Topic: 14. DRF Pagination

Speaker: Personal | Notebook: API Development using Django Framework

---

Django Pagination:

Documentation on Django Rest Framework Pagination

REST framework includes support for customizable pagination styles. This allows you to modify how large result sets are split into individual pages of data.

The pagination API can support either:

- Pagination links that are provided as part of the content of the response.
- Pagination links that are included in response headers, such as `Content-Range` or `Link`.

The built-in styles currently all use links included as part of the content of the response. This style is more accessible when using the browsable API.

Pagination is only performed automatically if you're using the generic views or viewsets. If you're using a regular `APIView`, you'll need to call into the pagination API yourself to ensure you return a paginated response. See the source code for the `mixins.ListModelMixin` and `generics.GenericAPIView` classes for an example.

Pagination can be turned off by setting the pagination class to `None`.

Two most commonly used pagination:

`PageNumberPagination` - takes a page_size as a parameter and returns a response accordingly.

example: /blogs/?page=10

`LimitOffsetPagination` -

The **limit** parameter sets the number of items you want to see on a single page.

The **offset** parameter tells the API where to start fetching the items from.

example for a blog post of 100:

/blogs/?limit=10&offset=0 * this means get the first 10 blog posts(items 1-10)

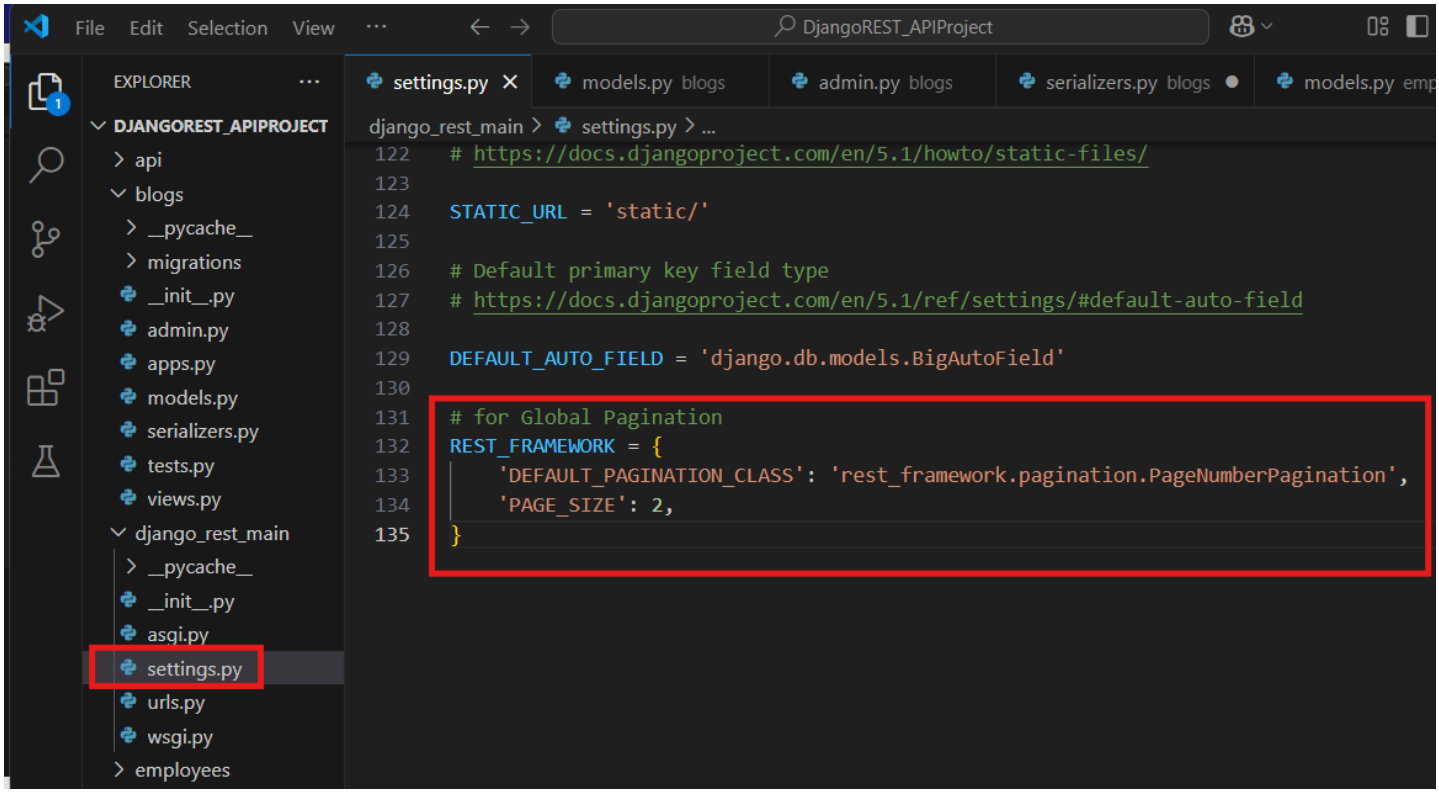
/blogs/?limit=10&offset=10 * this means get the next 10 blog posts(items 11-20)

/blogs/?limit=10&offset=90 * this means get the last 10 blog posts(items 91-100)

You can apply pagination using **Global Pagination** and using **Custom Pagination**.

