Response:

All routes and controllers should return a response to be sent back to the user's browser. Laravel provides several different ways to return responses. The most basic response is returning a string from a route or controller. The framework will automatically convert the string into a full HTTP response

And if you return an array the Laravel framework automaticly

Convert the array into the JSON response

Returning a full Response instance allows you to customize the response's HTTP status code and headers. A Response instance inherits from the Symfony\Component\HttpFoundation\Response class, which provides a variety of methods for building HTTP responses

The response helper may be used to generate other types of response instances. When the response helper is called without arguments, an implementation of the Illuminate\Contracts\Routing\ResponseFactory contract is returned. This contract provides several helpful methods for generating responses.

If you need control over the response's status and headers but also need to return a view as the response's content, you should use the view method:

return response()

->view('hello', $data, 200)

->header('Content-Type', $type);

Eloquent :

Laravel includes Eloquent, an object-relational mapper (ORM) that makes it enjoyable to interact with your database. When using Eloquent, each database table has a corresponding "Model" that is used to interact with that table. In addition to retrieving records from the database table, Eloquent models allow you to insert, update, and delete records from the table as well.

Eloquent Model :

Conventions Models generated by the make:model command will be placed in the app/Models directory. Let's examine a basic model class and discuss some of Eloquent's key conventions.

Eloquent will also assume that each model's corresponding database table has a primary key column named id. If necessary, you may define a protected $primaryKey property on your model to specify a different column that serves as your model's primary key

If you would like to generate a database migration when you generate the model, you may use the --migration or -m option

In addition, Eloquent assumes that the primary key is an incrementing integer value, which means that Eloquent will automatically cast the primary key to an integer. If you wish to use a non-incrementing or a non-numeric primary key you must define a public $incrementing property on your model that is set to false.

File upload :

In web applications, one of the most common use-cases for storing files is storing user uploaded files such as photos and documents. Laravel makes it very easy to store uploaded files using the store method on an uploaded file instance. Call the store method with the path at which you wish to store the uploaded file

There are a few important things to note about this example. Note that we only specified a directory name, not a filename. By default, the store method will generate a unique ID to serve as the filename. The file's extension will be determined by examining the file's MIME type. The path to the file will be returned by the store method so you can store the path, including the generated filename, in your database. You may also call the putFile method on the Storage facade to perform the same file storage operation as the example above.