

۱- برای سه موجودیت، جداول `users`, `articles`, `comments` در پایگاه داده `MySQL` پیاده‌سازی شده اند. جدول `articles` دارای فیلدهای مربوط به ساختار مقاله و `user_id` می‌باشد که یک کلید خارجی و ارجاعی به فیلد اصلی `id` در جدول `users` می‌باشد. جدول `comments` حاوی اطلاعات مربوط به یک دیدگاه و فیلد `article_id` که یک کلید خارجی و ارجاعی به فیلد اصلی و منحصر به فرد `id` در جدول `articles` می‌باشد. جدول `users` شامل اطلاعات کاربر و `username` که یک فیلد منحصر به فرد است، می‌باشد.

۲- برای جداول پیاده‌سازی شده، در قسمت `Model` لاراول، کلاس‌های `User`, `Article`, `Comment` در نظر گرفته شده است. در این کلاس‌ها فیلدهای قابل نوشتن در پایگاه داده و همچنین نوع روابط بین جداول پیاده‌سازی شده‌اند؛ مثلاً:

```
public function user(): BelongsTo
{
    return $this->belongsTo(related: User::class, foreignKey: 'user_id');
}
//Each article has many comments
0 references | 0 overrides
public function comments(): HasMany
{
    return $this->hasMany(related: Comment::class, foreignKey: 'article_id');
}
```

نوع رابطه بین `articles` و `users` یک به چند می‌باشد هر کاربر می‌تواند چندین مقاله داشته باشد ولی هر مقاله متعلق فقط به یک کاربر خاص می‌باشد. همچنین هر مقاله می‌تواند چندین دیدگاه داشته باشد ولی هر دیدگاه فقط متعلق به یک مقاله خاص می‌باشد.

۳- بخش `UserController` حاوی کلاس‌های `UserController`, `ArticleController`, `CommentController` می‌باشد.

```
//All articles
1 reference | 0 overrides
public function index(): JsonResponse
{
    $articles = $this->article->all();

    return response()->json(data: [
        'articles' => $articles
    ]);
}
```

در کلاس `ArticleController` متد `index` برای نمایش همه مقاله‌ها در `postman` می‌باشد:

The screenshot shows a Postman interface with a GET request to `http://127.0.0.1:8000/api`. The response is a 200 OK status with a JSON body. The JSON body contains an array of articles. The first article has an id of 1, a title, an introduction, and a content field. The second article has an id of 2, a title, and an introduction field.

```
{
  "articles": [
    {
      "id": 1,
      "title": "Et optio esse eius perferendis vero.",
      "introduction": "Reiciendis impedit laudantium voluptates inventore similique at dicta eveniet. Id inventore omnis voluptas quia corrupti.",
      "content": "Natus veniam sint repellendus. Sit et et officia omnis quae quaerat et. Ut ad laborum culpa quidem corrupti. Et quo sed eveniet rerum optio et.\n\nEius vitae ut consequatur voluptates velit. Natus eligendi aliquid eius doloremque id unde laudantium eos. Neque nihil laborum id adipisci qui consectetur. Dolores beatae sunt totam expedita.\n\nConsequuntur quaerat voluptate rerum. Voluptate aliquam reiciendis eaque temporibus repudiandae omnis neque. Quia ut aut quae quia quidem ullam consequatur. Aut assumenda eius quis error quam est. Quia qui porro ut et voluptas accusamus perferendis.\n\nOdio odio earum ut. Sapiente in magni reiciendis officiis in hic. Accusamus rerum quaerat illo ducimus reiciendis sapiente.\n\nMollitia id voluptates sint voluptatibus qui molestiae molestias. Ut et rerum laborum architecto reiciendis dolorem omnis. Ut itaque deserunt harum voluptas optio dolor sequi quisquam. Voluptates itaque quos fugiat eum animi eum.",
      "created_at": "2025-08-20T23:49:32.000000Z",
      "updated_at": "2025-08-20T23:49:32.000000Z"
    },
    {
      "id": 2,
      "title": "Sunt eveniet aliquid ipsam ipsam tempora asperiores eum voluptatem.",
      "introduction": "Rerum et autem earum. Rerum et est inventore minima fugit asperiores. Ad consequatur quasi et sunt dolorem"
    }
  ]
}
```

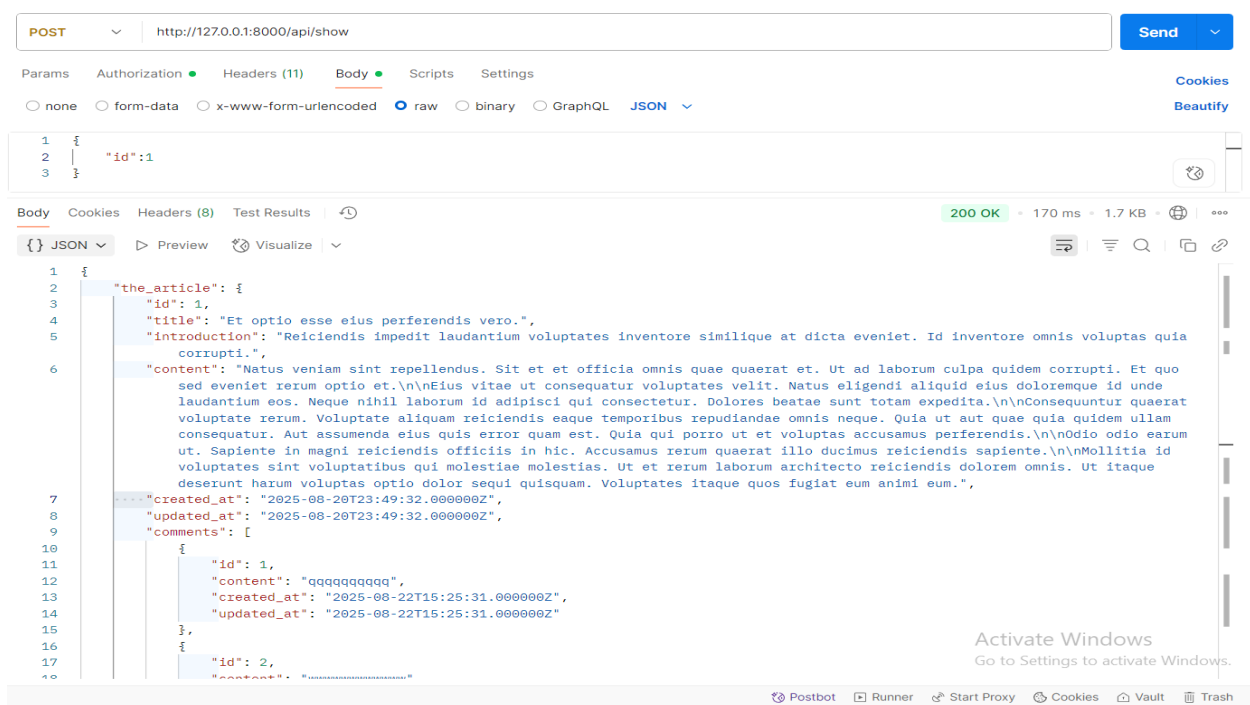
```
//Shows specefic article with post request
2 references | 0 overrides
public function show(Request $request): JsonResponse
{
    $id = $request->input(key: "id");

    $the_article = $this->article
        ->with(relations: 'comments')
        ->find(id: $id);

    if (!$the_article) {
        return response()->json(data: ['message' => 'Article not found'], status: 404);
    }

    return response()->json(data: [
        'the_article' => $the_article,
    ]);
}
```

متد **show** برای نمایش مقاله خاص می باشد ابتدا **id** مقاله توسط متد پست از طرف **postman** ارسال می شود سپس شی مقاله که از قبل توسط تابع سازنده ایجاد شده با جدول **comments** پیوند داده می شود سپس **\$the_article** اطلاعات مربوط به مقاله و دیدگاه های مربوط به آن را به صورت پاسخ نهایی به **postman** برمی گرداند:



-۵

```
Route::get(uri: "/", action: [ArticleController::class, "index"]);
Route::post(uri: "/register", action: [UserController::class, "register"]);
Route::post(uri: "/login", action: [UserController::class, "login"]);
Route::post(uri: "/show", action: [ArticleController::class, "show"]);
Route::post(uri: "/pdf", action: [ArticleController::class, "showPdf"]);

Route::middleware(middleware: "auth:sanctum")->group(callback: function (): void {
    Route::post(uri: '/store', action: [ArticleController::class, 'store']);
    Route::put(uri: '/update', action: [ArticleController::class, 'update']);
    Route::delete(uri: '/destroy', action: [ArticleController::class, 'destroy']);

    Route::post(uri: '/comment', action: [CommentController::class, 'store']);
});
```

متد store توسط sanctum middleware محدود شده و اگر کاربر api-token را به head درخواست http ضمیمه نکرده یا api-token نادرست باشد، درخواست با خطای ۴۰۱ رد می‌شود:

POST http://127.0.0.1:8000/api/store

Params Authorization Headers (11) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {}
2
3 {}
```

Body Cookies Headers (8) Test Results 401 Unauthorized

{ } JSON Preview Visualize

```
1 {
2   "message": "Unauthenticated."
3 }
```

-9

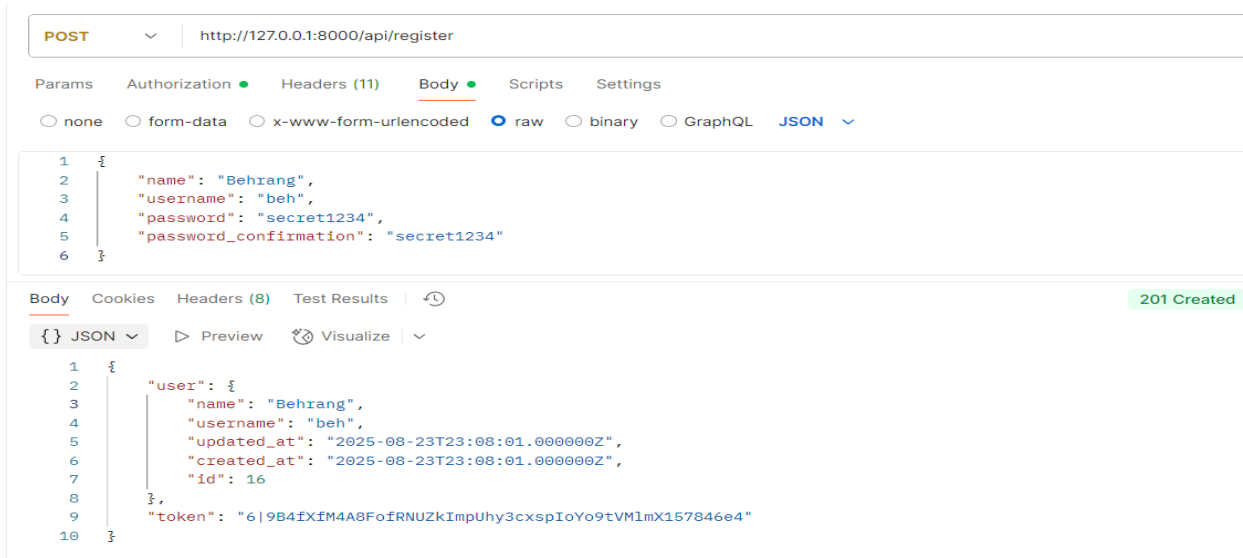
```
1 reference | 0 overrides
public function register(Request $request): JsonResponse
{
    $request->validate(rules: [
        'name' => 'required|string|max:255',
        'username' => 'required|string|max:255|unique:users,username',
        'password' => 'required|string|min:8|confirmed',
    ]);

    $user = User::create(attributes: [
        'name' => $request->name,
        'username' => $request->username,
        'password' => bcrypt(value: $request->password),
    ]);

    $token = $user->createToken(name: 'api-token')->plainTextToken;

    return response()->json(data: [
        'user' => $user,
        'token' => $token,
    ], status: 201);
}
```

متد register در کلاس UserController، \$request ارسال شده توسط postman را ابتدا با استفاده از validate اعتبارسنجی می‌کند و اگر درست باشد اطلاعات در پایگاه داده ثبت سپس api-token تولید می‌شود و اطلاعات کاربر، api-token و کد وضعیت ۲۰۱ به postman ارسال می‌شوند:



-۷

```

Reference [0] overrides
public function login(Request $request): JsonResponse
{
    $validated = $request->validate(rules: [
        'username' => 'required|string|min:3|max:50|regex:/^[a-zA-Z0-9-_.]+$/',
        'password' => 'required|string|min:6|max:255',
    ]);

    if (!Auth::attempt(credentials: $validated)) {
        return response()->json(data: ['message' => 'Invalid login details'], status: 401);
    }

    $user = User::where(column: 'username', operator: $validated['username'])->first();

    $token = $user->createToken(name: 'api-token')->plainTextToken;

    return response()->json(data: [
        'user' => $user,
        'token' => $token,
    ]);
}

```

متد login در کلاس UserController، \$request ورودی را همانند register اعتبارسنجی سپس اطلاعات ورودی را بررسی می‌کند اگر اطلاعات صحیح بودن، api-token و اطلاعات کاربر به postman

ارسال می شوند:

POST http://127.0.0.1:8000/api/login

Params Authorization Headers (11) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "username": "beh",
3   "password": "secret1234"
4 }
```

Body Cookies Headers (8) Test Results 200 OK

{ } JSON Preview Visualize

```
1 {
2   "user": {
3     "id": 16,
4     "name": "Behrang",
5     "username": "beh",
6     "username_verified_at": null,
7     "created_at": "2025-08-23T23:08:01.000000Z",
8     "updated_at": "2025-08-23T23:08:01.000000Z"
9   },
10  "token": "10|bKQBKw6kP7WjAwTtZAUzqHx8VMv18BkpjBX94eUad3d854a"
11 }
```

1 reference | 0 overrides

public function store(Request \$request): JsonResponse

{

```
$validated = $request->validate(rules: [
    'title' => 'required|string|max:255',
    'introduction' => 'required|string|max:500',
    'content' => 'required|string',
]);
```

```
$mentionParser = MentionFactory::create();
```

```
$article = $this->article->create(attributes: [
    'title' => $validated['title'],
    'introduction' => $validated['introduction'],
    'content' => $mentionParser->text_analyzer(text: $validated['content']),
    'user_id' => Auth::id(),
]);
```

```
return response()->json(data: $article, status: 201);
```

}

حال می‌توانیم از این توکن برای ذخیره مقاله جدید در پایگاه داده استفاده کنیم این توکن باید به **head** درخواست **http** ضمیمه شود متد **store** برای اعتبارسنجی (نیاز بودن فیلد، حداکثر کاراکتر و ...) و ذخیره اطلاعات در پایگاه داده پیاده‌سازی شده است:

Params Authorization ● Headers (11) Body ● Scripts Settings

Headers 8 hidden

	Key	Value
<input checked="" type="checkbox"/>	Accept	application/json
<input checked="" type="checkbox"/>	Authorization	Bearer 10 bKQBNkW6kP7WjAWtTZAuzqHx8VMv18Bkpj...

توکن ضمیمه شد حال می‌توانیم از **store** استفاده کنیم و به ما خطای ۴۰۱ برنمی‌گرداند!

POST http://127.0.0.1:8000/api/store

Params Authorization ● Headers (11) Body ● Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "title": "Store function usage",
3   "introduction": "This article about how to use api-tokens",
4   "content": "Api-token must attach to head of http request"
5 }
```

Body Cookies Headers (8) Test Results

201 Created

{ } JSON Preview Visualize

```
1 {
2   "title": "Store function usage",
3   "introduction": "This article about how to use api-tokens",
4   "content": "Api-token must attach to head of http request",
5   "updated_at": "2025-08-24T00:06:50.000000Z",
6   "created_at": "2025-08-24T00:06:50.000000Z",
7   "id": 40
8 }
```



```

public function update(Request $request): JsonResponse
{
    $id = $request->input(key: "id");

    $article = $this->article
        ->where(column: "id", operator: "=", value: $id)
        ->first();

    if ($article->user_id !== Auth::id()) {
        return response()->json(data: ['message' => 'Unauthorized'], status: 403);
    }

    $validated = $request->validate(rules: [
        'title' => 'sometimes|required|string|max:255',
        'introduction' => 'sometimes|required|string|max:500',
        'content' => 'sometimes|required|string',
    ]);

    $mentionParser = MentionFactory::create();

    $article = $article->update(attributes: [
        'title' => $validated['title'],
        'introduction' => $validated['introduction'],
        'content' => $mentionParser->text_analyzer(text: $validated['content']),
        'user_id' => Auth::id(),
    ]);

    return response()->json(data: $article);
}

```

متد update ابتدا بررسی می‌کند که user_id موجود در articles با id کاربر login کرده که حاوی api-token است، یکی می‌باشد یا خیر اگر یکی بود یعنی کاربری که در حال تغییرات بر روی مقاله می‌باشد، همان نویسنده اصلی است در غیر این صورت خطای ۴۰۳ می‌دهد و درخواست http رد می‌شود:

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://127.0.0.1:8000/api/update
- Body:**

```

1 {
2   "id": 1,
3   "title": "Update function usage",
4   "introduction": "This article is about update",
5   "content": "This request will be rejected!"
6 }

```
- Response:** 403 Forbidden
 - Body:**

```

1 {
2   "message": "Unauthorized"
3 }

```

PUT http://127.0.0.1:8000/api/update

Params Authorization Headers (11) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "id": 40,
3   "title": "Update function usage",
4   "introduction": "This article is about update",
5   "content": "This request will be accepted!"
6 }

```

Body Cookies Headers (8) Test Results 200 OK

{ } JSON Preview Visualize

```

1 {
2   "id": 40,
3   "title": "Update function usage",
4   "introduction": "This article is about update",
5   "content": "This request will be accepted!",
6   "created_at": "2025-08-24T00:06:50.000000Z",
7   "updated_at": "2025-08-24T07:57:46.000000Z"
8 }

```

این درخواست به دلیل برابر بودن id کاربر فعلی با user_id موجود در جدول articles قبول می‌شود.

-۹

```

1 reference | 0 overrides
public function destroy(Request $request): JsonResponse
{
    $id = $request->input(key: "id");

    $article = $this->article
        ->where(column: "id", operator: "=", value: $id)
        ->first();

    if ($article->user_id !== Auth::id()) {
        return response()->json(data: ['message' => 'Unauthorized'], status: 403);
    }

    $article->delete();

    return response()->json(data: ['message' => 'Deleted successfully']);
}

```

قوانین update برای متد destroy صادق می باشد!

DELETE http://127.0.0.1:8000/api/destroy

Params Authorization Headers (11) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": 1,
3   "title": "Update function usage",
4   "introduction": "This article is about update",
5   "content": "This request will be rejected!"
6 }
```

Body Cookies Headers (8) Test Results 403 Forbidden

JSON Preview Visualize

```
1 {
2   "message": "Unauthorized"
3 }
```

DELETE http://127.0.0.1:8000/api/destroy

Params Authorization Headers (11) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": 40,
3   "title": "Update function usage",
4   "introduction": "This article is about update",
5   "content": "This request will be rejected!"
6 }
```

Body Cookies Headers (8) Test Results 200 OK

JSON Preview Visualize

```
1 {
2   "message": "Deleted successfully"
3 }
```

-۱۰

```
1 reference | 0 overrides
public function showPdf(Request $request): JsonResponse|Response
{
    $id = $request->input(key: "id");

    $article = $this->article->find(id: $id);

    if (!$article) {
        return response()->json(data: ['message' => 'Article not found'], status: 404);
    }

    $html = '
    <h1 style="text-align:center;"> ' . htmlspecialchars(string: $article->title) . '</h1>
    <p><strong>Introduction:</strong> ' . htmlspecialchars(string: $article->introduction) . '</p>
    <hr>
    <p> ' . $article->content . '</p>
    ';

    $pdf = PDF::loadHTML(string: $html);

    return $pdf->stream(filename: 'article.pdf');
}
```

متد showPdf خروجی pdf تولید می کند:

POST http://127.0.0.1:8000/api/pdf

Params Authorization Headers (11) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": 1
3 }
```

200 OK • 1.44 s • 2.61 KB

Body Cookies Headers (9) Test Results

Hex Preview Visualize

Et optio esse eius perferendis vero.

Introduction: Reiciendis impedit laudantium voluptates inventore similique at dicta eveniet. Id inventore omnis voluptas quia corrupti.

Natus veniam sint repellendus. Sit et et officia omnis quae quaerat et. Ut ad laborum culpa quidem corrupti. Et quo sed eveniet rerum optio et. Eius vitae ut consequatur voluptates velit. Natus eligendi aliquid eius doloreque id unde laudantium eos. Neque nihil laborum id adipisci qui consectetur. Dolores beatae sunt totam expedita. Consequuntur quaerat voluptate rerum. Voluptate aliquam reiciendis eaque temporibus repudiandae omnis neque. Quia ut aut quae quia quidem ullam consequatur. Aut assumenda eius quis error quam est. Quia qui porro ut et voluptas accusamus perferendis. Odio odio earum ut. Sapiente in magni reiciendis officiis in hic. Accusamus rerum quaerat illo ducimus reiciendis sapiente. Mollitia id voluptates sint voluptatibus qui molestiae molestias. Ut et rerum laborum architecto reiciendis dolorem omnis. Ut itaque deserunt harum voluptas optio dolor sequi quisquam. Voluptates itaque quos fugiat eum animi eum.

Activate Windows
Go to Settings to activate Windows.

- ۱۱

```
<?php

namespace App;

2 references | 0 implementations
class Mention
{
    2 references | 0 overrides
    public function text_analyzer(string $text): string
    {
        return preg_replace_callback(
            pattern: '/@(\w+)\((.*)?\)/',
            callback: function ($matches): string {
                $username = $matches[1];
                $link = $matches[2];
                return "<a href=\"{$link}\">@{$username}</a>";
            },
            subject: $text
        );
    }
}
```

متد `text_analyzer` متن مقاله را دریافت می‌کند سپس می‌گردد دنبال کاراکتر `@` سپس اون عبارت را به صورت آرایه به تابع بی نام به عنوان پارامتر ارسال می‌کند قسمت اول تطابق در `$username` و قسمت دوم در `link` ریخته می‌شود سپس نتیجه نهایی به صورت تگ `a` به تابع `preg_replace_callback` برگشت داده می‌شود و تابع `preg_replace_callback` نتیجه نهایی را به `text_analyzer` بر می‌گرداند

```
<?php

namespace App;

use App\Mention;

3 references | 0 implementations
class MentionFactory
{
    2 references | 0 overrides
    public static function create(): Mention
    {
        return new Mention();
    }
}
```

کلاس `MentionFactory` شامل متد استاتیک `create` می‌باشد و شی `Mention` را بر می‌گرداند

```
$mentionParser = MentionFactory::create();

$article = $this->article->create(attributes: [
    'title' => $validated['title'],
    'introduction' => $validated['introduction'],
    'content' => $mentionParser->text_analyzer(text: $validated['content']),
    'user_id' => Auth::id(),
]);
```

در قسمت اول شی ساخته می‌شود در قسمت `content` محتوای مقاله به متد `text_analyzer` برای منشن کردن ارسال می‌شود برای متد `update` این قاعده صادق می‌باشد؛ مثلاً:

POST

▼

http://127.0.0.1:8000/api/store

Params

Authorization ●

Headers (11)

Body ●

Scripts

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON ▼

```

1  {
2    "title": "Mention",
3    "introduction": "In this article there is a mention",
4    "content": "Behrang trys Giltarah @beh(https://giltarah.com/) "
5  }

```

Body

Cookies

Headers (8)

Test Results

↺

201 Created

{ } JSON ▼

▶ Preview

🔗 Visualize ▼

```

1  {
2    "title": "Mention",
3    "introduction": "In this article there is a mention",
4    "content": "Behrang trys Giltarah <a href=\"https://giltarah.com/\">@beh</a>",
5    "updated_at": "2025-08-24T11:35:49.000000Z",
6    "created_at": "2025-08-24T11:35:49.000000Z",
7    "id": 49
8  }

```

Acti

Go to

-۱۲

```

1 reference | 0 overrides
public function store(Request $request): JsonResponse
{
    $request->validate(rules: [
        'article_id' => 'required|integer|exists:articles,id',
        'content' => 'required|string|max:1000',
    ]);

    $comment = Comment::create(attributes: [
        'article_id' => $request->input(key: "article_id"),
        "content" => $request->input(key: "content")
    ]);

    return response()->json(data: [
        'message' => 'Comment added successfully',
        'comment' => $comment
    ]);
}

```

متد `store` در کلاس `CommentController` برای اضافه کردن دیدگاه به یک مقاله با `id` منحصر فرد می‌باشد. `id` در `article_id` و `content` در `content` جدول `comments` ذخیره می‌شوند و هنگام نمایش یک مقاله خاص، آن مقاله با جدول `comments` پیوند زده می‌شود و دیدگاه‌های مربوط به کاربران به همراه آن مقاله نمایش داده می‌شوند؛ متد `show` برای همین منظور پیاده‌سازی شده است؛ مثلاً:

POST http://127.0.0.1:8000/api/comment

Params Authorization Headers (11) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "article_id": 49,
3   "content": "This is a comment"
4 }
```

Body Cookies Headers (8) Test Results 200 OK

{ } JSON Preview Visualize

```
1 {
2   "message": "Comment added successfully",
3   "comment": {
4     "content": "This is a comment",
5     "updated_at": "2025-08-24T14:11:52.000000Z",
6     "created_at": "2025-08-24T14:11:52.000000Z",
7     "id": 11
8   }
9 }
```

Acti
Go to

بهرنگ آباد فومنی