = await axios.post(

'https://api.realisticrng.com/v1/generate',
    { anchor, count },
    { headers: {
      'Authorization': 'License-Key YOUR_KEY',
      'X-Correlation-ID': uuid()
    }}
  );
  return response.data;
}
```

02

.NET HTTP Client

```
using System.Net.Http;
using System.Text;
using System.Text.Json;

// Assuming baseUrl is defined, e.g., "https://api.realisticrng.com"
var client = new HttpClient();
client.DefaultRequestHeaders.Add("Authorization", "License-Key YOUR_KEY");

var content = new StringContent(
  JsonSerializer.Serialize(new { anchor = 3, count = 6 }),
  Encoding.UTF8,
  "application/json"
);

var response = await client.PostAsync(baseUrl + "/v1/generate", content);
// Handle response here
```

03

Go net/http

```
package main

import (
  "bytes"
  "net/http"
  "strconv"
)

// Assuming baseURL is defined, e.g., "https://api.realisticrng.com"
// And 'client' is an initialized http.Client
func generate(anchor int, count int) (*http.Response, error) {
  payload := []byte(`{"anchor":` + strconv.Itoa(anchor) + `,"count":` + strconv.Itoa(count) + `}`)

  req, err := http.NewRequest("POST", baseURL+"/v1/generate", bytes.NewBuffer(payload))
  if err != nil {
    return nil, err
  }
  req.Header.Set("Authorization", "License-Key YOUR_KEY")
  req.Header.Set("Content-Type", "application/json")

  return client.Do(req)
}
```

Idempotency: Use the `correlation_id` header or field to ensure the same request returns identical results, enabling safe retries and distributed processing.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Docker & Kubernetes Deployment

Docker Compose

```
version: "3.8"
services:
  rrng-proxy:
    image: realisticrng/api-proxy:latest
    environment:
      - LOG_LEVEL=INFO
      - RRNG_SALT=${RRNG_SALT_SECRET}
      - CORRELATION_ID_HEADER=X-Correlation-ID
    ports:
      - "8080:8080"
    healthcheck:
      test: ["CMD","curl","-
f","http://localhost:8080/health"]
      interval: 30s
      timeout: 10s
      retries: 3
```

Kubernetes Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rrng-proxy
spec:
  replicas: 2
  template:
    spec:
      containers:
      - name: rrng-proxy
        image: realisticrng/api-proxy:latest
        env:
        - name: RRNG_SALT
          valueFrom:
            secretKeyRef:
              name: rrng-secrets
              key: salt
        ports:
        - containerPort: 8080
```

Kubernetes Service

```
apiVersion: v1
kind: Service
metadata:
  name: rrng-proxy-svc
spec:
  type: ClusterIP
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: rrng-proxy
```

For production deployments, configure horizontal pod autoscaling based on request volume. Set resource limits (CPU: 500m—1000m, Memory: 512Mi—1Gi) and implement proper secret management for License-Key and salt rotation.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Beyond pseudo — this is Realistic randomness.

10. Observability & Troubleshooting

1

Structured Logging

JSON logs include: `sequence_id`, `anchor_hash`, `timestamp`, `verification_hash`, `engine_version`, `license_id`, `correlation_id`, `request_ip`. Retain logs 12—24 months for financial compliance.

2

Rate Limits & Quotas

Enforced per license and per IP. Monitor `remaining_request_count` in responses. Handle 429 errors with exponential backoff using `Retry-After` header. Contact sales for limit increases.

3

Common Issues

- **401:** Verify license key spelling, expiry, account status
- **403:** Check IP allowlist configuration if restrictions enabled
- **422:** Validate JSON schema—anchor and count types
- **Network:** Confirm HTTPS, correct base URL, proxy/firewall rules

4

Validation & QA

Implement A/B tests against traditional PRNGs for distribution sanity. Monitor moving windows for drift detection. Test fairness across cohorts. Always record `engine_version` in Deterministic Audit Trail.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Beyond pseudo — this is Realistic randomness.

