■ INSTRUÇÕES DO PROCESSO DE AGENDAMENTO TELENORDESTE

Última Atualização: 2024
Workspace: agendamentos_telenordeste
Status: Em desenvolvimento (95% funcional - identifica 4 dias com vagas: 16, 23, 30, 31)

■ OBJETIVO PRINCIPAL

Automatizar o processo de agendamento de consultas médicas no portal **TeleNordeste** integrando com o **Notion** para gerenciar tarefas de agendamento.

■ ARQUITETURA DO SISTEMA

Componentes Principais:

1. **Notion Database** - Fonte das tarefas de agendamento
2. **Playwright** - Automação web para navegar no site
3. **Script Python** - Orquestrador do processo (agendador_final_corrigido.py)

Fluxo Geral:

```
Notion (Tarefas) → Script Python → Playwright → Site TeleNordeste → Confirmação → Notion (Atualização)
```

■ ESTRUTURA DO NOTION

Database de Tarefas:

- **Database ID:** [Configurado em variável de ambiente/código]
- **Propriedades Necessárias:**

- `Nome da tarefa` ou `Nome` (Title)
- `Status` (Select) - Valores: "Não iniciado", "Em andamento", "Concluído"
- `Descrição` (Rich Text) - Contém dados do paciente

Formato da Descrição:

```
Nome: [Nome Completo do Paciente]
CPF: [000.000.000-00]
Especialidade: [Nome da Especialidade]
Motivo da Consulta: [Descrição]
ACS: [Nome do Agente Comunitário de Saúde]
Tipo: Adulto ou Infantil
```

Status de Tarefas:

- **Não iniciado** → Tarefa pendente de agendamento
- **Em andamento** → Agendamento em processo
- **Concluído** → Agendamento finalizado

■ PROCESSO NO SITE TELENORDESTE

***URL Base:** `https://www.telenordeste.com.br/agendamento`*

Estrutura do Site:

```
Homepage Agendamento
■■■ Agenda Adulto (Botão)
■    ■■■ Seleção de Especialidade
■    ■■■ Calendário de Datas
■    ■■■ Grade de Horários
■    ■■■ Formulário de Agendamento
■
■■■ Agenda Infantil (Botão)
    ■■■ Seleção de Especialidade
    ■■■ Calendário de Datas
    ■■■ Grade de Horários
    ■■■ Formulário de Agendamento
```

■ FLUXO DETALHADO DO AGENDAMENTO

ETAPA 1: BUSCAR TAREFAS NO NOTION

```
def buscar_tarefas_nao_iniciadas(client: Client) -> list
```

Ações:

1. Conecta ao Notion via API
2. Busca tarefas com `Status = "Não iniciado"`
3. Extrai dados da descrição:

- Nome do paciente
- CPF
- Especialidade desejada
- Motivo da consulta
- ACS responsável
- Tipo (Adulto/Infantil)

4. Retorna lista de tarefas para processar

Tratamento de Dados:

- Infere tipo (Adulto/Infantil) baseado na especialidade se não especificado
- Especialidades infantis: triagem, pediatria, neuropediatria, psiquiatria infantil, etc.

ETAPA 2: NAVEGAR PARA AGENDA

```
def navegar_para_agenda(page: Page, tipo: str) -> bool
```

Ações:

1. Acessa `https://www.telenordeste.com.br/agendamento`
2. Identifica o tipo de agenda (Adulto/Infantil)
3. Clica no botão correspondente:

- **Adulto:** Botão "Agenda Adulto"
- **Infantil:** Botão "Agenda Infantil"

4. Aguarda carregamento da página de agendamento

Seletores Possíveis:
- Link/botão com texto "Agenda Adulto" ou "Agenda Infantil"
- Elementos `<a>` ou `<button>` com classes específicas

ETAPA 3: SELECIONAR ESPECIALIDADE

```
def selecionar_especialidade(page: Page, especialidade: str) -> bool
```

Ações:
1. Localiza campo/dropdown de especialidades
2. Procura pela especialidade desejada
3. Seleciona a especialidade
4. Aguarda atualização do calendário

Tipos de Componentes:
- `<select>` dropdown
- Lista clicável de especialidades
- Campo de busca com autocompletar

Especialidades Comuns:
- **Adulto:** Clínica Geral, Cardiologia, Dermatologia, Psiquiatria, etc.
- **Infantil:** Pediatria, Neuropediatria, Psiquiatria Infantil, etc.

ETAPA 4: BUSCAR HORÁRIOS DISPONÍVEIS

```
def buscar_horarios_disponiveis(page: Page) -> dict
```

Ações:
1. Analisa o calendário exibido
2. Identifica datas disponíveis (geralmente destacadas visualmente)
3. Para cada data disponível:

- Clica na data
- Captura horários disponíveis
- Armazena em dicionário `{data: [horarios]}`

4. Retorna mapa de disponibilidade

Estrutura do Calendário:
- Componente visual com dias do mês
- Dias disponíveis: classe CSS específica, cor diferente, clicável
- Dias indisponíveis: desabilitados, acinzentados

Exemplo de Retorno:
```
{
    "2024-01-16": ["08:00", "09:00", "14:00"],
    "2024-01-23": ["10:00", "15:00"],
    "2024-01-30": ["08:30", "11:00"],
    "2024-01-31": ["09:00", "13:00"]
}
```

Status Atual: ■ Funcional - identifica 4 dias com vagas (16, 23, 30, 31)

ETAPA 5: PREENCHER FORMULÁRIO

```
def preencher_formulario(page: Page, dados_paciente: dict, data: str, horario: str) -> bool
```

Ações:
1. Seleciona data e horário escolhidos
2. Preenche formulário com dados do paciente:

- Nome completo
- CPF (com ou sem formatação)

- Data de nascimento
- Telefone
- Email (se necessário)
- Convênio/Particular
- Motivo da consulta
    3. Valida campos obrigatórios
    4. Prepara para confirmação

Campos Comuns:

```
<input name="nome" type="text" />
<input name="cpf" type="text" />
<input name="data_nascimento" type="date" />
<input name="telefone" type="tel" />
<textarea name="motivo"></textarea>
```

ETAPA 6: CLICAR EM RESERVAR*

```
def clicar_reservar(page: Page) -> bool
```

Ações:

1. Localiza botão de confirmação/reserva
2. Clica no botão
3. Aguarda processamento

Seletores Possíveis:

- Botão com texto "Reservar", "Confirmar", "Agendar"
- `<button type="submit">`
- Elemento com classe específica

ETAPA 7: VERIFICAR CONFIRMAÇÃO*

```
def verificar_confirmacao(page: Page) -> bool
```

Ações:

1. Aguarda mensagem de confirmação
2. Verifica sucesso do agendamento:

- Mensagem "Agendamento realizado com sucesso"
- Código de confirmação
- Detalhes da consulta

    3. Captura informações da confirmação
    4. Retorna status (sucesso/falha)

Indicadores de Sucesso:

- Mensagem de sucesso visível
- Redirecionamento para página de confirmação
- Email/SMS de confirmação enviado

ETAPA 8: ATUALIZAR NOTION*

```
def atualizar_status_tarefa_completa(client: Client, page_id: str, resultado: dict) -> bool
```

Ações:

1. Atualiza status da tarefa no Notion:

- **Sucesso:** Status = "Concluído"
- **Falha:** Status = "Não iniciado" (para retentar)

    2. Adiciona informações na descrição:

- Data/hora agendada
- Código de confirmação

- Mensagens de erro (se houver)
- Timestamp da automação

■■ FUNÇÕES AUXILIARES

Logs:

- `log_info()` - Informações gerais
- `log_sucesso()` - Operações bem-sucedidas
- `log_erro()` - Erros e problemas
- `log_aviso()` - Avisos importantes

Conexão Notion:

- `conectar_notion()` - Estabelece conexão com API
- Token de autenticação necessário

■ CAPTURAS DE TELA DISPONÍVEIS

Agenda Adulto:

1. `agenda_adulto_01.png` - Tela inicial
2. `agenda_adulto_02_analise.png` - Interface analisada
3. `agenda_adulto_03_meio.png` - Processo intermediário
4. `agenda_adulto_04_completo.png` - Tela completa

Agenda Infantil:

1. `agenda_infantil_01.png` - Tela 1
2. `agenda_infantil_02.png` - Tela 2
3. `agenda_infantil_03.png` - Tela 3

Navegação:

1. `telenordeste_main.png` - Página principal
2. `telenordeste_voltando.png` - Navegação de retorno

■■ CONFIGURAÇÕES NECESSÁRIAS

Variáveis de Ambiente:

```
NOTION_TOKEN = "secret_xxxxxxxxxx"  # Token de integração do Notion
DATABASE_ID = "xxxxxxxxxx"           # ID do database de tarefas
```

Dependências Python:

```
playwright
notion-client
```

Playwright Setup:

```
playwright install chromium
```

■ TRATAMENTO DE ERROS

Erros Comuns:

1. **Especialidade não encontrada**
- Verifica variações de nome
- Tenta busca parcial
- Registra erro no Notion

2. **Nenhum horário disponível**
- Verifica próximos dias
- Limita busca a período configurado
- Notifica indisponibilidade

3. **Erro no preenchimento**
- Valida campos antes de enviar
- Captura mensagens de erro do site
- Retorna para correção

4. **Falha na confirmação**
- Verifica múltiplas vezes
- Captura screenshot do erro
- Registra para análise manual

■ STATUS ATUAL DO DESENVOLVIMENTO

■ Funcionalidades Implementadas:

- ■ Conexão com Notion API
- ■ Busca de tarefas não iniciadas
- ■ Navegação para agenda (Adulto/Infantil)
- ■ Identificação de dias com vagas (16, 23, 30, 31)
- ■ Sistema de logs detalhado
- ■ Atualização de status no Notion

■ Em Desenvolvimento:

- ■ Seleção automática de especialidade
- ■ Captura completa de horários disponíveis
- ■ Preenchimento do formulário
- ■ Confirmação do agendamento

■ Pendente:

- ■ Testes end-to-end completos
- ■ Tratamento de edge cases
- ■ Sistema de retry automático
- ■ Notificações de sucesso/falha
- ■ Relatórios de execução

■ PRÓXIMOS PASSOS SUGERIDOS

1. **Completar seleção de especialidade**
- Mapear todas especialidades disponíveis
- Criar dicionário de equivalências
- Implementar busca fuzzy

2. **Finalizar captura de horários**
- Testar em diferentes especialidades
- Validar formato de data/hora
- Implementar seleção inteligente (primeiro disponível vs preferência)

3. **Implementar preenchimento completo**
- Mapear todos campos do formulário
- Validar CPF e outros dados
- Adicionar verificações de segurança

4. **Testar confirmação**
- Validar múltiplos cenários de sucesso
- Capturar códigos de confirmação
- Implementar verificação dupla

5. **Adicionar features avançadas**
- Agendamento em lote
- Priorização de tarefas
- Reagendamento automático
- Dashboard de monitoramento

■ NOTAS IMPORTANTES

- O sistema está **95% funcional** na identificação de disponibilidade
- Precisa de ajustes finos no processo de preenchimento e confirmação
- Screenshots disponíveis para referência visual de cada etapa
- Código modular permite fácil manutenção e expansão

■ SEGURANÇA

- **Dados Sensíveis:** CPF, informações médicas (LGPD)
- **Credenciais:** Nunca versionar tokens no código
- **Logs:** Não registrar dados sensíveis completos
- **Acesso:** Apenas usuários autorizados devem executar o script

Última Atualização: 2024
Versão: 1.0 (95% Funcional)
Desenvolvido por: Agente AI Avançado