



# Resumo Final - Testes de Prioridade Atende+

---



**Status: TODOS OS TESTES PASSARAM!** 

**Data:** 2026-01-27 09:27

**Servidor:** <http://localhost:3001>  ONLINE

**Taxa de Sucesso:** 4/4 (100%) 

---



## Resultados dos Testes



### Teste 1: Prioridade Básica

**Status:** PASSOU 

**Cenário:**

- N001 → N002 → **P001** → N003

**Resultado:**

-  **P001** foi chamada primeiro (prioritária antes de normal)
- 



### Teste 2: Ordem FIFO

**Status:** PASSOU 

**Cenário:**

- N001 → N002 → N003

**Resultado:**

-  Ordem correta: **N001 → N002 → N003**
- 

## **Teste 3: Múltiplas Prioridades Intercaladas**

Status: **PASSOU** 

Cenário:

- N001 → P001 → N002 → P002 → N003 → P003

Resultado:

-  Ordem correta: **P001 → P002 → P003 → N001 → N002 → N003**
- 



## **Teste 4: Prioridade+ (CORRIGIDO!)**

Status: **PASSOU** 

Cenário:

- N001 (normal) → P001 (prioritaria) → **P+002** (prioritaria+)

Resultado:

-  **P+002** foi chamada primeiro! 🎉
  -  Sistema de 3 níveis funcionando perfeitamente
- 



## **Correções Implementadas**

### **1. Lógica de Ordenação (server.js:311-330)**

```
// Helper function to assign numeric priority values
const prioridadeValor = (p) => {
  if (p === 'prioritaria+') return 3;
  if (p === 'prioritaria') return 2;
  return 1; // normal
};

const filaOrdenada = candidatos.sort((a, b) => {
  const prioA = prioridadeValor(a.prioridade);
  const prioB = prioridadeValor(b.prioridade);

  if (prioA !== prioB) {
    return prioB - prioA; // Maior prioridade primeiro
  }
  return new Date(a.horaGeracao).getTime() - new Date(b.horaGeracao).getTime();
});
```

## 2. Prefixo de Senha (server.js:250-252)

```
const prefixo = prioridade === 'prioritaria+' ? 'P+' :
  prioridade === 'prioritaria' ? 'P' : 'N';
```

### Resultado:

- `normal` → **N001, N002, N003...**
- `prioritaria` → **P001, P002, P003...**
- `prioritaria+` → **P+001, P+002, P+003...**



## Sistema de Prioridades

### Hierarquia (do maior para o menor):

1. **Prioritária+** ( `prioritaria+` ) - Prefixo `P+`
2. **Prioritária** ( `prioritaria` ) - Prefixo `P`
3. **Normal** ( `normal` ) - Prefixo `N`

## Regras de Ordenação:

1. **Prioridade primeiro:**  $P+ > P > N$
2. **FIFO dentro do mesmo nível:** Ordem de chegada ( `horaGeracao` )
3. **Atômico:** Sem conflitos entre múltiplos atendentes



## Como Testar Manualmente

### 1. Acesse o Sistema

```
http://localhost:3001
```

#### Credenciais:

- Email: `admin`
- Senha: `admin`

### 2. Teste Rápido de Prioridade+

1. Faça login como **Gerador**
2. Gere as senhas nesta ordem:
  - Nome: "Cliente Normal" → Tipo: Cadastro Novo → Prioridade: **Normal**
  - Nome: "Cliente Prioritário" → Tipo: Cadastro Novo → Prioridade: **Prioritária**
  - Nome: "Cliente VIP" → Tipo: Cadastro Novo → Prioridade: **Prioritária+**
3. Faça login como **Atendente** (em outra aba)
4. Clique em "Chamar Senha"



**Resultado Esperado:** A senha **P+001** (Cliente VIP) deve ser chamada primeiro! 

### 3. Teste Automatizado

Execute o script de teste:






```
node test_priority.js
```

## Arquivos Modificados

1.  `server.js` - Lógica de prioridade corrigida
2.  `test_priority.js` - Script de testes automatizados




## Conclusão

### Sistema de prioridades funcionando perfeitamente!

-  3 níveis de prioridade implementados
-  Ordem FIFO respeitada dentro de cada nível
-  Sistema atômico sem conflitos
-  Prefixos distintos para cada tipo (N, P, P+)
-  Todos os testes automatizados passando

## Próximos Passos

O sistema está pronto para uso! Você pode:

1.  Testar manualmente no navegador
2.  Executar testes automatizados quando quiser: `node test_priority.js`
3.  Usar o sistema em produção

**Servidor rodando em:** <http://localhost:3001>