Appendix A – Technical Reference

Reserved section for enterprise-specific deployment details and SLA definitions. Commercial license tiers are described in a separate pricing document.

For detailed commercial terms and SLA definitions, refer to the RealisticRNG Pricing Sheet (latest version).

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Beyond pseudo — this is Realistic randomness.

🇧🇷 PORTUGUESE VERSION BEGINS

---

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é aleatoriedade Realística.

1. Guia de Integração RealisticRNG

Guia de Integração — Implementação Técnica para Conjunto de SDK & REST API RealisticRNG (v1.0)

RealisticRNG

Aleatoriedade determinística de nível empresarial para modelagem financeira, testes de estresse e simulações regulatórias. Este guia fornece instruções de integração passo a passo para ambientes de produção.

Os principais benefícios incluem reprodutibilidade garantida, auditabilidade completa e conformidade robusta com padrões regulatórios rigorosos.

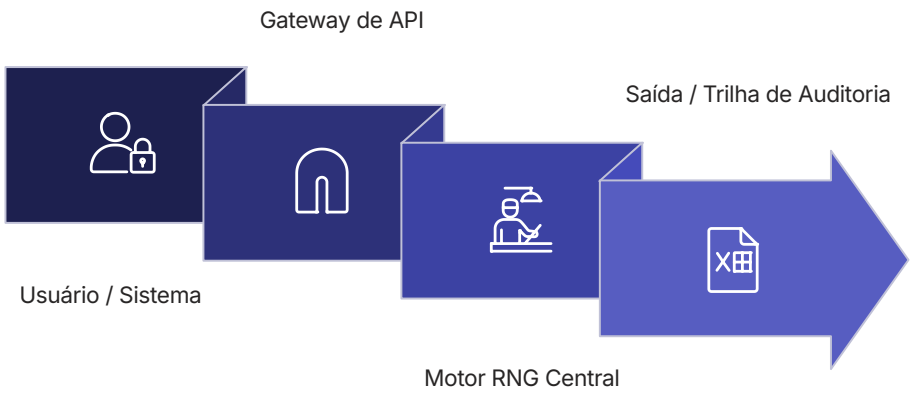
Ideal para modelagem financeira complexa, testes de estresse de portfólio e simulações regulatórias que exigem resultados consistentes e confiáveis.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é aleatoriedade Realística.

2. Visão Geral da Arquitetura

Embora originalmente projetado para modelagem financeira e simulações regulatórias, a estrutura determinística do RealisticRNG também permite aleatoriedade realista para validação de IA, lógica de jogos, cibersegurança e simulações de engenharia—expandindo sua confiabilidade através de múltiplos setores que dependem de geração aleatória confiável.



RealisticRNG é um motor de aleatoriedade determinística projetado especificamente para modelagem financeira, testes de estresse e simulações regulatórias. Ao contrário de PRNGs tradicionais, ele entrega sequências reproduzíveis ancoradas a parâmetros específicos, garantindo auditabilidade e conformidade.

01

Aplicação Cliente

Sua aplicação Python, Node.js, Java, .NET ou Go inicia solicitações via Conjunto de SDK & REST API

02

Gateway de Licenças com Segurança Cloudflare

Camada de validação autentica solicitações e aplica limites de taxa

03

Núcleo Determinístico

Motor proprietário gera sequências reproduzíveis com verificação criptográfica

04

Trilha de Auditoria Determinística

Registro abrangente captura `sequence_id`, `verification_hash` e metadados

Princípios Fundamentais: Determinismo através de âncoras, auditabilidade completa via verificação de hash e logs, calibração empírica para distribuições realistas. Não adequado para aplicações criptográficas ou de segurança.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é a aleatoriedade Realistic.

