Manual de Boas Práticas de Codificação

Projeto: Clínica Veterinária Animal Health Center

1. Introdução

Este documento estabelece as diretrizes de codificação adotadas no desenvolvimento do sistema *Animal Health Center*. O objetivo é garantir a legibilidade, manutenibilidade e escalabilidade do código, facilitando a colaboração entre os membros da equipe e futuras evoluções do software.

O projeto segue rigorosamente os padrões da comunidade PHP (PSR) e as convenções do framework Laravel 11.

2. Padrões de Estilo (PSR-12)

O código deve aderir à especificação **PSR-12 (Extended Coding Style Guide)**.

- **Identação:** Utilizar 4 espaços (não tabs).
- **Chaves:**
  - Classes e Métodos: A chave de abertura { deve estar na **próxima linha**.
  - Estruturas de controle (if, foreach): A chave de abertura { deve estar na **mesma linha**.
- **Visibilidade:** Todas as propriedades e métodos devem ter a visibilidade declarada (public, protected, private).

Exemplo (Aplicado no AgendamentoController.php):

```
class AgendamentoController extends Controller
{ // Chave na próxima linha (Classe)
    public function index()
    { // Chave na próxima linha (Método)
        if ($user->isCliente()) { // Chave na mesma linha (Controle)
            // ...
        }
    }
}
```

3. Convenções de Nomenclatura

A consistência nos nomes é crucial para a previsibilidade do código.

Elemento	Padrão	Idioma	Exemplo
Classes	PascalCase	Inglês (Preferencial) ou Português (Domínio)	AgendamentoController, User
Métodos	camelCase	Inglês (Verbo + Substantivo)	store, create, isAdmin
Variáveis	camelCase	Inglês ou Português (Descritivo)	$agendamentos, $duracaoHoras
Tabelas (DB)	snake_case (Plural)	Português (Domínio)	agendamentos, clientes
Colunas (DB)	snake_case	Português	data_hora, nome_completo
Rotas (Name)	kebab-case (dot notation)	Inglês/Português	agendamentos.index, profile.edit

4. Práticas de Legibilidade e Clareza

4.1. Métodos Helper no Model (Fat Models)

Evitar lógica condicional repetida nas Views ou Controllers. Encapsular verificações booleanas dentro do Model.

Aplicação no Projeto (User.php):

Em vez de verificar if ($user->perfil == 'Admin') espalhado pelo código, utilizamos:

```
// No Model User
public function isAdmin(): bool
{
    return $this->perfil === 'Admin';
}

// No uso (Controller/View)
if ($user->isAdmin()) { ... }
```

4.2. Eager Loading (Prevenção do Problema N+1)

Para garantir a performance e evitar centenas de consultas ao banco de dados ao listar registros, deve-se utilizar o carregamento antecipado (with) sempre que houver relacionamentos.

Aplicação no Projeto (AgendamentoController.php):

```
// CERTO: Carrega Pet, Cliente e Veterinário em apenas 2 queries
$agendamentos = Agendamento::with(['pet.cliente', 'veterinario'])->get();

// ERRADO: Faria uma query extra para cada linha da tabela na View
// $agendamentos = Agendamento::all();
```

4.3. Type Hinting e Retorno de Tipos

Sempre que possível, tipar os argumentos e retornos das funções (PHP 8+). Isso serve como documentação viva e previne erros de tipo.

Exemplo:

```
public function destroy(string $id) // Argumento tipado
{
    // ...
}
```

5. Organização e Arquitetura (MVC)

5.1. Controllers "Magros"

Os Controllers devem focar em receber a requisição e devolver a resposta. Regras de negócio complexas devem ser comentadas ou extraídas.

- **Validação:** A validação dos dados de entrada ($request->validate) deve ser feita no início do método store ou update para falhar precocemente (*Fail Fast*).

5.2. Rotas Nomeadas

Nunca utilizar URLs "hardcoded" (/agendamentos/create) no código. Utilizar sempre o helper route(). Isso permite alterar a URL sem quebrar a aplicação.

Aplicação no Projeto (web.php):

```
// Definição
Route::resource('agendamentos', AgendamentoController::class);
```

```
// Uso nas Views/Controllers
redirect()->route('agendamentos.index');
```

6. Comentários e Documentação

O código deve ser autoexplicativo, mas comentários são obrigatórios em blocos de lógica de negócio complexa ("Por que" foi feito, não "O que" foi feito).

Exemplo de Boa Prática (AgendamentoController.php):

```
// --- REGRA 3: DISPONIBILIDADE DO PET (NOVO) ---
// O mesmo pet não pode estar em dois lugares ao mesmo tempo
$conflitosPet = Agendamento::where('pet_id', $request->pet_id)
   ->whereDate('data_hora', $inicio->toDateString())
   // ...
```

7. Segurança

- **CSRF:** Todos os formulários POST, PUT, DELETE devem conter a diretiva @csrf.
- **Mass Assignment:** Todos os Models devem ter a propriedade $fillable definida para evitar injeção de dados maliciosos.
- **Middleware:** Rotas sensíveis devem estar protegidas pelo middleware auth e, quando necessário, pelo middleware admin.