اند. جدول articles دارای فیلدهای مربوط به ساختار مقاله و user_id میباشد که یک کلید خارجی و اند. جدول articles دارای فیلدهای مربوط به ساختار مقاله و comments میباشد که یک کلید خارجی و ارجاعی به فیلد اصلی id در جدول users میباشد. جدول article دارای فیلد اصلی و منحصر به فرد id در جدول دیدگاه و فیلد اصلی و منحصر به فرد id در جدول دیدگاه و فیلد اصلی و منحصر به فرد id در جدول article_id که یک کلید خارجی و ارجاعی به فیلد اصلی و منحصر به فرد است، articles میباشد. جدول username که یک فیلد منحصر به فرد است، میباشد.

۲- برای جداول پیادهسازی شده، در قسمت Model لاراول، کلاس های Model در نظر گرفته شده است. در این کلاس ها 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 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 Controller, حاوی کلاسهای CommentController

در کلاس ArticleController متد index برای نمایش همه مقالهها در

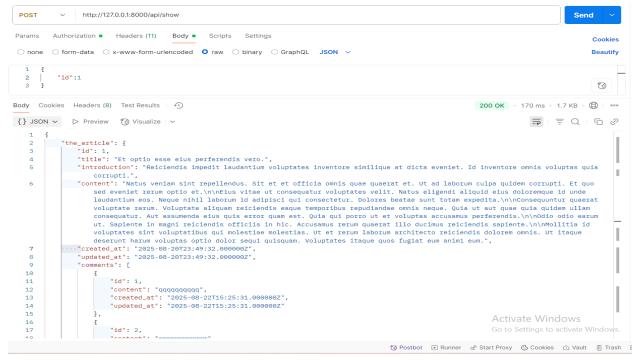
```
GET
           http://127.0.0.1:8000/api
        Authorization • Headers (9) Body Scripts Settings
Params
                                                                                                                                            Cookies
 ○ none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL JSON ∨
                                                                                                                                            Beautify
  1 Ctrl+Alt+P for Postbot
                                                                                                                                             (*)
Body Cookies Headers (8) Test Results
                                                                                                               200 OK • 172 ms • 32.2 KB • 🖨 👓
{} JSON ✓ ▷ Preview 🍪 Visualize ✓
                                                                                                                             = Q 1 6 0
            "articles": [
                   "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 molestiae. 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"
   10
   11
                   "id": 2,
   12
                   "title": "Sunt eveniet aliquid ipsam ipsam tempora asperiores eum voluptatem.",
   13
                   "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 میباشد ابتدا id مقاله توسط متد پست از طرف postman ارسال میشود میشود میشود سپس شی مقاله که از قبل توسط تابع سازنده ایجاد شده با جدول comments پیوند داده میشود سپس sthe_article اطلاعات مربوط به مقاله و دیدگاه های مربوط به آن را به صورت پاسخ نهایی به postman برمی گرداند:



```
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 sanctum middleware متد درخواست با فطای ۴۰۱ رد می شود:

```
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 \$request ،UserController ارسال شده توسط postman را ابتدا با استفاده api-token و اگر درست باشد اطلاعات در پایگاه داده ثبت سپس api-token تولید می کند و اگر درست باشد اطلاعات در پایگاه داده ثبت سپس postman و کد وضعیت ۲۰۱ به postman ارسال می شوند:

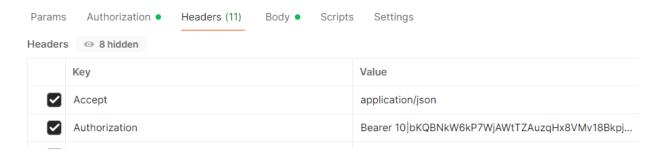
```
POST
           http://127.0.0.1:8000/api/register
Params
         Authorization • Headers (11) Body • Scripts Settings
○ none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL JSON ∨
  1
  2
          "name": "Behrang",
          "username": "beh",
  3
          "password": "secret1234",
  5
          "password_confirmation": "secret1234"
  6
Body Cookies Headers (8) Test Results
                                                                                                          201 Created
{} JSON ∨ ▷ Preview 🖔 Visualize ∨
   2
           "user": {
   3
              "name": "Behrang",
              "username": "beh",
              "updated_at": "2025-08-23T23:08:01.000000Z",
   5
             "created_at": "2025-08-23T23:08:01.000000Z",
             "id": 16
   7
   8
           "token": "6|9B4fXfM4A8FofRNUZkImpUhy3cxspIoYo9tVMlmX157846e4"
   9
   10 }
```

_٧

متد login در کلاس register اعتبارسنجی سپس پاrequest ،UserController ورودی را همانند register اعتبارسنجی سپس postman و اطلاعات ورودی را بررسی می کند اگر اطلاعات صحیح بودن، api-token و اطلاعات کاربر به ارسال می شوند:

