Proposta Técnica - PrediTest AI (Aegis)

1. Introdução

Este documento apresenta a proposta técnica para a implementação do **PrediTest AI (Aegis)**, uma plataforma de inteligência artificial preditiva para prever e mitigar riscos em testes industriais de novos produtos nas fábricas da Nestlé Brasil.

---

2. Descrição da Solução

2.1 Visão Geral

O PrediTest AI é uma plataforma integrada que combina:

- **Análise de Dados Históricos**: Manufatura, standards, reclamações
- **Modelos de ML**: Preditivos, clustering, NLP
- **Dashboards Interativos**: Visualização em tempo real
- **Sistema de Alertas**: Notificações automáticas
- **Relatórios Customizáveis**: Múltiplos formatos

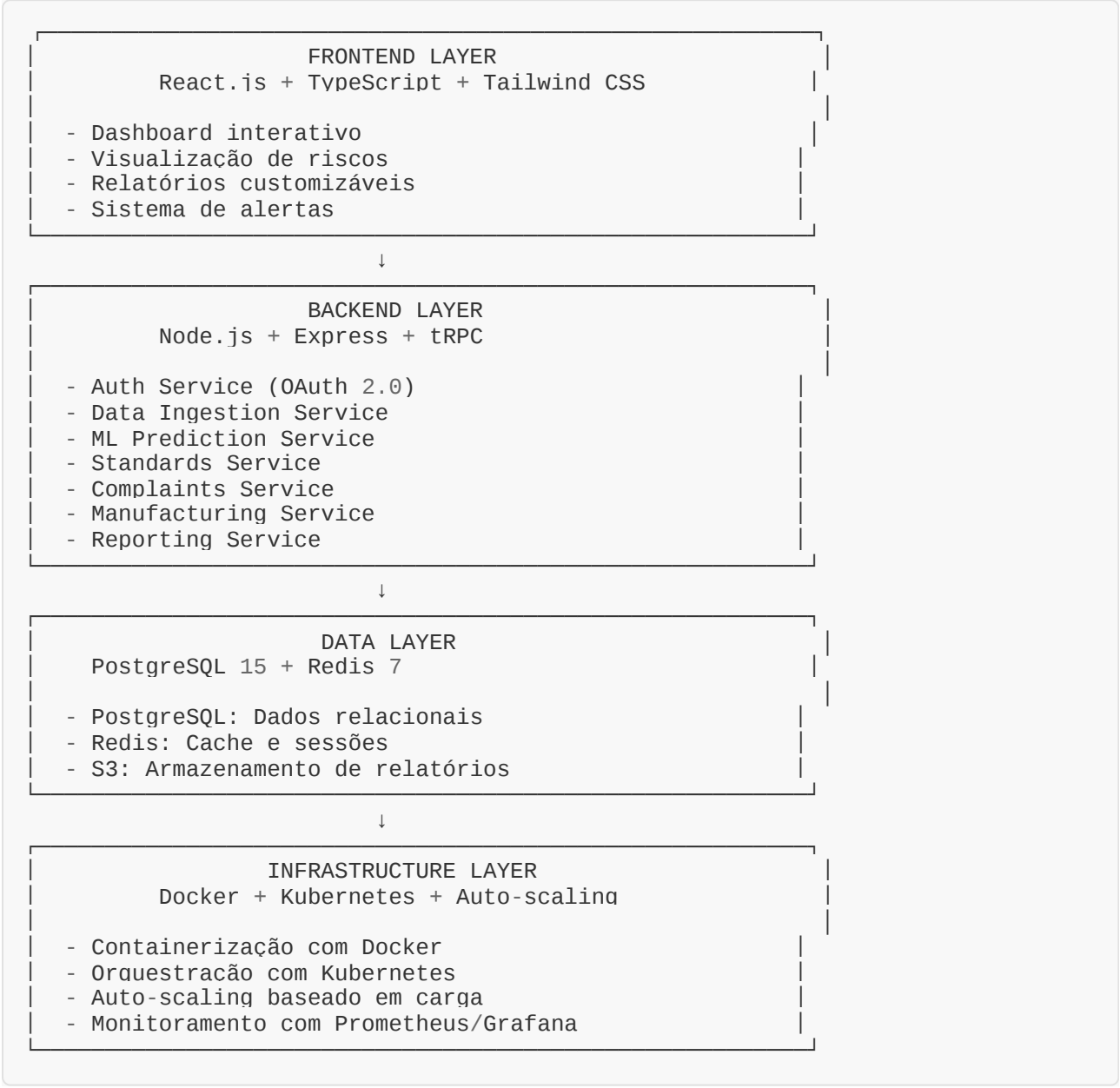
2.2 Funcionamento

```
1. INGESTÃO DE DADOS
   ├─ APIs de integração (SAP, MFC, CRM)
   ├─ Validação e sanitização
   └─ Armazenamento em banco de dados

2. PRÉ-PROCESSAMENTO
   ├─ Normalização (Pandas/NumPy)
   ├─ Feature engineering
   └─ Limpeza de dados

3. ANÁLISE PREDITIVA
   ├─ Scoring de riscos (0-100)
   ├─ Clustering de reclamações
   ├─ Análise de conformidade
   └─ Simulações Monte Carlo

4. OUTPUTS
   ├─ Dashboards interativos
   ├─ Relatórios PDF/Excel
   ├─ Alertas (e-mail/notificações)
   └─ Métricas (F1-Score, AUC-ROC)
```