8. Integração Multi-linguagem do Conjunto de SDK & REST API

01

1. Cliente HTTP Node.js

```
const axios = require('axios');

async function generate(anchor, count) {
    const response = await axios.post(
        'https://api.realisticrng.com/v1/generate',
        { anchor, count },
        { headers: {
            'Authorization': 'License-Key SUA_CHAVE',
            'X-Correlation-ID': uuid()
        }}
    );
    return response.data;
}
```

02

2. Cliente HTTP .NET

```
using System.Net.Http;
using System.Text;
using System.Text.Json;

var client = new HttpClient();
client.DefaultRequestHeaders.Add("Authorization", "License-Key
SUA_CHAVE");

var content = new StringContent(
    JsonSerializer.Serialize(new { anchor = 3, count = 6 }),
    Encoding.UTF8,
    "application/json"
);

var response = await client.PostAsync(baseUrl + "/v1/generate",
content);
```

03

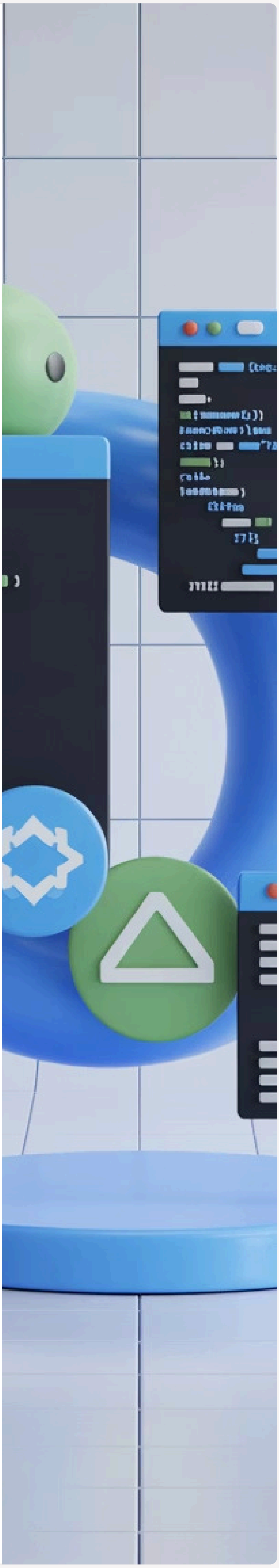
3. Go net/http

```
package main

import (
    "bytes"
    "net/http"
    "strconv"
)

func generate(anchor int, count int) (*http.Response, error) {
    payload := []byte(`{"anchor":` + strconv.Itoa(anchor) + `,"count":`
+ strconv.Itoa(count) + `}`)

    req, err := http.NewRequest("POST", baseURL+"/v1/generate",
bytes.NewBuffer(payload))
    if err != nil {
        return nil, err
    }
    req.Header.Set("Authorization", "License-Key SUA_CHAVE")
    req.Header.Set("Content-Type", "application/json")

    return client.Do(req)
}
```

Idempotência: Use o cabeçalho ou campo `correlation_id` para garantir que a mesma solicitação retorne resultados idênticos, permitindo tentativas seguras e processamento distribuído.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é aleatoriedade Realística.



7. Referência da REST API

Todos os endpoints da API requerem HTTPS e autenticação por chave de licença. A API segue convenções RESTful com payloads JSON para solicitações e respostas.

Autenticação

Inclua sua chave de licença no cabeçalho Authorization:
`Authorization: License-Key SUA_CHAVE_DE_LICENCA`