```
http://127.0.0.1:8000/api/login
 POST
Params
       Authorization • Headers (11) Body • Scripts Settings
○ none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL JSON ∨
  2
        "username": "beh",
  3
        "password": "secret1234"
Body Cookies Headers (8) Test Results
                                                                                            200 OK
{} JSON ✓ ▷ Preview 🍪 Visualize ✓
         "user": {
   3
            "id": 16,
            "name": "Behrang",
            "username": "beh",
            "username_verified_at": null,
            "created at": "2025-08-23T23:08:01.000000Z",
            "updated_at": "2025-08-23T23:08:01.0000000Z"
   8
         "token": "10|bKQBNkW6kP7WjAWtTZAuzqHx8VMv18BkpjBX94eUad3d854a"
  10
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(),
     1);
     return response()->json(data: $article, status: 201);
```

حال می توانیم 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 می توانیم از store استفاده کنیم و به ما خطای ۴۰۱ برنمی گرداند!

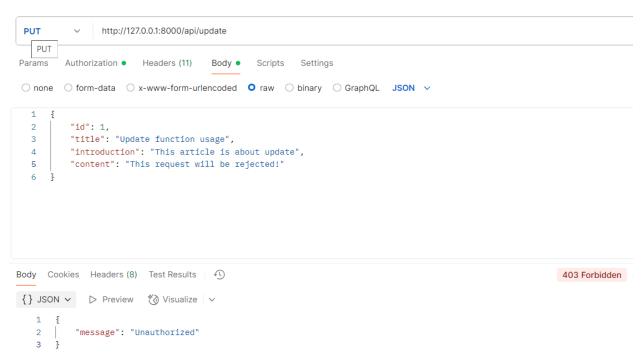
```
POST
           http://127.0.0.1:8000/api/store
Params
         Authorization • Headers (11)
                                      Body •
                                               Scripts
                                                       Settings
○ none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL JSON ∨
  1
          "title": "Store function usage",
          "introduction": "This article about how to use api-tokens",
  3
          "content":"Api-token must attach to head of http request"
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",
           "updated_at": "2025-08-24T00:06:50.000000Z",
```

5

7 8 "created_at": "2025-08-24T00:06:50.000000Z",

```
$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',
  1);
  $mentionParser = MentionFactory::create();
  $article = $article->update(attributes: [
   'title' => $validated['title'],
   'introduction' => $validated['introduction'],
       'content' => $mentionParser->text_analyzer(text: $validated['content']),
       'user_id' => Auth::id(),
  1);
  return response()->json(data: $article);
```

متد update ابتدا بررسی می کند که user_id موجود در update با login کرده که حاوی user_id کرده که حاوی articles با api-token است، یکی می باشد یا خیر اگر یکی بود یعنی کاربری که در حال تغییرات بر روی مقاله می باشد، همان نویسنده اصلی است در غیر این صورت خطای ۴۰۳ می دهد و در خواست http رد می شود:



```
PUT
           http://127.0.0.1:8000/api/update
         Authorization • Headers (11) Body • Scripts Settings
Params
○ none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL JSON ∨
          "id": 40.
  2
  3
          "title": "Update function usage",
  4
          "introduction": "This article is about update",
  5
          "content": "This request will be accepted!"
Body Cookies Headers (8) Test Results
                                                                                                              200 OK
{} JSON ∨ ▷ Preview 🍪 Visualize ∨
           "id": 40,
   2
   3
           "title": "Update function usage",
   4
           "introduction": "This article is about update",
           "content": "This request will be accepted!",
           "created_at": "2025-08-24T00:06:50.000000Z",
   6
           "updated_at": "2025-08-24T07:57:46.000000Z"
```

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

-٩

```
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);
    }

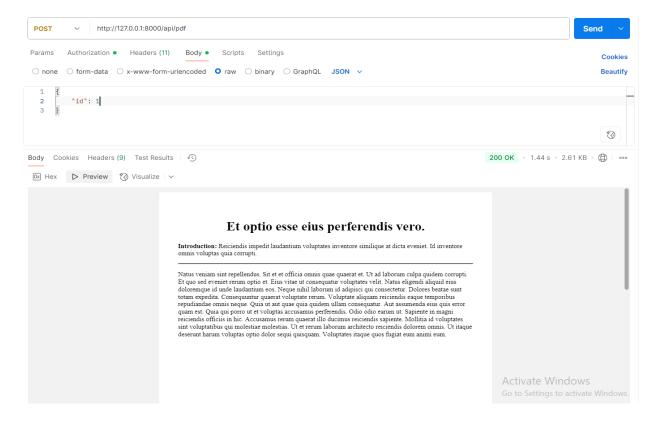
    $article->delete();

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

قوانین update برای متد destroy صادق می باشد!

```
DELETE V 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 ∨
           "id": 1,
           "title": "Update function usage",
           "introduction": "This article is about update",
           "content": "This request will be rejected!"
Body Cookies Headers (8) Test Results
                                                                                                              403 Forbidden
{} JSON ∨ ▷ Preview 👸 Visualize ∨
            "message": "Unauthorized"
           http://127.0.0.1:8000/api/destroy
  DELETE
 Pa DELETE uthorization • Headers (11) Body • Scripts Settings
 \bigcirc none \bigcirc form-data \bigcirc x-www-form-urlencoded \bigcirc raw \bigcirc binary \bigcirc GraphQL JSON \checkmark
   2
           "id": 40,
           "title": "Update function usage",
   3
           "introduction": "This article is about update",
   4
           "content": "This request will be rejected!"
Body Cookies Headers (8) Test Results
                                                                                                                    200 OK
 {} JSON ✓ ▷ Preview 🍪 Visualize ✓
            "message": "Deleted successfully"
```

متد showPdf خروجی pdf تولید می کند:



-11

متد text_analyzer متن مقاله را دریافت می کند سپس می گردد دنبال کاراکتر @ سپس اون عبارت را به مورت آرایه به تابع بی نام به عنوان پارامتر ارسال می کند قسمت اول تطابق در susername و قسمت دوم در link ریخته می شود سپس نتیجه نهایی به صورت تگ a به تابع preg_replace_callback بر گشت داده می شود و تابع preg_replace_callback بر می گرداند

کلاس Mention شامل متد استاتیک create میباشد و شی Mention را بر می گرداند

```
$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 می شود در قسمت content محتوای مقاله به متد text_analyzer برای منشن کردن ارسال می شود برای متد update این قاعده صادق می باشد؛ مثلاً:

```
Params Authorization • Headers (11) Body • Scripts Settings

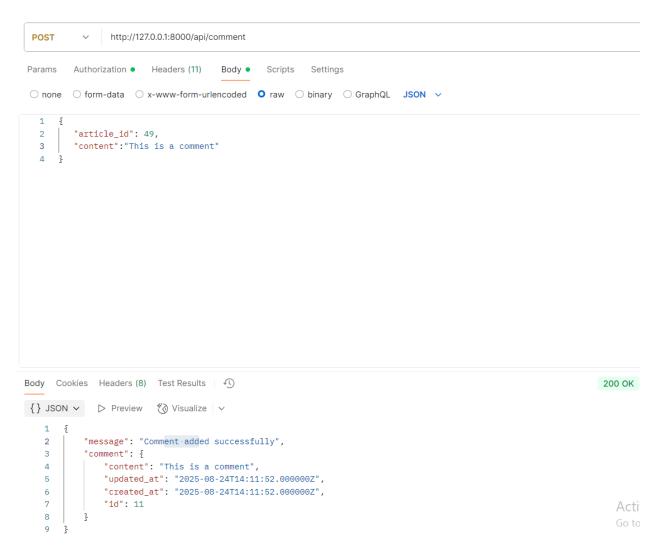
one of form-data ox-www-form-urlencoded oraw binary of GraphQL JSON v

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

```
201 Created
Body Cookies Headers (8) Test Results
{} JSON ✓ ▷ Preview 🍪 Visualize ✓
   1
           "title": "Mention",
   2
   3
           "introduction": "In this article there is a mention",
           "content": "Behrang trys Giltarah <a href=\"https://giltarah.com/\">@beh</a>",
   4
           "updated_at": "2025-08-24T11:35:49.000000Z",
   5
           "created_at": "2025-08-24T11:35:49.000000Z",
   7
           "id": 49
                                                                                                                 Acti
```

-17

متد store در کلاس CommentController برای اضافه کردن دیدگاه به یک مقاله با id منحصر فرد می store در کلاس content در content در content در content دخیره می شوند و هنگام نمایش می اشد. id در article_id در content و content در content بیوند زده می شود و دیدگاه های مربوط به کاربران به همراه آن مقاله نامیش داده می شوند؛ متد show برای همین منظور پیاده سازی شده است؛ مثلاً:



بهرنگ آباد فومنی