Setup Completo de Infraestrutura - BestStag v9.1

Resumo da Implementação

O setup completo de infraestrutura de desenvolvimento foi implementado com sucesso para o projeto BestStag v9.1. Todos os arquivos de configuração, Docker, CI/CD e monitoramento foram criados e estão prontos para uso.

Arquivos Criados

Docker & Containerização

- `Dockerfile.backend` - Container Python/FastAPI
- `Dockerfile.frontend` - Container React/Nginx
- `docker-compose.yml` - Orquestração completa dos serviços
- `docker-compose.dev.yml` - Configurações para desenvolvimento
- `.dockerignore` - Otimização de build

⚙ Configurações de Ambiente

- `.env.example` - Template de variáveis de ambiente
- `config/requirements_backend.txt` - Dependências específicas do backend
- `config/logging_config.py` - Sistema de logs estruturados
- `config/health_checks.py` - Monitoramento de saúde
- `config/init.sql` - Inicialização do banco PostgreSQL

CI/CD Pipeline

- `.github/workflows/ci-cd.yml` - Pipeline principal
- `.github/workflows/security.yml` - Verificações de segurança

Backend Application

- `src/backend/app.py` - Aplicação FastAPI principal
- `src/backend/__init__.py` - Inicialização do package
- `src/frontend/package.json` - Configuração React atualizada

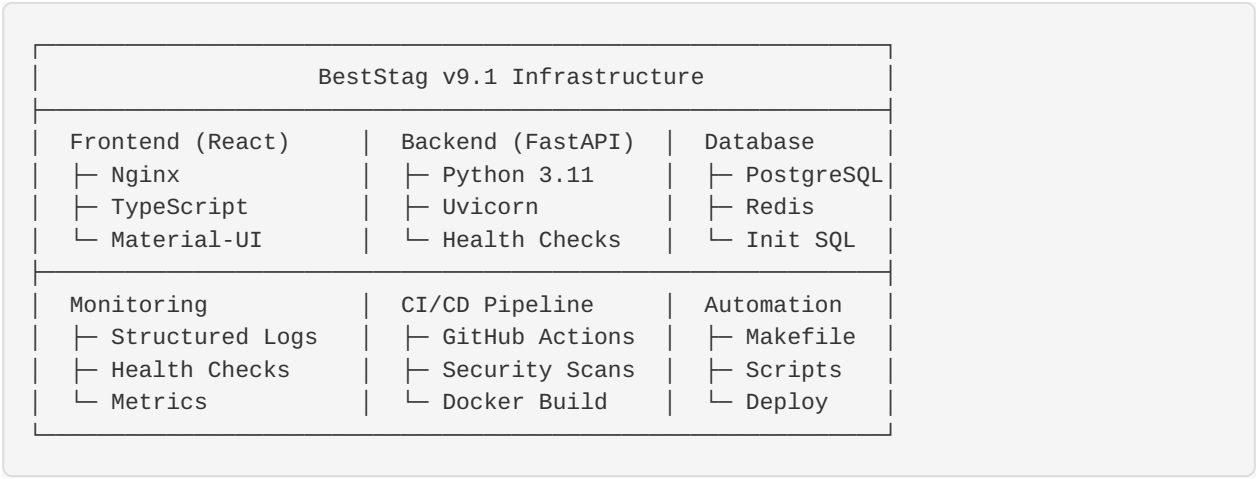
Automação & Scripts

- `Makefile` - Comandos de automação (30+ comandos)
- `scripts/setup.sh` - Setup automático inicial
- `scripts/deploy.sh` - Deploy automatizado
- `scripts/monitor.sh` - Monitoramento contínuo

Documentação

- `README.md` - Documentação principal atualizada

Arquitetura Implementada



Serviços Configurados

Core Services

- **Backend**: FastAPI + Python 3.11 (porta 8000)
- **Frontend**: React + Nginx (porta 3000)
- **Database**: PostgreSQL 15 (porta 5432)
- **Cache**: Redis 7 (porta 6379)
- **Workflows**: N8N (porta 5678)

Development Services

- **Adminer**: Interface de banco (porta 8080)
- **Redis Commander**: Interface Redis (porta 8081)
- **MailHog**: Email testing (porta 8025)

Funcionalidades Implementadas

Monitoramento

- Health checks automáticos para todos os serviços
- Logs estruturados em JSON
- Métricas de sistema (CPU, memória, disco)
- Alertas configuráveis
- Relatórios de status

Segurança

- Scans automáticos de vulnerabilidades
- Verificação de dependências
- Análise de código estático
- Container security scanning
- Secrets management

CI/CD

- Testes automatizados (backend + frontend)

- Build e push de imagens Docker
- Deploy automático por ambiente
- Quality gates
- Security scanning

Desenvolvimento

- Hot reload para desenvolvimento
- Debugging configurado
- Linting e formatação automática
- Type checking
- Coverage reports

Comandos Principais

```
# Setup inicial
./scripts/setup.sh

# Desenvolvimento
make dev                 # Iniciar ambiente completo
make test                # Executar todos os testes
make lint                # Verificar código
make format              # Formatar código

# Docker
make docker-build        # Build das imagens
make docker-up           # Subir serviços
make docker-down         # Parar serviços
make docker-logs         # Ver logs

# Monitoramento
make health              # Verificar saúde
make status              # Status do projeto
./scripts/monitor.sh     # Monitoramento contínuo

# Deploy
./scripts/deploy.sh staging     # Deploy staging
./scripts/deploy.sh production  # Deploy produção

# Banco de dados
make backup-db           # Backup
make db-migrate          # Migrações
make db-reset            # Reset completo
```

URLs de Acesso

Desenvolvimento

- **Frontend**: http://localhost:3000
- **Backend API**: http://localhost:8000
- **API Docs**: http://localhost:8000/docs
- **Health Check**: http://localhost:8000/health
- **N8N**: http://localhost:5678

- **Adminer**: http://localhost:8080
- **Redis Commander**: http://localhost:8081
- **MailHog**: http://localhost:8025

Configuração Necessária

1. Variáveis de Ambiente

```
cp .env.example .env
# Editar .env com suas configurações
```

2. Chaves de API

- Configurar `ABACUS_API_KEY` no .env
- Configurar webhooks do N8N
- Configurar credenciais de banco se necessário

3. Dependências do Sistema

- Docker & Docker Compose
- Python 3.11+
- Node.js 20+
- Make (para comandos automatizados)

Próximos Passos

1. **Configurar Ambiente**:
   `bash`
   `./scripts/setup.sh`
2. **Iniciar Desenvolvimento**:
   `bash`
   `make dev`
3. **Verificar Saúde**:
   `bash`
   `make health`
4. **Executar Testes**:
   `bash`
   `make test`
5. **Deploy**:
   `bash`
   `./scripts/deploy.sh staging`

Benefícios Implementados

Para Desenvolvimento

- ⚡ Setup automático em minutos
- Hot reload para mudanças rápidas

- Testes automatizados
- Monitoramento em tempo real
- Debugging facilitado

Para Produção

- Deploy automatizado
- Monitoramento completo
- Segurança integrada
- Containerização otimizada
- CI/CD robusto

Para Operações

- Métricas detalhadas
- Alertas configuráveis
- Logs estruturados
- Automação completa
- Documentação abrangente

Status Final

SETUP COMPLETO DE INFRAESTRUTURA IMPLEMENTADO COM SUCESSO!

O projeto BestStag v9.1 agora possui uma infraestrutura de desenvolvimento e produção completa, moderna e robusta, pronta para suportar o desenvolvimento e operação do assistente virtual inteligente.

Data de Implementação: 04 de Junho de 2025
Versão: BestStag v9.1
Status: Concluído
Próxima Fase: Desenvolvimento de funcionalidades (Semana 3-4)