request life cycle

1. The entry point for all requests to a Laravel application is the index file.
2. All requests are directed to this file by your web server .
3. The index file doesn't contain much code. Rather, it is a starting point for loading the rest of the framework.
4. The index file loads the Composer generated autoloader definition, and then retrieves an instance of the Laravel application from app.php The first action taken by Laravel itself is to create an instance of the application
5. The incoming request is sent to the HTTP kernel. This kernel serves as the central location that all requests flow through.
6. This kernel bootstrapping configures error handling, configure logging, detect the application environment, and perform other tasks that need to be done before the request is actually handled. One of the most important kernel bootstrapping actions is loading the service providers for your application
7. Essentially every major feature offered by Laravel is bootstrapped and configured by a service provider
8. One of the most important service providers in your application is the App\Providers\RouteServiceProvider. This service provider loads the route files contained within your application's routes directory
9. Service providers are truly the key to bootstrapping a Laravel application. The application instance is created, the service providers are registered, and the request is handed to the bootstrapped application

facade

- Facade is a structural design pattern that provides a simplified interface to a complex system of classes, library or framework.
- Façade decrease overall complexity of the application, it also helps to move unwanted dependencies to one place

Façade use in

1. Simplification: Facades simplify the interaction with a complex system by providing high-level, easy to understand interface.
2. Abstraction: Facades abstract the details and intricacies of the subsystems, allowing the client code to interact with the system without needing to understand its internal workings.
3. Encapsulation: The facade encapsulates the interactions with the subsystems, keeping them separate from the client code.
4. Improved Maintainability: it becomes easier to make changes or updates to the subsystems without affecting the client code.
5. Security: Facades can also be used to control access to sensitive parts of a system, ensuring that only authorized interactions occur.

   Overall, facades are a valuable design pattern for making complex systems more manageable and accessible.

Service provider	service container
1. Service providers are the central place to configure your application. 2. In Laravel, service providers are an integral part of the framework's service container and dependency injection system. 3. They are responsible for binding classes and dependencies into the container and bootstrapping various components of your application during the Laravel application's bootstrapping process.	1. is simply a PHP object that manages the instantiation of services 2. is a design pattern used to manage and organize dependencies within an application. 3. Dependency Management: The primary purpose of a service container is to manage the dependencies required by various parts of your application.

validation
Validation is a critical aspect of software development, especially when dealing with user input Validation use to 1. Data Integrity: Ensuring that data is accurate 2. Security: It guards against various types of attacks, such as SQL injection, by validating input and ensuring that it adheres to expected formats, you can significantly reduce the risk of security breaches and unauthorized access. 3. Error Prevention: Validation helps prevent errors and exceptions from occurring during data processing. By rejecting invalid data at the input stage, you avoid downstream issues that can be harder to diagnose and correct. 4. Data Consistency: Validation ensures that data entered a system or database adheres to predefined rules and constraints. This consistency is crucial for data integrity and for preventing data anomalies that can lead to application errors. 5. Efficiency: By validating data upfront, you can avoid unnecessary processing and database queries for invalid or irrelevant data, thus improving system performance. 6. Reliability: Validation contributes to the overall reliability of your application. It helps identify and handle exceptional cases gracefully, reducing the likelihood of application crashes or unexpected behavior. 7. Maintenance: Validated code is typically easier to maintain. When validation is done consistently, it makes

it easier for developers to understand, update, and extend the codebase.

8. Preventing Business Logic Flaws: Validation can also help prevent flaws in your application's business logic. By verifying that data adheres to specific business rules, you ensure that your application operates correctly according to your intended logic.

- Laravel provides a wide range of validation rules and features that you can use to validate form data, HTTP requests.

1. Create a Validation Request (Form Request).
2. Define Validation Rules.
3. Use the Validation Request in Your Controller.
4. Display Validation Errors.
5. Custom Error Messages.
6. Additional Validation Features.

REQUEST

$_REQUEST is a super global variable in PHP that contains input from $_POST, $_GET, and $_COOKIE. These correspond to HTTP POST, GET, and Cookies.

Resources

1. https://laravel.com/docs/10.x/lifecycle#lifecycle-overview

2. https://refactoring.guru/design-patterns/facade/php/example
3. https://www.codementor.io/@decodeweb/laravel-service-providers-explained-in-depth-12uu86s2pq
4. https://www.codemag.com/Article/2212041/Dependency-Injection-and-Service-Container-in-Laravel
5. https://www.educative.io/answers/what-is-request-in-php
6. ChatGpt,arabyAI,peo