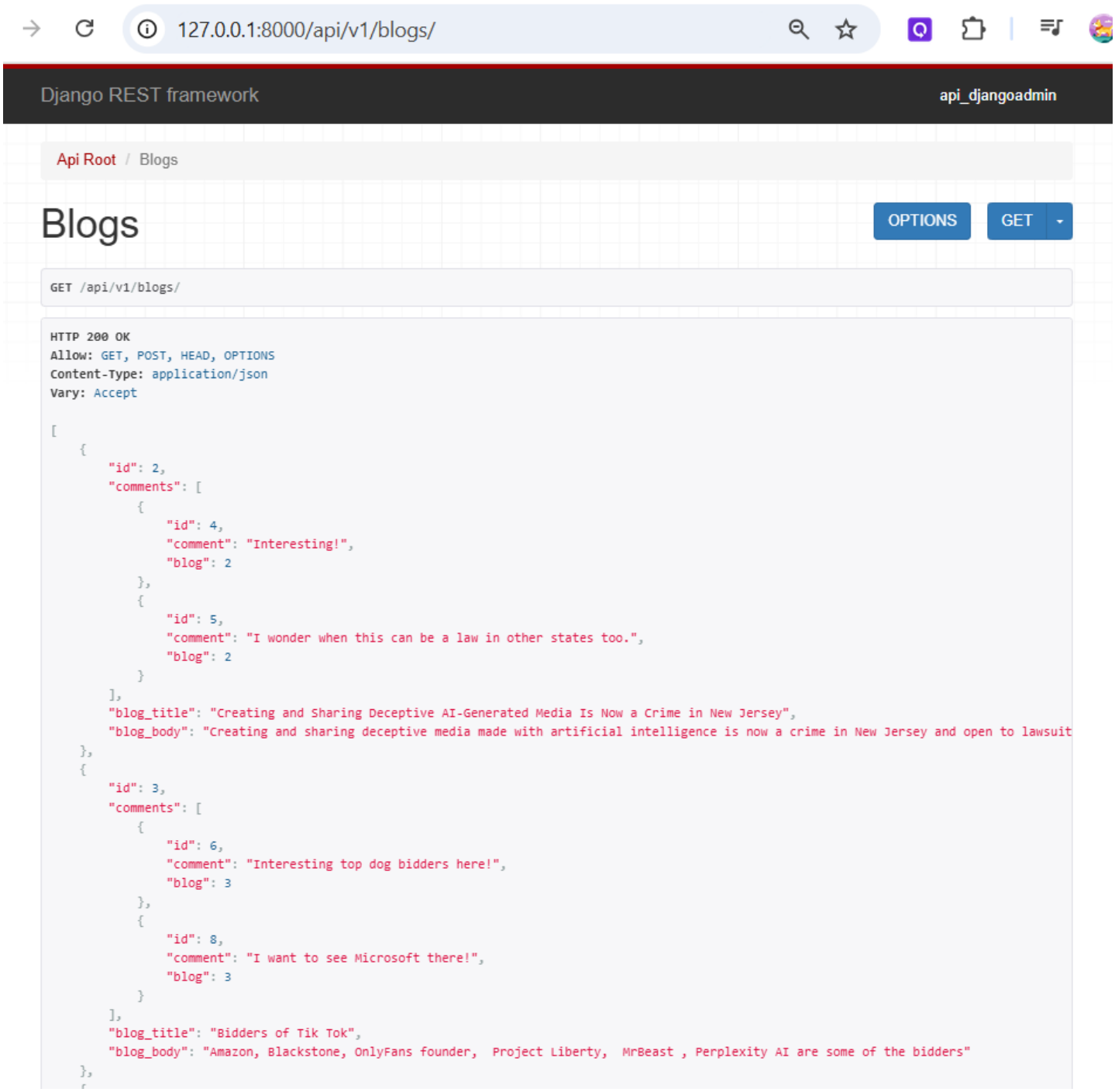
Global Pagination:

1. Go to SETTINGS.PY, and add the `default_pagination_class`: This works only with generics and viewsets.



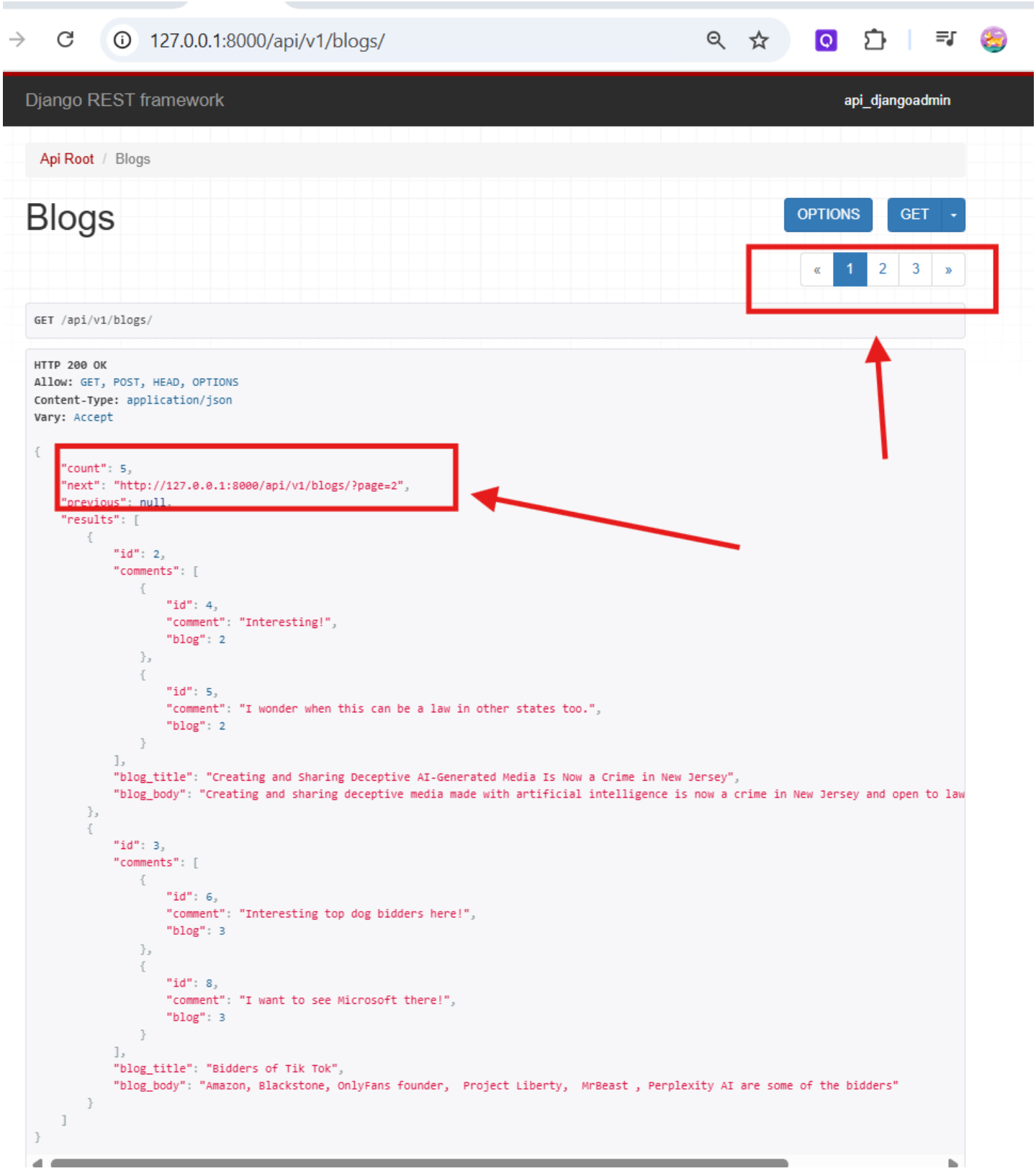
2. So, now when you reload your blog page:

BEFORE GENERAL PAGINATION, it lists all the blog posts on one page.