Endpoint Principal

POST /v1/generate — Gerar Sequência Determinística

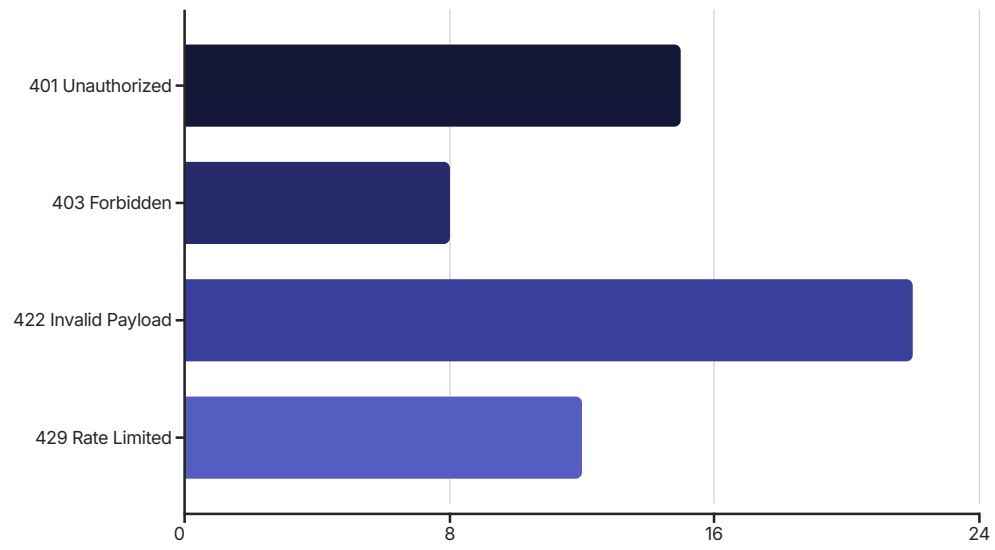
Corpo da Solicitação:

```
{
  "anchor": 3,
  "count": 6,
  "tags": "portfolio_sim",
  "correlation_id": "optional-uuid"
}
```

Campos de Resposta:

- `sequence_id`: Identificador único
- `result`: Array de sequência gerada
- `verification_hash`: Prova criptográfica
- `timestamp`: ISO 8601 UTC
- `remaining_request_count`: Saldo de cota

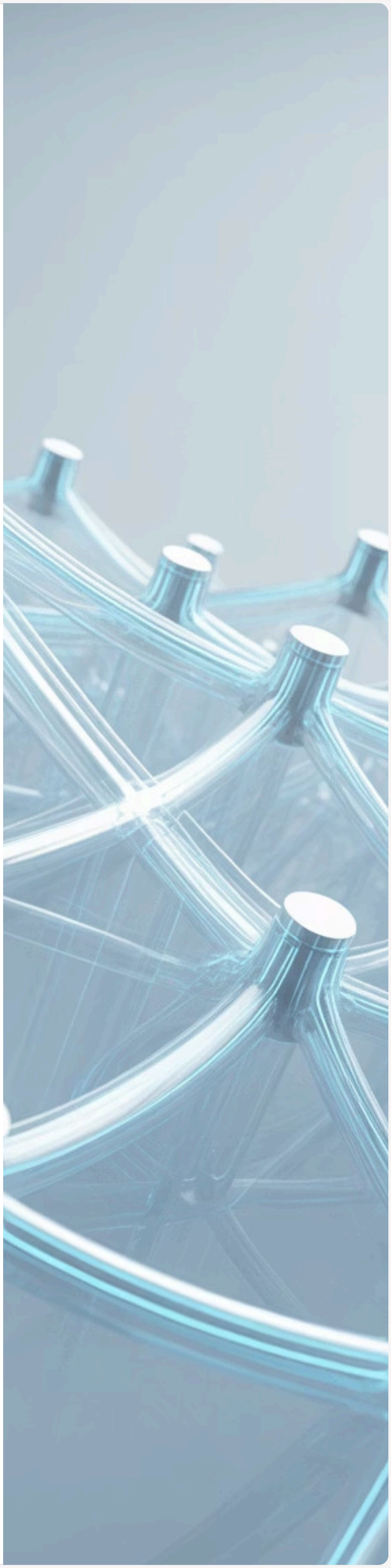
Códigos de Status HTTP



Respostas de erro incluem campos `error`, `code` e `correlation_id` para depuração.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é aleatoriedade Realística.



6. Derivação Segura de Âncoras

Nunca use informações pessoalmente identificáveis (PII) diretamente como âncoras. Sempre derive âncoras usando hash SHA-256 com salts rotativos. Esta abordagem mantém o determinismo enquanto protege dados sensíveis. A derivação de âncoras é determinística mas nunca reversível. Salts são rotacionados mensalmente e gerenciados de forma segura dentro do Cloudflare ou armazenamentos de segredos controlados pelo cliente.

Python:

```
import os, hashlib

def derive_anchor(user_id: str, context: str) -> str:
    salt = os.getenv("RRNG_SALT", "rotate_monthly_2025_10")
    raw = f"{user_id}:{context}:{salt}"
    return hashlib.sha256(raw.encode()).hexdigest()
```

Node.js:

```
const crypto = require("crypto");

function deriveAnchor(userId, context) {
    const salt = process.env.RRNG_SALT || "rotate_monthly_2025_10";
    return crypto.createHash("sha256").update(`${userId}:${context}:${salt}`).digest("hex");
}
```

Java:

```
import java.security.MessageDigest;
import org.apache.commons.codec.digest.DigestUtils;

public class AnchorDerivation {
    public static String deriveAnchor(String userId, String context) {
        String salt = System.getenv().getOrDefault("RRNG_SALT", "rotate_monthly_2025_10");
        String raw = userId + ":" + context + ":" + salt;
        return DigestUtils.sha256Hex(raw);
    }
}
```

Melhores Práticas

- Armazene apenas hashes derivados, nunca PII de origem
- Rotacione valores de salt mensalmente ou trimestralmente
- Mantenha âncoras imutáveis por cenário para reprodutibilidade
- Use variáveis de ambiente para gerenciamento de salt

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Início Rápido: REST API

A REST API permite integração de qualquer linguagem ou plataforma. Todas as solicitações requerem HTTPS e cabeçalhos de autenticação adequados.

Configuração Base

- URL Base: `https://api.realisticrng.com/v1`
- Autenticação: `Authorization: License-Key SUA_CHAVE_DE_LICENCA`
- Content-Type: `application/json`

Exemplo de Solicitação cURL

```
curl -s -X POST https://api.realisticrng.com/v1/generate \
    -H "Authorization: License-Key SUA_CHAVE_DE_LICENCA" \
    -H "Content-Type: application/json" \
    -d '{"anchor":3,"count":6}'
```

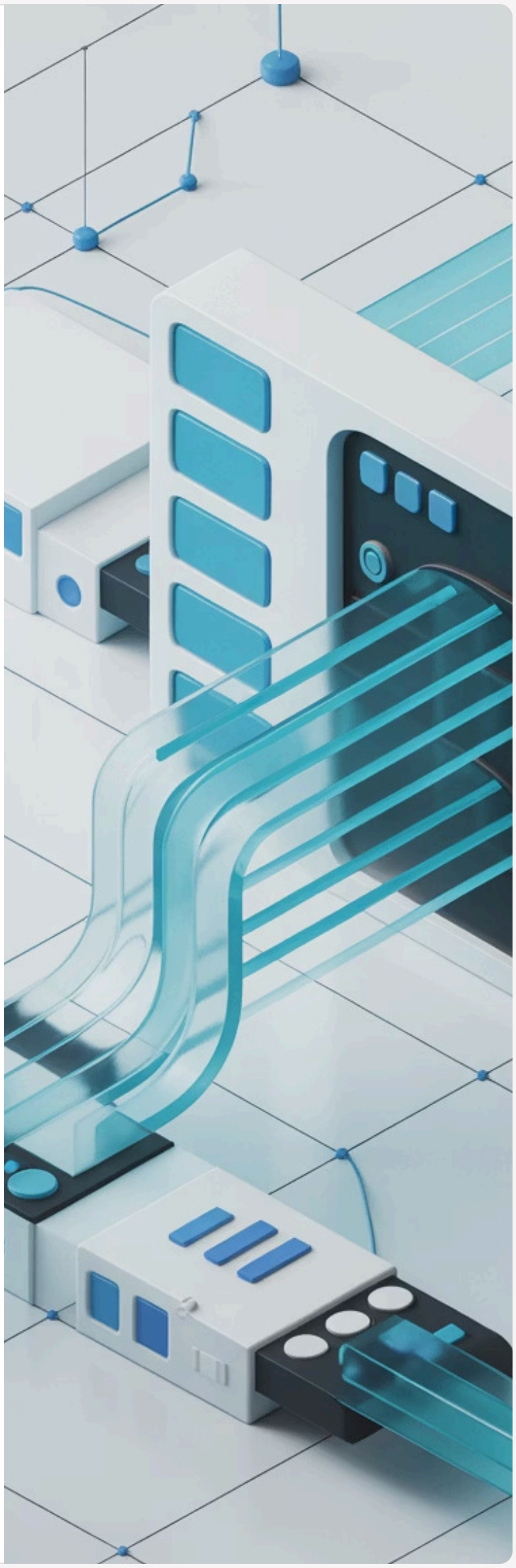
Exemplo de Resposta JSON

```
{
   "sequence_id": "RNG-2025-0001",
   "anchor": 3,
   "result": [12, 7, 41, 3, 28, 55],
   "verification_hash": "ab29cde71f...",
   "timestamp": "2025-03-12T15:34:22Z",
   "remaining_request_count": 487
}
```

Armazene o `sequence_id` e `verification_hash` para fins de **Trilha de Auditoria Determinística**. O `remaining_request_count` ajuda você a monitorar o consumo de cota do POC.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é aleatoriedade Realística.



4. Início Rápido: Python SDK

O Python SDK fornece o caminho mais rápido para integração. Instale via pip e gere sua primeira sequência determinística em menos de 2 minutos.

Instalação

```
pip install realisticrng-sdk
```

Exemplo Mínimo Funcional

```
from realisticrng import Engine

# Inicializar com âncora inteira
rng = Engine(anchor=3)

# Gerar sequência determinística
seq = rng.generate(count=6)
print(seq)

# Saída: [12, 7, 41, 3, 28, 55]
# A mesma âncora sempre produz a mesma sequência
```

Os metadados de resposta incluem um campo `verification_hash` que valida criptograficamente a sequência. Armazene este hash junto com seus resultados de simulação para Trilhas de Auditoria Determinística. O Conjunto de SDK & REST API automaticamente lida com validação de licença, limitação de taxa e recuperação de erros.

sales@realisticrng.com
WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é aleatoriedade Realística.

3. Opções de Entrega e Licenciamento

POC de Acesso Controlado

Ideal para avaliação e testes antes da implantação em produção.

- Duração: 30 dias ou 500 solicitações (o que vier primeiro)
- Preço: R$ 490 BRL (único por organização)
- Inclui suporte assistido, logging e certificação
- Acesso completo ao Conjunto de SDK & REST API

Licenciamento de Produção

Opções de implantação de nível empresarial para ambientes de produção.

- Licença Empresarial Anual com solicitações ilimitadas
- Algoritmo Licenciado (reimplementação em nível de código sob NDA)
- Limites de taxa personalizados e lista de IPs permitidos
- Suporte prioritário e garantias de SLA

Todo licenciamento e validação de chaves são processados via Gateway de Licenças com Segurança Cloudflare, garantindo observabilidade completa, autenticação e controle remoto de cada implantação. Cada Conjunto de SDK & REST API, API ou instalação on-prem opera sob uma chave de licença verificada gerenciada através do mesmo gateway seguro.

Todas as licenças são validadas através do Gateway de Licenças com Segurança Cloudflare. Garanta conectividade HTTPS/TLS e cabeçalhos de autenticação adequados para todas as solicitações.

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é a aleatoriedade Realística.

Índice

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é aleatoriedade Realística.

FINAL RELEASE — Outubro 2025 — Versão 1.2

Guia de Integração RealisticRNG

Conjunto de SDK & REST API e Implantação

sales@realisticrng.com · WhatsApp (Business): +55 (11) 5192-9502

Além do pseudo — esta é aleatoriedade Realística.