HEFESTO — Manual Técnico

Sistema de Controle Orçamentário para Facilities

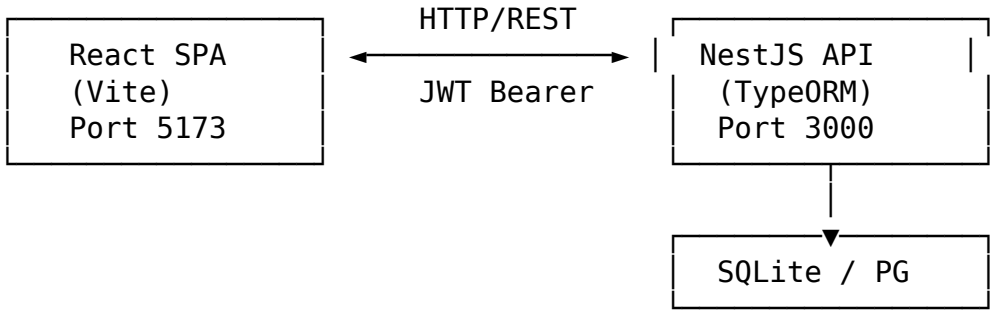
Versão 1.0 | Fevereiro 2025

---

1. Visão Geral da Arquitetura

O HEFESTO é uma aplicação web fullstack composta por:

- **Backend:** API REST em NestJS (Node.js) com TypeORM
- **Frontend:** SPA em React + TypeScript com Vite
- **Banco de Dados:** SQLite (desenvolvimento) / PostgreSQL (produção)
- **Autenticação:** JWT com RBAC (Role-Based Access Control)



2. Stack Completa

Componente	Tecnologia	Versão
Runtime	Node.js	22.x LTS
Backend Framework	NestJS	10.x
ORM	TypeORM	0.3.x
Frontend Framework	React	18.x
Build Tool	Vite	5.x
Linguagem	TypeScript	5.x
UI Components	Tailwind CSS	3.x
BD Desenvolvimento	SQLite3	5.x
BD Produção	PostgreSQL	16.x
Auth	@nestjs/jwt + passport	10.x
HTTP Client	Axios	1.x
Validação	class-validator	0.14.x
Documentação API	@nestjs/swagger	7.x

3. Estrutura de Pastas

3.1 Backend

```
backend/
```

```
├── src/
│   ├── main.ts                    # Bootstrap da aplicação
│   ├── app.module.ts              # Módulo raiz
│   ├── common/                  # Guards, decorators, pipes, interceptors
│   │   ├── guards/
│   │   │   ├── jwt-auth.guard.ts
│   │   │   └── roles.guard.ts
│   │   ├── decorators/
│   │   │   ├── roles.decorator.ts
│   │   │   └── current-user.decorator.ts
│   │   └── interceptors/
│   │       └── audit.interceptor.ts
│   ├── database/                # Configuração TypeORM, migrations, seeds
│   │   ├── database.module.ts
│   │   ├── migrations/
│   │   └── seeds/
│   └── modules/
│       ├── auth/                # Autenticação JWT, login, refresh token
│       │   ├── auth.controller.ts
│       │   ├── auth.service.ts
│       │   ├── auth.module.ts
│       │   ├── strategies/
│       │   │   └── jwt.strategy.ts
│       │   └── dto/
│       ├── users/                 # CRUD de usuários e perfis
│       │   ├── users.controller.ts
│       │   ├── users.service.ts
│       │   ├── entities/
│       │   │   ├── usuario.entity.ts
│       │   │   └── perfil.entity.ts
│       │   └── dto/
│       ├── locais/                # Gestão de locais/unidades
│       ├── centros-custo/        # Centros de custo vinculados a locais
│       ├── categorias/            # Categorias de serviço
│       ├── fornecedores/         # Cadastro de fornecedores e certidões
│       ├── demandas/              # Abertura e gestão de demandas
│       ├── propostas/            # Recebimento e comparação de propostas
│       ├── orcamento/             # Orçamento planejado vs realizado
│       ├── workflow/              # Máquina de estados de aprovação
│       ├── dashboard/             # Indicadores e relatórios
│       └── ordens-servico/        # Emissão e acompanhamento de OS
├── test/
├── nest-cli.json
├── tsconfig.json
└── package.json
```

3.2 Frontend

```
frontend/
├── src/
│   ├── main.tsx                    # Entry point
│   ├── App.tsx                     # Router + Layout
│   ├── assets/                     # Imagens, ícones
│   ├── components/                 # Componentes reutilizáveis
│   │   ├── Layout/
│   │   ├── Sidebar/
│   │   ├── Header/
│   │   ├── DataTable/
│   │   ├── StatusBadge/
│   │   └── Charts/
│   ├── pages/
│   │   ├── Login.tsx               # Tela de autenticação
│   │   ├── Dashboard.tsx           # Painel de indicadores
│   │   ├── Demandas.tsx            # Lista de demandas com filtros
│   │   ├── Landing.tsx             # Página inicial / nova demanda
│   │   ├── Fornecedores.tsx        # Gestão de fornecedores
│   │   ├── Orcamentos.tsx          # Orçamento planejado vs realizado
│   │   ├── OrdensServico.tsx       # Ordens de serviço
│   │   ├── Relatorios.tsx          # Relatórios gerenciais
│   │   └── Usuarios.tsx            # Administração de usuários
│   ├── services/                   # Axios clients e API calls
│   │   └── api.ts
│   ├── types/                      # Interfaces TypeScript
│   └── index.css                   # Tailwind directives
├── vite.config.ts
├── tailwind.config.js
├── tsconfig.json
└── package.json
```

4. Modelo de Dados

4.1 Diagrama de Entidades

O sistema possui 16 entidades principais:

```
perfis ──< usuarios ──< demandas ──< itens_linha
                  │               │
                  │               ├──< propostas
                  │               │
                  │               ├──< workflow_aprovacao
                  │               │
                  │               └──< ordens_servico ──< avaliacoes
                  │
                  └──< audit_log
```

```
locais ──< centros_custo ──< orcamento_planejado

categorias ──< demandas

fornecedores ──< certidoes
fornecedores ──< propostas

alertas (standalone)
```

4.2 Descrição das Entidades

#	Entidade	Descrição	Campos Principais
1	**perfis**	Perfis de acesso (RBAC)	id, nome, descricao, permissoes (JSON)
2	**usuarios**	Usuários do sistema	id, nome, email, senha_hash, perfil_id, ativo, ultimo_acesso
3	**locais**	Unidades/edifícios	id, nome, endereco, cidade, estado, cnpj, ativo
4	**centros_custo**	Centros de custo por local	id, codigo, descricao, local_id, ativo
5	**categorias**	Categorias de serviço	id, nome, descricao, sla_dias, ativo
6	**fornecedores**	Cadastro de fornecedores	id, razao_social, cnpj, contato, email, telefone, ativo, rating
7	**certidoes**	Certidões de fornecedores	id, fornecedor_id, tipo, arquivo_url, validade, status
8	**orcamento_planejado**	Budget por centro de custo/ano	id, centro_custo_id, ano, mes, valor_planejado, valor_realizado
9	**demandas**	Demandas de serviço	id, titulo, descricao, local_id, categoria_id, solicitante_id, status, prioridade, valor_estimado, created_at

#	Entidade	Descrição	Campos Principais
10	**itens_linha**	Itens detalhados da demanda	id, demanda_id, descricao, quantidade, unidade, valor_unitario
11	**propostas**	Propostas de fornecedores	id, demanda_id, fornecedor_id, valor_total, arquivo_url, data_validade, status, observacoes
12	**workflow_aprovacao**	Etapas de aprovação	id, demanda_id, etapa, aprovador_id, status, comentario, data_acao
13	**ordens_servico**	Ordens de serviço emitidas	id, demanda_id, proposta_id, numero_os, data_inicio, data_fim_prevista, data_fim_real, status
14	**avaliacoes**	Avaliação pós-execução	id, ordem_servico_id, avaliador_id, nota, comentario, created_at
15	**audit_log**	Log de auditoria	id, usuario_id, acao, entidade, entidade_id, dados_antes, dados_depois, ip, created_at
16	**alertas**	Notificações e alertas	id, usuario_id, tipo, mensagem, lido, referencia_tipo, referencia_id, created_at

4.3 Perfis de Acesso (RBAC)

Perfil	Permissões
Admin	Acesso total ao sistema, gestão de usuários e configurações
Gestor Facilities	CRUD de demandas, fornecedores, propostas, OS; aprovação nível 1
Aprovador Financeiro	Aprovação nível 2 (financeiro), visualização de orçamento
Diretoria	Aprovação nível 3 (alçada máxima), dashboard executivo
Solicitante	Abertura de demandas, acompanhamento do próprio pedido
Fornecedor	Envio de propostas, acompanhamento de OS atribuídas

5. API — Endpoints

5.1 Autenticação (`/api/auth`)

Método	Rota	Descrição
POST	`/api/auth/login`	Login com email/senha → JWT
POST	`/api/auth/refresh`	Renovar token
POST	`/api/auth/logout`	Invalidar token
GET	`/api/auth/me`	Dados do usuário logado
POST	`/api/auth/forgot-password`	Solicitar reset de senha
POST	`/api/auth/reset-password`	Resetar senha com token

5.2 Usuários (`/api/users`)

Método	Rota	Descrição
GET	`/api/users`	Listar usuários (paginado)
GET	`/api/users/:id`	Detalhes do usuário
POST	`/api/users`	Criar usuário
PATCH	`/api/users/:id`	Atualizar usuário
DELETE	`/api/users/:id`	Desativar usuário
GET	`/api/perfis`	Listar perfis

5.3 Locais (`/api/locais`)

Método	Rota	Descrição
GET	`/api/locais`	Listar locais
GET	`/api/locais/:id`	Detalhes do local
POST	`/api/locais`	Criar local

Método	Rota	Descrição
PATCH	`/api/locais/:id`	Atualizar local
DELETE	`/api/locais/:id`	Desativar local

5.4 Centros de Custo (`/api/centros-custo`)

Método	Rota	Descrição
GET	`/api/centros-custo`	Listar centros de custo
GET	`/api/centros-custo/:id`	Detalhes
POST	`/api/centros-custo`	Criar centro de custo
PATCH	`/api/centros-custo/:id`	Atualizar
DELETE	`/api/centros-custo/:id`	Desativar

5.5 Categorias (`/api/categorias`)

Método	Rota	Descrição
GET	`/api/categorias`	Listar categorias
GET	`/api/categorias/:id`	Detalhes
POST	`/api/categorias`	Criar categoria
PATCH	`/api/categorias/:id`	Atualizar
DELETE	`/api/categorias/:id`	Desativar

5.6 Fornecedores (`/api/fornecedores`)

Método	Rota	Descrição
GET	`/api/fornecedores`	Listar fornecedores
GET	`/api/fornecedores/:id`	Detalhes com certidões
POST	`/api/fornecedores`	Cadastrar fornecedor
PATCH	`/api/fornecedores/:id`	Atualizar fornecedor
DELETE	`/api/fornecedores/:id`	Desativar
POST	`/api/fornecedores/:id/certidoes`	Upload de certidão
GET	`/api/fornecedores/:id/certidoes`	Listar certidões
DELETE	`/api/fornecedores/:id/certidoes/:certidaoId`	Remover certidão

5.7 Demandas (`/api/demandas`)

Método	Rota	Descrição
GET	`/api/demandas`	Listar demandas (filtros: status, local, categoria, período)

Método	Rota	Descrição
GET	`/api/demandas/:id`	Detalhes completos da demanda
POST	`/api/demandas`	Criar nova demanda
PATCH	`/api/demandas/:id`	Atualizar demanda
DELETE	`/api/demandas/:id`	Cancelar demanda
POST	`/api/demandas/:id/itens`	Adicionar item de linha
PATCH	`/api/demandas/:id/itens/:itemId`	Atualizar item
DELETE	`/api/demandas/:id/itens/:itemId`	Remover item
POST	`/api/demandas/:id/enviar-cotacao`	Enviar para cotação
GET	`/api/demandas/:id/comparativo`	Comparativo de propostas

5.8 Propostas (`/api/propostas`)

Método	Rota	Descrição
GET	`/api/propostas`	Listar propostas
GET	`/api/propostas/:id`	Detalhes da proposta
POST	`/api/propostas`	Submeter proposta (fornecedor)
PATCH	`/api/propostas/:id`	Atualizar proposta
DELETE	`/api/propostas/:id`	Retirar proposta
POST	`/api/propostas/:id/aceitar`	Aceitar proposta
POST	`/api/propostas/:id/rejeitar`	Rejeitar proposta
POST	`/api/propostas/:id/ocr`	Processar OCR do arquivo

5.9 Orçamento (`/api/orcamento`)

Método	Rota	Descrição
GET	`/api/orcamento`	Orçamento geral (filtros: ano, local, centro_custo)
GET	`/api/orcamento/:id`	Detalhes da linha orçamentária
POST	`/api/orcamento`	Criar linha orçamentária
PATCH	`/api/orcamento/:id`	Atualizar valores
GET	`/api/orcamento/resumo`	Resumo planejado vs realizado
GET	`/api/orcamento/projecao`	Projeção de gastos

5.10 Workflow (`/api/workflow`)

Método	Rota	Descrição
GET	`/api/workflow/pendentes`	Aprovações pendentes do usuário logado
GET	`/api/workflow/demanda/:demandaId`	Histórico de aprovações da demanda
POST	`/api/workflow/aprovar`	Aprovar etapa
POST	`/api/workflow/rejeitar`	Rejeitar etapa
POST	`/api/workflow/devolver`	Devolver para correção
GET	`/api/workflow/alçadas`	Consultar alçadas configuradas

5.11 Dashboard (`/api/dashboard`)

Método	Rota	Descrição
GET	`/api/dashboard/indicadores`	KPIs gerais
GET	`/api/dashboard/demandas-por-status`	Demandas agrupadas por status
GET	`/api/dashboard/gastos-por-local`	Gastos por unidade
GET	`/api/dashboard/gastos-por-categoria`	Gastos por categoria
GET	`/api/dashboard/evolucao-mensal`	Série temporal de gastos
GET	`/api/dashboard/top-fornecedores`	Ranking de fornecedores
GET	`/api/dashboard/sla`	Indicadores de SLA

5.12 Ordens de Serviço (`/api/ordens-servico`)

Método	Rota	Descrição
GET	`/api/ordens-servico`	Listar OS
GET	`/api/ordens-servico/:id`	Detalhes da OS
POST	`/api/ordens-servico`	Emitir OS
PATCH	`/api/ordens-servico/:id`	Atualizar OS
POST	`/api/ordens-servico/:id/iniciar`	Marcar início da execução
POST	`/api/ordens-servico/:id/concluir`	Marcar conclusão
POST	`/api/ordens-servico/:id/avaliar`	Avaliar serviço prestado

5.13 Alertas (`/api/alertas`)

Método	Rota	Descrição
GET	`/api/alertas`	Listar alertas do usuário
PATCH	`/api/alertas/:id/lido`	Marcar como lido
DELETE	`/api/alertas/:id`	Remover alerta

Total: 68 endpoints

6. Autenticação e Autorização

6.1 Fluxo JWT

1. Usuário faz POST `/api/auth/login` com email e senha
2. Backend valida credenciais e retorna `{ accessToken, refreshToken }`
3. Frontend armazena tokens e envia `Authorization: Bearer <accessToken>` em todas as requests
4. Token expira em 1h; refresh token expira em 7d
5. Frontend usa `/api/auth/refresh` para renovar automaticamente

6.2 Guards NestJS

```
@UseGuards(JwtAuthGuard, RolesGuard)
@Roles('admin', 'gestor_facilities')
@Get('demandas')
findAll() { ... }
```

- **JwtAuthGuard**: valida o token JWT
- **RolesGuard**: verifica se o perfil do usuário está na lista de roles permitidas
- **@CurrentUser()**: decorator customizado que injeta o usuário logado

7. Workflow de Aprovação

7.1 Máquina de Estados

```
RASCUNHO → EM_ESCOPO → EM_COTAÇÃO → PROPOSTAS_RECEBIDAS
    → EM_COMPARAÇÃO → AGUARDANDO_APROVAÇÃO → APROVADA
    → OS_EMITIDA → EM_EXECUÇÃO → CONCLUÍDA → AVALIADA

Estados alternativos:
    AGUARDANDO_APROVAÇÃO → REJEITADA
    AGUARDANDO_APROVAÇÃO → DEVOLVIDA → EM_ESCOPO
    Qualquer estado → CANCELADA
```

7.2 Alçadas de Aprovação

Faixa de Valor	Aprovador
Até R$ 5.000	Gestor Facilities
R$ 5.001 — R$ 50.000	Gestor Facilities + Aprovador Financeiro
R$ 50.001 — R$ 200.000	Gestor + Financeiro + Diretoria
Acima de R$ 200.000	Gestor + Financeiro + Diretoria + CEO

Cada etapa gera um registro em `workflow_aprovacao` com timestamp, aprovador e comentário.

8. Como Rodar Localmente

8.1 Pré-requisitos

- Node.js 22.x
- npm 10.x
- Git

8.2 Backend

```
cd backend
cp .env.example .env          # Ajustar variáveis
npm install
npm run build
npm run migration:run         # Criar tabelas
npm run seed                  # Dados iniciais
npm start                     # Porta 3000
```

8.3 Frontend

```
cd frontend
cp .env.example .env          # VITE_API_URL=http://localhost:3000/api
npm install
npm run dev                   # Porta 5173
```

8.4 Acesso Inicial

- **Admin:** admin@hefesto.com.br / admin123
- **Swagger:** http://localhost:3000/api/docs

9. Deploy em Produção

9.1 Infraestrutura

- **Servidor:** DigitalOcean Droplet (Ubuntu 24.04, 2 vCPU, 4GB RAM)
- **Proxy reverso:** Nginx
- **SSL:** Let’s Encrypt (Certbot)

- **Processo:** PM2 (backend) / Nginx serve build estático (frontend)
- **BD:** PostgreSQL 16

9.2 Nginx Config

```
server {
    listen 443 ssl;
    server_name hefesto.exemplo.com.br;

    ssl_certificate /etc/letsencrypt/live/hefesto.exemplo.com.br/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/hefesto.exemplo.com.br/privkey.pem;

    # Frontend (build estático)
    location / {
        root /var/www/hefesto/frontend/dist;
        try_files $uri $uri/ /index.html;
    }

    # API Backend
    location /api {
        proxy_pass http://127.0.0.1:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

9.3 PM2

```
cd /var/www/hefesto/backend
pm2 start dist/main.js --name hefesto-api
pm2 save
pm2 startup
```

10. Variáveis de Ambiente

Backend (`.env`)

Variável	Descrição	Exemplo
`NODE_ENV`	Ambiente	`production`
`PORT`	Porta da API	`3000`
`DB_TYPE`	Tipo de banco	`postgres`
`DB_HOST`	Host do banco	`localhost`
`DB_PORT`	Porta do banco	`5432`
`DB_USERNAME`	Usuário do banco	`hefesto`
`DB_PASSWORD`	Senha do banco	`***`

Variável	Descrição	Exemplo
DB_DATABASE	Nome do banco	hefesto_prod
JWT_SECRET	Chave secreta JWT	minha-chave-secreta-256bits
JWT_EXPIRATION	Tempo de expiração do access token	3600s
JWT_REFRESH_EXPIRATION	Tempo de expiração do refresh token	7d
CORS_ORIGIN	Origem permitida	https://hefesto.exemplo.com.br
OCR_API_KEY	Chave da API de OCR	***
SMTP_HOST	Servidor SMTP	smtp.gmail.com
SMTP_PORT	Porta SMTP	587
SMTP_USER	Usuário SMTP	noreply@hefesto.com.br
SMTP_PASS	Senha SMTP	***

Frontend (`.env`)

Variável	Descrição	Exemplo
VITE_API_URL	URL base da API	https://hefesto.exemplo.com.br/api
VITE_APP_NAME	Nome da aplicação	HEFESTO

Documento gerado automaticamente — HEFESTO v1.0