PHP REST API Part 2

Code improvements and Implementation of Reviews

**.htaccess**

1. RewriteEngine On
2. RewriteCond %{REQUEST\_FILENAME} !-f
3. RewriteCond %{REQUEST\_FILENAME} !-d
4. RewriteCond %{REQUEST\_FILENAME} !-l
5. RewriteRule . index.php [L]

**Index.php**

1. <?php
2. **declare**(strict\_types=1);
3. **require** \_\_DIR\_\_ . "/vendor/autoload.php";
5. set\_error\_handler("ErrorHandler::handleError");
6. set\_exception\_handler("ErrorHandler::handleException");
8. header("Content-type: application/json; charset=UTF-8");
9. $parts = explode("/", $\_SERVER["REQUEST\_URI"]);
11. **switch** ($parts[2]) {
12. **case** 'products':
13. $id = $parts[3] ?? null;
15. $database = **new** Database("localhost", "ecommercedb", "root", "");
16. $database->getConnection();
17. $gateway = **new** ProductGateway($database);
19. $controller = **new** ProductController($gateway);
20. $controller->processRequest($\_SERVER["REQUEST\_METHOD"], $id);
21. **break**;
23. **case** 'reviews':
24. $id = $parts[3] ?? null;
25. $productid = $parts[5] ?? null;
27. $database = **new** Database("localhost", "ecommercedb", "root", "");
28. $database->getConnection();
29. $gateway = **new** ReviewGateway($database);
31. $controller = **new** ReviewController($gateway);
32. $controller->processRequest($\_SERVER["REQUEST\_METHOD"], $id, $productid);
33. **break**;
35. **default**:
36. http\_response\_code(404);
37. exit;
38. }

**ProductGateway.php (inside src folder)**

1. <?php
3. **class** ProductGateway
4. {
5. **private** PDO $conn;
6. **public** **function** \_\_construct(Database $database)
7. {
8. $this->conn = $database->getConnection();
9. }
11. **public** **function** getAll(): **array**
12. {
13. $sql = "SELECT \* FROM products";
14. $res = $this->conn->query($sql);
15. $data = [];
17. **while** ($row = $res->fetch(PDO::FETCH\_ASSOC)) {
18. $row["is\_available"] = (bool) $row["is\_available"];
19. $data[] = $row;
20. }
22. **return** $data;
23. }
25. **public** **function** create(**array** $data): string
26. {
27. $sql = "INSERT INTO products (name, size, is\_available)
28. VALUES (:name, :size, :is\_available)";
29. $res = $this->conn->prepare($sql);
30. $res->bindValue(":name", $data["name"], PDO::PARAM\_STR);
31. $res->bindValue(":size", $data["size"] ?? 0, PDO::PARAM\_INT);
32. $res->bindValue(":is\_available", (bool) $data["is\_available"] ?? false, PDO::PARAM\_BOOL);
34. $res->execute();
35. **return** $this->conn->lastInsertId();
36. }
38. **public** **function** get(string $id)
39. {
40. $sql = "SELECT \* FROM products WHERE id = :id";
41. $res = $this->conn->prepare($sql);
42. $res->bindValue(":id", $id, PDO::PARAM\_INT);
43. $res->execute();
44. $data = $res->fetch(PDO::FETCH\_ASSOC);
46. **if** ($data !== false) {
47. $data["is\_available"] = (bool) $data["is\_available"];
49. $sqlReviews = "SELECT \* FROM reviews where productid = :productid LIMIT 10";
50. $resReviews = $this->conn->prepare($sqlReviews);
51. $resReviews->bindValue(":productid", $id, PDO::PARAM\_INT);
52. $resReviews->execute();
53. $dataReviews = $resReviews->fetchAll(PDO::FETCH\_ASSOC);
54. **if** ($dataReviews !== false) {
55. $data['reviews'] = $dataReviews;
56. }
57. }
59. **return** $data;
60. }
62. **public** **function** update(**array** $current, **array** $new): int
63. {
64. $sql = "UPDATE products SET name = :name, size = :size, is\_available = :is\_available WHERE id =:id";
65. $res = $this->conn->prepare($sql);
66. $res->bindValue(":name", $new["name"] ?? $current["name"], PDO::PARAM\_STR);
67. $res->bindValue(":size", $new["size"] ?? $current["size"], PDO::PARAM\_INT);
68. $res->bindValue(":is\_available", $new["is\_available"] ?? $current["is\_available"], PDO::PARAM\_BOOL);
69. $res->bindValue(":id", $current["id"], PDO::PARAM\_INT);
71. $res->execute();
73. **return** $res->rowCount();
74. }
76. **public** **function** delete(string $id): int
77. {
78. $sql = "DELETE FROM products WHERE id = :id";
79. $res = $this->conn->prepare($sql);
80. $res->bindValue(":id", $id, PDO::PARAM\_INT);
81. $res->execute();
83. **return** $res->rowCount();
84. }
85. }

**ProductController.php (inside src folder)**

1. <?php
2. **class** ProductController
3. {
4. **public** **function** \_\_construct(**private** ProductGateway $gateway)
5. {
6. }
7. **public** **function** processRequest(string $method, ?string $id): void
8. {
9. **if** ($id) {
10. $this->processResourcetRequest($method, $id);
11. } **else** {
12. $this->processCollectionRequest($method);
14. }
15. }

18. **private** **function** processResourcetRequest(string $method, string $id): void
19. {
20. $product = $this->gateway->get($id);
21. **if** (!$product) {
22. http\_response\_code(404);
23. echo json\_encode(["message" => "Product not found"]);
24. **return**;
25. }
27. **switch** ($method) {
28. **case** "GET":
29. echo json\_encode($product);
30. **break**;
32. **case** "PATCH":
33. $data = (**array**) json\_decode(file\_get\_contents("php://input"), true);
34. $errors = $this->getValidationErrors($data, false);
36. **if** (!empty($errors)) {
37. http\_response\_code(422);
38. echo json\_encode(["errors" => $errors]);
39. **break**;
40. }
42. $rows = $this->gateway->update($product, $data);
44. echo json\_encode([
45. "message" => "Product $id updated",
46. "rows" => $rows
47. ]);
48. **break**;
50. **case** "DELETE":
51. $rows = $this->gateway->delete($id);
52. echo json\_encode([
53. "message" => "Product $id deleted",
54. "rows" => $rows
55. ]);
56. **break**;
58. **default**:
59. http\_response\_code(405);
60. header("Allow: GET, PATCH, DELETE");
62. }
63. }
65. **private** **function** processCollectionRequest(string $method): void
66. {
67. **switch** ($method) {
68. **case** "GET":
69. echo json\_encode($this->gateway->getAll());
70. **break**;
72. **case** "POST":
73. $data = (**array**) json\_decode(file\_get\_contents("php://input"), true);
74. $errors = $this->getValidationErrors(($data));
76. **if** (!empty($errors)) {
77. http\_response\_code(422);
78. echo json\_encode(["errors" => $errors]);
79. **break**;
80. }
82. $id = $this->gateway->create($data);
84. http\_response\_code(201);
85. echo json\_encode([
86. "message" => "Product created",
87. "id" => $id
88. ]);
89. **break**;
91. **default**:
92. http\_response\_code(405);
93. header("Allow: GET, POST");
94. }
95. }
97. **private** **function** getValidationErrors(**array** $data, bool $is\_new = true): **array**
98. {
99. $errors = [];
100. **if** ($is\_new && empty($data["name"])) {
101. $errors[] = "name is requred";
102. }
104. **if** (array\_key\_exists("size", $data)) {
105. **if** (filter\_var($data["size"], FILTER\_VALIDATE\_INT) === false) {
106. $errors[] = "size must be an integer";
107. }
108. }
110. **return** $errors;
111. }
112. }

**\*\***Download and install Composer on your computer, **https://getcomposer.org/download/  
\***url for windows setup.exe **(**[**https://getcomposer.org/Composer-Setup.exe**](https://getcomposer.org/Composer-Setup.exe)**)**

I also added a new file named **composer.json** for autoloading of files with the same class names.

After creating the composer.json with the code below, run this on command prompt inside your project folder to generate the vendor folder.

**composer dump-autoload**

1. {
2. "autoload": {
3. "psr-4": {
4. "": "src/"
5. }
6. }
7. }

Graphical user interface

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

In my phpMyAdmin, create a new table named **reviews,** set the id index = primary, auto increment(A\_I) to true.

**A screenshot of a computer

Description automatically generated**

**User Authentication**

Create a new table: **users**, set the id index as primary and auto increment(A\_I) to true. Next, we need to have a unique email, set the index of email as primary.

**Graphical user interface

Description automatically generated**

For user authentication, create a new file inside **src folder** and named it as **UserGateway.php.**

Set the $database inside public function \_\_construct() so that we can pass the database to the UserGateway. Then create new function for **register** and **login.**

1. <?php
3. **class** UserGateway
4. {
5. **private** PDO $conn;
6. **public** **function** \_\_construct(Database $database)
7. {
8. $this->conn = $database->getConnection();
9. }
11. **public** **function** register(**array** $data): string
12. {
13. $sql = "INSERT INTO users (email, password, firstName, middleName, lastName, contactNo)
14. VALUES (:email, :password, :firstName, :middleName, :lastName, :contactNo)";
15. $res = $this->conn->prepare($sql);
17. $password = md5($data["password"]);
19. $res->bindValue(":email", $data["email"], PDO::PARAM\_STR);
20. $res->bindValue(":password", $password, PDO::PARAM\_STR);
21. $res->bindValue(":firstName", $data["firstName"], PDO::PARAM\_STR);
22. $res->bindValue(":middleName", $data["middleName"], PDO::PARAM\_STR);
23. $res->bindValue(":lastName", $data["lastName"], PDO::PARAM\_STR);
24. $res->bindValue(":contactNo", $data["contactNo"], PDO::PARAM\_STR);
26. $res->execute();
27. **return** $this->conn->lastInsertId();
28. }
30. **public** **function** login(string $email, string $password)
31. {
32. $sql = "SELECT \* FROM users WHERE email = :email AND password = :password";
33. $res = $this->conn->prepare($sql);
34. $res->bindValue(":email", $email, PDO::PARAM\_STR);
35. $res->bindValue(":password", md5($password), PDO::PARAM\_STR);
37. $res->execute();
38. $data = $res->fetch(PDO::FETCH\_ASSOC);
39. unset($data['password']);
40. **return** $data;
41. }
43. **public** **function** changePassword(**array** $current, **array** $new): int
44. {
45. $sql = "UPDATE users SET password = :password";
46. $res = $this->conn->prepare($sql);
47. $res->bindValue(":password", md5($new["password"]) ?? md5($current["password"]), PDO::PARAM\_STR);
49. $res->execute();
51. **return** $res->rowCount();
52. }
54. }

After creating the UserGate, create a new file inside **src** folder for **UserController.php**

1. <?php
2. **class** UserController
3. {
4. **public** **function** \_\_construct(**private** UserGateway $gateway)
5. {
6. }
8. **public** **function** processRequest(string $method, string $action): void
9. {
10. **if** ($method === 'POST') {
11. **switch** ($action) {
12. **case** "login":
13. $this->processLoginRequest();
14. **break**;
16. **case** "register":
17. $this->processRegistrationRequest();
18. **break**;
19. }
20. } **else** {
21. http\_response\_code(405);
22. header("Allow: POST");
23. }
24. }
26. **private** **function** processLoginRequest(): void
27. {
29. $data = (**array**) json\_decode(file\_get\_contents("php://input"), true);
30. $errors = $this->getLogInValidationErrors(($data));
32. **if** (!empty($errors)) {
33. http\_response\_code(422);
34. echo json\_encode(["errors" => $errors]);
35. }
37. $user = $this->gateway->login($data['email'], $data['password']);
39. **if** (!$user) {
40. http\_response\_code(404);
41. echo json\_encode(["message" => "Incorrect email/password"]);
42. **return**;
43. }
45. echo json\_encode($user);
47. }
49. **private** **function** processRegistrationRequest(): void
50. {
52. $data = (**array**) json\_decode(file\_get\_contents("php://input"), true);
53. $errors = $this->getRegistrationValidationErrors(($data));
55. **if** (!empty($errors)) {
56. http\_response\_code(422);
57. echo json\_encode(["errors" => $errors]);
58. }
60. $id = $this->gateway->register($data);
62. http\_response\_code(201);
63. echo json\_encode([
64. "message" => "User created",
65. "id" => $id
66. ]);
68. }
70. **private** **function** getLogInValidationErrors(**array** $data): **array**
71. {
72. $errors = [];
73. **if** (empty($data["email"])) {
74. $errors[] = "Email is requred";
75. }
77. **if** (empty($data["password"])) {
78. $errors[] = "Password is requred";
79. }
81. **return** $errors;
82. }
84. **private** **function** getRegistrationValidationErrors(**array** $data): **array**
85. {
86. $errors = [];
87. **if** (empty($data["email"])) {
88. $errors[] = "Email is requred";
89. }
91. **if** (empty($data["password"])) {
92. $errors[] = "Password is requred";
93. }
95. **if** (empty($data["firstName"])) {
96. $errors[] = "First name is requred";
97. }
99. **if** (empty($data["lastName"])) {
100. $errors[] = "Last name is requred";
101. }
103. **return** $errors;
104. }
105. }

To use the UserController, modify the **index.php.** Add a new case inside the switch

1. **case** 'user':
2. $action = $parts[3] ?? null;
3. **if** ($action === null) {
4. http\_response\_code(404);
5. }
7. $database = **new** Database("localhost", "ecommercedb", "root", "");
8. $database->getConnection();
9. $gateway = **new** UserGateway($database);
11. $controller = **new** UserController($gateway);
12. $controller->processRequest($\_SERVER["REQUEST\_METHOD"], $action);
13. **break**;