Library Management System Restful API Application Documentation

This Library Management System was built using the PHP programming language with the Laravel 10 framework. This system is used to manage Author and Book data. The reason I chose the Laravel framework was because it implemented the MVC (Model-View-Controller) and ORM (Object Relation Mapping) architecture.

Database Design :

API Endpoint :

```
GET|HEAD   api/authors ..... authors.index › AuthorController@index
POST       api/authors ..... authors.store › AuthorController@store
GET|HEAD   api/authors/{author} ..... authors.show › AuthorController@show
PUT|PATCH  api/authors/{author} ..... authors.update › AuthorController@update
DELETE     api/authors/{author} ..... authors.destroy › AuthorController@destroy
GET|HEAD   api/authors/{id}/books ..... AuthorController@books
GET|HEAD   api/books ..... books.index › BookController@index
POST       api/books ..... books.store › BookController@store
GET|HEAD   api/books/{book} ..... books.show › BookController@show
PUT|PATCH  api/books/{book} ..... books.update › BookController@update
DELETE     api/books/{book} ..... books.destroy › BookController@destroy
```

Performance Tuning :

For performance tuning, I use query optimization techniques by selecting only the required fields and cache techniques provided by Laravel. For cases of millions of data, maybe you can use pagination techniques and Redis for the cache.

```php
//Method untuk listing semua Author
no usages
public function index(): \Illuminate\Http\JsonResponse
{
    //Implementasi cache untuk mempecepat laod data Author
    //Cache akan disimpan selama 60 menit, ketika lebih dari 60 detik maka,
    //Cache akan diisi dengan request terakhir
    $authors = Cache::remember( key: 'authors_all',  ttl: 60, function () {
        return Author::select('id', 'name', 'bio', 'birth_date')->get();
    });

    return response()->json($authors);
}
```

Endpoint Testing :

1.Author - Creating: Ensure the author is created with correct attributes. - Retrieving: Fetch an author by ID and verify its details. - Updating: Modify an author's attributes and check the changes. - Deleting: Remove an author and verify it no longer exists. - Edge Case: Test for a 404 response when the author is not found.	2. Book - Creating: Ensure the book is created with the correct attributes and associated with an author. - Retrieving: Fetch a book by ID and verify its details. - Updating: Modify a book's attributes and check the changes. - Deleting: Remove a book and verify it no longer exists. - Edge Case: Test for a 404 response when the book is not found. - Books by Author: Verify that books are correctly returned when querying by author.

```
PASS  Tests\Feature\AuthorControllerTest
✓ it can create an author                                              3.04s
✓ it can retrieve an author                                            0.90s
✓ it can update an author                                              0.77s
✓ it can delete an author                                              0.77s
✓ it returns 404 for nonexistent author                                0.85s
PASS  Tests\Feature\BookControllerTest
✓ it can create a book                                                 0.91s
✓ it can retrieve a book                                               1.07s
✓ it can update a book                                                 0.87s
✓ it can delete a book                                                 0.76s
✓ it returns 404 for nonexistent book                                  0.77s
✓ it can retrieve books by author                                      0.73s

PASS  Tests\Feature\ExampleTest
✓ the application returns a successful response                        0.76s

PASS  Tests\Unit\ExampleTest
✓ that true is true

Tests:    13 passed (33 assertions)
Duration: 14.03s
```