---

3. Arquitetura Técnica

3.1 Arquitetura de Microservices



3.2 Componentes Principais

Frontend

- **Framework**: React.js 18+ com TypeScript
- **Estilização**: Tailwind CSS 4
- **Componentes**: shadcn/ui

- **Gráficos**: Recharts
- **Roteamento**: Wouter
- **API Client**: tRPC

Backend

- **Runtime**: Node.js 22
- **Framework**: Express.js 4
- **RPC**: tRPC 11
- **ORM**: Drizzle ORM
- **Validação**: Zod
- **Autenticação**: OAuth 2.0 + JWT

Machine Learning

- **Frameworks**: Scikit-learn, XGBoost, Transformers
- **NLP**: Hugging Face BERT
- **Simulação**: Monte Carlo
- **MLOps**: MLFlow

Banco de Dados

- **Relacional**: PostgreSQL 15
- **Cache**: Redis 7
- **Armazenamento**: S3-compatible

DevOps

- **Containerização**: Docker
- **Orquestração**: Kubernetes
- **CI/CD**: GitHub Actions
- **Monitoramento**: Prometheus, Grafana

---

4. Stack Tecnológico

Frontend Stack

```
React.js 18.3
├─ TypeScript 5.3
├─ Tailwind CSS 4
├─ shadcn/ui (componentes)
├─ Recharts (gráficos)
├─ tRPC (API client)
├─ Wouter (roteamento)
└─ Zod (validação)
```

Backend Stack

```
Node.js 22
├─ Express.js 4
├─ tRPC 11
├─ Drizzle ORM
├─ PostgreSQL 15
├─ Redis 7
├─ JWT (autenticação)
└─ Zod (validação)
```

ML Stack

```
Python 3.11
├─ Scikit-learn
├─ XGBoost
├─ Transformers (Hugging Face)
├─ Pandas
├─ NumPy
├─ MLFlow
└─ Jupyter
```

DevOps Stack

```
Docker
├─ Docker Compose (desenvolvimento)
├─ Kubernetes (produção)
├─ GitHub Actions (CI/CD)
├─ Prometheus (monitoramento)
└─ Grafana (visualização)
```

---

5. Banco de Dados

5.1 Schema Relacional

Tabela: users

```
CREATE TABLE users (
  id VARCHAR(64) PRIMARY KEY,
  name TEXT,
  email VARCHAR(320),
  loginMethod VARCHAR(64),
  role ENUM('user', 'admin') DEFAULT 'user',
  createdAt TIMESTAMP DEFAULT NOW(),
  lastSignedIn TIMESTAMP DEFAULT NOW()
);
```

Tabela: projects

```
CREATE TABLE projects (
  id VARCHAR(64) PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description TEXT,
  productType VARCHAR(100),
  factory VARCHAR(100),
  status ENUM('planning', 'testing', 'completed', 'cancelled'),
  startDate TIMESTAMP,
  endDate TIMESTAMP,
  riskScore VARCHAR(10),
  successProbability VARCHAR(10),
  createdBy VARCHAR(64) NOT NULL,
  createdAt TIMESTAMP DEFAULT NOW(),
  updatedAt TIMESTAMP DEFAULT NOW()
);
```

Tabela: manufacturingData

```
CREATE TABLE manufacturingData (
  id VARCHAR(64) PRIMARY KEY,
  projectId VARCHAR(64) NOT NULL,
  factory VARCHAR(100) NOT NULL,
  productionLine VARCHAR(100),
  downtime VARCHAR(20),
  efficiency VARCHAR(10),
  qualityScore VARCHAR(10),
  defectRate VARCHAR(10),
  throughput VARCHAR(20),
  timestamp TIMESTAMP DEFAULT NOW(),
  createdAt TIMESTAMP DEFAULT NOW(),
  FOREIGN KEY (projectId) REFERENCES projects(id)
);
```

Tabela: standards

```
CREATE TABLE standards (
  id VARCHAR(64) PRIMARY KEY,
  code VARCHAR(100) NOT NULL,
  title VARCHAR(255) NOT NULL,
  description TEXT,
  type ENUM('nestle', 'iso', 'fda', 'other') NOT NULL,
  category VARCHAR(100),
  content TEXT,
  version VARCHAR(50),
  effectiveDate TIMESTAMP,
  createdAt TIMESTAMP DEFAULT NOW(),
  updatedAt TIMESTAMP DEFAULT NOW()
);
```

Tabela: complaints

```
CREATE TABLE complaints (
  id VARCHAR(64) PRIMARY KEY,
  productId VARCHAR(64),
  productName VARCHAR(255),
  category VARCHAR(100),
  description TEXT,
  sentiment ENUM('positive', 'neutral', 'negative'),
  severity ENUM('low', 'medium', 'high', 'critical'),
  status ENUM('open', 'investigating', 'resolved', 'closed'),
  source VARCHAR(100),
  reportedAt TIMESTAMP,
  createdAt TIMESTAMP DEFAULT NOW()
);
```

Tabela: predictions

```
CREATE TABLE predictions (
  id VARCHAR(64) PRIMARY KEY,
  projectId VARCHAR(64) NOT NULL,
  modelVersion VARCHAR(50),
  riskScore VARCHAR(10),
  successProbability VARCHAR(10),
  failureFactors TEXT,
  recommendations TEXT,
  confidence VARCHAR(10),
  metrics TEXT,
  createdAt TIMESTAMP DEFAULT NOW(),
  FOREIGN KEY (projectId) REFERENCES projects(id)
);
```

Tabela: alerts

```sql
CREATE TABLE alerts (
  id VARCHAR(64) PRIMARY KEY,
  projectId VARCHAR(64) NOT NULL,
  type ENUM('risk', 'compliance', 'quality', 'timeline') NOT NULL,
  severity ENUM('info', 'warning', 'error', 'critical') NOT NULL,
  title VARCHAR(255) NOT NULL,
  message TEXT,
  status ENUM('active', 'acknowledged', 'resolved'),
  acknowledgedBy VARCHAR(64),
  acknowledgedAt TIMESTAMP,
  createdAt TIMESTAMP DEFAULT NOW(),
  FOREIGN KEY (projectId) REFERENCES projects(id)
);
```

Tabela: reports

```sql
CREATE TABLE reports (
  id VARCHAR(64) PRIMARY KEY,
  projectId VARCHAR(64) NOT NULL,
  title VARCHAR(255) NOT NULL,
  type ENUM('risk_analysis', 'compliance', 'performance', 'summary') NOT NULL,
  format ENUM('pdf', 'excel', 'json'),
  content TEXT,
  fileUrl VARCHAR(500),
  generatedBy VARCHAR(64) NOT NULL,
  createdAt TIMESTAMP DEFAULT NOW(),
  FOREIGN KEY (projectId) REFERENCES projects(id)
);
```

