🚀 Deploy Instructions - Orchestrator Agent

✅ Status: READY FOR PRODUCTION DEPLOY

O Orchestrator Agent foi **completamente corrigido** e está pronto para deploy no Render!

🔧 Problemas Técnicos Resolvidos

- ✅ **LangGraph Schema Issues**: Corrigido OrchestratorState com campos Optional
- ✅ **Node Functions**: Funções agora retornam Dict updates corretamente
- ✅ **Graph Configuration**: Removido nó inalcançável, StateGraph usando dict
- ✅ **Server Integration**: Campo messages adicionado ao estado inicial
- ✅ **All Endpoints Working**: `/task` e `/status/{task_id}` funcionando perfeitamente

🧪 Testes Locais Realizados

```
# ✅ Graph Creation
Graph created successfully!

# ✅ Task Execution
POST /task → {"task_id": "6c4bde4b-f604-4151-b6c5-213d338ce308", "status": "running"}

# ✅ Status Check
GET /status/6c4bde4b-f604-4151-b6c5-213d338ce308 → {"status": "completed"}
```

🌐 Deploy no Render

Opção 1: Deploy Automático via GitHub (Recomendado)

1. **Conectar Repositório**
   - Acesse Render Dashboard (https://dashboard.render.com)
   - Clique em “New +” → “Web Service”
   - Conecte o repositório: `https://github.com/paranhospr/web-next`
   - Branch: `main` (após merge do PR #3)

2. **Configuração Automática**
   - O Render detectará automaticamente o `render.yaml`
   - Nome do serviço: `orchestrator-agent`
   - Build Command: `pip install -r requirements.txt`
   - Start Command: `uvicorn app.server:app --host 0.0.0.0 --port 8080`

Opção 2: Deploy Manual

1. **Criar Web Service**
   - Runtime: Python
   - Build Command: `pip install -r requirements.txt`
   - Start Command: `uvicorn app.server:app --host 0.0.0.0 --port 8080`

🔑 Variáveis de Ambiente Obrigatórias

Configure estas variáveis no painel do Render:

Essenciais

```
AGENT_API_KEY=your-secret-api-key-here
GITHUB_TOKEN=ghp_your_github_token_here
GITHUB_OWNER=paranhospr
GITHUB_REPO=web-next
```

Integrações (Configure conforme necessário)

```
# Render Deploy Hooks
RENDER_DEPLOY_HOOK_API=https://api.render.com/deploy/srv-your-api-service-id
RENDER_DEPLOY_HOOK_WEB=https://api.render.com/deploy/srv-your-web-service-id

# Cloudflare DNS
CLOUDFLARE_API_TOKEN=your-cloudflare-api-token
CLOUDFLARE_ZONE_ID=your-cloudflare-zone-id

# Supabase Database
SUPABASE_URL=https://your-project.supabase.co
SUPABASE_SERVICE_ROLE=your-supabase-service-role-key

# Google Calendar (Service Account)
GOOGLE_PROJECT_ID=your-google-project-id
GOOGLE_CLIENT_EMAIL=your-service-account@your-project.iam.gserviceaccount.com
GOOGLE_PRIVATE_KEY="-----BEGIN PRIVATE KEY-----\nYour private key here\n-----END
PRIVATE KEY-----\n"
GOOGLE_CALENDAR_ID=your-calendar-id@group.calendar.google.com
```

Opcionais (Valores padrão funcionam)

```
AUTO_FIX=false
MAX_FIX_ATTEMPTS=3
ENABLE_HUMAN_GATE=false
ENVIRONMENT=production
PORT=8080
DATABASE_PATH=/tmp/orchestrator.db
VERIFY_TIMEOUT=30
VERIFY_MAX_RETRIES=3
HUMAN_GATE_TIMEOUT=300
HUMAN_GATE_AUTO_APPROVE=false
```

🧪 Testando o Deploy

Após o deploy, teste os endpoints:

1. Health Check

```
curl https://your-service.onrender.com/
```

2. Task Execution

```
curl -X POST https://your-service.onrender.com/task \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer your-secret-api-key-here" \
  -d '{"type": "verify", "targets": ["https://httpbin.org/status/200"]}'
```

3. Status Check

```
curl -H "Authorization: Bearer your-secret-api-key-here" \
  https://your-service.onrender.com/status/{task_id}
```

📋 Próximos Passos

1. **Merge PR #3** para branch main
2. **Deploy no Render** usando as instruções acima
3. **Configurar variáveis de ambiente** no painel do Render
4. **Testar endpoints** conforme exemplos acima
5. **Integrar com seus sistemas** usando a API REST

🎯 URL Final

Após o deploy, sua URL será algo como:

```
https://orchestrator-agent.onrender.com
```

🔒 Segurança

- ✅ Autenticação Bearer Token obrigatória
- ✅ Validação de entrada com Pydantic
- ✅ Logs estruturados para monitoramento
- ✅ Tratamento de erros robusto

Status: ✅ PRONTO PARA PRODUÇÃO
Última atualização: 25/09/2025 18:32 UTC
Commit: 24b8701 - Fix LangGraph schema issues