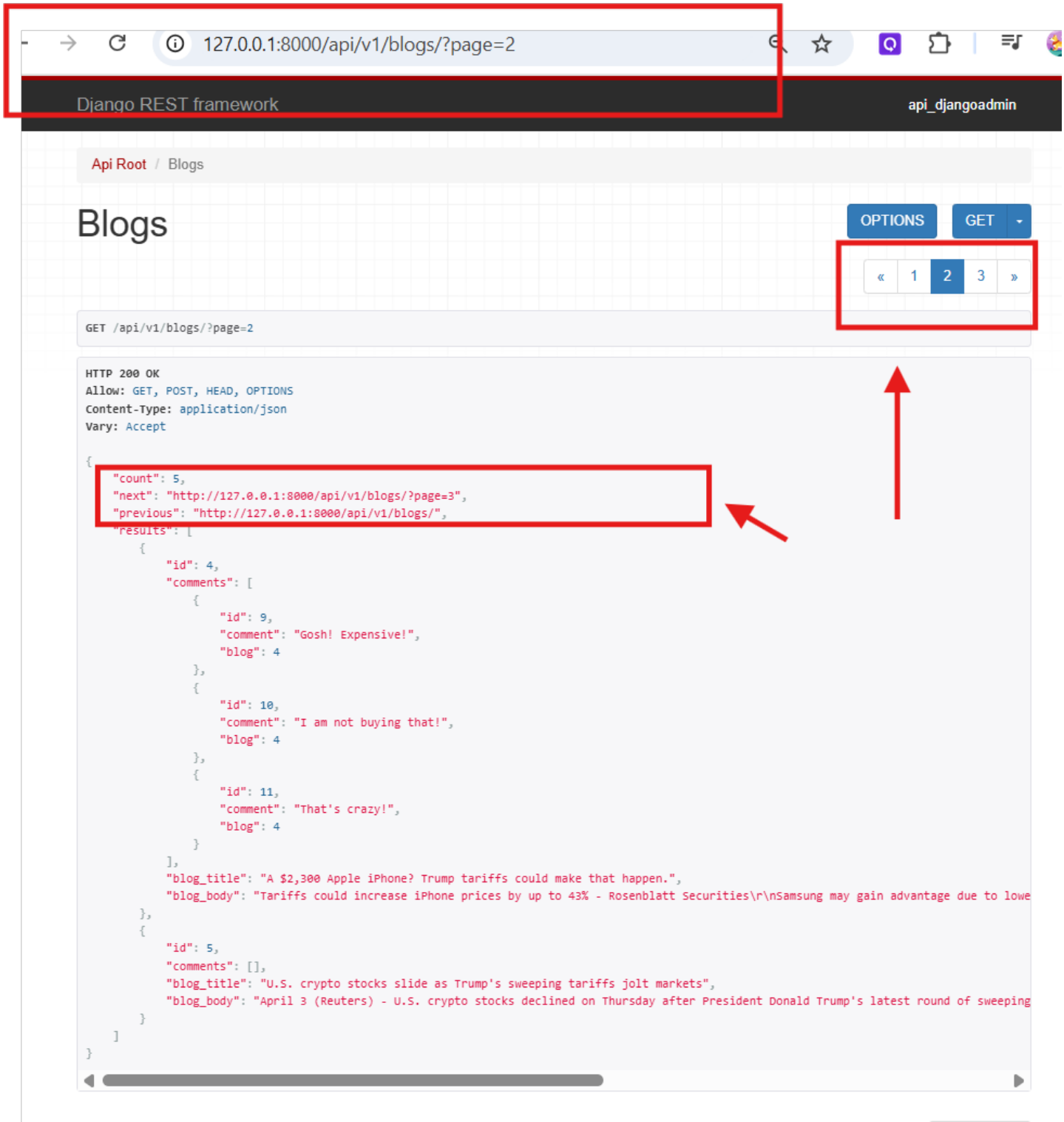


AFTER GENERAL PAGINATION, it lists 2 blog posts per page and shows page numbers. There were a total of 5 blog posts saved in a database model, BLOGS.

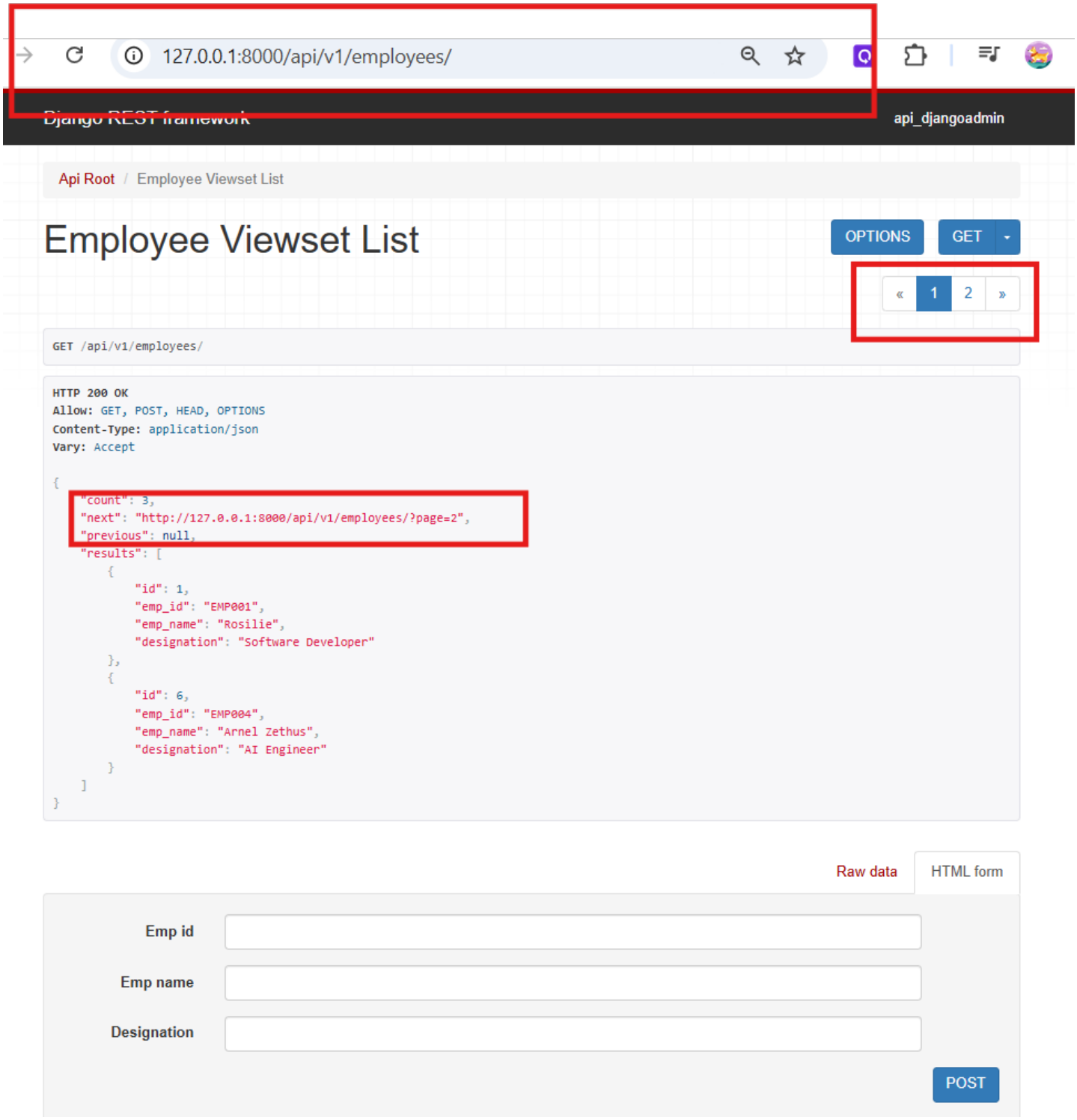
Page 1:


127.0.0.1:8000/api/v1/blogs/
Django REST framework
api_djangoadmin
Api Root / Blogs
Blogs
OPTIONS
GET
« 1 2 3 »
GET /api/v1/blogs/
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"count": 5,
"next": "http://127.0.0.1:8000/api/v1/blogs/?page=2",
"previous": null,
"results": [
{
"id": 2,
"comments": [
{
"id": 4,
"comment": "Interesting!",
"blog": 2
},
{
"id": 5,
"comment": "I wonder when this can be a law in other states too.",
"blog": 2
}
],
"blog_title": "Creating and Sharing Deceptive AI-Generated Media Is Now a Crime in New Jersey",
"blog_body": "Creating and sharing deceptive media made with artificial intelligence is now a crime in New Jersey and open to law
},
{
"id": 3,
"comments": [
{
"id": 6,
"comment": "Interesting top dog bidders here!",
"blog": 3
},
{
"id": 8,
"comment": "I want to see Microsoft there!",
"blog": 3
}
],
"blog_title": "Bidders of Tik Tok",
"blog_body": "Amazon, Blackstone, OnlyFans founder, Project Liberty, MrBeast , Perplexity AI are some of the bidders"
}
]
}

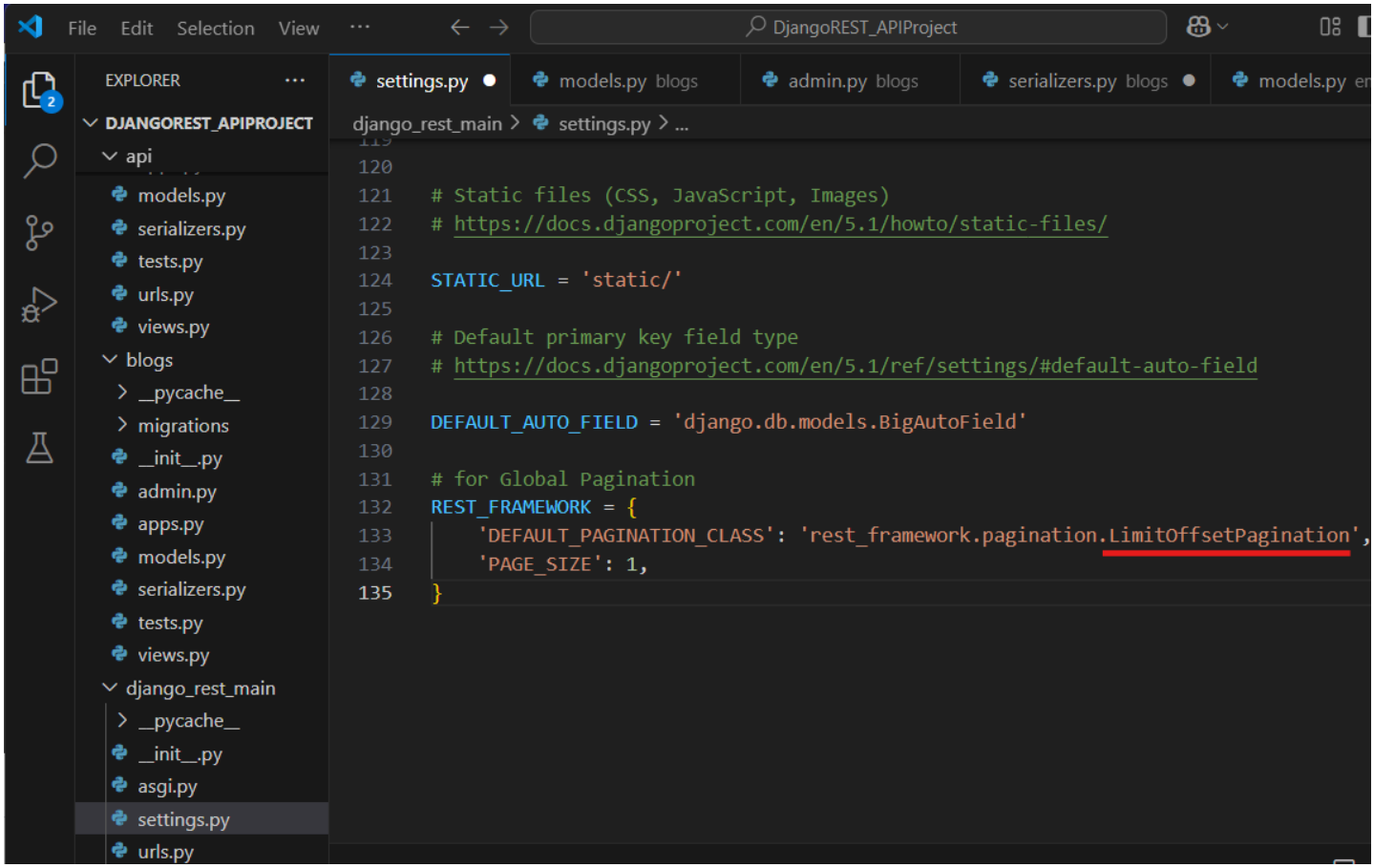
Page 2:



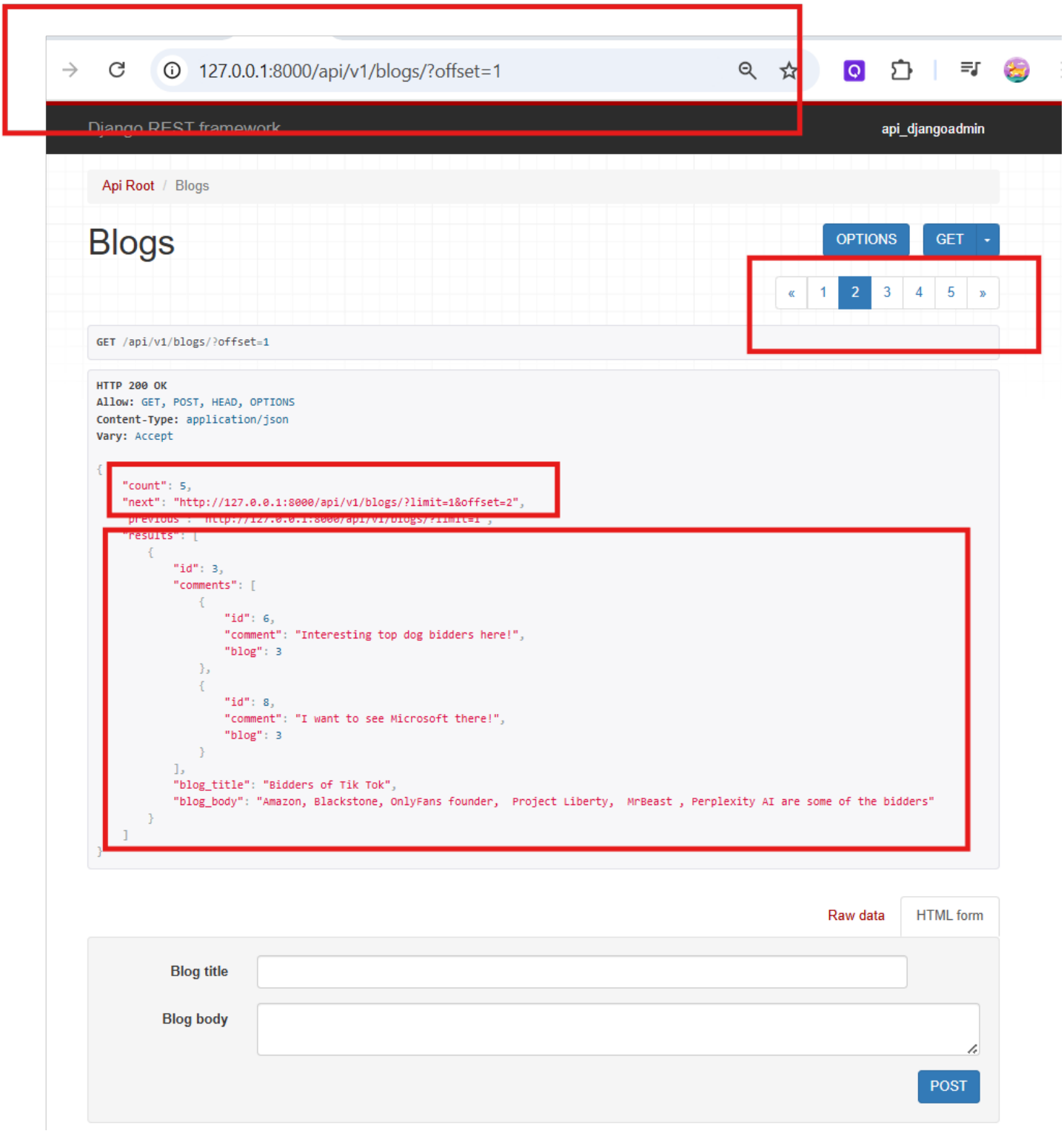
3. Also, when we run our path for EMPLOYEES model, we see this pagination as well.



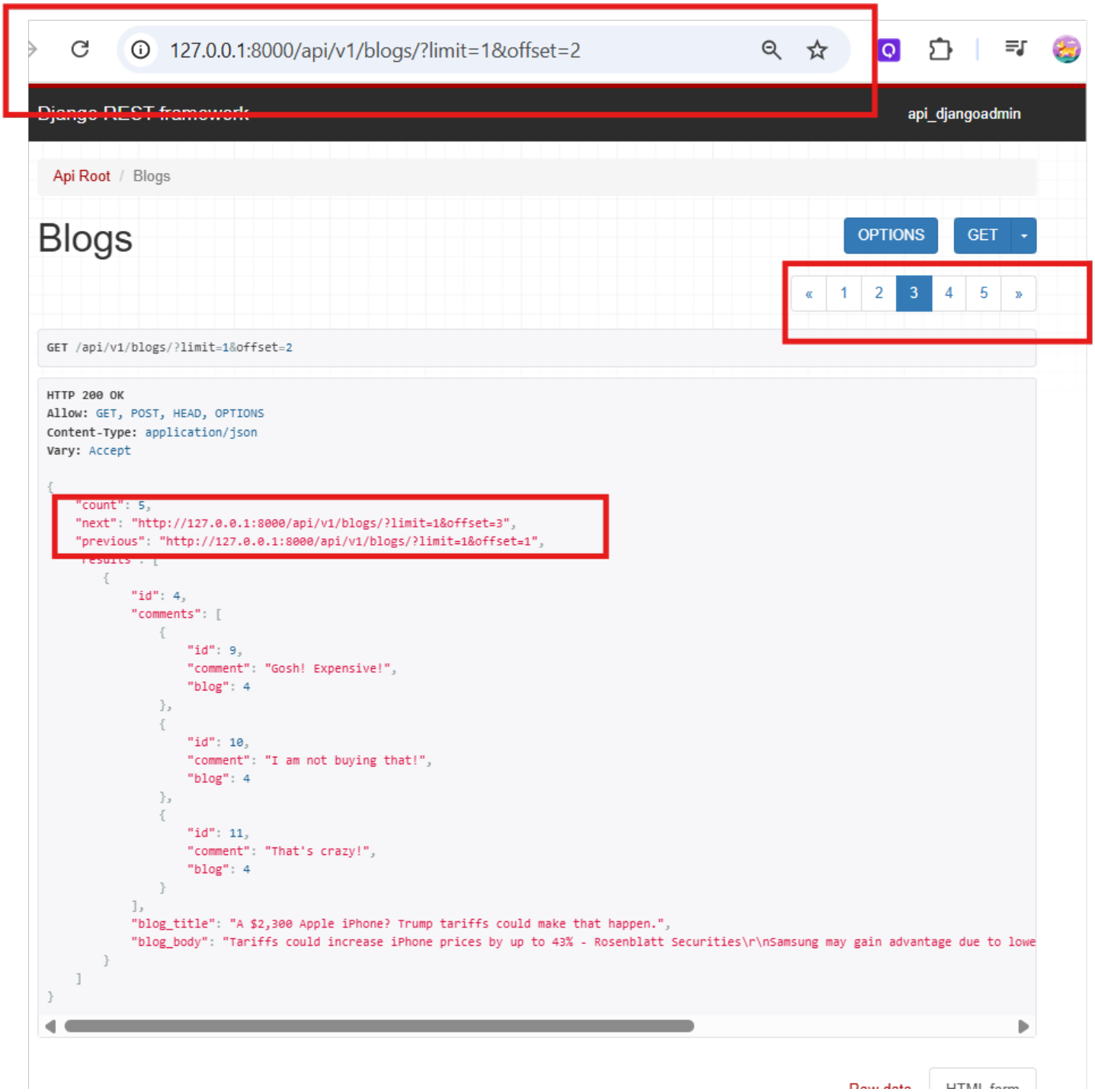
4. Now if we use LimitOffsetPagination, we update our SETTINGS.PY:



Our blog pages shall look like this where only one blog post per page and it sets the offset to 1. If you click on NEXT, it shall display OFFSET=2 on your URL path.



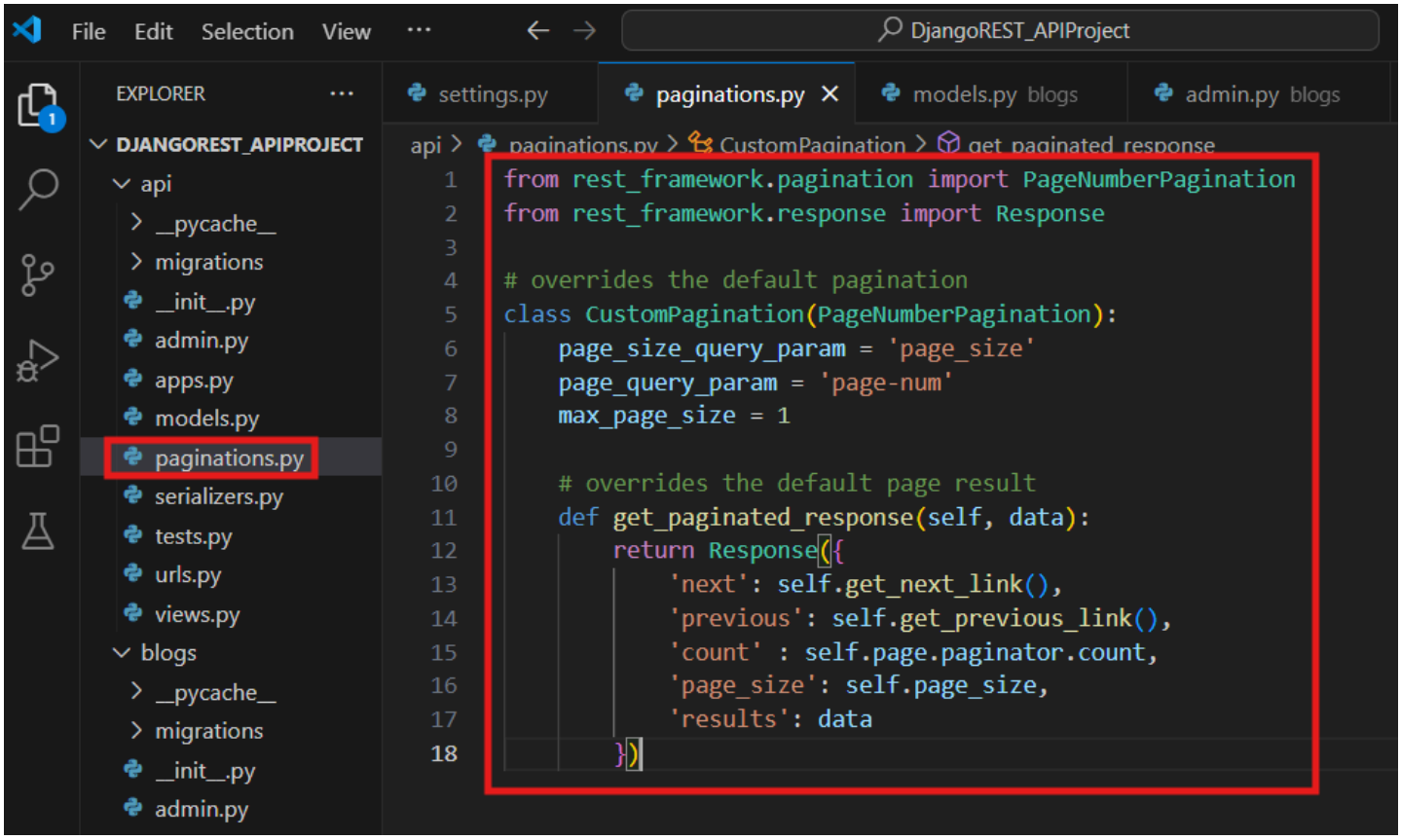
If the offset is 2:



