Inside your project folder, create a new file named **index.php**

index.php

```php
<?php

var_dump($_SERVER["REQUEST_URI"]);
```

Create a new file named **.htaccess** to specify server configuration. This will says that any url will now call the index.php script, these rules are only valid for apache.
.htaccess

```
RewriteEngine On
RewriteRule . index.php
```

Modify the **index.php**

```php
<?php

$parts = explode("/", $_SERVER["REQUEST_URI"]);
print_r($parts);
```

Output:

URL: localhost/php-rest-api

```
Array ( [0] => [1] => php-rest-api[2] => )
```

File: **Index.php**

```php
<?php

$parts = explode("/", $_SERVER["REQUEST_URI"]);

if ($parts[2] != "products") {
    http_response_code(404);
    exit;
}

$id = $parts[3] ?? null;

var_dump($id);
```

Create a new folder named **src**, and create a new file named **ProductController.php**

```php
<?php
class ProductController
{
    public function processRequest(string $method, ?string $id): void
    {
        var_dump($method, $id);
    }
}
```

Modify the **index.php**

```php
<?php
declare(strict_types=1);
spl_autoload_register(function ($class) {
    require __DIR__ . "/src/$class.php";
});

set_exception_handler("ErrorHandler::handleException");

header("Content-type: application/json; charset=UTF-8");
$parts = explode("/", $_SERVER["REQUEST_URI"]);

if ($parts[2] != "products") {
    http_response_code(404);
    exit;
}

$id = $parts[3] ?? null;
$controller = new ProductController;
$controller->processRequest($_SERVER["REQUEST_METHOD"], $id);
```

Access the localhost/phpMyAdmin, create a new database named **productDb** and create a new table **product** or run the script below in SQL tab

```sql
CREATE DATABASE productDb;

CREATE TABLE product (
 id INT NOT NULL AUTO_INCREMENT,
 name VARCHAR(128) NOT NULL,
 size INT NOT NULL DEFAULT 0,
 is_available BOOLEAN NOT NULL DEFAULT FALSE,
 PRIMARY KEY (id)
);
```

Manually insert value into the **product** table or run the script below

```sql
INSERT INTO `product`(`name`, `size`, `is_available`) VALUES ('Product one','10',0);
INSERT INTO `product`(`name`, `size`, `is_available`) VALUES ('Product two','20',1);
```

Modify the **ProductController.php**

```php
<?php
class ProductController
{
    public function processRequest(string $method, ?string $id): void
    {
        if ($id) {
            $this->processResourcetRequest($method, $id);
        } else {
            $this->processCollectionRequest($method);

        }
    }


    private function processResourcetRequest(string $method, string $id): void
    {

    }

    private function processCollectionRequest(string $method): void
    {
        switch ($method) {
            case "GET":
                echo json_encode(["id" => 123]);
                break;
        }
    }
}
```

Create a new file named **Databased.php** inside **src** folder

```php
<?php
class Database
{
    public function __construct(
        private string $host,
        private string $name,
        private string $user,
        private string $password
    )
    {
    }
    public function getConnection(): PDO
    {
        $dsn = "mysql:host={$this->host};dbname={$this->name};charset=utf8";
        return new PDO($dsn, $this->user, $this->password);
    }
}
```

Modify **index.php** and insert the code below line 17

```php
17. $id = $parts[3] ?? null;
18.
19. $database = new Database("localhost", "productDb", "root", "");
20. $database->getConnection();
```

Create **ErrorHandler.php** inside **src** folder

```
<?php

class ErrorHandler
{
    public static function handleException(Throwable $exception): void
    {
        http_response_code(500);
        echo json_encode([
            "code" => $exception->getCode(),
            "message" => $exception->getMessage(),
            "file" => $exception->getFile(),
            "line" => $exception->getLine()
        ]);
    }
}
```

Modify the **index.php** and insert the code between spl_autoload_register and header

```
spl_autoload_register(function ($class) {
    require __DIR__ . "/src/$class.php";
});

set_exception_handler("ErrorHandler::handleException");

header("Content-type: application/json; charset=UTF-8");
```

Create a new file named **ProductGateway.php** inside **src** folder

```
<?php

class ProductGateway
{
    private PDO $conn;
    public function __construct(Database $database)
    {
        $this->con = $database->getConnection();
    }

    public function getAll(): array
    {
        $sql = "SELECT * FROM product";
        $res = $this->conn->query($sql);
        $data = [];

        while ($row = $res->fetch(PDO::FETCH_ASSOC)) {
            $data[] = $row;
            ;
        }

        return $data;
    }
}
```

Modify the **ProductController.php** to use the **ProductGateway**

```php
<?php
class ProductController
{
    public function __construct(private ProductGateway $gateway)
    {
    }
```

```php
private function processCollectionRequest(string $method): void
{
    switch ($method) {
        case "GET":
            echo json_encode($this->gateway->getAll());
            break;
    }
}
```

Modify the **index.php** to use the ProductGateway

```php
$database = new Database("localhost", "productDb", "root", "");
~~$database->getConnection();~~

$gateway = new ProductGateway($database);

$controller = new ProductController($gateway);
$controller->processRequest($_SERVER["REQUEST_METHOD"], $id);
```

In API Response, you will see that the is_available should be a Boolean value (true/false), in PDO, the default response is set to stringify, this converts all value to string. We should fix this by setting the PDO Stringify attribute to false, to do this, modify the **getConnection** function in the **Database.php** and **getAll** function in **ProductGateway.php**

Database.php

```php
public function getConnection(): PDO
{
    $dsn = "mysql:host={$this->host};dbname={$this->name};charset=utf8";
    return new PDO($dsn, $this->user, $this->password, [PDO::ATTR_EMULATE_PREPARES => false, PDO::ATTR_STRINGIFY_FETCHES => false]);
}
```

ProductGateway.php

```php
while ($row = $res->fetch(PDO::FETCH_ASSOC)) {
    $row["is_available"] = (bool) $row["is_available"];
    $data[] = $row;
}
```

Modify the **ProductController.php** to add the POST method

```
private function processCollectionRequest(string $method): void
    {
        switch ($method) {
            case "GET":
                echo json_encode($this->gateway->getAll());
                break;

            case "POST":
                $data = (array) json_decode(file_get_contents("php://input"), true);
                var_dump($data);
                break;
        }
    }
```

Modify the **ProductGateway.php** and create a new function for creating a new row

```
public function create(array $data): string
{
    $sql = "INSERT INTO product (name, size, is_available)
            VALUES (:name, :size, :is_available)";
    $res = $this->conn->prepare($sql);
    $res->bindValue(":name", $data["name"], PDO::PARAM_STR);
    $res->bindValue(":size", $data["size"] ?? 0, PDO::PARAM_INT);
    $res->bindValue(":is_available", (bool) $data["is_available"] ?? false, PDO::PARAM_BOOL);

    $res->execute();
    return $this->conn->lastInsertId();
}
```

Back to the **ProductController.php,** hook the new function **create()** from **ProductGateway.php**

```
case "POST":
   $data = (array) json_decode(file_get_contents("php://input"), true);
~~   var_dump($data);~~

   $id = $this->gateway->create($data);

   http_response_code(201);
   echo json_encode([
     "message" => "Product created",
     "id" => $id
   ]);
   break;
```

To handle the error in ProductGateway, we have to create a new function for error handler, open the **ErrorHandler.php** and add the new function below.

```php
public static function handdleError(
    int $errno,
    string $errstr,
    string $errfile,
    int $errline
): bool
{
    throw new ErrorException($errstr, 0, $errno, $errfile, $errline);
}
```

Set the error handler above the set_exception_handler inside **index.php**

```php
set_error_handler("ErrorHandler::handleError");
set_exception_handler("ErrorHandler::handleException");
```

Modify the **ProductController.php** and create a new function getValidationErrors()

```php
private function getValidationErrors(array $data): array
{
    $errors = [];
    if (empty($data["name"])) {
        $errors[] = "name is requred";
    }

    if (array_key_exists("size", $data)) {
        if (filter_var($data["size"], FILTER_VALIDATE_INT) === false) {
            $errors[] = "size must be an integer";
        }
    }

    return $errors;
}
```

After creating the validation error, use it inside the case “POST” of **ProductController.php**

```php
case "POST":
    $data = (array) json_decode(file_get_contents("php://input"), true);
    $errors = $this->getValidationErrors(($data));

    if (!empty($errors)) {
        http_response_code(422);
        echo json_encode(["errors" => $errors]);
        break;
    }

    $id = $this->gateway->create($data);

    http_response_code(201);
    echo json_encode([
        "message" => "Product created",
        "id" => $id
    ]);
    break;

default:
    http_response_code(405);
    header("Allow: GET, POST");
```

To get the product by id, create a new function get() inside **ProductGateway.php**

```php
public function get(string $id)
{
    $sql = "SELECT * FROM product WHERE id = :id";
    $res = $this->conn->prepare($sql);
    $res->bindValue(":id", $id, PDO::PARAM_INT);
    $res->execute();
    $data = $res->fetch(PDO::FETCH_ASSOC);

    if ($data !== false) {
        $data["is_available"] = (bool) $data["is_available"];
    }

    return $data;
}
```

Then in the **ProductController.php**, add the code below inside the **processResourceRequest()**

```php
private function processResourcetRequest(string $method, string $id): void
{
    $product = $this->gateway->get($id);
    if (!$product) {
        http_response_code(404);
        echo json_encode(["message" => "Product not found"]);
        return;
    }

    switch ($method) {
        case "GET":
            echo json_encode($product);
            break;
    }
}
```

If you run the **localhost/php-rest-api/products/1**, you should get the data similar to the sample below.

** localhost/[project name] /products/[id of the product]

```json
{
  "id": 1,
  "name": "test",
  "size": 1,
  "is_available": true
}
```

Implement the PATCH method by adding a new case inside **processResourceRequest** inside **ProductController.php**

```php
case "PATCH":
    $data = (array) json_decode(file_get_contents("php://input"), true);
    $errors = $this->getValidationErrors(($data));

    if (!empty($errors)) {
        http_response_code(422);
        echo json_encode(["errors" => $errors]);
        break;
    }

    $rows = $this->gateway->update($product, $data);

    echo json_encode([
        "message" => "Product $id updated",
        "rows" => $rows
    ]);
    break;
```

Create a new function **update()** inside **ProductGateway.php**

```php
public function update(array $current, array $new): int
{
    $sql = "UPDATE product SET name = :name, size = :size, is_available = :is_available WHERE id =:id";
    $res = $this->conn->prepare($sql);
    $res->bindValue(":name", $new["name"] ?? $current["name"], PDO::PARAM_STR);
    $res->bindValue(":size", $new["size"] ?? $current["size"], PDO::PARAM_INT);
    $res->bindValue(":is_available", $new["is_available"] ?? $current["is_available"],
PDO::PARAM_BOOL);
    $res->bindValue(":id", $current["id"], PDO::PARAM_INT);

    $res->execute();

    return $res->rowCount();
}
```

If we try to update one field of products, we are getting the validation error. We need to update the condition inside **getValidationErrors** of **ProductController.php**.

```php
private function getValidationErrors(array $data, bool $is_new = true): array
{
    $errors = [];
    if ($is_new && empty($data["name"])) {
        $errors[] = "name is requred";
    }

    if (array_key_exists("size", $data)) {
        if (filter_var($data["size"], FILTER_VALIDATE_INT) === false) {
            $errors[] = "size must be an integer";
        }
    }

    return $errors;
}
```

We can pass false as the second argument in **getValidationErrors** inside the **PATCH case** of **ProductController.php**.

```php
case "PATCH":
    $data = (array) json_decode(file_get_contents("php://input"), true);
    $errors = $this->getValidationErrors($data, false);
```

Create a new function **delete()** inside **ProductGateway.php**

```php
public function delete(string $id): int
{
    $sql = "DELETE FROM product WHERE id = :id";
    $res = $this->conn->prepare($sql);
    $res->bindValue(":id", $id, PDO::PARAM_INT);
    $res->execute();

    return $res->rowCount();
}
```

The last method that we need to create in **ProductController.php** is the **DELETE** method, add a new case named DELETE.

```php
case "DELETE":
    $rows = $this->gateway->delete($id);
    echo json_encode([
        "message" => "Product $id deleted",
        "rows" => $rows
    ]);
    break;

default:
    http_response_code(405);
    header("Allow: GET, PATCH, DELETE");
```