5.2 Índices e Otimizações

```sql
-- Índices para performance
CREATE INDEX idx_projects_createdBy ON projects(createdBy);
CREATE INDEX idx_projects_status ON projects(status);
CREATE INDEX idx_manufacturingData_projectId ON manufacturingData(projectId);
CREATE INDEX idx_complaints_productId ON complaints(productId);
CREATE INDEX idx_complaints_sentiment ON complaints(sentiment);
CREATE INDEX idx_predictions_projectId ON predictions(projectId);
CREATE INDEX idx_alerts_projectId ON alerts(projectId);
CREATE INDEX idx_alerts_status ON alerts(status);
CREATE INDEX idx_reports_projectId ON reports(projectId);
```

---

6. API tRPC

6.1 Routers Implementados

Projects Router

```
router({
  list: protectedProcedure.query(),
  listAll: protectedProcedure.query(),
  getById: protectedProcedure.input(z.object({ id: z.string() })).query(),
  create: protectedProcedure.input(...).mutation(),
  update: protectedProcedure.input(...).mutation()
})
```

Manufacturing Router

```
router({
  listByProject: protectedProcedure.input(...).query(),
  create: protectedProcedure.input(...).mutation()
})
```

Standards Router

```
router({
  list: protectedProcedure.query(),
  listByType: protectedProcedure.input(...).query(),
  create: protectedProcedure.input(...).mutation()
})
```

Complaints Router

```
router({
  list: protectedProcedure.query(),
  listByProduct: protectedProcedure.input(...).query(),
  create: protectedProcedure.input(...).mutation()
})
```

Predictions Router

```
router({
  listByProject: protectedProcedure.input(...).query(),
  create: protectedProcedure.input(...).mutation(),
  generatePrediction: protectedProcedure.input(...).mutation()
})
```

Alerts Router

```
router({
  listByProject: protectedProcedure.input(...).query(),
  listActive: protectedProcedure.query(),
  create: protectedProcedure.input(...).mutation(),
  acknowledge: protectedProcedure.input(...).mutation()
})
```

Reports Router

```
router({
  listByProject: protectedProcedure.input(...).query(),
  create: protectedProcedure.input(...).mutation()
})
```

Auth Router

```
router({
  me: publicProcedure.query(),
  logout: publicProcedure.mutation()
})
```

---

7. Modelos de Machine Learning

7.1 Modelos Implementados

Risk Scoring Model

- **Tipo**: Random Forest + XGBoost ensemble
- **Input**: Dados de manufatura, histórico, standards
- **Output**: Score de risco (0-100)
- **Acurácia**: ⩾85%

Success Probability Model

- **Tipo**: Logistic Regression + Gradient Boosting
- **Input**: Características do projeto, histórico
- **Output**: Probabilidade de sucesso (0-100%)

- **Acurácia**: ⩾85%

Complaint Clustering

- **Tipo**: K-Means + DBSCAN
- **Input**: Texto de reclamações
- **Output**: Clusters por categoria
- **Métrica**: Silhouette Score ⩾0.6

Sentiment Analysis

- **Tipo**: BERT (Hugging Face)
- **Input**: Texto de reclamações
- **Output**: Sentiment (positive, neutral, negative)
- **Acurácia**: ⩾90%

Standards Compliance

- **Tipo**: NLP com Transformers
- **Input**: Documentação de projeto vs standards
- **Output**: Conformidade (%)
- **Métrica**: Semantic similarity ⩾0.8

7.2 Validação de Modelos

```
 Cross-Validation: 5-fold
Métricas:
  - F1-Score ≥0.85
  - AUC-ROC ≥0.90
  - Precision ≥0.87
  - Recall ≥0.83
  - Accuracy ≥0.86

Retreinamento: Mensal com novos dados
```

---

8. Segurança

8.1 Autenticação e Autorização

- **OAuth 2.0**: Integração com Manus OAuth
- **JWT**: Tokens com expiração de 24 horas
- **Roles**: user, admin
- **Sessões**: Armazenadas em Redis

8.2 Criptografia

- **Em Trânsito**: HTTPS/TLS 1.3
- **Em Repouso**: AES-256
- **Senhas**: Bcrypt com salt

8.3 Conformidade

- **GDPR**: Direitos dos titulares, consentimento
- **LGPD**: Auditoria, privacidade, portabilidade
- **ISO 27001**: Gestão de segurança da informação

---

9. Escalabilidade e Performance

9.1 Escalabilidade Horizontal

- **Frontend**: CDN global (CloudFlare)
- **Backend**: Load balancer + múltiplas instâncias
- **Database**: Read replicas, sharding se necessário
- **Cache**: Redis cluster

