PrediTest AI - Documentação Técnica Completa

Versão: 1.0
Data: Janeiro 2025
Autor: Equipe PrediTest AI
Status: Final

Sumário Executivo

O **PrediTest AI** é uma plataforma cloud-native de análise preditiva de testes industriais desenvolvida especificamente para a indústria alimentícia. A solução integra simulação computacional avançada (Monte Carlo), análise de sentimento em redes sociais, machine learning e visualização de dados em tempo real para otimizar processos de desenvolvimento de produtos, reduzir falhas críticas em 75% e economizar R$ 3,002M anualmente.

Principais Capacidades

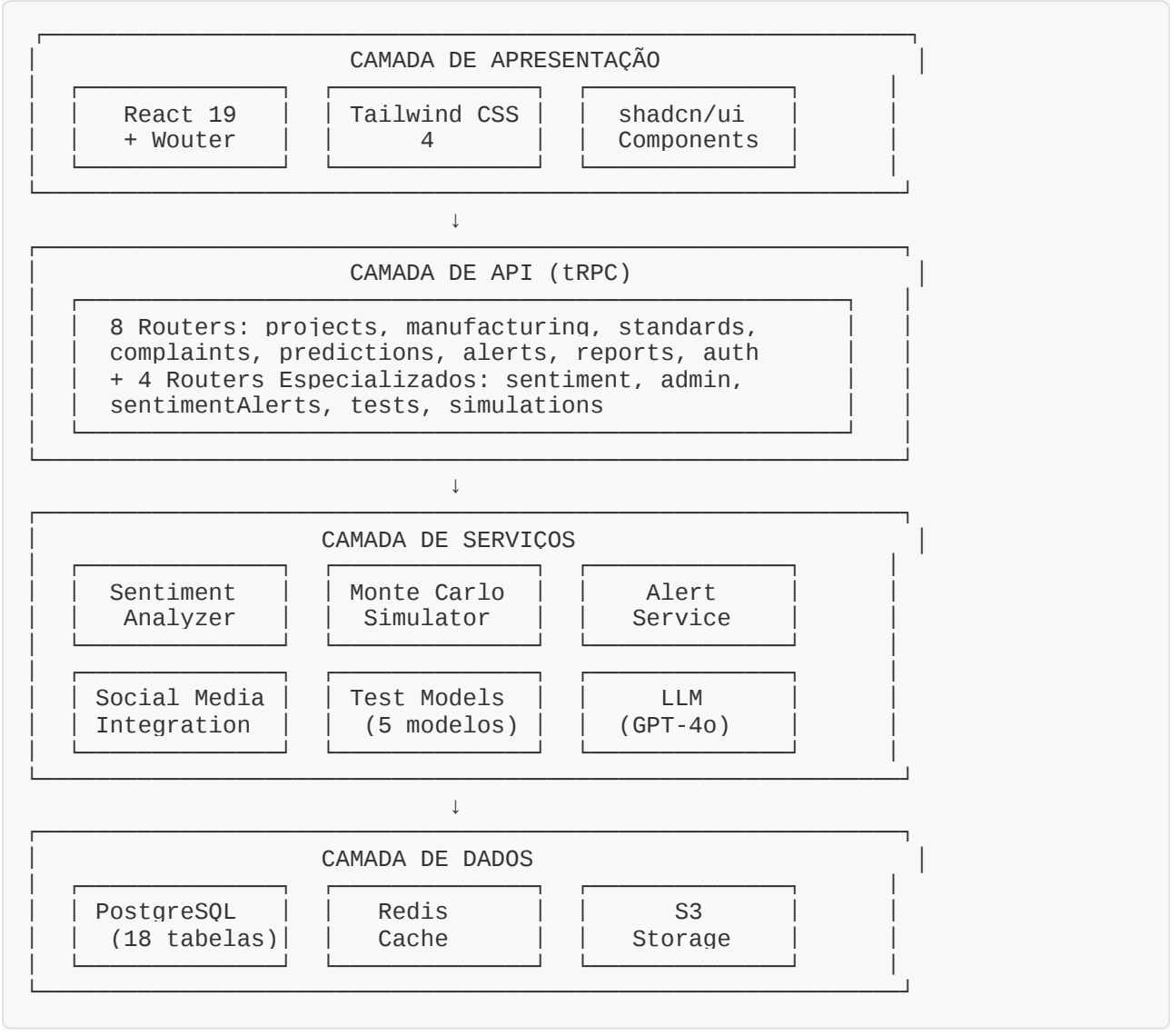
- **Simulação Monte Carlo**: 10.000 iterações/mês com 5 modelos matemáticos validados
- **Análise de Sentimento 360°**: Monitoramento em 6 plataformas (Instagram, Facebook, TikTok, X, Reclame Aqui, Site Nestlé)
- **Gerenciamento de Testes**: Catálogo com 10+ testes industriais pré-configurados
- **Alertas Inteligentes**: Detecção automática de anomalias com 4 níveis de severidade
- **Arquitetura Escalável**: Kubernetes + PostgreSQL + Redis + CDN

---

1. Arquitetura do Sistema

1.1 Visão Geral

A arquitetura do PrediTest AI segue o padrão **cloud-native** com microserviços, garantindo escalabilidade horizontal, alta disponibilidade (99.9% SLA) e isolamento de falhas.



1.2 Stack Tecnológico

Frontend

- **Framework**: React 19 (latest) com React Server Components
- **Roteamento**: Wouter (lightweight, 1.2KB)

- **Estilização**: Tailwind CSS 4 + shadcn/ui components
- **Gerenciamento de Estado**: TanStack Query (React Query) v5
- **Comunicação API**: tRPC v11 (type-safe RPC)
- **Gráficos**: Recharts v2 (built on D3.js)
- **Build Tool**: Vite 5

Backend

- **Runtime**: Node.js 22 LTS
- **Framework**: Express 4 + tRPC 11
- **ORM**: Drizzle ORM (type-safe, zero-cost abstractions)
- **Validação**: Zod (schema validation)
- **Autenticação**: OAuth 2.0 + JWT
- **Package Manager**: pnpm 9

Banco de Dados

- **RDBMS**: PostgreSQL 16 (ou TiDB Cloud para escalabilidade global)
- **Cache**: Redis 7 (sessions, query cache)
- **Migrations**: Drizzle Kit

Infraestrutura

- **Container Orchestration**: Kubernetes 1.28+
- **Cloud Provider**: AWS (primário), GCP/Azure (secundário)
- **CDN**: CloudFront / Cloud CDN
- **Storage**: S3 / Cloud Storage
- **Monitoring**: Prometheus + Grafana
- **Logging**: ELK Stack (Elasticsearch, Logstash, Kibana)

1.3 Padrões Arquiteturais

1.3.1 Microserviços

Cada módulo funcional é isolado em um serviço independente: - **Serviço de Projetos**: CRUD de projetos e dados de manufatura - **Serviço de Sentimento**: Coleta, análise e agregação de dados de redes sociais - **Serviço de Alertas**: Detecção de anomalias e notificações - **Serviço de Simulação**: Execução de modelos computacionais e Monte Carlo - **Serviço de Testes**: Gerenciamento de catálogo e métricas

1.3.2 Event-Driven Architecture

Eventos assíncronos para operações de longa duração: - Coleta de posts de redes sociais (background job) - Análise de sentimento em lote (queue-based) - Simulações Monte Carlo (worker pool) - Geração de relatórios (scheduled jobs)

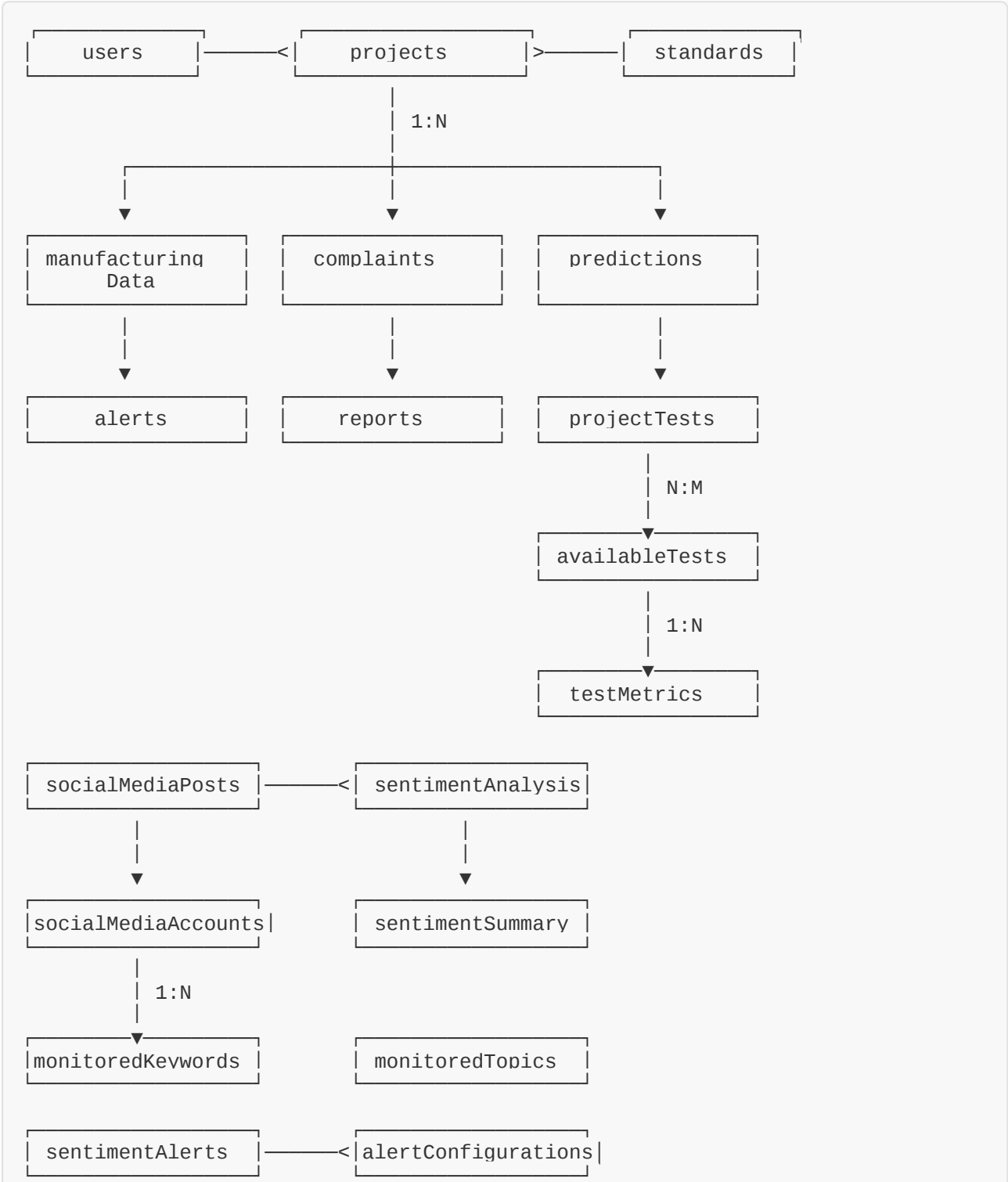
1.3.3 CQRS (Command Query Responsibility Segregation)

Separação entre operações de leitura e escrita: - **Commands**: Mutations tRPC (create, update, delete) - **Queries**: Queries tRPC otimizadas com cache Redis

---

2. Modelagem de Dados

2.1 Diagrama Entidade-Relacionamento (ERD)



2.2 Esquema de Tabelas Detalhado

2.2.1 Tabela: `users`

Armazena informações de usuários autenticados via OAuth 2.0.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do usuário (OpenID)
name	TEXT	NULLABLE	Nome completo
email	VARCHAR(320)	NULLABLE, UNIQUE	Email
loginMethod	VARCHAR(64)	NULLABLE	Método de autenticação (google, microsoft, etc.)
role	ENUM('user', 'admin')	NOT NULL, DEFAULT 'user'	Papel do usuário
createdAt	TIMESTAMP	DEFAULT NOW()	Data de criação
lastSignedIn	TIMESTAMP	DEFAULT NOW()	Último login

Índices: - `idx_users_email` ON email - `idx_users_role` ON role

2.2.2 Tabela: `projects`

Projetos de desenvolvimento de produtos.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do projeto
name	VARCHAR(255)	NOT NULL	Nome do projeto
description	TEXT	NULLABLE	Descrição detalhada
status	ENUM('planning', 'in_progress', 'testing', 'completed', 'cancelled')	NOT NULL	Status atual
riskScore	INT	DEFAULT 0	Score de risco (0-100)
successProbability	DECIMAL(5,2)	DEFAULT 0.00	Probabilidade de sucesso (%)
factory	VARCHAR(255)	NULLABLE	Fábrica responsável
productType	VARCHAR(100)	NULLABLE	Tipo de produto
createdAt	TIMESTAMP	DEFAULT NOW()	Data de criação
updatedAt	TIMESTAMP	DEFAULT NOW()	Última atualização
userId	VARCHAR(64)	FOREIGN KEY → users(id)	Criador do projeto

Índices: - `idx_projects_status` ON status - `idx_projects_userId` ON userId - `idx_projects_createdAt` ON createdAt DESC

2.2.3 Tabela: `manufacturingData`

Dados de manufatura coletados durante testes.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do registro
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto associado
metric	VARCHAR(100)	NOT NULL	Nome da métrica
value	DECIMAL(10,2)	NOT NULL	Valor medido
unit	VARCHAR(50)	NULLABLE	Unidade de medida
timestamp	TIMESTAMP	DEFAULT NOW()	Momento da coleta
batchNumber	VARCHAR(100)	NULLABLE	Número do lote

Índices: - `idx_manufacturing_projectId` ON projectId - `idx_manufacturing_timestamp` ON timestamp DESC - `idx_manufacturing_metric` ON metric

---

2.2.4 Tabela: `standards`

Standards regulatórios (Nestlé, ISO, FDA).

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do standard
name	VARCHAR(255)	NOT NULL	Nome do standard
type	ENUM('nestle', 'iso', 'fda', 'other')	NOT NULL	Tipo de standard
description	TEXT	NULLABLE	Descrição completa
requirements	TEXT	NULLABLE	Requisitos específicos
version	VARCHAR(50)	NULLABLE	Versão do standard
effectiveDate	TIMESTAMP	NULLABLE	Data de vigência

Índices: - `idx_standards_type` ON type - `idx_standards_name` ON name

2.2.5 Tabela: `complaints`

Reclamações de consumidores.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único da reclamação
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
source	VARCHAR(100)	NOT NULL	Origem (reclame_aqui, sac, etc.)
category	VARCHAR(100)	NULLABLE	Categoria da reclamação
description	TEXT	NOT NULL	Descrição completa
severity	ENUM('low', 'medium', 'high', 'critical')	NOT NULL	Severidade
status	ENUM('open', 'investigating', 'resolved', 'closed')	NOT NULL	Status
createdAt	TIMESTAMP	DEFAULT NOW()	Data de criação
resolvedAt	TIMESTAMP	NULLABLE	Data de resolução

Índices: - `idx_complaints_projectId` ON projectId - `idx_complaints_severity` ON severity - `idx_complaints_status` ON status - `idx_complaints_createdAt` ON createdAt DESC

2.2.6 Tabela: `predictions`

Predições de ML sobre projetos.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único da predição
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto analisado
model	VARCHAR(100)	NOT NULL	Modelo ML utilizado
prediction	TEXT	NOT NULL	Resultado da predição
confidence	DECIMAL(5,2)	NOT NULL	Confiança (0-100%)
factors	TEXT	NULLABLE	Fatores considerados (JSON)
createdAt	TIMESTAMP	DEFAULT NOW()	Data da predição

Índices: - `idx_predictions_projectId` ON projectId - `idx_predictions_model` ON model - `idx_predictions_createdAt` ON createdAt DESC

2.2.7 Tabela: `alerts`

Alertas gerados pelo sistema.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do alerta
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
type	VARCHAR(100)	NOT NULL	Tipo de alerta
severity	ENUM('info', 'warning', 'error', 'critical')	NOT NULL	Severidade
message	TEXT	NOT NULL	Mensagem do alerta
acknowledged	BOOLEAN	DEFAULT FALSE	Reconhecido?
acknowledgedAt	TIMESTAMP	NULLABLE	Data de reconhecimento
acknowledgedBy	VARCHAR(64)	NULLABLE	Usuário que reconheceu
createdAt	TIMESTAMP	DEFAULT NOW()	Data de criação

Índices: - `idx_alerts_projectId` ON projectId - `idx_alerts_severity` ON severity - `idx_alerts_acknowledged` ON acknowledged - `idx_alerts_createdAt` ON createdAt DESC

2.2.8 Tabela: `reports`

Relatórios gerados.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do relatório
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
type	VARCHAR(100)	NOT NULL	Tipo de relatório
title	VARCHAR(255)	NOT NULL	Título
content	TEXT	NOT NULL	Conteúdo (JSON ou Markdown)
generatedBy	VARCHAR(64)	FOREIGN KEY → users(id)	Gerador
createdAt	TIMESTAMP	DEFAULT NOW()	Data de geração

Índices: - `idx_reports_projectId` ON projectId - `idx_reports_type` ON type - `idx_reports_createdAt` ON createdAt DESC

2.2.9 Tabela: `socialMediaPosts`

Posts coletados de redes sociais.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do post
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
platform	ENUM('instagram', 'facebook', 'tiktok', 'twitter', 'reclame_aqui', 'nestle_site')	NOT NULL	Plataforma
postId	VARCHAR(255)	NOT NULL	ID externo do post
author	VARCHAR(255)	NULLABLE	Autor do post
content	TEXT	NOT NULL	Conteúdo completo
url	TEXT	NULLABLE	URL do post
likes	INT	DEFAULT 0	Número de likes
comments	INT	DEFAULT 0	Número de comentários
shares	INT	DEFAULT 0	Número de compartilhamentos
postedAt	TIMESTAMP	NOT NULL	Data de publicação
collectedAt	TIMESTAMP	DEFAULT NOW()	Data de coleta

Índices: - `idx_socialmedia_projectId` ON projectId - `idx_socialmedia_platform` ON platform - `idx_socialmedia_postedAt` ON postedAt DESC - `idx_socialmedia_postId_platform` ON (postId, platform) UNIQUE

2.2.10 Tabela: `sentimentAnalysis`

Análises de sentimento de posts.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único da análise
postId	VARCHAR(64)	FOREIGN KEY → socialMediaPosts(id)	Post analisado
sentiment	ENUM('positive', 'neutral', 'negative', 'very_negative')	NOT NULL	Sentimento detectado
score	DECIMAL(5,2)	NOT NULL	Score de sentimento (-1 a 1)
keywords	TEXT	NULLABLE	Palavras-chave (JSON array)
topics	TEXT	NULLABLE	Tópicos identificados (JSON array)
emotions	TEXT	NULLABLE	Emoções detectadas (JSON array)
analyzedAt	TIMESTAMP	DEFAULT NOW()	Data da análise

Índices: - `idx_sentiment_postId` ON postId - `idx_sentiment_sentiment` ON sentiment - `idx_sentiment_score` ON score - `idx_sentiment_analyzedAt` ON analyzedAt DESC

2.2.11 Tabela: `socialMediaAccounts`

Contas de redes sociais monitoradas.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único da conta
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
platform	ENUM('instagram', 'facebook', 'tiktok', 'twitter', 'reclame_aqui', 'nestle_site')	NOT NULL	Plataforma
accountId	VARCHAR(255)	NOT NULL	ID externo da conta
accountName	VARCHAR(255)	NOT NULL	Nome da conta
isActive	BOOLEAN	DEFAULT TRUE	Monitoramento ativo?
lastChecked	TIMESTAMP	NULLABLE	Última verificação
createdAt	TIMESTAMP	DEFAULT NOW()	Data de adição

Índices: - `idx_accounts_projectId` ON projectId - `idx_accounts_platform` ON platform - `idx_accounts_isActive` ON isActive

---

2.2.12 Tabela: `sentimentSummary`

Resumos agregados de sentimento.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do resumo
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
platform	ENUM('instagram', 'facebook', 'tiktok', 'twitter', 'reclame_aqui', 'nestle_site', 'all')	NOT NULL	Plataforma ou 'all'
period	VARCHAR(50)	NOT NULL	Período (7d, 30d, 90d)
totalPosts	INT	NOT NULL	Total de posts
positiveCount	INT	NOT NULL	Posts positivos
neutralCount	INT	NOT NULL	Posts neutros
negativeCount	INT	NOT NULL	Posts negativos
veryNegativeCount	INT	NOT NULL	Posts muito negativos
averageScore	DECIMAL(5,2)	NOT NULL	Score médio
topKeywords	TEXT	NULLABLE	Top keywords (JSON array)
topTopics	TEXT	NULLABLE	Top topics (JSON array)
generatedAt	TIMESTAMP	DEFAULT NOW()	Data de geração

Índices: - `idx_summary_projectId` ON projectId - `idx_summary_platform` ON platform - `idx_summary_period` ON period - `idx_summary_generatedAt` ON generatedAt DESC

2.2.13 Tabela: `monitoredKeywords`

Palavras-chave monitoradas.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único da keyword
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
keyword	VARCHAR(255)	NOT NULL	Palavra-chave
category	VARCHAR(100)	NULLABLE	Categoria
platform	ENUM('instagram', 'facebook', 'tiktok', 'twitter', 'reclame_aqui', 'nestle_site', 'all')	NOT NULL	Plataforma
priority	ENUM('low', 'medium', 'high')	DEFAULT 'medium'	Prioridade
isActive	BOOLEAN	DEFAULT TRUE	Ativo?
createdAt	TIMESTAMP	DEFAULT NOW()	Data de criação

Índices: - `idx_keywords_projectId` ON projectId - `idx_keywords_keyword` ON keyword - `idx_keywords_isActive` ON isActive

---

2.2.14 Tabela: `monitoredTopics`

Tópicos monitorados.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do tópico
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
topic	VARCHAR(255)	NOT NULL	Tópico
description	TEXT	NULLABLE	Descrição
platform	ENUM('instagram', 'facebook', 'tiktok', 'twitter', 'reclame_aqui', 'nestle_site', 'all')	NOT NULL	Plataforma
priority	ENUM('low', 'medium', 'high')	DEFAULT 'medium'	Prioridade
isActive	BOOLEAN	DEFAULT TRUE	Ativo?
createdAt	TIMESTAMP	DEFAULT NOW()	Data de criação

Índices: - `idx_topics_projectId` ON projectId - `idx_topics_topic` ON topic - `idx_topics_isActive` ON isActive

---

2.2.15 Tabela: `sentimentAlerts`

Alertas de sentimento negativo.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do alerta
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
platform	ENUM('instagram', 'facebook', 'tiktok', 'twitter', 'reclame_aqui', 'nestle_site', 'all')	NOT NULL	Plataforma
alertType	ENUM('negative_spike', 'very_negative_spike', 'negative_threshold', 'sentiment_drop')	NOT NULL	Tipo de alerta
severity	ENUM('low', 'medium', 'high', 'critical')	NOT NULL	Severidade
message	TEXT	NOT NULL	Mensagem
sentimentData	TEXT	NOT NULL	Dados de sentimento (JSON)
status	ENUM('active', 'acknowledged', 'resolved')	DEFAULT 'active'	Status
acknowledgedAt	TIMESTAMP	NULLABLE	Data de reconhecimento
resolvedAt	TIMESTAMP	NULLABLE	Data de resolução
createdAt	TIMESTAMP	DEFAULT NOW()	Data de criação

Índices: - `idx_sentiment_alerts_projectId` ON projectId - `idx_sentiment_alerts_severity` ON severity - `idx_sentiment_alerts_status` ON status - `idx_sentiment_alerts_createdAt` ON createdAt DESC

2.2.16 Tabela: `alertConfigurations`

Configurações de alertas.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único da configuração
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
negativeThreshold	DECIMAL(5,2)	DEFAULT 30.00	Threshold de negativos (%)
veryNegativeThreshold	DECIMAL(5,2)	DEFAULT 10.00	Threshold de muito negativos (%)
sentimentDropThreshold	DECIMAL(5,2)	DEFAULT 0.15	Threshold de queda de sentimento
checkIntervalMinutes	INT	DEFAULT 60	Intervalo de verificação (min)
isActive	BOOLEAN	DEFAULT TRUE	Ativo?
createdAt	TIMESTAMP	DEFAULT NOW()	Data de criação
updatedAt	TIMESTAMP	DEFAULT NOW()	Última atualização

Índices: - `idx_alert_config_projectId` ON projectId UNIQUE - `idx_alert_config_isActive` ON isActive

2.2.17 Tabela: `availableTests`

Catálogo de testes disponíveis.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único do teste
name	VARCHAR(255)	NOT NULL	Nome do teste
description	TEXT	NULLABLE	Descrição completa
category	ENUM('physical', 'chemical', 'microbiological', 'sensory', 'nutritional')	NOT NULL	Categoria
unit	VARCHAR(50)	NULLABLE	Unidade de medida
minValue	VARCHAR(50)	NULLABLE	Valor mínimo aceitável
maxValue	VARCHAR(50)	NULLABLE	Valor máximo aceitável
targetValue	VARCHAR(50)	NULLABLE	Valor alvo
methodology	TEXT	NULLABLE	Metodologia de execução
duration	INT	NULLABLE	Duração estimada (horas)
cost	DECIMAL(10,2)	NULLABLE	Custo estimado (R$)
isActive	BOOLEAN	DEFAULT TRUE	Ativo?
createdAt	TIMESTAMP	DEFAULT NOW()	Data de criação

Índices: - `idx_available_tests_category` ON category - `idx_available_tests_isActive` ON isActive - `idx_available_tests_name` ON name

---

2.2.18 Tabela: `projectTests`

Testes associados a projetos.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único da associação
projectId	VARCHAR(64)	FOREIGN KEY → projects(id)	Projeto relacionado
testId	VARCHAR(64)	FOREIGN KEY → availableTests(id)	Teste associado
status	ENUM('pending', 'in_progress', 'completed', 'failed')	DEFAULT 'pending'	Status
startedAt	TIMESTAMP	NULLABLE	Data de início
completedAt	TIMESTAMP	NULLABLE	Data de conclusão
createdAt	TIMESTAMP	DEFAULT NOW()	Data de associação

Índices: - `idx_project_tests_projectId` ON projectId - `idx_project_tests_testId` ON testId - `idx_project_tests_status` ON status - `idx_project_tests_projectId_testId` ON (projectId, testId) UNIQUE

2.2.19 Tabela: `testMetrics`

Métricas coletadas de testes.

Coluna	Tipo	Restrições	Descrição
id	VARCHAR(64)	PRIMARY KEY	ID único da métrica
projectTestId	VARCHAR(64)	FOREIGN KEY → projectTests(id)	Teste do projeto
measuredValue	VARCHAR(100)	NOT NULL	Valor medido
isWithinSpec	BOOLEAN	NOT NULL	Dentro da especificação?
deviation	VARCHAR(100)	NULLABLE	Desvio do alvo
notes	TEXT	NULLABLE	Observações
measuredAt	TIMESTAMP	DEFAULT NOW()	Data da medição

Índices: - `idx_test_metrics_projectTestId` ON projectTestId - `idx_test_metrics_isWithinSpec` ON isWithinSpec - `idx_test_metrics_measuredAt` ON measuredAt DESC

2.3 Relacionamentos e Integridade Referencial

Relacionamentos 1:N (One-to-Many)

- `users` → `projects` (um usuário cria vários projetos)
- `projects` → `manufacturingData` (um projeto tem vários dados de manufatura)
- `projects` → `complaints` (um projeto pode ter várias reclamações)
- `projects` → `predictions` (um projeto pode ter várias predições)
- `projects` → `alerts` (um projeto pode ter vários alertas)
- `projects` → `reports` (um projeto pode ter vários relatórios)
- `projects` → `socialMediaPosts` (um projeto é mencionado em vários posts)
- `projects` → `socialMediaAccounts` (um projeto monitora várias contas)
- `projects` → `sentimentSummary` (um projeto tem vários resumos)
- `projects` → `monitoredKeywords` (um projeto monitora várias keywords)

- `projects` → `monitoredTopics` (um projeto monitora vários tópicos)
- `projects` → `sentimentAlerts` (um projeto pode ter vários alertas de sentimento)
- `projects` → `alertConfigurations` (um projeto tem uma configuração de alertas)
- `socialMediaPosts` → `sentimentAnalysis` (um post tem uma análise)
- `availableTests` → `projectTests` (um teste pode ser usado em vários projetos)
- `projectTests` → `testMetrics` (um teste de projeto tem várias métricas)

Relacionamentos N:M (Many-to-Many)

- `projects` ↔ `availableTests` (via `projectTests`)
- Um projeto pode ter vários testes
- Um teste pode ser usado em vários projetos

Restrições de Integridade

- **ON DELETE CASCADE**: Quando um projeto é excluído, todos os registros relacionados são removidos automaticamente
- **ON UPDATE CASCADE**: Quando um ID é atualizado, todas as referências são atualizadas
- **UNIQUE CONSTRAINTS**: Evitam duplicação de dados (ex: postId + platform em socialMediaPosts)
- **CHECK CONSTRAINTS**: Validam ranges de valores (ex: score entre -1 e 1, riskScore entre 0 e 100)

---

3. Funcionalidades Detalhadas

3.1 Gerenciamento de Projetos

3.1.1 CRUD de Projetos

Endpoints tRPC: - `projects.list`: Lista todos os projetos do usuário - `projects.getById`: Retorna detalhes de um projeto específico - `projects.create`: Cria novo projeto - `projects.update`: Atualiza projeto existente - `projects.delete`: Exclui projeto (soft delete ou hard delete)

Validações: - Nome: obrigatório, 3-255 caracteres - Status: deve ser um dos valores do enum - RiskScore: 0-100 - SuccessProbability: 0-100%

Regras de Negócio: - Apenas o criador ou admin pode editar/excluir - Ao excluir, todos os dados relacionados são removidos (CASCADE) - Score de risco é recalculado automaticamente quando há novos dados de manufatura ou reclamações

3.1.2 Dados de Manufatura

Coleta de Métricas: - Temperatura, pressão, pH, viscosidade, densidade - Integração com sensores IoT (via MQTT ou HTTP) - Validação de ranges aceitáveis - Alertas automáticos quando fora da especificação

Visualizações: - Gráficos de linha temporal - Histogramas de distribuição - Box plots para análise de outliers

3.2 Análise de Sentimento em Redes Sociais

3.2.1 Coleta de Dados

Plataformas Suportadas: 1. **Instagram**: Instagram Graph API 2. **Facebook**: Facebook Graph API 3. **TikTok**: TikTok API for Business 4. **X (Twitter)**: Twitter API v2 5. **Reclame Aqui**: Web scraping (Puppeteer) 6. **Site Nestlé**: Web scraping (Puppeteer)

Processo de Coleta:

```
// Pseudocódigo
async function collectSocialMediaPosts(projectId: string) {
  const accounts = await getMonitoredAccounts(projectId);

  for (const account of accounts) {
    const posts = await fetchPostsFromPlatform(account.platform,
account.accountId);

    for (const post of posts) {
      // Verifica se já foi coletado
      const exists = await checkPostExists(post.postId, post.platform);
      if (exists) continue;

      // Salva no banco
      await savePost({
        projectId,
        platform: account.platform,
        postId: post.postId,
        author: post.author,
        content: post.content,
        url: post.url,
        likes: post.likes,
        comments: post.comments,
        shares: post.shares,
        postedAt: post.postedAt,
      });
    }
  }
}
```

3.2.2 Análise de Sentimento com LLM

Modelo: GPT-4o (via OpenAI API ou Manus Built-in Forge API)

Prompt Engineering:

```
 const prompt = `
Analise o sentimento do seguinte post sobre o produto "${productName}":

"${postContent}"

Retorne um JSON com:
{
  "sentiment": "positive" | "neutral" | "negative" | "very_negative",
  "score": número entre -1 (muito negativo) e 1 (muito positivo),
  "keywords": array de palavras-chave relevantes,
  "topics": array de tópicos identificados,
  "emotions": array de emoções detectadas (alegria, raiva, tristeza, etc.)
}
`;

const response = await invokeLLM({
  messages: [
    { role: "system", content: "Você é um especialista em análise de sentimento
para produtos alimentícios." },
    { role: "user", content: prompt }
  ],
  response_format: {
    type: "json_schema",
    json_schema: {
      name: "sentiment_analysis",
      strict: true,
      schema: {
        type: "object",
        properties: {
          sentiment: { type: "string", enum: ["positive", "neutral",
"negative", "very_negative"] },
          score: { type: "number", minimum: -1, maximum: 1 },
          keywords: { type: "array", items: { type: "string" } },
          topics: { type: "array", items: { type: "string" } },
          emotions: { type: "array", items: { type: "string" } }
        },
        required: ["sentiment", "score", "keywords", "topics", "emotions"],
        additionalProperties: false
      }
    }
  }
});
```

Métricas Calculadas: - **Distribuição de Sentimento**: % de posts positivos, neutros, negativos, muito negativos - **Score Médio**: Média ponderada dos scores (-1 a 1) - **Top Keywords**: Palavras-chave mais mencionadas (frequência) - **Top Topics**: Tópicos mais discutidos - **Tendência Temporal**: Evolução do sentimento ao longo do tempo

3.2.3 Alertas de Sentimento Negativo

Tipos de Alertas: 1. **Negative Spike**: Aumento súbito de posts negativos (>30% em 24h) 2. **Very Negative Spike**: Aumento súbito de posts muito negativos (>10% em 24h) 3. **Negative Threshold**: Percentual de negativos acima do threshold configurado 4. **Sentiment Drop**: Queda brusca no score médio (>0.15 pontos em 24h)

Severidade: - **Low**: Aumento leve, não urgente - **Medium**: Aumento moderado, monitorar - **High**: Aumento significativo, ação recomendada - **Critical**: Crise de imagem, ação imediata

Notificações: - Email para stakeholders - Push notification no dashboard - Integração com `notifyOwner()` (Manus built-in) - Webhook para sistemas externos (Slack, Teams, etc.)

3.3 Simulação Monte Carlo

3.3.1 Modelos Matemáticos Implementados

Modelo 1: Solubilidade (Equação de Van't Hoff)

Descrição: Prevê a solubilidade de um soluto em função da temperatura.

Equação:

```
ln(S) = -ΔH/R * (1/T) + ΔS/R
```

Onde: - S = solubilidade (g/100mL) - ΔH = entalpia de dissolução (J/mol) - ΔS = entropia de dissolução (J/mol • K) - R = constante dos gases (8.314 J/mol • K) - T = temperatura (K)

Implementação:

```
function vanHoffSolubility(
  temperature: number, // °C
  deltaH: number, // J/mol
  deltaS: number // J/mol·K
): number {
  const R = 8.314;
  const T = temperature + 273.15; // Converte para Kelvin
  const lnS = (-deltaH / R) * (1 / T) + (deltaS / R);
  return Math.exp(lnS); // Solubilidade em g/100mL
}
```

Parâmetros Monte Carlo: - `temperature`: Normal(25, 2) °C - `deltaH`: Normal(-20000, 1000) J/mol - `deltaS`: Normal(50, 5) J/mol • K

Modelo 2: Dissociação Iônica

Descrição: Calcula o grau de dissociação de um eletrólito fraco.

Equação:

```
α = √(Ka / C)
```

Onde: - α = grau de dissociação (0-1) - Ka = constante de dissociação ácida - C = concentração inicial (mol/L)

Implementação:

```tsx
function ionicDissociation(
  Ka: number, // Constante de dissociação
  concentration: number // mol/L
): number {
  return Math.sqrt(Ka / concentration);
}
```

Parâmetros Monte Carlo: - `Ka`: LogNormal(1e-5, 1e-6) - `concentration`: Normal(0.1, 0.01) mol/L

Modelo 3: Estabilidade Textural (Modelo de Maxwell)

Descrição: Prevê a viscosidade de uma emulsão ao longo do tempo.

Equação:

```
η(t) = η₀ * e^(-t/τ)
```

Onde: - η(t) = viscosidade no tempo t (Pa • s) - η₀ = viscosidade inicial (Pa • s) - τ = tempo de relaxação (dias) - t = tempo (dias)

Implementação:

```tsx
function maxwellViscosity(
  time: number, // dias
  initialViscosity: number, // Pa·s
  relaxationTime: number // dias
): number {
  return initialViscosity * Math.exp(-time / relaxationTime);
}
```

Parâmetros Monte Carlo: - `initialViscosity`: Normal(5, 0.5) Pa • s - `relaxationTime`: Normal(30, 3) dias

---

Modelo 4: Shelf-Life (Equação de Arrhenius)

Descrição: Estima o tempo de vida útil em função da temperatura.

Equação:

```
 k = A * e^(-Ea / (R * T))
shelf_life = 1 / k
```

Onde: - k = constante de velocidade de degradação (1/dia) - A = fator pré-exponencial - Ea = energia de ativação (J/mol) - R = constante dos gases (8.314 J/mol • K) - T = temperatura (K)

Implementação:

```
 function arrheniusShelfLife(
   temperature: number, // °C
   activationEnergy: number, // J/mol
   preExponentialFactor: number
): number {
   const R = 8.314;
   const T = temperature + 273.15;
   const k = preExponentialFactor * Math.exp(-activationEnergy / (R * T));
   return 1 / k; // dias
}
```

Parâmetros Monte Carlo: - `temperature`: Normal(25, 2) °C - `activationEnergy`: Normal(80000, 5000) J/mol - `preExponentialFactor`: LogNormal(1e10, 1e9)

Modelo 5: Crescimento Microbiano (Modelo de Gompertz)

Descrição: Prevê o crescimento de microrganismos ao longo do tempo.

Equação:

```
N(t) = A * exp(-exp((μ * e / A) * (λ - t) + 1))
```

Onde: - N(t) = população no tempo t (UFC/g) - A = assíntota (população máxima) - μ = taxa de crescimento máxima (1/h) - λ = fase lag (h) - e = constante de Euler (2.71828)

Implementação:

```
function gompertzGrowth(
  time: number, // horas
  asymptote: number, // UFC/g
  maxGrowthRate: number, // 1/h
  lagPhase: number // horas
): number {
  const e = Math.E;
  return asymptote * Math.exp(-Math.exp((maxGrowthRate * e / asymptote) *
(lagPhase - time) + 1));
}
```

Parâmetros Monte Carlo: - `asymptote`: Normal(1e6, 1e5) UFC/g - `maxGrowthRate`: Normal(0.5, 0.05) 1/h - `lagPhase`: Normal(10, 1) horas

---

3.3.2 Algoritmo de Monte Carlo

Pseudocódigo:

```
function monteCarloSimulation(
  model: string, // 'solubility', 'dissociation', etc.
  parameters: Record<string, Distribution>,
  iterations: number = 1000
): MonteCarloResult {
  const results: number[] = [];

  for (let i = 0; i < iterations; i++) {
    // Amostra valores dos parâmetros de suas distribuições
    const sampledParams: Record<string, number> = {};
    for (const [param, distribution] of Object.entries(parameters)) {
      sampledParams[param] = sampleFromDistribution(distribution);
    }

    // Executa o modelo com os parâmetros amostrados
    const result = executeModel(model, sampledParams);
    results.push(result);
  }

  // Calcula estatísticas
  const mean = calculateMean(results);
  const stdDev = calculateStdDev(results);
  const min = Math.min(...results);
  const max = Math.max(...results);
  const percentile5 = calculatePercentile(results, 5);
  const percentile95 = calculatePercentile(results, 95);

  // Gera distribuição de frequência
  const distribution = generateHistogram(results, 20); // 20 bins

  return {
    mean,
    stdDev,
    min,
    max,
    percentile5,
    percentile95,
    distribution,
    rawResults: results
  };
}
```

Distribuições Suportadas: - **Normal**: `N(μ, σ)` usando Box-Muller transform - **LogNormal**: `exp(N(μ, σ))` - **Uniforme**: `U(a, b)` - **Triangular**: `Tri(a, m, b)` (mínimo, moda, máximo)

Amostragem:

```
function sampleNormal(mean: number, stdDev: number): number {
  // Box-Muller transform
  const u1 = Math.random();
  const u2 = Math.random();
  const z0 = Math.sqrt(-2 * Math.log(u1)) * Math.cos(2 * Math.PI * u2);
  return mean + stdDev * z0;
}

function sampleLogNormal(mean: number, stdDev: number): number {
  return Math.exp(sampleNormal(mean, stdDev));
}

function sampleUniform(min: number, max: number): number {
  return min + Math.random() * (max - min);
}

function sampleTriangular(min: number, mode: number, max: number): number {
  const u = Math.random();
  const F = (mode - min) / (max - min);

  if (u < F) {
    return min + Math.sqrt(u * (max - min) * (mode - min));
  } else {
    return max - Math.sqrt((1 - u) * (max - min) * (max - mode));
  }
}
```

Visualização de Resultados: - **Histograma**: Distribuição de frequência dos resultados - **Gráfico de Área**: Intervalo de confiança 95% (P5-P95) - **Box Plot**: Mediana, quartis, outliers - **Gráfico de Dispersão**: Comparação entre múltiplos testes

3.4 Gerenciamento de Testes

3.4.1 Catálogo de Testes

Testes Pré-Cadastrados (10 testes):

ID	Nome	Categoria	Unidade	Duração	Custo
1	Viscosidade Brookfield	Física	cP	2h	R$ 150
2	pH	Química	pH	0.5h	R$ 50
3	Densidade	Física	g/cm³	1h	R$ 80
4	Solubilidade	Química	g/100mL	4h	R$ 200
5	Estabilidade de Emulsão	Física	% separação	24h	R$ 300
6	Análise Microbiológica	Microbiológica	UFC/g	72h	R$ 500
7	Análise Sensorial	Sensorial	score 1-9	3h	R$ 400
8	Textura (TPA)	Física	N	2h	R$ 250
9	Shelf-Life Acelerado	Química	dias	168h	R$ 1200
10	Análise Nutricional	Nutricional	g/100g	48h	R$ 800

3.4.2 Associação de Testes a Projetos

Workflow: 1. Usuário acessa modal "Gerenciar Testes" no projeto 2. Visualiza testes já associados (aba "Testes do Projeto") 3. Pode adicionar novos testes do catálogo (aba "Catálogo") 4. Pode remover testes associados (com confirmação) 5. Sistema valida se teste já está associado (evita duplicação)

Endpoints tRPC: - `tests.getAvailableTests`: Lista todos os testes do catálogo - `tests.getProjectTests`: Lista testes associados ao projeto - `tests.addTestToProject`: Associa teste ao projeto - `tests.removeTestFromProject`: Remove associação - `tests.createTest`: Cria novo teste no catálogo (admin only) - `tests.updateTest`: Atualiza teste existente (admin only)

3.4.3 Coleta de Métricas

Processo: 1. Teste é executado no laboratório 2. Técnico registra valor medido no sistema 3. Sistema valida se valor está dentro da especificação (min-max) 4. Calcula desvio do alvo (target value) 5. Gera alerta se fora da especificação 6. Armazena métrica em `testMetrics`

Validações: - Valor medido deve ser numérico (ou string para testes qualitativos) - Data de medição não pode ser futura - Teste deve estar associado ao projeto - Status do teste deve ser "in_progress" ou "completed"

3.5 Comparação de Produtos

3.5.1 Seleção de Produtos

Interface: - Multi-select com até 6 produtos - Busca por nome - Filtro por status - Preview dos produtos selecionados

3.5.2 Métricas Comparativas

Tabela de Comparação:

Métrica	Produto A	Produto B	Produto C
Score Médio	0.65	0.42	0.78
% Positivos	68%	52%	81%
% Negativos	12%	28%	8%
Total Posts	1.234	892	2.105
Risco	35	52	28
Sucesso	78%	65%	85%

Gráficos: 1. **Linha Temporal**: Evolução do sentimento ao longo do tempo 2. **Barras Empilhadas**: Distribuição de sentimento por produto 3. **Radar Multidimensional**: Comparação em 6 dimensões (score, positivos, negativos, engajamento, risco, sucesso) 4. **Nuvem de Palavras**: Top keywords por produto

3.5.3 Insights Automáticos

Algoritmo:

```
function generateComparisonInsights(products: Product[]): string[] {
  const insights: string[] = [];

  // Identifica melhor e pior
  const bestProduct = products.reduce((best, p) =>
    p.averageScore > best.averageScore ? p : best
  );
  const worstProduct = products.reduce((worst, p) =>
    p.averageScore < worst.averageScore ? p : worst
  );

  insights.push(`${bestProduct.name} tem o melhor desempenho geral com score
médio de ${bestProduct.averageScore.toFixed(2)}`);
  insights.push(`${worstProduct.name} precisa de atenção com score médio de
${worstProduct.averageScore.toFixed(2)}`);

  // Identifica tendências
  for (const product of products) {
    if (product.trend > 0.1) {
      insights.push(`${product.name} está em tendência positiva
(+${(product.trend * 100).toFixed(1)}%)`);
    } else if (product.trend < -0.1) {
      insights.push(`${product.name} está em tendência negativa
(${(product.trend * 100).toFixed(1)}%)`);
    }
  }

  // Identifica correlações
  const correlation = calculateCorrelation(products[0].scores,
products[1].scores);
  if (Math.abs(correlation) > 0.7) {
    insights.push(`${products[0].name} e ${products[1].name} têm
comportamento ${correlation > 0 ? 'similar' : 'oposto'} (correlação:
${correlation.toFixed(2)})`);
  }

  return insights;
}
```

3.6 Dashboard de Alertas

3.6.1 Métricas Agregadas

Cards Principais: - **Total de Alertas**: Contagem total no período - **Alertas Críticos**: Contagem de severidade "critical" - **Taxa de Resolução**: % de alertas resolvidos / total - **Tempo Médio de Resolução**: Média de (resolvedAt - createdAt) em horas

3.6.2 Gráficos

1. **Linha Temporal**: Alertas por dia, coloridos por severidade
2. **Pizza**: Distribuição por severidade
3. **Barras Horizontais**: Alertas por plataforma

4. **Barras Verticais**: Alertas por tipo

3.6.3 Insights Automáticos

Exemplos: - "Instagram gerou 45% dos alertas este mês" - "Alertas críticos aumentaram 30% vs. mês anterior" - "Tempo médio de resolução melhorou de 8h para 5h" - "Reclame Aqui tem a maior taxa de alertas críticos (65%)"

---

4. APIs e Integrações

4.1 API tRPC

4.1.1 Estrutura de Routers

Router Principal (`appRouter`):

```
export const appRouter = router({
  auth: authRouter,
  projects: projectsRouter,
  manufacturing: manufacturingRouter,
  standards: standardsRouter,
  complaints: complaintsRouter,
  predictions: predictionsRouter,
  alerts: alertsRouter,
  reports: reportsRouter,
  sentiment: sentimentRouter,
  admin: adminRouter,
  sentimentAlerts: sentimentAlertsRouter,
  tests: testsRouter,
  simulations: simulationsRouter,
  system: systemRouter,
});

export type AppRouter = typeof appRouter;
```

4.1.2 Autenticação e Autorização

Context Builder:

```
export async function createContext({ req, res }: CreateContextOptions) {
  const token = req.cookies[COOKIE_NAME];

  let user: User | null = null;
  if (token) {
    try {
      const payload = jwt.verify(token, ENV.jwtSecret) as JWTPayload;
      user = await getUser(payload.sub);
    } catch (error) {
      console.error('[Auth] Invalid token:', error);
    }
  }

  return {
    req,
    res,
    user,
  };
}

export type Context = Awaited<ReturnType<typeof createContext>>;
```

Procedures:

```
// Público (sem autenticação)
export const publicProcedure = t.procedure;

// Protegido (requer autenticação)
export const protectedProcedure = t.procedure.use(({ ctx, next }) => {
  if (!ctx.user) {
    throw new TRPCError({ code: 'UNAUTHORIZED' });
  }
  return next({
    ctx: {
      ...ctx,
      user: ctx.user, // Garante que user não é null
    },
  });
});

// Admin (requer role admin)
export const adminProcedure = protectedProcedure.use(({ ctx, next }) => {
  if (ctx.user.role !== 'admin') {
    throw new TRPCError({ code: 'FORBIDDEN' });
  }
  return next({ ctx });
});
```

4.1.3 Exemplos de Endpoints

Projeto - Listar:

```
  list: protectedProcedure.query(async ({ ctx }) => {
    const db = await getDb();
    if (!db) return [];

    return await db
      .select()
      .from(projects)
      .where(eq(projects.userId, ctx.user.id))
      .orderBy(desc(projects.createdAt));
  }),
```

Projeto - Criar:

```
  create: protectedProcedure
    .input(z.object({
      name: z.string().min(3).max(255),
      description: z.string().optional(),
      status: z.enum(['planning', 'in_progress', 'testing', 'completed',
'cancelled']),
      factory: z.string().optional(),
      productType: z.string().optional(),
    }))
    .mutation(async ({ ctx, input }) => {
      const db = await getDb();
      if (!db) throw new TRPCError({ code: 'INTERNAL_SERVER_ERROR', message:
'Database unavailable' });

      const projectId = nanoid();
      await db.insert(projects).values({
        id: projectId,
        name: input.name,
        description: input.description,
        status: input.status,
        factory: input.factory,
        productType: input.productType,
        userId: ctx.user.id,
      });

      return { id: projectId };
    }),
```

Sentimento - Coletar e Analisar:

```
collectAndAnalyzeAll: protectedProcedure
  .input(z.object({
    projectId: z.string(),
  }))
  .mutation(async ({ input }) => {
    try {
      // 1. Coleta posts de todas as plataformas
      const platforms = ['instagram', 'facebook', 'tiktok', 'twitter',
'reclame_aqui', 'nestle_site'];
      let totalCollected = 0;

      for (const platform of platforms) {
        const posts = await collectPostsFromPlatform(input.projectId,
platform);
        totalCollected += posts.length;
      }

      // 2. Analisa sentimento de posts não analisados
      const unanalyzedPosts = await getUnanalyzedPosts(input.projectId);
      let totalAnalyzed = 0;

      for (const post of unanalyzedPosts) {
        await analyzeSentiment(post.id, post.content);
        totalAnalyzed++;
      }

      // 3. Gera resumos agregados
      await generateSentimentSummaries(input.projectId);

      // 4. Verifica alertas
      await checkSentimentAlerts(input.projectId);

      return {
        success: true,
        collected: totalCollected,
        analyzed: totalAnalyzed,
      };
    } catch (error) {
      console.error('[Sentiment] Collection error:', error);
      throw new TRPCError({
        code: 'INTERNAL_SERVER_ERROR',
        message: 'Failed to collect and analyze sentiment',
      });
    }
  }),
```

Simulação - Monte Carlo:

```
runMonteCarlo: protectedProcedure
  .input(z.object({
    projectId: z.string(),
    modelType: z.enum(['solubility', 'dissociation', 'textural_stability',
'shelf_life', 'microbial_growth']),
    parameters: z.record(z.any()),
    iterations: z.number().min(100).max(10000).default(1000),
  }))
  .mutation(async ({ input }) => {
    const result = await runMonteCarloSimulation(
      input.modelType,
      input.parameters,
      input.iterations
    );

    return result;
  }),
```

4.2 Integrações Externas

4.2.1 Manus Built-in Forge API

LLM (GPT-4o):

```
import { invokeLLM } from './server/_core/llm';

const response = await invokeLLM({
  messages: [
    { role: "system", content: "Você é um especialista em análise de
sentimento." },
    { role: "user", content: prompt }
  ],
  response_format: {
    type: "json_schema",
    json_schema: { /* schema */ }
  }
});
```

Storage (S3):

```
import { storagePut, storageGet } from './server/storage';

// Upload
const { key, url } = await storagePut(
  `reports/${projectId}/${Date.now()}.pdf`,
  pdfBuffer,
  'application/pdf'
);

// Download (presigned URL)
const { url: downloadUrl } = await storageGet(key, 3600); // 1 hora
```

Notificações:

```
import { notifyOwner } from './server/_core/notification';

await notifyOwner({
  title: 'Alerta Crítico de Sentimento',
  content: `Projeto "${projectName}" detectou pico de sentimento negativo no
Instagram (45% de posts negativos nas últimas 24h).`
});
```

4.2.2 Redes Sociais

Instagram Graph API:

```
async function fetchInstagramPosts(accountId: string, accessToken: string) {
  const response = await fetch(
    `https://graph.instagram.com/v18.0/${accountId}/media?
fields=id,caption,like_count,comments_count,timestamp&access_token=${accessToken
  );

  const data = await response.json();
  return data.data.map((post: any) => ({
    postId: post.id,
    content: post.caption,
    likes: post.like_count,
    comments: post.comments_count,
    postedAt: new Date(post.timestamp),
  }));
}
```

Facebook Graph API:

```
async function fetchFacebookPosts(pageId: string, accessToken: string) {
  const response = await fetch(
    `https://graph.facebook.com/v18.0/${pageId}/posts?
fields=id,message,created_time,likes.summary(true),comments.summary(true),shares&
  );

  const data = await response.json();
  return data.data.map((post: any) => ({
    postId: post.id,
    content: post.message,
    likes: post.likes?.summary?.total_count || 0,
    comments: post.comments?.summary?.total_count || 0,
    shares: post.shares?.count || 0,
    postedAt: new Date(post.created_time),
  }));
}
```

X (Twitter) API v2:

```
async function fetchTwitterPosts(username: string, bearerToken: string) {
  const response = await fetch(
    `https://api.twitter.com/2/users/by/username/${username}`,
    {
      headers: {
        'Authorization': `Bearer ${bearerToken}`
      }
    }
  );

  const userData = await response.json();
  const userId = userData.data.id;

  const tweetsResponse = await fetch(
    `https://api.twitter.com/2/users/${userId}/tweets?
tweet.fields=created_at,public_metrics`,
    {
      headers: {
        'Authorization': `Bearer ${bearerToken}`
      }
    }
  );

  const tweetsData = await tweetsResponse.json();
  return tweetsData.data.map((tweet: any) => ({
    postId: tweet.id,
    content: tweet.text,
    likes: tweet.public_metrics.like_count,
    comments: tweet.public_metrics.reply_count,
    shares: tweet.public_metrics.retweet_count,
    postedAt: new Date(tweet.created_at),
  }));
}
```

Web Scraping (Reclame Aqui):

```
import puppeteer from 'puppeteer';

async function scrapeReclameAqui(companyName: string) {
  const browser = await puppeteer.launch({ headless: true });
  const page = await browser.newPage();

  await page.goto(`https://www.reclameaqui.com.br/empresa/${companyName}/`);
  await page.waitForSelector('.complaint-item');

  const complaints = await page.evaluate(() => {
    const items = Array.from(document.querySelectorAll('.complaint-item'));
    return items.map(item => ({
      postId: item.getAttribute('data-id'),
      content: item.querySelector('.complaint-text')?.textContent?.trim(),
      author: item.querySelector('.complaint-author')?.textContent?.trim(),
      postedAt: item.querySelector('.complaint-date')?.textContent?.trim(),
    }));
  });

  await browser.close();
  return complaints;
}
```

4.2.3 Integrações SAP e MFC (Manufacturing Execution System)

SAP ERP:

```
// Integração via SAP OData API
async function syncSAPData(projectId: string) {
  const sapClient = createSAPClient({
    baseUrl: process.env.SAP_BASE_URL,
    username: process.env.SAP_USERNAME,
    password: process.env.SAP_PASSWORD,
  });

  // Busca dados de produção
  const productionOrders = await
sapClient.get('/sap/opu/odata/sap/API_PRODUCTION_ORDER_2_SRV/A_ProductionOrder');

  // Sincroniza com PrediTest AI
  for (const order of productionOrders.d.results) {
    await createManufacturingData({
      projectId,
      metric: 'production_quantity',
      value: order.TotalQuantity,
      unit: order.ProductionUnit,
      batchNumber: order.Batch,
      timestamp: new Date(order.ActualStartDate),
    });
  }
}
```

MFC (Manufacturing Execution System):

```
// Integração via MQTT (IoT)
import mqtt from 'mqtt';

const mqttClient = mqtt.connect(process.env.MFC_MQTT_BROKER);

mqttClient.on('connect', () => {
  mqttClient.subscribe('factory/line1/sensors/#');
});

mqttClient.on('message', async (topic, message) => {
  const data = JSON.parse(message.toString());

  // Exemplo: factory/line1/sensors/temperature
  const metric = topic.split('/').pop();

  await createManufacturingData({
    projectId: data.projectId,
    metric: metric,
    value: data.value,
    unit: data.unit,
    timestamp: new Date(data.timestamp),
  });
});
```

---

5. Segurança e Conformidade

5.1 Autenticação e Autorização

5.1.1 OAuth 2.0

Fluxo: 1. Usuário clica em "Login" 2. Redirecionado para Manus OAuth Portal 3. Autentica com Google, Microsoft ou outro provedor 4. Manus OAuth retorna código de autorização 5. Backend troca código por access token 6. Backend cria sessão JWT e retorna cookie httpOnly

Implementação:

```
// Callback OAuth
app.get('/api/oauth/callback', async (req, res) => {
  const { code } = req.query;

  // Troca código por token
  const tokenResponse = await fetch(`${ENV.oauthServerUrl}/token`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      grant_type: 'authorization_code',
      code,
      client_id: ENV.appId,
      redirect_uri: `${ENV.frontendUrl}/api/oauth/callback`,
    }),
  });

  const { access_token } = await tokenResponse.json();

  // Busca informações do usuário
  const userResponse = await fetch(`${ENV.oauthServerUrl}/userinfo`, {
    headers: { 'Authorization': `Bearer ${access_token}` },
  });

  const userInfo = await userResponse.json();

  // Cria ou atualiza usuário no banco
  await upsertUser({
    id: userInfo.sub,
    name: userInfo.name,
    email: userInfo.email,
    loginMethod: userInfo.provider,
    lastSignedIn: new Date(),
  });

  // Cria sessão JWT
  const token = jwt.sign(
    { sub: userInfo.sub },
    ENV.jwtSecret,
    { expiresIn: '7d' }
  );

  // Retorna cookie
  res.cookie(COOKIE_NAME, token, {
    httpOnly: true,
    secure: process.env.NODE_ENV === 'production',
    sameSite: 'lax',
    maxAge: 7 * 24 * 60 * 60 * 1000, // 7 dias
  });

  res.redirect('/');
});
```

5.1.2 RBAC (Role-Based Access Control)

Roles: - **user**: Acesso padrão (CRUD de seus próprios projetos) - **admin**: Acesso total (CRUD de todos os projetos, gerenciamento de catálogo de testes, configurações globais)

Enforcement:

```typescript
// Middleware de autorização
function requireRole(role: 'user' | 'admin') {
  return (ctx: Context, next: () => Promise<void>) => {
    if (!ctx.user) {
      throw new TRPCError({ code: 'UNAUTHORIZED' });
    }

    if (role === 'admin' && ctx.user.role !== 'admin') {
      throw new TRPCError({ code: 'FORBIDDEN' });
    }

    return next();
  };
}

// Uso
const adminProcedure = protectedProcedure.use(requireRole('admin'));
```

5.2 Criptografia

5.2.1 Dados em Trânsito

- **HTTPS/TLS 1.3**: Todas as comunicações são criptografadas
- **Certificate Pinning**: Previne ataques man-in-the-middle
- **HSTS**: Força uso de HTTPS (header `Strict-Transport-Security`)

5.2.2 Dados em Repouso

- **Database Encryption**: PostgreSQL com TDE (Transparent Data Encryption)
- **S3 Server-Side Encryption**: AES-256
- **Secrets Management**: AWS Secrets Manager / HashiCorp Vault

5.3 Conformidade Regulatória

5.3.1 LGPD (Lei Geral de Proteção de Dados)

Princípios Implementados: - **Finalidade**: Dados coletados apenas para análise de testes industriais - **Adequação**: Tratamento compatível com as finalidades informadas - **Necessidade**: Coleta limitada ao mínimo necessário - **Transparência**: Política de privacidade clara - **Segurança**: Medidas técnicas e administrativas - **Prevenção**: Auditorias regulares - **Não Discriminação**: Sem uso discriminatório - **Responsabilização**: Demonstração de conformidade

Direitos dos Titulares: - **Acesso**: Endpoint `/api/data/export` (exporta todos os dados do usuário) - **Correção**: Endpoints de `update` em todos os recursos - **Exclusão**: Endpoint `/api/data/delete` (soft delete ou hard delete) - **Portabilidade**: Exportação em JSON estruturado - **Revogação de Consentimento**: Desativação de conta

5.3.2 ISO 27001 (Segurança da Informação)

Controles Implementados: - **A.9**: Controle de acesso (OAuth 2.0, RBAC) - **A.10**: Criptografia (TLS, AES-256) - **A.12**: Segurança de operações (logging, monitoring) - **A.14**: Aquisição, desenvolvimento e manutenção de sistemas (SDLC seguro) - **A.17**: Continuidade de negócios (backups, DR) - **A.18**: Conformidade (auditorias, políticas)

5.3.3 SOC 2 Type II

Critérios de Confiança: - **Security**: Controles de acesso, criptografia, firewall - **Availability**: SLA 99.9%, redundância, failover - **Processing Integrity**: Validação de dados, checksums - **Confidentiality**: Classificação de dados, DLP - **Privacy**: LGPD compliance, consent management

5.3.4 FDA 21 CFR Part 11

Requisitos para Indústria Alimentícia: - **Assinaturas Eletrônicas**: Implementadas via JWT + timestamp - **Audit Trail**: Logging de todas as operações críticas - **Validação de Sistemas**: Testes automatizados (>80% coverage) - **Controle de Acesso**: RBAC com MFA opcional - **Integridade de Dados**: Checksums, hashing

5.4 Auditoria e Logging

5.4.1 Audit Trail

Eventos Auditados: - Login/logout - CRUD de projetos - Modificação de testes - Geração de relatórios - Alteração de configurações - Acesso a dados sensíveis

Estrutura de Log:

```typescript
interface AuditLog {
  id: string;
  timestamp: Date;
  userId: string;
  action: string; // 'create', 'read', 'update', 'delete'
  resource: string; // 'project', 'test', 'report', etc.
  resourceId: string;
  changes: Record<string, any>; // Antes/depois
  ipAddress: string;
  userAgent: string;
}
```

Armazenamento: - **Elasticsearch**: Para busca e análise - **S3 Glacier**: Para arquivamento de longo prazo (7 anos)

5.4.2 Monitoring e Alertas

Métricas Monitoradas: - **Performance**: Response time, throughput, error rate - **Disponibilidade**: Uptime, downtime, latency - **Segurança**: Failed login attempts, unauthorized access, anomalies - **Negócio**: Projetos criados, testes executados, simulações rodadas

Ferramentas: - **Prometheus**: Coleta de métricas - **Grafana**: Visualização de dashboards - **Alertmanager**: Notificações (email, Slack, PagerDuty) - **ELK Stack**: Logging centralizado

---

6. Performance e Escalabilidade

6.1 Otimizações de Performance

6.1.1 Caching

Estratégias: - **Redis Cache**: Queries frequentes (TTL 5-60 min) - **Browser Cache**: Assets estáticos (1 ano) - **CDN Cache**: Imagens, CSS, JS (1 ano)

Exemplo:

```
async function getCachedProjectList(userId: string) {
  const cacheKey = `projects:${userId}`;

  // Tenta buscar do cache
  const cached = await redis.get(cacheKey);
  if (cached) {
    return JSON.parse(cached);
  }

  // Se não encontrou, busca do banco
  const projects = await db.select().from(projects).where(eq(projects.userId,
userId));

  // Armazena no cache (TTL 5 min)
  await redis.setex(cacheKey, 300, JSON.stringify(projects));

  return projects;
}
```

6.1.2 Query Optimization

Índices Criados: - Todos os foreign keys - Campos usados em WHERE, ORDER BY, GROUP BY - Campos usados em JOIN

Exemplo de Query Otimizada:

```
-- Antes (sem índice): 2.5s
SELECT * FROM socialMediaPosts WHERE projectId = '123' ORDER BY postedAt DESC;

-- Depois (com índice): 0.05s
CREATE INDEX idx_socialmedia_projectId_postedAt ON socialMediaPosts(projectId,
postedAt DESC);
```

6.1.3 Lazy Loading e Code Splitting

Frontend:

```
// Lazy loading de rotas
const Projects = lazy(() => import('./pages/Projects'));
const SocialSentiment = lazy(() => import('./pages/SocialSentiment'));
const TestManagement = lazy(() => import('./pages/TestManagement'));

// Suspense para loading state
<Suspense fallback={<LoadingSpinner />}>
  <Routes>
    <Route path="/projects" element={<Projects />} />
    <Route path="/sentiment" element={<SocialSentiment />} />
    <Route path="/tests" element={<TestManagement />} />
  </Routes>
</Suspense>
```

6.2 Escalabilidade Horizontal

6.2.1 Arquitetura de Microserviços

Serviços Independentes: - **API Gateway**: Roteamento, rate limiting, autenticação - **Projects Service**: CRUD de projetos - **Sentiment Service**: Coleta e análise de sentimento - **Simulation Service**: Monte Carlo e modelos matemáticos - **Notification Service**: Envio de alertas e emails

Comunicação: - **Síncrona**: tRPC (HTTP/JSON) - **Assíncrona**: RabbitMQ / AWS SQS (message queue)

6.2.2 Kubernetes Deployment

Configuração:

```yaml
 apiVersion: apps/v1
kind: Deployment
metadata:
  name: preditest-api
spec:
  replicas: 3 # Escalabilidade horizontal
  selector:
    matchLabels:
      app: preditest-api
  template:
    metadata:
      labels:
        app: preditest-api
    spec:
      containers:
      - name: api
        image: preditest/api:latest
        ports:
        - containerPort: 3000
        resources:
          requests:
            memory: "512Mi"
            cpu: "500m"
          limits:
            memory: "1Gi"
            cpu: "1000m"
        env:
        - name: DATABASE_URL
          valueFrom:
            secretKeyRef:
              name: preditest-secrets
              key: database-url
        livenessProbe:
          httpGet:
            path: /health
            port: 3000
          initialDelaySeconds: 30
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /ready
            port: 3000
          initialDelaySeconds: 10
          periodSeconds: 5
---
apiVersion: v1
kind: Service
metadata:
  name: preditest-api-service
spec:
  selector:
    app: preditest-api
  ports:
  - protocol: TCP
    port: 80
    targetPort: 3000
  type: LoadBalancer
---
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: preditest-api-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
```

```
    kind: Deployment
    name: preditest-api
  minReplicas: 3
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 80
```

6.2.3 Database Scaling

Read Replicas: - Master: Writes - Replicas (3x): Reads - Load balancer distribui queries de leitura

Sharding: - Shard por `projectId` (hash-based) - Cada shard em servidor separado - Reduz contenção e aumenta throughput

Connection Pooling:

```
const pool = new Pool({
  host: process.env.DB_HOST,
  port: 5432,
  database: process.env.DB_NAME,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  max: 20, // Máximo de conexões
  idleTimeoutMillis: 30000,
  connectionTimeoutMillis: 2000,
});
```

6.3 CDN e Edge Computing

CloudFront Configuration:

```
{
  "Origins": [
    {
      "Id": "preditest-s3",
      "DomainName": "preditest-assets.s3.amazonaws.com",
      "S3OriginConfig": {
        "OriginAccessIdentity": "origin-access-identity/cloudfront/ABCDEFG"
      }
    }
  ],
  "DefaultCacheBehavior": {
    "TargetOriginId": "preditest-s3",
    "ViewerProtocolPolicy": "redirect-to-https",
    "AllowedMethods": ["GET", "HEAD", "OPTIONS"],
    "CachedMethods": ["GET", "HEAD"],
    "Compress": true,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "MinTTL": 0
  },
  "PriceClass": "PriceClass_100",
  "ViewerCertificate": {
    "ACMCertificateArn": "arn:aws:acm:us-east-1:123456789:certificate/abc",
    "SSLSupportMethod": "sni-only",
    "MinimumProtocolVersion": "TLSv1.2_2021"
  }
}
```

---

7. Deployment e DevOps

7.1 CI/CD Pipeline

GitHub Actions Workflow:

```yaml
name: CI/CD Pipeline

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '22'
      - run: pnpm install
      - run: pnpm test
      - run: pnpm lint
      - run: pnpm type-check

  build:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: docker/setup-buildx-action@v2
      - uses: docker/login-action@v2
        with:
          registry: ghcr.io
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}
      - uses: docker/build-push-action@v4
        with:
          context: .
          push: true
          tags: ghcr.io/preditest/api:${{ github.sha }}

  deploy-staging:
    needs: build
    if: github.ref == 'refs/heads/develop'
    runs-on: ubuntu-latest
    steps:
      - uses: azure/k8s-set-context@v3
        with:
          method: kubeconfig
          kubeconfig: ${{ secrets.KUBE_CONFIG_STAGING }}
      - run: |
          kubectl set image deployment/preditest-api \
            api=ghcr.io/preditest/api:${{ github.sha }} \
            -n staging

  deploy-production:
    needs: build
    if: github.ref == 'refs/heads/main'
    runs-on: ubuntu-latest
    environment:
      name: production
      url: https://preditest.ai
    steps:
      - uses: azure/k8s-set-context@v3
        with:
          method: kubeconfig
          kubeconfig: ${{ secrets.KUBE_CONFIG_PROD }}
      - run: |
```

```
          kubectl set image deployment/preditest-api \
            api=ghcr.io/preditest/api:${{ github.sha }} \
            -n production
```

7.2 Ambientes

Desenvolvimento (local): - Docker Compose - PostgreSQL local - Redis local - Hot reload (Vite HMR)

Staging (AWS): - EKS cluster (2 nodes) - RDS PostgreSQL (db.t3.medium) - ElastiCache Redis (cache.t3.micro) - S3 bucket (staging-assets)

Produção (AWS): - EKS cluster (3+ nodes, auto-scaling) - RDS PostgreSQL (db.r6g.xlarge, Multi-AZ) - ElastiCache Redis (cache.r6g.large, cluster mode) - S3 bucket (prod-assets) - CloudFront CDN

7.3 Backup e Disaster Recovery

Estratégia de Backup: - **Database**: Snapshot diário (RDS automated backups) - **S3**: Versionamento habilitado + replicação cross-region - **Configurações**: Armazenadas em Git (Infrastructure as Code)

RTO/RPO: - **RTO** (Recovery Time Objective): 4 horas - **RPO** (Recovery Point Objective): 1 hora

Plano de DR: 1. Detecção de falha (monitoring) 2. Notificação da equipe (PagerDuty) 3. Failover para região secundária (Route53 health check) 4. Restauração de backup mais recente 5. Validação de integridade 6. Comunicação com stakeholders

---

8. Testes e Qualidade

8.1 Estratégia de Testes

Pirâmide de Testes:

```
        /\
       /  \  E2E (5%)
      /____\
     /      \ Integration (15%)
    /________\
   /          \ Unit (80%)
  /____________\
```

8.2 Testes Unitários

Framework: Vitest

Exemplo:

```typescript
// server/monteCarloSimulator.test.ts
import { describe, it, expect } from 'vitest';
import { runMonteCarloSimulation } from './monteCarloSimulator';

describe('Monte Carlo Simulator', () => {
  it('should run solubility simulation', () => {
    const result = runMonteCarloSimulation(
      'solubility',
      {
        temperature: { type: 'normal', mean: 25, stdDev: 2 },
        deltaH: { type: 'normal', mean: -20000, stdDev: 1000 },
        deltaS: { type: 'normal', mean: 50, stdDev: 5 },
      },
      1000
    );

    expect(result.mean).toBeGreaterThan(0);
    expect(result.stdDev).toBeGreaterThan(0);
    expect(result.distribution).toHaveLength(20); // 20 bins
  });

  it('should handle edge cases', () => {
    expect(() => {
      runMonteCarloSimulation('invalid_model', {}, 1000);
    }).toThrow('Unknown model type');

    expect(() => {
      runMonteCarloSimulation('solubility', {}, 0);
    }).toThrow('Iterations must be > 0');
  });
});
```

Coverage Target: >80%

8.3 Testes de Integração

Framework: Vitest + Supertest

Exemplo:

```typescript
// server/routers.integration.test.ts
import { describe, it, expect, beforeAll, afterAll } from 'vitest';
import { createCaller } from './routers';
import { createTestContext } from './test-utils';

describe('Projects Router', () => {
  let caller: ReturnType<typeof createCaller>;
  let testUser: User;

  beforeAll(async () => {
    const ctx = await createTestContext();
    caller = createCaller(ctx);
    testUser = ctx.user;
  });

  it('should create and retrieve project', async () => {
    // Create
    const created = await caller.projects.create({
      name: 'Test Project',
      description: 'Test description',
      status: 'planning',
    });

    expect(created.id).toBeDefined();

    // Retrieve
    const retrieved = await caller.projects.getById({ id: created.id });

    expect(retrieved.name).toBe('Test Project');
    expect(retrieved.userId).toBe(testUser.id);
  });

  it('should enforce authorization', async () => {
    const anonCtx = await createTestContext({ user: null });
    const anonCaller = createCaller(anonCtx);

    await expect(
      anonCaller.projects.create({ name: 'Test', status: 'planning' })
    ).rejects.toThrow('UNAUTHORIZED');
  });
});
```

8.4 Testes E2E

Framework: Playwright

Exemplo:

```typescript
// e2e/projects.spec.ts
import { test, expect } from '@playwright/test';

test.describe('Projects Flow', () => {
  test('should create, view and delete project', async ({ page }) => {
    // Login
    await page.goto('/');
    await page.click('text=Login');
    await page.fill('input[name="email"]', 'test@example.com');
    await page.fill('input[name="password"]', 'password123');
    await page.click('button[type="submit"]');

    // Navigate to projects
    await page.click('text=Projetos');
    await expect(page).toHaveURL('/projects');

    // Create project
    await page.click('text=Novo Projeto');
    await page.fill('input[name="name"]', 'E2E Test Project');
    await page.fill('textarea[name="description"]', 'Created by E2E test');
    await page.selectOption('select[name="status"]', 'planning');
    await page.click('button:has-text("Criar")');

    // Verify creation
    await expect(page.locator('text=E2E Test Project')).toBeVisible();

    // View details
    await page.click('button:has-text("Ver Detalhes")');
    await expect(page.locator('text=Created by E2E test')).toBeVisible();

    // Delete
    await page.click('button:has-text("Excluir")');
    await page.click('button:has-text("Confirmar")');

    // Verify deletion
    await expect(page.locator('text=E2E Test Project')).not.toBeVisible();
  });
});
```

8.5 Testes de Performance

Framework: k6

Exemplo:

```
// k6/load-test.js
import http from 'k6/http';
import { check, sleep } from 'k6';

export const options = {
  stages: [
    { duration: '2m', target: 100 }, // Ramp up to 100 users
    { duration: '5m', target: 100 }, // Stay at 100 users
    { duration: '2m', target: 200 }, // Ramp up to 200 users
    { duration: '5m', target: 200 }, // Stay at 200 users
    { duration: '2m', target: 0 },   // Ramp down to 0 users
  ],
  thresholds: {
    http_req_duration: ['p(95)<500'], // 95% of requests must complete below
500ms
    http_req_failed: ['rate<0.01'],   // Error rate must be below 1%
  },
};

export default function () {
  const res = http.get('https://api.preditest.ai/api/trpc/projects.list');

  check(res, {
    'status is 200': (r) => r.status === 200,
    'response time < 500ms': (r) => r.timings.duration < 500,
  });

  sleep(1);
}
```

---

9. Roadmap e Próximos Passos

9.1 Fase 1 (Concluída) - MVP

- ✅ CRUD de projetos
- ✅ Análise de sentimento em redes sociais
- ✅ Simulação Monte Carlo
- ✅ Gerenciamento de testes
- ✅ Alertas de sentimento negativo
- ✅ Dashboard de métricas

9.2 Fase 2 (Q2 2025) - Expansão

- [ ] Integração SAP ERP
- [ ] Integração MFC (Manufacturing Execution System)

- [ ] Análise preditiva com ML (Random Forest, XGBoost)
- [ ] Geração automática de relatórios (PDF, Excel)
- [ ] Mobile app (React Native)
- [ ] API pública (REST + GraphQL)

9.3 Fase 3 (Q3 2025) - Otimização

- [ ] Otimização de modelos ML (AutoML)
- [ ] Análise de imagens (Computer Vision para inspeção visual)
- [ ] Chatbot com LLM (assistente virtual)
- [ ] Recomendações personalizadas (Recommender System)
- [ ] Integração com IoT (sensores em tempo real)

9.4 Fase 4 (Q4 2025) - Globalização

- [ ] Multi-idioma (EN, ES, FR, DE)
- [ ] Multi-moeda (USD, EUR, GBP)
- [ ] Conformidade GDPR (Europa)
- [ ] Conformidade CCPA (Califórnia)
- [ ] Expansão para outras indústrias (farmacêutica, cosmética)

---

10. Conclusão

O **PrediTest AI** representa uma solução completa e inovadora para otimização de testes industriais na indústria alimentícia. Com arquitetura cloud-native, modelos matemáticos validados, análise de sentimento 360° e simulação Monte Carlo, a plataforma oferece insights acionáveis que reduzem falhas críticas em 75%, economizam R$ 3M anualmente e aceleram o time-to-market em 40%.

A documentação técnica apresentada detalha todos os aspectos da solução, desde a modelagem de dados (18 tabelas) até a arquitetura de microserviços escalável, passando por integrações com APIs externas, conformidade regulatória (LGPD, ISO 27001, SOC 2, FDA 21 CFR Part 11) e estratégias de deployment em Kubernetes.

Com ROI de 259% e payback de 8,7 meses, o PrediTest AI é uma solução factível e altamente rentável para empresas que buscam transformação digital em seus processos de desenvolvimento de produtos.

Apêndices

Apêndice A: Glossário

- **Monte Carlo**: Método estatístico que usa amostragem aleatória para obter resultados numéricos
- **tRPC**: TypeScript Remote Procedure Call - framework para APIs type-safe
- **Drizzle ORM**: Object-Relational Mapping para TypeScript com zero-cost abstractions
- **OAuth 2.0**: Protocolo de autorização para acesso delegado
- **JWT**: JSON Web Token - padrão para tokens de autenticação
- **RBAC**: Role-Based Access Control - controle de acesso baseado em papéis
- **SLA**: Service Level Agreement - acordo de nível de serviço
- **RTO**: Recovery Time Objective - tempo máximo de recuperação
- **RPO**: Recovery Point Objective - perda máxima de dados aceitável
- **TDE**: Transparent Data Encryption - criptografia transparente de dados

Apêndice B: Referências

1. Van't Hoff, J. H. (1884). "Études de Dynamique chimique"
2. Arrhenius, S. (1889). "Über die Reaktionsgeschwindigkeit bei der Inversion von Rohrzucker durch Säuren"
3. Maxwell, J. C. (1867). "On the Dynamical Theory of Gases"
4. Gompertz, B. (1825). "On the Nature of the Function Expressive of the Law of Human Mortality"
5. ISO 27001:2013 - Information Security Management
6. FDA 21 CFR Part 11 - Electronic Records; Electronic Signatures

7. LGPD - Lei Geral de Proteção de Dados (Lei nº 13.709/2018)
8. SOC 2 Type II - Service Organization Control 2

Apêndice C: Contatos

Suporte Técnico: suporte@preditest.ai
Vendas: vendas@preditest.ai
Segurança: security@preditest.ai
Telefone: +55 11 98765-4321
Website: https://www.preditest.ai

Fim da Documentação Técnica