CacheService - Padronização das chaves cache

Documentação - CacheService

Visão Geral

O **CacheService** é uma camada de abstração sobre o sistema de cache do Laravel que oferece:

- **Chaves estruturadas e legíveis** com nome, descrição e parâmetros
- **Geração determinística de hashes** para unicidade
- **Suporte a modo nativo** (compatível com Laravel)
- **Cascade delete** por família e usuário
- **CLI poderoso** para inspeção e gerenciamento

Estrutura das Chaves

Formato Físico (Redis)

```
laravel:{key_name:X, description:Y, {params:[{key:value}]}, hash:XXXXXX}
```

Exemplo real:

```
laravel:{key_name:user_get_my_courses_ids,description:lista_dos_cursos_do_usuario,{params:[{userid:215814}]},hash:0a46b3}
```

Componentes

Campo	Descrição	Exemplo
`key_name`	Nome da família/categoria do cache	`user_my_courses`
`description`	Descrição legível do que é cacheado	`cursos do usuário logado`
`params`	Parâmetros que tornam a chave única	`{userid:215814, page:1}`
`hash`	Hash MD5 dos 6 primeiros caracteres	`0a46b3`

Métodos Disponíveis

1. `setKey()` - Gerar chave física

Gera uma chave física determinística sem salvar no cache.

```
$cacheKey = $this->cacheService->setKey(
    name: "user_my_courses",
    description: "cursos do usuário logado",
    params: [
        'userId' => $userId,
        'page' => $page,
        'perPage' => $perPage,
        'search' => $search,
        'params' => $params
    ]
);
```

Resultado:

```
{key_name:user_my_courses, description:cursos_do_usuario_logado, {params:[{userid:215814},{page:1}]}, hash:5e49b2}
```

---

2. `remember()` - Cache com callback

Busca no cache ou executa callback se não existir.

Modo Nativo (Laravel padrão)

```
$key = $this->cacheService->setKey(
                'meeting_plan_simultaneous_rooms',
                'cache de número máximo de salas simultâneas por plano',
                ['plan_id' => $plan_id]
            );




$data = $this->cacheService->remember(
    $key,60 * 3, // TTL em segundos
  function () {
        return .....
    });
```

Modo Lógico (Estruturado)

```
$courses = $this->cacheService->remember(
    'user_my_courses',
    'cursos do usuário logado',
    [
        'userId' => $userId,
        'page' => $page,
        'perPage' => $perPage,
        'search' => $search,
        'params' => $params
    ],
    60 * 3, // TTL: 3 minutos
    function () use ($page, $perPage, $search, $params) {
        return $this->enrollmentService->getMyCourses(
            returnQuery: true,
            $page,
            $perPage,
            $search,
            $params
        )->toArray();
    }
);
```

Assinatura:

```
remember(
    string $name,           // Nome da família
    string $description,    // Descrição legível
    array $params,          // Parâmetros únicos
    int $ttl,               // TTL em segundos
    Closure $callback       // Função a executar se não houver cache
)
```

---

3. `rememberForever()` - Cache permanente

Igual ao `remember()`, mas sem expiração.

Modo Nativo

```
$key = $this->cacheService->setKey(
              'meeting_plan_simultaneous_rooms',
              'cache de número máximo de salas simultâneas por plano',
              ['plan_id' => $plan_id]
          );



$data = $this->cacheService->rememberForever(
    $key, function () {
        return Config::all();
    }
);
```

Modo Lógico

```
$coursesIds = $this->cacheService->rememberForever(
    'user_get_my_courses_ids',
    'lista dos cursos do usuario',
    [
        'userId' => $userId,
        'favorites' => $favoriteIds,
        'params' => $params,
```

```
    ],
    function () use ($favoriteIds) {
        return $this->enrollmentService
            ->getMyCourses(
                returnQuery: false,
                page: 1,
                perPage: 100,
                search: null,
                ['include_hidden_courses' => true, 'favorites' => $favoriteIds]
            )
            ->pluck(value: 'id')
            ->toArray();
    }
);
```

Assinatura:

```
rememberForever(
    string $name,           // Nome da família
    string $description,    // Descrição legível
    array $params,          // Parâmetros únicos
    Closure $callback       // Função a executar
)
```

---

4. `get()` - Buscar valor

Recupera um valor do cache.

Modo Nativo

```
$key = $this->cacheService->setKey(
                'meeting_plan_simultaneous_rooms',
                'cache de número máximo de salas simultâneas por plano',
                ['plan_id' => $plan_id]
            );
```

```
$value = $this->cacheService->get($key);
```

Modo Lógico

```
$configs = $this->cacheService->get(
    'user_global_configs',
    'configuracoes globais do usuario',
    ['user_id' => 123],
    $default = null
);
```

Assinatura:

```
get(
    string $name,
    string $description,
    array $params,
    mixed $default = null
)
```

---

5. `put()` - Salvar valor

Salva um valor no cache com TTL.

Modo Nativo

```
$key = $this->cacheService->setKey(
            'meeting_plan_simultaneous_rooms',
            'cache de número máximo de salas simultâneas por plano',
            ['plan_id' => $plan_id]
        );



$this->cacheService->put($key, $value, 60 * 5);
```

Modo Lógico

```
$this->cacheService->put(
    'user_my_courses',
    'chave cache dos cursos do usuario logado',
    ['userid' => 123, 'page' => 1],
    $coursesData,
    60 * 3 // 3 minutos
);
```

Assinatura:

```
put(
    string $name,
    string $description,
    array $params,
    mixed $value,
    int $ttl
)
```

---

6. `has()` - Verificar existência

Verifica se uma chave existe no cache.

Modo Nativo

```
$key = $this->cacheService->setKey(
                'meeting_plan_simultaneous_rooms',
                'cache de número máximo de salas simultâneas por plano',
                ['plan_id' => $plan_id]
            );



if ($this->cacheService->has($key)) {
    // exists
}
```

Modo Lógico

```
$exists = $this->cacheService->has(
    'user_my_courses',
```

```
    'chave cache dos cursos do usuario logado',
    ['userid' => 123, 'page' => 1]
);
```

---

7. `forget()` - Remover chave

Remove uma chave específica ou todas de uma família (cascade).

Modo Nativo

```
$key = $this->cacheService->setKey(
            'meeting_plan_simultaneous_rooms',
            'cache de número máximo de salas simultâneas por plano',
            ['plan_id' => $plan_id]
        );



$this->cacheService->forget($key);
```

Modo Lógico - Específico

```
// Remove apenas esta combinação específica
$this->cacheService->forget(
    'user_my_courses',
    'cursos do usuario',
    ['userid' => 123, 'page' => 1]
);
```

Modo Lógico - Cascade (todas as físicas da família)

```
// Remove TODAS as chaves físicas de 'user_my_courses' para o userId 123
$this->cacheService->forget(
    'user_my_courses',
    'cursos do usuario',
    ['userid' => 123],
    $cascade = true  // □□ Remove todas as variações
);
```

Cascade: Quando `true`, remove todas as chaves físicas que:

- Pertencem à mesma família (`key_name`)
- Possuem o mesmo identificador de usuário (`userId`, `user_id`, `auth`, etc.)

Assinatura:

```
forget(
    string $name,
    ?string $description,  // Pode ser null
    array $params,
    bool $cascade = false
)
```

Exemplos Práticos

Exemplo 1: Cache de listagem com paginação

```
public function getMyCourses($page, $perPage, $search, $params)
{
    $userId = auth()->id();

    $cacheKey = $this->cacheService->setKey(
        name: "user_my_courses",
        description: "cursos do usuário logado",
        params: [
            'userId' => $userId,
            'page' => $page,
            'perPage' => $perPage,
            'search' => $search,
            'params' => $params
        ]
    );

    $courses = $this->cacheService->remember(
        $cacheKey,
```

```
        60 * 3,
        function () use ($page, $perPage, $search, $params) {
            return $this->enrollmentService->getMyCourses(
                returnQuery: true,
                $page,
                $perPage,
                $search,
                $params
            )->toArray();
        }
    );

    return response()->json(['data' => $courses]);
}
```

Exemplo 2: Cache permanente de IDs

```
public function getMyCoursesIds($favoriteIds, $params)
{
    $userId = auth()->id();

    $myCoursesIds = $this->cacheService->rememberForever(
        'user_get_my_courses_ids',
        'lista dos cursos do usuario',
        [
            'userId' => $userId,
            'favorites' => $favoriteIds,
            'params' => $params,
        ],
        function () use ($favoriteIds) {
            return $this->enrollmentService
                ->getMyCourses(
                    returnQuery: false,
                    page: 1,
                    perPage: 100,
                    search: null,
                    [
                        'include_hidden_courses' => true,
```

```
                    'favorites' => $favoriteIds
                ]
            )
            ->pluck('id')
            ->toArray();
        }
    );

    return $myCoursesIds;
}
```

Exemplo 3: Invalidação seletiva

```
// Quando um usuário adiciona um curso aos favoritos:
public function addToFavorites($userId, $courseId)
{
    // Adiciona aos favoritos...
    Favorite::create(['user_id' => $userId, 'course_id' => $courseId]);

    // Invalida TODAS as chaves de 'user_get_my_courses_ids' deste usuário
    $this->cacheService->forget(
        'user_get_my_courses_ids',
        null, // descrição opcional
        ['userId' => $userId],
        true  // CASCADE: remove todas as variações
    );

    // Também limpa a listagem paginada
    $this->cacheService->forget(
        'user_my_courses',
        null,
        ['userId' => $userId],
        true
    );
}
```

□□ Comandos Artisan

php artisan cache:keys

Lista todas as chaves do Redis de forma organizada.

Saída:

□□ Chaves Estruturadas (15)

#	Tipo	Nome	Desc	Params	Hash
1	□□	user_my_courses	cursos do usuario	userid:215814...	0a46b3
2	□□	user_get_my_courses_ids	lista dos cursos	userid:215814...	5e49b2

□□ Chaves Simples (3)

#	Tipo	Key
1	□□	permissions
2	□□	all_configs

---

php artisan cache:keys --name=user_my_courses

Filtra chaves por nome de família.

Saída:

#	Família	Description	Params	Hash
1	user_my_courses	cursos do usuario	userid:215814, pa...	0a46b3
2	user_my_courses	cursos do usuario	userid:215814, pa...	8b72c1

php artisan cache:keys --stats

Mostra estatísticas agrupadas por família.

Saída:

```
┌──────────────────────────┬──────────────┬──────────────┬────────────┐
│ Família (name)           │ Total físicas │ Hashes únicos │ Duplicadas │
├──────────────────────────┼──────────────┼──────────────┼────────────┤
│ user_my_courses          │ 5            │ 3            │ 2          │
│ user_get_my_courses_ids  │ 3            │ 3            │ 0          │
│ total_active_users       │ 1            │ 1            │ 0          │
└──────────────────────────┴──────────────┴──────────────┴────────────┘
```

---

php artisan cache:keys --purge=user_my_courses --force

Remove **TODAS** as chaves de uma família (global).

```
php artisan cache:keys --purge=user_my_courses --force
# 🗑 15 chaves removidas da família 'user_my_courses' (global).
```

---

php artisan cache:keys --purge=user_my_courses --userId=215814 --force

Remove apenas chaves de um usuário específico.

```
php artisan cache:keys --purge=user_my_courses --userId=215814 --force
# 🗑 5 chaves removidas da família 'user_my_courses' para userId=215814.
```

---

php artisan cache:keys --commands

Lista todos os comandos disponíveis com exemplos.

Boas Práticas

DO: Use nomes descritivos

```
// ✅ BOM
$this->cacheService->remember(
    'user_my_courses',
    'cursos do usuário logado',
    ['userId' => $userId, 'page' => $page],
    //...
);
```

```
// ❌ RUIM
$this->cacheService->remember(
    'umc',
    'cache',
    ['u' => $userId, 'p' => $page],
    //...
);
```

DO: Sempre inclua identificadores únicos

```
// ✅ BOM - inclui userId para invalidação seletiva
['userId' => $userId, 'courseId' => $courseId]

// ❌ RUIM - não tem como invalidar por usuário
['courseId' => $courseId]
```

DO: Use cascade para invalidar famílias

```
// Quando o usuário atualiza perfil, limpa TODOS os caches dele
$this->cacheService->forget(
    'user_profile',
    null,
```

```
    ['userId' => $userId],
    true  // CASCADE
);
```

DO: Normalize os parâmetros

O `CacheService` normaliza automaticamente, mas mantenha consistência:

```
// □ BOM - sempre use as mesmas chaves
['userId' => $id, 'page' => $page]

// □ RUIM - às vezes 'userId', às vezes 'user_id'
['userId' => $id]  // em um lugar
['user_id' => $id] // em outro lugar
```

Debug e Inspeção

Ver chaves de um usuário específico

```
// No código
$keys = $this->cacheService->findUserKeys(); // usuário logado

// Via CLI
php artisan cache:keys --name=user_ --stats
```

Ver todas as chaves de uma família

```
php artisan cache:keys --name=user_my_courses
```

Verificar duplicatas

```
php artisan cache:keys --stats
```

Procure por valores altos na coluna "Duplicadas".

Limitações

- **Apenas Redis:** Métodos de inspeção (`getKey`, `findUserKeys`, etc.) só funcionam com driver Redis
- **Cascade só no Redis:** O forget com cascade requer Redis
- **Modo nativo vs lógico:** Não misture os dois modos na mesma chamada

---

Migração do Cache Antigo

Se você tem código usando cache tradicional do Laravel:

Antes:

```
Cache::remember('user_courses_' . $userId . '_' . $page, 60, function() {
    return Course::where('user_id', $userId)->get();
});
```

Depois:

```
$this->cacheService->remember(
    'user_courses',
    'cursos do usuario',
    ['userId' => $userId, 'page' => $page],
    60,
    function() use ($userId) {
        return Course::where('user_id', $userId)->get();
    }
);
```

Vantagens:

- □ Chaves legíveis no Redis
- □ Fácil de debugar
- □ Invalidação seletiva
- □ Estatísticas por família
- □ CLI poderoso

Referências

- Documentação Laravel Cache
- Redis Documentation
- Código fonte: `App\Services\CacheService`
- CLI: `App\Console\Commands\CacheKeysCommand`

Última atualização: Dezembro 2025
Versão: 2.0 (Nova Estrutura)

Revisão #1
Criado 5 dezembro 2025 15:23:50 por Suporte EduStore
Atualizado 5 dezembro 2025 15:51:22 por Suporte EduStore