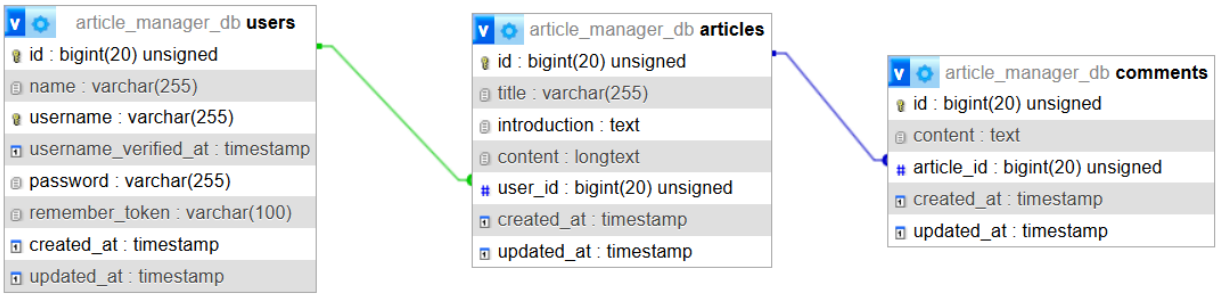



۱- برای سه موجودیت، جداول users, articles, comments در پایگاه داده MySql پیاده‌سازی شده اند. جدول articles دارای فیلدهای مربوط به ساختار مقاله و user_id می‌باشد که یک کلید خارجی و ارجاعی به فیلد اصلی id در جدول users می‌باشد. جدول comments حاوی اطلاعات مربوط به یک دیدگاه و فیلد article_id که یک کلید 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، کلاس های User, Article, Comment در نظر گرفته شده است. در این کلاس ها فیلدهای قابل نوشتن در پایگاه داده و همچنین نوع روابط بین جداول پیاده‌سازی شده‌اند؛ مثلاً:

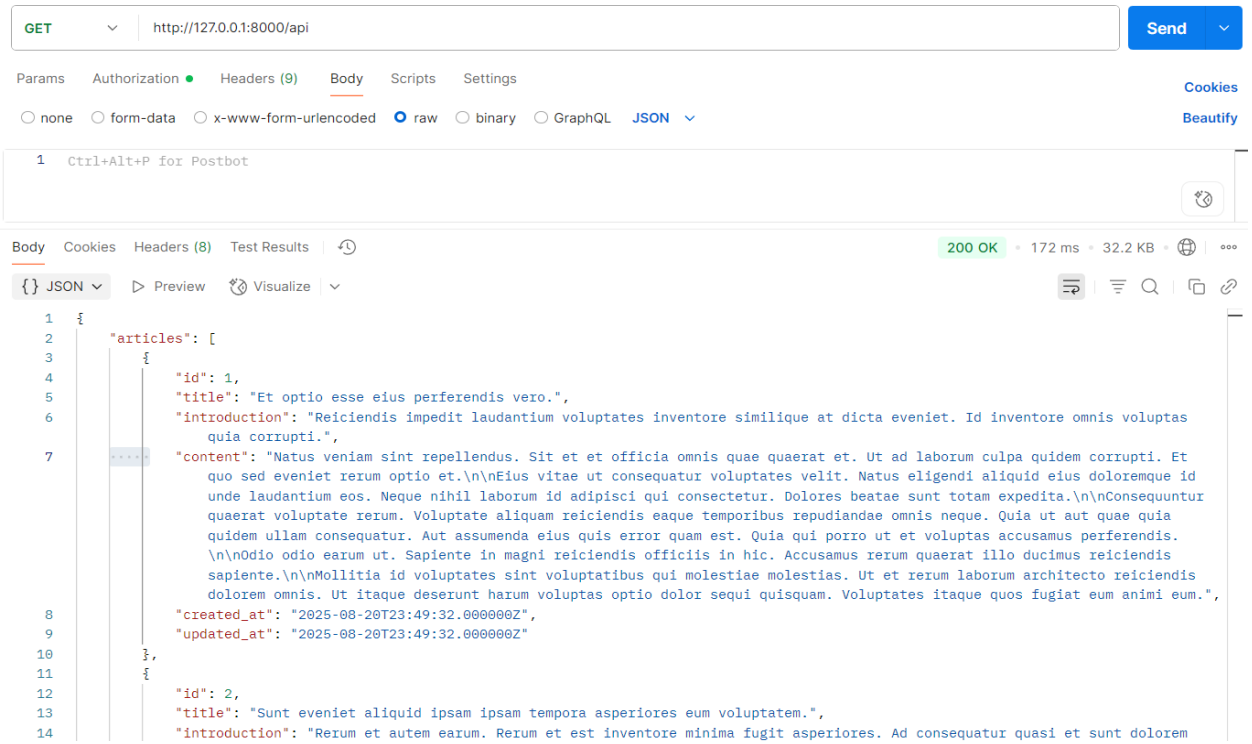
```php
public function user(): BelongsTo
{
    return $this->belongsTo(related: User::class, foreignKey: 'user_id');
}
//Each article has many comments
0 references | 0 overrides
public function comments(): HasMany
{
    return $this->hasMany(related: Comment::class, foreignKey: 'article_id');
}
```

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

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

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

    return response()->json(data: [
        'articles' => $articles
    ]);
}
//Shows specefic article with post reque
```

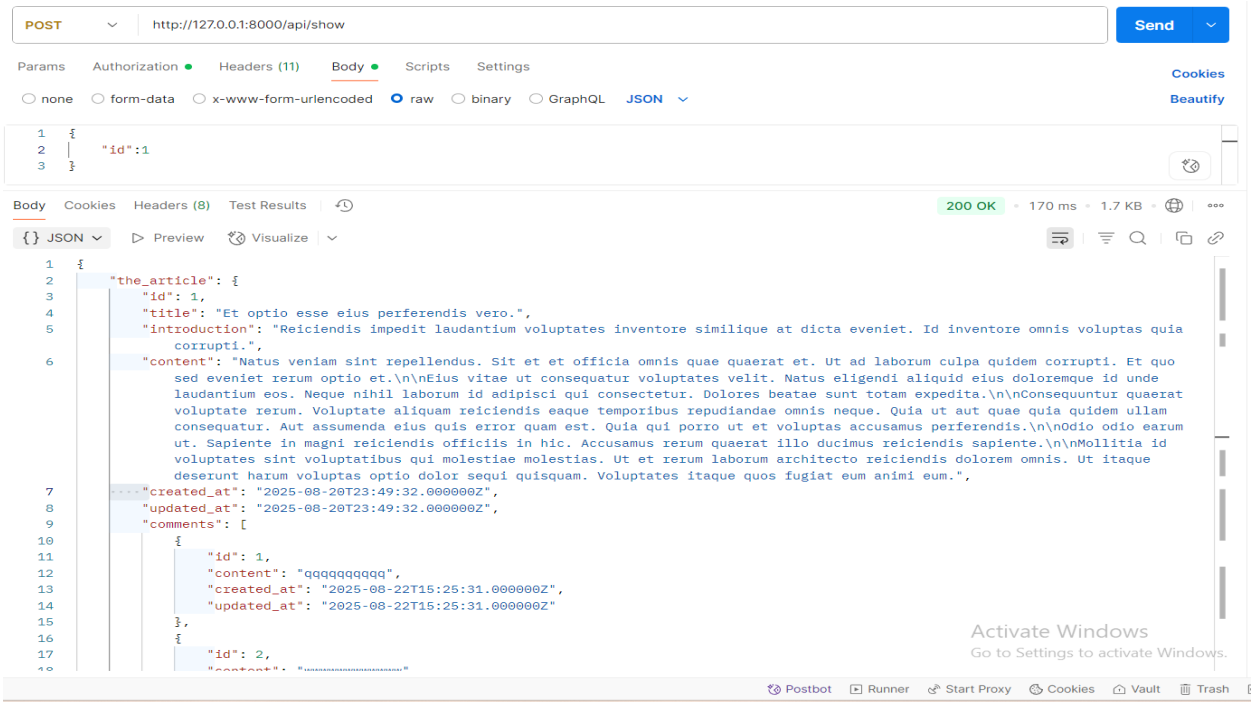
در کلاس ArticleController متد index برای نمایش همه مقاله‌ها در postman می‌باشد:



۴-

```php
//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 می‌باشد ابتدا id مقاله توسط متد پست از طرف postman ارسال می‌شود سپس شی مقاله که از قبل توسط تابع سازنده ایجاد شده با جدول comments پیوند داده می‌شود

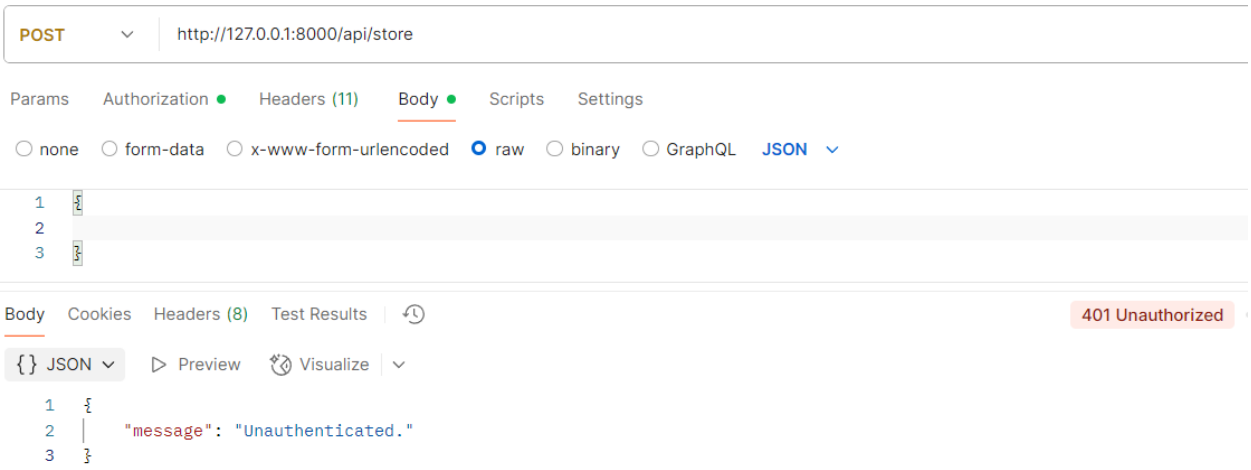


سپس $the_article اطلاعات مربوط به مقاله و دیدگاه های مربوط به آن را به صورت پاسخ نهایی به postman برمی‌گرداند:

۵-

```php
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']);

});
```

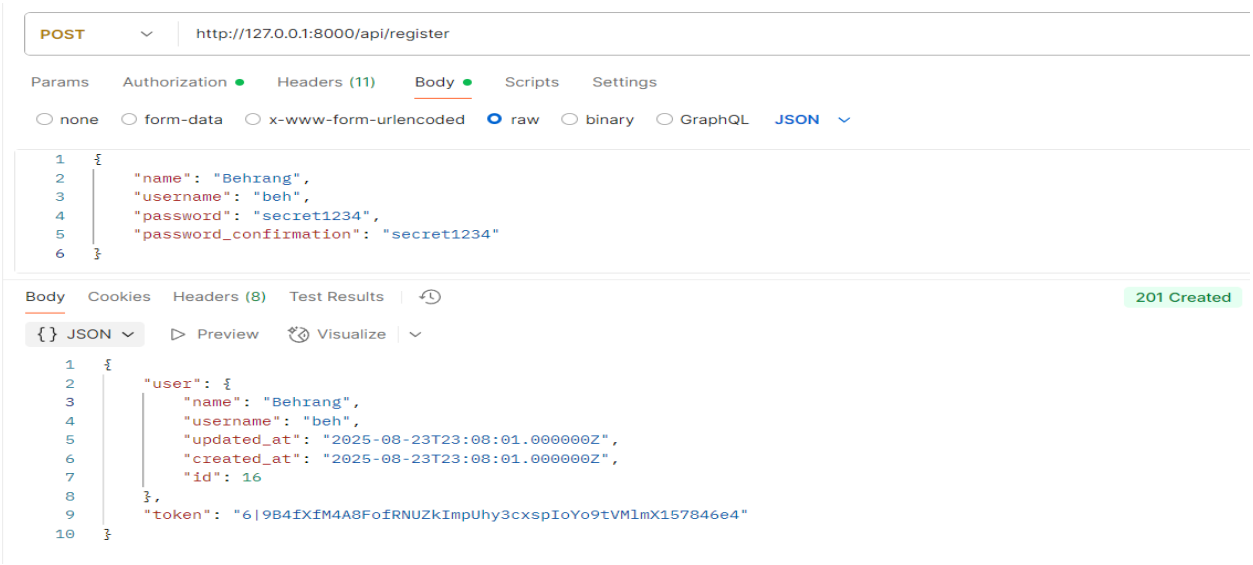
متد store توسط sanctum middleware محدود شده و اگر کاربر api-token را به head درخواست http ضمیمه نکرده یا api-token نادرست باشد، درخواست با خطای ۴۰۱ رد می‌شود:




۶-

```php
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);
}
```

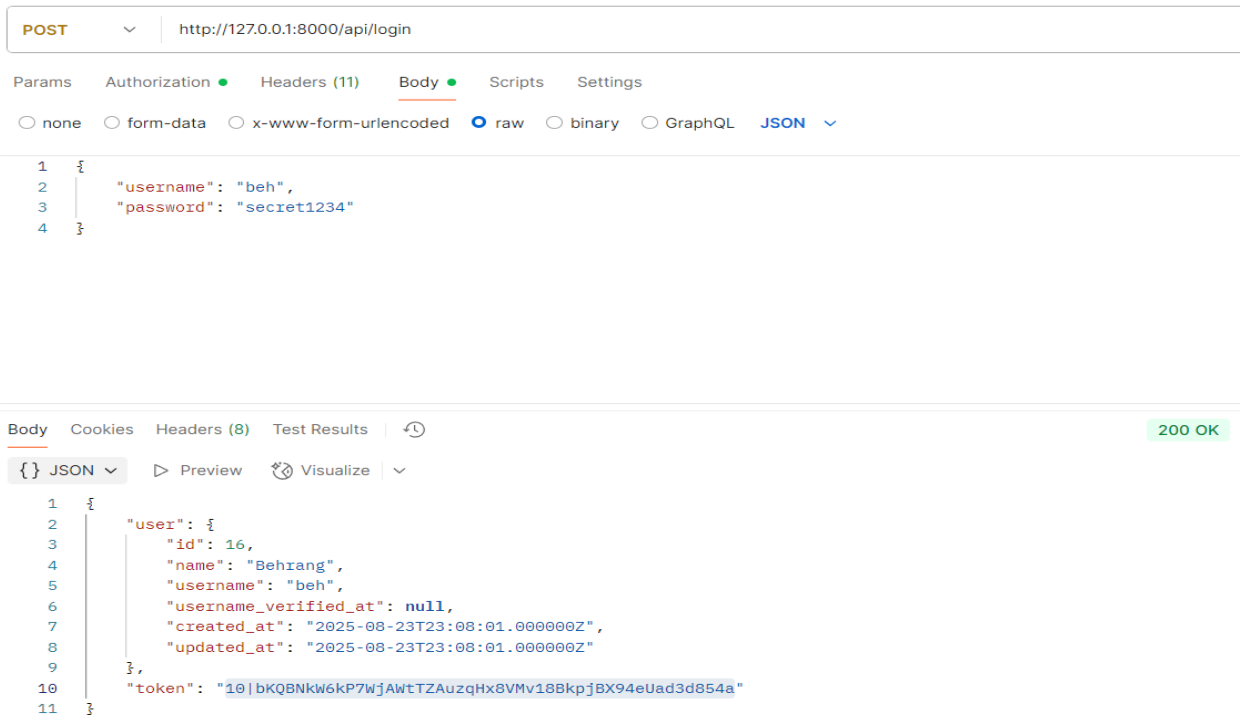
متد register در کلاس UserController، $request ارسال شده توسط postman را ابتدا با استفاده validate اعتبارسنجی می‌کند و اگر درست باشد اطلاعات در پایگاه داده ثبت سپس api-token تولید می‌شود و اطلاعات کاربر، api-token و کد وضعیت ۲۰۱ به postman ارسال می‌شوند:



۷-

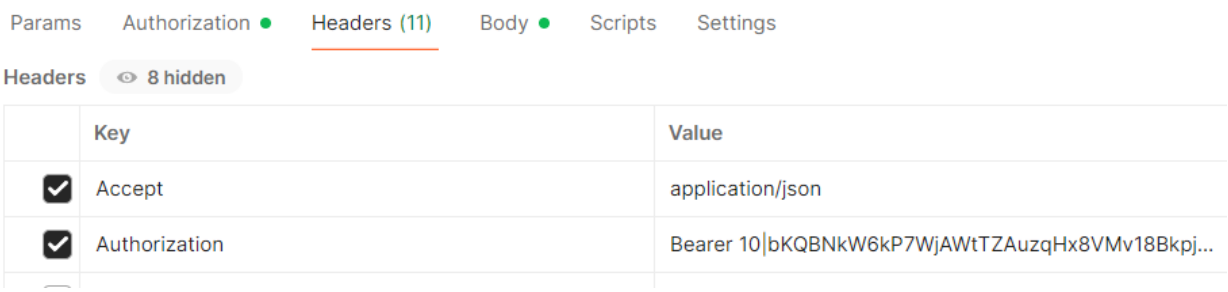
```php
1 reference | 0 overrides
public function login(Request $request): JsonResponse
{
    $validated = $request->validate(rules: [
        'username' => 'required|string|min:3|max:50|regex:/^[a-zA-Z0-9._-]+$/',
        'password' => 'required|string|min:6|max:255',
    ]);

    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 می‌شوند:



```php
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 ار 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:




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

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

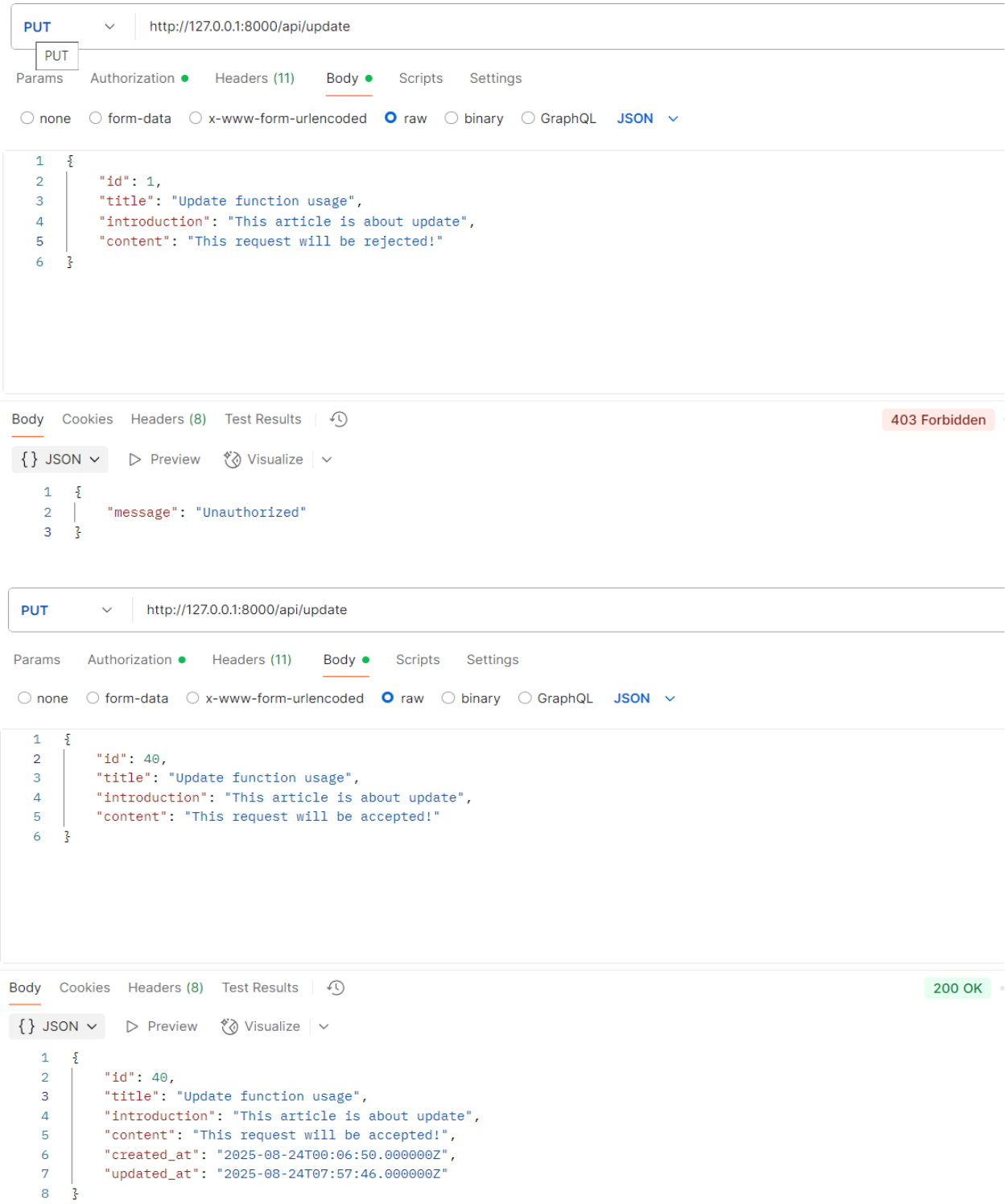
Body Cookies Headers (8) Test Results 201 Created

{} JSON Preview Visualize

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

```php
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 می دهد و درخواست http رد می‌شود:

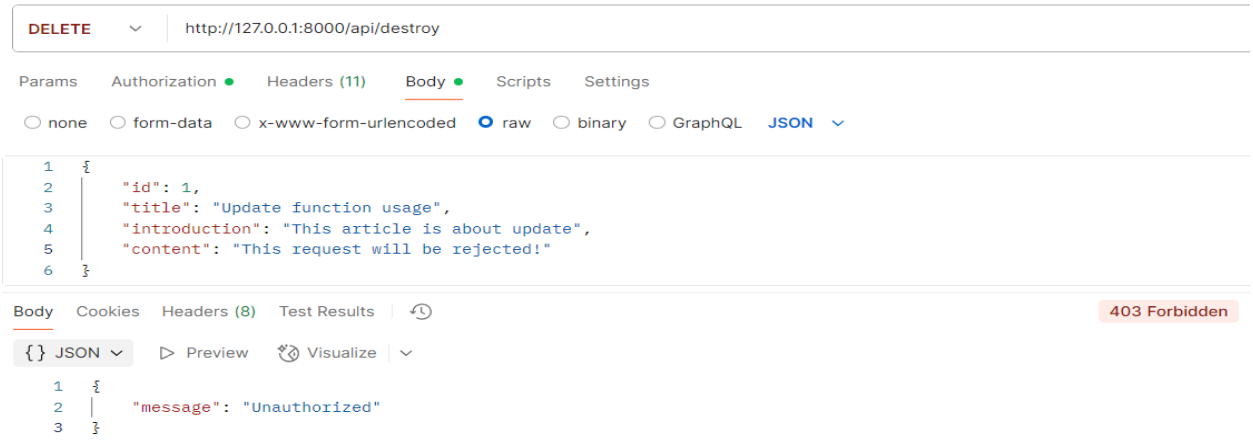


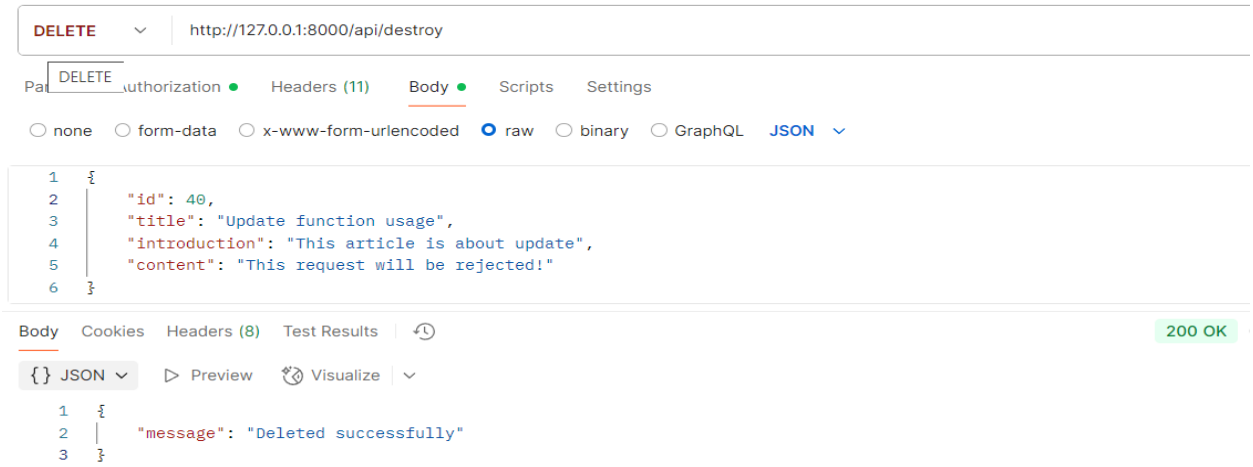
این درخواست به دلیل برابر بودن id کاربر فعلی با user_id موجود در جدول articles قبول می‌شود.

۹-

```php
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']);
}
```

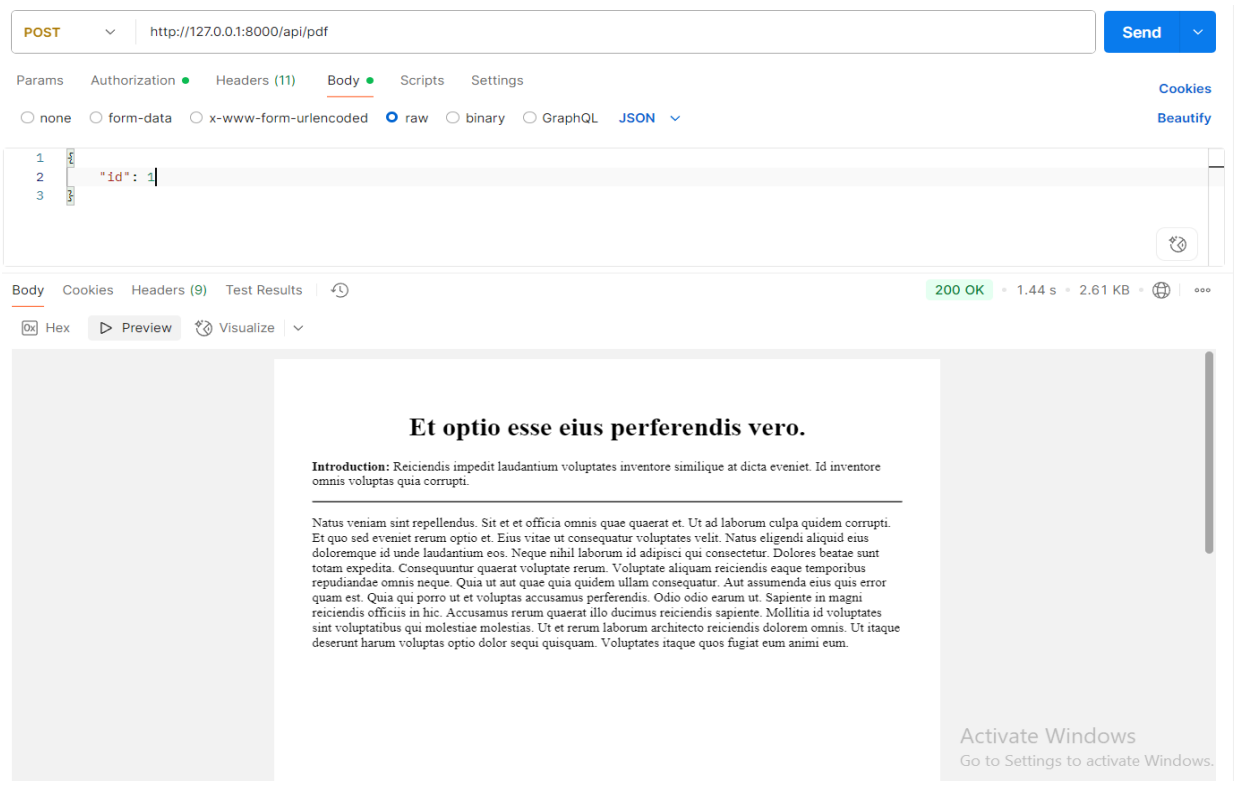
قوانین update برای متد destroy صادق می‌باشد!






```php
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 می‌کند:




-۱۱

```php
<?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
<?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

```php
$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(),
]);
```

درقسمت 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

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

Body Cookies Headers (8) Test Results 201 Created

{} JSON Preview Visualize

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



-۱۲

```php
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: