

SPELL

Spec Programming & Execution Logic Language

Manual do Arquiteto

Versão 2.0.0

Marcos Fiore

1. O que é o SPELL

SPELL é uma linguagem de especificação técnica com compilador. Você escreve a spec do sistema em sintaxe estruturada, o spellcomp valida, audita e gera um conjunto de arquivos .md que o agente executor segue como contrato — sem inferências, sem ambiguidades, sem decisões arquiteturais delegadas à IA.

O fluxo completo com SPELL é:

Arquiteto escreve	→	spellcomp compila	→	Agente implementa
agenda.spell		Validação estrutural		-dados.md
		Auditoria semântica		-regras.md
		Detecção de gaps		-endpoints.md
		Geração de .md por módulo		-telas.md

O que diferencia o SPELL de qualquer documento de especificação manual é exatamente o spellcomp: um compilador que detecta contradições, referências mortas e gaps lógicos que revisão humana não garante capturar. A spec não é apenas escrita — ela é provada.

2. SPELL, BMAD e Spec-Driven Development

O problema que todas essas ferramentas tentam resolver

Pedir código diretamente para uma IA sem especificação prévia é a principal causa de retrabalho no desenvolvimento assistido por IA. A IA infere o que não foi dito, toma decisões arquiteturais sem autoridade para isso, e produz código que precisa ser corrigido — às vezes do zero.

O SPELL resolve esse problema na raiz.

BMAD: quando o processo também precisa de estrutura

O BMAD Method organiza o processo de desenvolvimento com IA. Ele define papéis (Analista, Arquiteto, Desenvolvedor), sequências de trabalho e agentes especializados que colaboram em cadeia para levar uma ideia até o código.

O BMAD não especifica sistemas — ele orquestra quem faz o quê e em que ordem.

Equipes que usam BMAD para organizar o processo e SPELL para especificar tecnicamente obtêm o melhor dos dois: processo estruturado com especificação precisa. Mas é importante entender que o BMAD não substitui o SPELL — sem uma especificação técnica formal, os agentes do BMAD trabalham sobre documentos em linguagem natural, sujeitos às mesmas ambiguidades que o Spec-Driven Development foi criado para eliminar.

O SPELL pode ser adotado independentemente do BMAD. O inverso não oferece as mesmas garantias.

OpenSpec e Spec Kit: mesma filosofia, menos profundidade

OpenSpec e Spec Kit implementam Spec-Driven Development (SDD) — a mesma filosofia do SPELL: especificação antes do código. Elas estruturam a intenção do desenvolvedor, quebram a spec em tarefas menores e orientam o agente a executar de forma controlada.

É uma abordagem correta. Mas há diferenças fundamentais em relação ao SPELL:

	OpenSpec / Spec Kit	SPELL
Formato	Documentos estruturados e templates	Linguagem formal com sintaxe própria
Validação	Revisão humana	Compilador com validação estrutural
Auditoria semântica	Não possui	Detecta gaps, contradições e referências mortas via IA
Output	Tarefas e histórias de usuário	Arquivos .md por domínio, otimizados para agentes
Público-alvo	Desenvolvedores	Arquitetos de sistema
Domínio	Genérico	Específico para arquitetura de software

A diferença central está na auditoria semântica. OpenSpec e Spec Kit transformam uma spec em tarefas — mas não questionam se a spec é logicamente consistente. O spellcomp questiona. Uma entidade com campo status sem regra de transição declarada, um endpoint sem resposta de erro para um recurso que pode não existir, uma tela que navega para outra tela não declarada — tudo isso é detectado antes que qualquer agente comece a trabalhar.

Equipes que já usam OpenSpec ou Spec Kit podem adotar o SPELL como camada anterior: a spec compilada e auditada pelo spellcomp alimenta o fluxo de tarefas dessas ferramentas como fonte da verdade imutável. Mas equipes que adotam o SPELL do zero não precisam de mais nada para ter um processo de desenvolvimento assistido por IA confiável e rastreável.

3. Sintaxe: hierarquia por numeração

O SPELL usa numeração hierárquica para inferir estrutura. Você pode escrever no Word e salvar como texto simples (.spell).

```
# Hierarquia
1.          → bloco raiz (nome fixo — ver tabela abaixo)
1.1.        → sub-bloco dentro do bloco raiz
1.1.1.      → sub-item dentro do sub-bloco

# Atributos
campo: valor          → atributo estruturado
campo: recomendar     → delegação à IA

# Instruções livres
- texto              → instrucoes implícitas

# Comentários
```

```
// texto          → ignorado pelo parser

# Modificadores inline
campo: tipo(N), modificador1, modificador2
```

Regras do parser:

- O parser reconhece blocos raiz pelo nome, não pelo número.
- A numeração não precisa ser sequencial — 2. pode vir antes de 1. sem erro.
- Linhas com - dentro de qualquer bloco são tratadas como instruções implícitas.
- A palavra adaptar sozinha em uma linha dentro de Wireframes indica que a IA pode adaptar o layout.

4. Blocos raiz

Número	Nome fixo	Obrigatório
1.	Projeto	Sim
2.	Banco de Dados	Recomendado
3.	Regras de Negócio	Recomendado
4.	Endpoints	Recomendado
5.	Telas	Recomendado
6.	Segurança	Opcional
7.	Serviços	Opcional

5. Bloco Projeto

```
1. Projeto
  nome: "AgendaFácil"
  versao: "1.0"
  descricao: "Sistema para clínicas agendarem consultas"
```

1.1. Stack

```
1.1. Stack
  frontend: React
  backend: Node.js
  banco: PostgreSQL
  autenticacao: contas com login e senha
  agente: claude-code
```

O campo agente controla o nome do arquivo de entrada gerado:

Valor	Arquivo gerado
claude-code (padrão)	CLAUDE.md
cursor	.cursor/rules
aider	CONVENTIONS.md
copilot	.github/copilot-instructions.md
nenhum	(nenhum arquivo de entrada)

1.2. Convenções

Declara restrições organizacionais — não convenções técnicas que a IA já conhece.

```
1.2. Convenções
  idioma-do-codigo: português
  - Todos os serviços devem ter sufixo Service
  - Métodos de busca devem começar com buscar
```

Campo	Descrição
idioma-do-codigo	Idioma dos identificadores no código gerado
- texto	Convenção organizacional em linguagem natural

1.3. Skills

Declara capacidades esperadas do agente executor. O spellcomp detecta skills adicionais automaticamente.

```
1.3. Skills
  - pdf-reading
  - xlsx
  - charts
```

Skills disponíveis no catálogo: pdf-reading, xlsx, charts, docx, frontend-design, oauth-integration, email, storage, websocket, queue.

6. Bloco Banco de Dados

2.0. Configuração

Sub-bloco especial com número fixo 2.0. para configurações globais do banco.

```
2. Banco de Dados
```

```
2.0. Configuração
  multitenancy: schema-por-tenant
  escopo: hibrido
  migracao: goose
  - Tabelas globais ficam no schema public
  - Tabelas de tenant usam prefixo tenant_{id}
```

Campo	Valores possíveis
multitenancy	schema-por-tenant, banco-por-tenant, coluna-tenant, nao
escopo	global, por-tenant, hibrido
migracao	flyway, liquibase, goose, etc.

2.N. Tabela NomeDaTabela

```
2.1. Tabela Medico
  id: inteiro, chave-primaria, auto-incremento
  nome: texto(100), obrigatorio
  crm: texto(20), unico, obrigatorio
  ativo: booleano, padrao: sim
```

Tipos disponíveis:

inteiro, decimal, decimal(P,S), texto, texto(N), booleano, data, data-hora, hora, arquivo, json

Modificadores:

chave-primaria, auto-incremento, unico, obrigatorio, opcional, padrao: valor, privado, indexado

Estrutura delegada à IA:

```
2.2. Tabela Paciente
  estrutura: recomendar
  - Paciente deve ter nome, data de nascimento e telefone
  - CPF é opcional mas deve ser único se informado
```

2.N.1. Relacionamentos

Usado quando a convenção de nomes não resolve (dois campos apontando para a mesma tabela).

```
2.4. Tabela Mensagem
  remetente_id: inteiro, obrigatorio
  destinatario_id: inteiro, obrigatorio
```

```
2.4.1. Relacionamentos
  remetente_id -> Usuario.id
  destinatario_id -> Usuario.id
```

7. Bloco Regras de Negócio

As regras de negócio são o coração da especificação. A IA não inventa regras — tudo que não estiver aqui será ignorado.

3.1. Regra HorárioDisponível

contexto: Agendamento de consulta

condicao: Médico já tem consulta no mesmo horário

acao: Recusar o agendamento com mensagem de horário indisponível

motivo: Política da clínica para reduzir conflitos de agenda

Campo	Obrigatório	Descrição
contexto	Sim	Em qual operação ou momento a regra se aplica
condicao	Sim	Quando exatamente a regra dispara
acao	Sim	O que deve acontecer
excecao	Não	Caso em que a regra não se aplica
motivo	Não	Por que esta regra existe

8. Bloco Endpoints

4.1. Endpoint AgendarConsulta

rota: POST /consultas

autenticacao: obrigatoria

regras: HorárioDisponível

4.1.1. Dados de Entrada

medico_id: inteiro, obrigatorio

paciente_id: inteiro, obrigatorio

data_hora: data-hora, obrigatorio

4.1.2. Respostas

201: Consulta agendada com sucesso

400: Horário indisponível para este médico

404: Médico ou paciente não encontrado

Campo	Descrição
rota	Método HTTP + caminho (ex: POST /consultas)
autenticacao	obrigatoria ou nao-requerida
regras	Nome(s) de regras declaradas em 3. Regras de Negócio

9. Bloco Telas

```
5.1. Tela AgendaDoMedico
  rota: /agenda/{medico_id}
  autenticacao: obrigatoria
  layout: recomendar
  - Exibir calendário semanal com horários ocupados e livres
  - Clicar em horário livre deve abrir formulário de agendamento

5.1.1. Wireframes
  mobile: wireframes/agenda-mobile.png
  desktop: wireframes/agenda-desktop.png
  adaptar
```

A palavra adaptar sozinha dentro de Wireframes indica que a IA pode adaptar o layout para mobile ou acessibilidade.

10. Bloco Segurança

O bloco Segurança é escrito em linguagem de intenção de negócio — não técnica. O arquiteto declara o que precisa ser protegido. A IA decide como implementar.

```
6. Segurança
  acesso: usuários precisam criar conta com apelido e senha
  - Login deve ser protegido contra tentativas automáticas de senha
  - Cadastro deve ser protegido contra criação em massa de contas
```

11. Bloco Serviços

```
7.1. Serviço LimpezaDeSessoes
  periodicidade: 24 horas
  horario: 03:00
  - Remover sessões de usuários expiradas há mais de 30 dias
  - Enviar relatório de limpeza por e-mail para o administrador
```

12. Comandos do spellcomp

```
spellcomp new      <nome>          # Cria arquivo .spell com template
spellcomp compile <arquivo.spell> # Compila e gera arquivos por módulo
spellcomp check   <arquivo.spell> # Valida sem gerar nada
spellcomp info    <arquivo.spell> # Exibe resumo do arquivo
spellcomp --version
```



```
spellcomp --help
```

Opções do compile:

```
--output <pasta/>    # Pasta de saída (padrão: ./<nome-do-projeto>/)
--semantic           # Executa auditoria semântica via API do Claude
```

Símbolos de diagnóstico:

Símbolo	Significado
✖ ERRO	Impede geração correta — deve ser corrigido
⚠ AVISO	Ambíguo — arquiteto deve revisar
ℹ INFO	Inferência aplicada — verificar se está correto
☑ OK	Tudo em ordem

13. Auditoria semântica

O auditor semântico é a camada mais poderosa do spellcomp. Após construir o AST, ele envia a spec para a API do Claude e recebe um relatório de diagnóstico.

O que o auditor detecta:

- Gaps: entidade com campo status sem regra que governe transições
- Contradições: duas regras com comportamentos conflitantes para o mesmo contexto
- Referências mortas: endpoints referenciando regras não declaradas; telas navegando para telas inexistentes
- Entidades órfãs: declaradas sem nenhum endpoint que as escreva
- Cobertura de erros: endpoint que busca por ID sem resposta 404 declarada

Ativar a auditoria:

```
spellcomp compile agenda.spell --semantic
```

A auditoria semântica é desativada por padrão. Use `--semantic` quando quiser validação semântica completa da spec antes de gerar os arquivos.

14. Autenticação com a API

O spellcomp nunca pede que você salve credenciais em arquivos do projeto. A autenticação segue uma cascata:

```
1. Variável de ambiente ANTHROPIC_API_KEY
   ↓
2. Credenciais do Claude Code (~/.claude/.credentials.json)
   ↓
3. Chave salva pelo spellcomp (~/.spell_api_key)
   ↓
   Prompt interativo → oferece salvar para uso futuro
```

Configuração mais simples (shell profile):

```
export ANTHROPIC_API_KEY=sk-ant-...
```

15. Output modular

O spellcomp compile gera uma pasta com arquivos separados por domínio:

```
agendafacil/
├─ CLAUDE.md                ← arquivo de entrada do agente
├─ agendafacil-projeto.md   ← stack, convenções, skills
├─ agendafacil-dados.md     ← banco de dados
├─ agendafacil-regras.md    ← regras de negócio
├─ agendafacil-endpoints.md ← endpoints
├─ agendafacil-telas.md    ← telas
├─ agendafacil-servicos.md  ← serviços (se declarado)
└─ agendafacil-seguranca.md ← segurança (se declarado)
```

O arquivo de entrada (CLAUDE.md ou equivalente) contém:

- Índice de todos os arquivos e o que cada um contém
- Skills requeridas e sugeridas
- Instruções de dependência entre módulos
- Avisos do compilador e relatório semântico (se houver)

Ao final de toda compilação, o spellcomp exibe uma tabela de referência com os nomes esperados por cada agente — útil para equipes que usam agentes diferentes.

16. Exemplo completo

```
1. Projeto
  nome: "AgendaFácil"
  versao: "1.0"
  descricao: "Sistema para clínicas agendarem consultas"

1.1. Stack
  frontend: React
```

backend: Node.js
banco: PostgreSQL
autenticacao: contas com login e senha
agente: claude-code

1.2. Convenções

idioma-do-codigo: português
- Todos os serviços devem ter sufixo Service

1.3. Skills

- xlsx

2. Banco de Dados

2.0. Configuração

multitenancy: nao
escopo: global

2.1. Tabela Medico

id: inteiro, chave-primaria, auto-incremento
nome: texto(100), obrigatorio
especialidade: texto(100), obrigatorio
crm: texto(20), unico, obrigatorio
ativo: booleano, padrao: sim

2.2. Tabela Paciente

estrutura: recomendar
- Paciente deve ter nome, data de nascimento e telefone
- CPF é opcional mas deve ser único se informado

2.3. Tabela Consulta

id: inteiro, chave-primaria, auto-incremento
medico_id: inteiro, obrigatorio
paciente_id: inteiro, obrigatorio
data_hora: data-hora, obrigatorio
status: texto, padrao: "Agendada"
observacoes: texto, opcional

3. Regras de Negócio

3.1. Regra HorarioDisponivel

contexto: Agendamento de consulta
condicao: Médico já tem consulta no mesmo horário
acao: Recusar o agendamento com mensagem de horário indisponível

3.2. Regra CancelamentoComAntecedencia

contexto: Cancelamento de consulta
condicao: Cancelamento com menos de 24 horas de antecedência
acao: Exibir aviso de política de cancelamento e pedir confirmação

motivo: Política da clínica para reduzir faltas de última hora

4. Endpoints

4.1. Endpoint AgendarConsulta

rota: POST /consultas
autenticacao: obrigatoria
regras: HorarioDisponivel

4.1.1. Dados de Entrada

medico_id: inteiro, obrigatorio
paciente_id: inteiro, obrigatorio
data_hora: data-hora, obrigatorio
observacoes: texto, opcional

4.1.2. Respostas

201: Consulta agendada com sucesso
400: Horário indisponível para este médico
404: Médico ou paciente não encontrado

5. Telas

5.1. Tela AgendaDoMedico

rota: /agenda/{medico_id}
autenticacao: obrigatoria

- Exibir calendário semanal com horários ocupados e livres
- Clicar em horário livre deve abrir formulário de agendamento
- Consultas canceladas devem aparecer riscadas, não removidas

5.1.1. Wireframes

mobile: wireframes/agenda-mobile.png
desktop: wireframes/agenda-desktop.png
adaptar

6. Segurança

acesso: usuários precisam criar conta com apelido e senha

- Login deve ser protegido contra tentativas automáticas de senha

7. Serviços

7.1. Serviço LimpezaDeSessoes

periodicidade: 24 horas
horario: 03:00

- Remover sessões de usuários expiradas há mais de 30 dias
- Enviar relatório por e-mail para o administrador

Sobre o autor

Empreendedor serial com mais de 30 anos de experiência em tecnologia, fundou e liderou uma das primeiras plataformas de e-mail marketing e automação de canais digitais do Brasil, construída do zero em 2005 e vendida a um dos maiores players nacionais de Customer Experience em 2022. Hoje comanda um grupo tecnológico através do qual desenvolve múltiplos projetos nas áreas de transformação digital, cloud computing e automação de processos, além de atuar como investidor em negócios de tecnologia. Ao longo da carreira, ajudou mais de 5 mil empresas brasileiras a se comunicarem melhor com seus clientes por meio da tecnologia.