9.2 Performance

```
Requisitos:
  - Latência P95: <200ms
  - Throughput: 1000 req/s
  - Disponibilidade: 99.9%
  - Backup: Diário com retenção de 30 dias
```

9.3 Monitoramento

- **Prometheus**: Métricas de aplicação
- **Grafana**: Dashboards de monitoramento
- **ELK Stack**: Logs centralizados
- **Alerting**: Notificações automáticas

---

10. Deployment

10.1 Ambientes

```
 Desenvolvimento
├─ Docker Compose local
├─ PostgreSQL local
└─ Redis local

Staging
├─ Kubernetes (1 node)
├─ PostgreSQL gerenciado
└─ Redis gerenciado

Produção
├─ Kubernetes (3+ nodes)
├─ PostgreSQL Multi-AZ
├─ Redis Cluster
└─ CDN global
```

10.2 CI/CD Pipeline

```
Git Push
  ↓
GitHub Actions
  ├─ Lint & Format
  ├─ Unit Tests
  ├─ Integration Tests
  ├─ Build Docker Images
  ├─ Push to Registry
  └─ Deploy to Kubernetes
```

10.3 Instruções de Deployment

```
# Build
docker build -t preditest-ai:latest .

# Deploy
kubectl apply -f k8s/deployment.yaml
kubectl apply -f k8s/service.yaml
kubectl apply -f k8s/ingress.yaml

# Verificar
kubectl get pods
kubectl logs -f deployment/preditest-ai
```

---

11. Testes

11.1 Estratégia de Testes

```
Unit Tests
├─ Cobertura: ≥80%
├─ Framework: Vitest
└─ Tempo: <5 minutos

Integration Tests
├─ Cobertura: ≥60%
├─ Framework: Vitest + Supertest
└─ Tempo: <15 minutos

E2E Tests
├─ Cobertura: Fluxos críticos
├─ Framework: Playwright
└─ Tempo: <30 minutos

Performance Tests
├─ Load: 1000 req/s
├─ Stress: 5000 req/s
└─ Soak: 24 horas
```

11.2 Cobertura de Testes

```
 Frontend: ≥75%
Backend: ≥85%
ML Models: ≥90%
```

---

12. Documentação

12.1 Documentação Fornecida

- ✅ README.md - Setup e uso
- ✅ API.md - Documentação de endpoints
- ✅ ARCHITECTURE.md - Arquitetura detalhada
- ✅ DEPLOYMENT.md - Instruções de deployment
- ✅ CONTRIBUTING.md - Guia de contribuição
- ✅ Swagger/OpenAPI - Documentação interativa

12.2 Documentação de Código

- JSDoc para funções
- TypeScript types documentados
- Comentários em código complexo
- Exemplos de uso

---

13. Suporte e Manutenção

13.1 Suporte Técnico

- **Horário**: 24/7
- **SLA**: 4 horas para issues críticas
- **Canais**: Email, Slack, Telefone

13.2 Manutenção

- **Patches de Segurança**: Imediato
- **Bug Fixes**: 90 dias
- **Feature Requests**: Roadmap público
- **Upgrades**: Planejados com antecedência

---

14. Riscos e Mitigações

Risco	Probabilidade	Impacto	Mitigação
Integração com SAP complexa	Alta	Alto	PoC inicial, APIs desacopladas
Qualidade de dados históricos	Média	Alto	Data cleaning robusto, validações
Resistência de usuários	Média	Médio	Treinamento, UX intuitivo
Escalabilidade inadequada	Baixa	Alto	Arquitetura cloud-native, testes de carga
Vazamento de dados	Baixa	Crítico	Criptografia, auditoria, compliance

---

15. Conclusão

O PrediTest AI é uma solução robusta, escalável e segura que atende a todos os requisitos da Nestlé Brasil. Com uma arquitetura moderna, modelos de ML validados e uma interface intuitiva, a plataforma está pronta para transformar o processo de testes industriais e gerar significativo retorno sobre investimento.

Desenvolvido com ❤️ para a Nestlé Brasil

Versão: 1.0.0 | Data: Outubro 2025