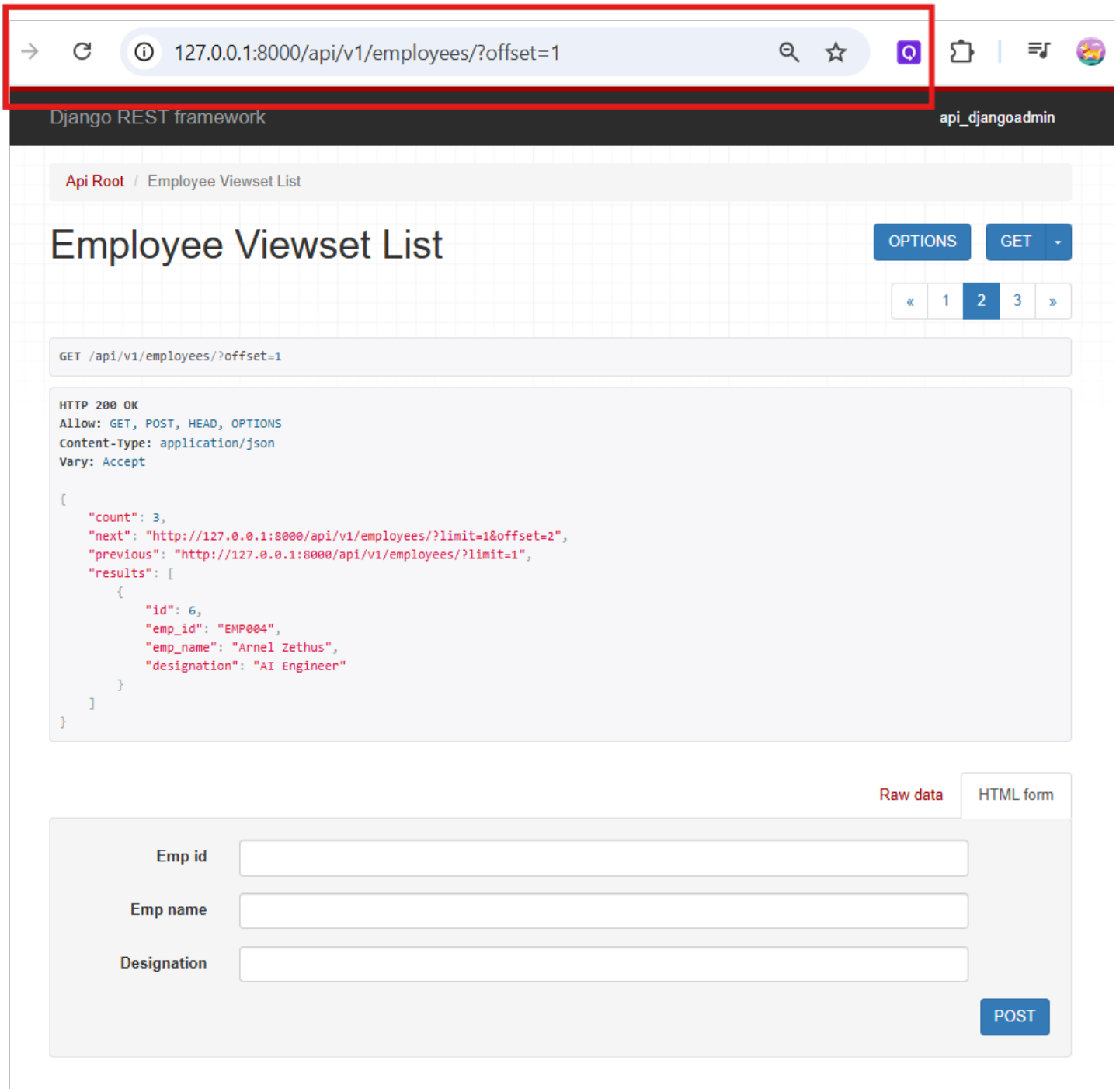
Customized Pagination:

5. If you want only a certain model to have a certain pagination, you can customize the pagination for this model alone. So if we want Employee model to have a customized pagination, create a new file under API or under root directory PAGINATIONS.PY.

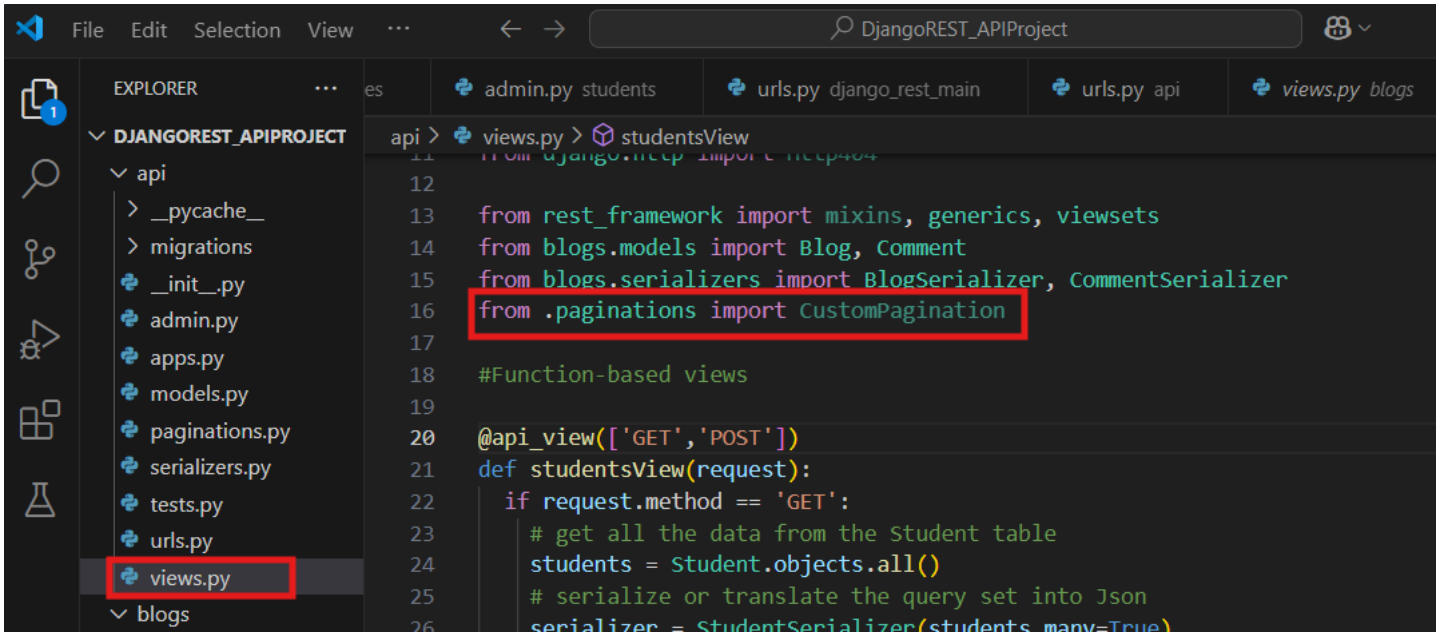
The class below will override the default class and method for pagination.



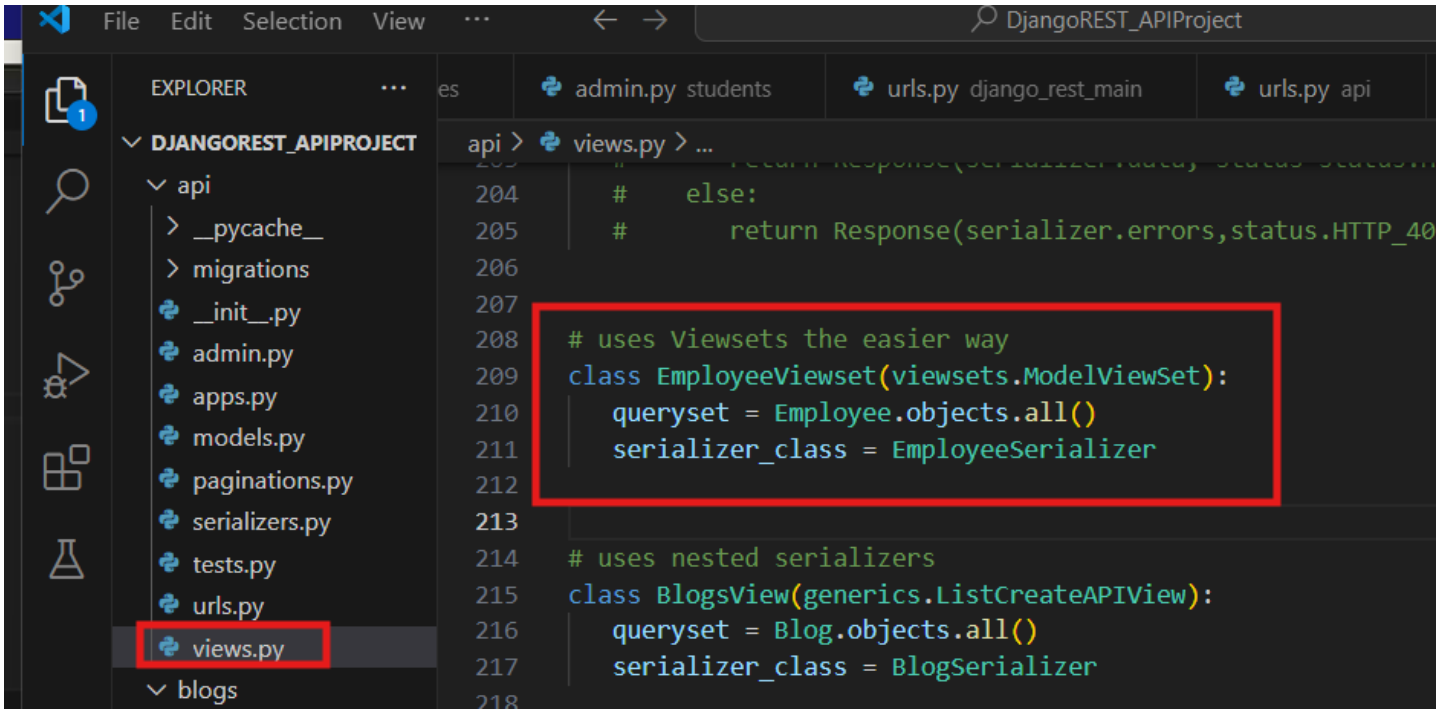
6. So currently, EMPLOYEES page looks like this because, in SETTINGS.PY, we have the default_pagination_class.



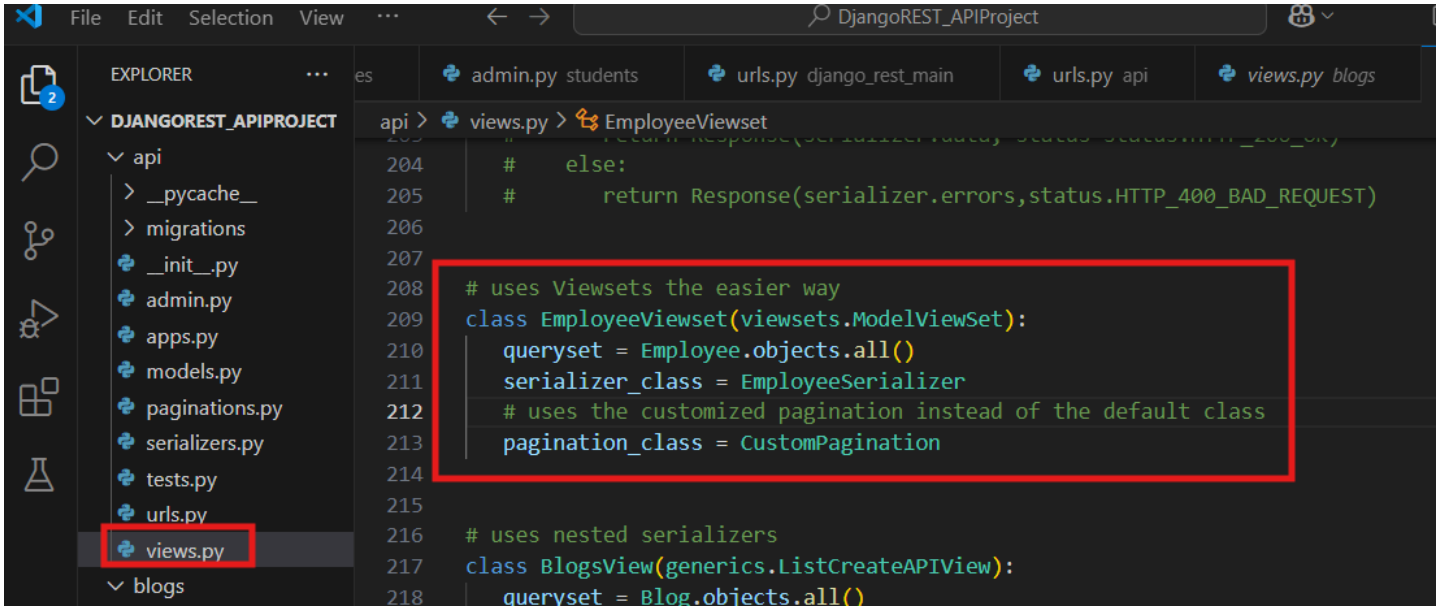
So to use the customized pagination, PAGINATIONS.PY, we import this in VIEWS.PY:



Update the EmployeeView as:



TO:



This will result to: