Introduction

Welcome to the GuardMe Code Architecture Guide! 📚 This document serves as an introduction to the coding standards and architectural principles that ensure our codebase remains clean, maintainable, and scalable. Adhering to these guidelines will help maintain consistency and quality across all projects within the company. Let's build something amazing together! 🎉

Contents

Models

Controllers

Services

Migrations

Requests

Routes

Enums

Models

Models represent the data and the business logic of your application. This section covers:

Defining models and their attributes.

Using relationships and eager loading.

Implementing scopes and accessors/mutators.

Adhering to the single responsibility principle.

Read more

Controllers

Controllers handle the HTTP requests coming into your application and return the appropriate responses. This guide covers:

Naming conventions for controllers and methods.

Using form requests for validation.

Leveraging resource controllers for CRUD operations.

Best practices for organizing your controllers.

Read more

Services

Service classes contain the business logic of your application. This guide covers:

Creating service classes to handle complex logic.

Keeping controllers thin by delegating logic to services.

Ensuring low coupling and high cohesion.

Examples of good and bad practices for service classes.

Read more

Migrations

Migrations manage the changes to your database schema over time. This guide includes:

Best practices for creating and managing migrations.

Naming conventions for migration files, tables, columns, foreign keys, and indexes.

Handling complex schema changes and data migrations.

Read more

Requests

Requests are used to encapsulate validation logic. This guide explains:

Creating and using form request classes.

Validating request data.

Using custom validation rules.

Handling authorization within requests.

Read more

Routes

Routes define the endpoints for your application. This guide discusses:

Best practices for naming and organizing routes.

Grouping routes by functionality.

Using named routes for easier reference.

Appropriate usage of HTTP methods.

Read more

Enums

Enums provide a way to define a set of named values, improving code readability and reducing bugs. This guide includes:

Creating enums using PHP artisan commands.

Using enums in models and requests.

Best practices for defining and using enums.

Examples of good and bad practices.

Read more

Conclusion

By following the guidelines outlined in this guide, you will contribute to a codebase that is maintainable, scalable, and consistent with GuardMe's standards. Thank you for your commitment to coding excellence. Together, we can achieve greatness! 🙌

For more detailed information, please refer to the individual sections linked above. Happy coding! 💻✨