Topic: 15. DRF Filtering

Speaker: Personal | Notebook: API Development using Django Framework

---

Django Filtering:

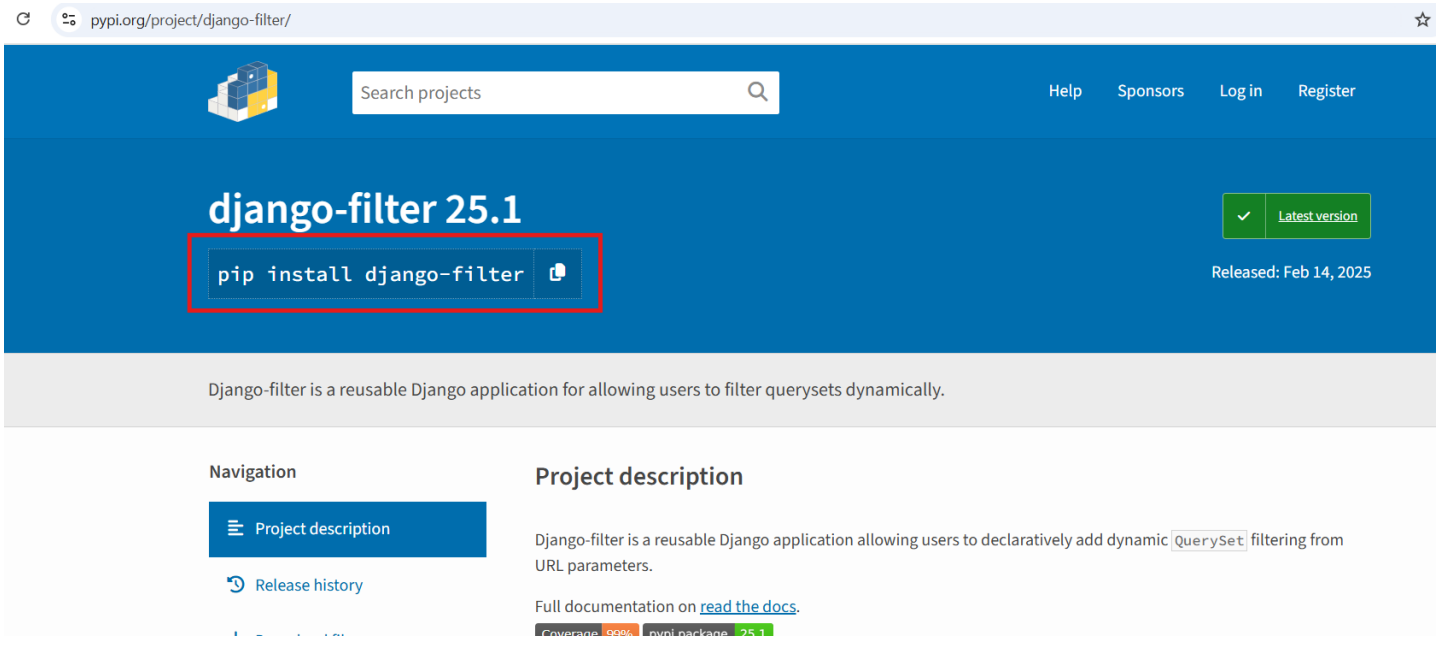
1. Filtering allows searching for certain record(s) based on given criteria. According to Django Rest API Framework documentation:

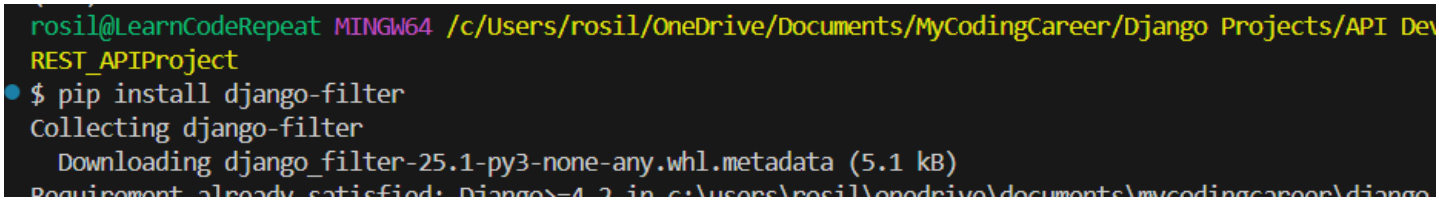
The default behavior of REST framework's generic list views is to return the entire queryset for a model manager. Often you will want your API to restrict the items that are returned by the queryset.

The simplest way to filter the queryset of any view that subclasses `GenericAPIView` is to override the `.get_queryset()` method.

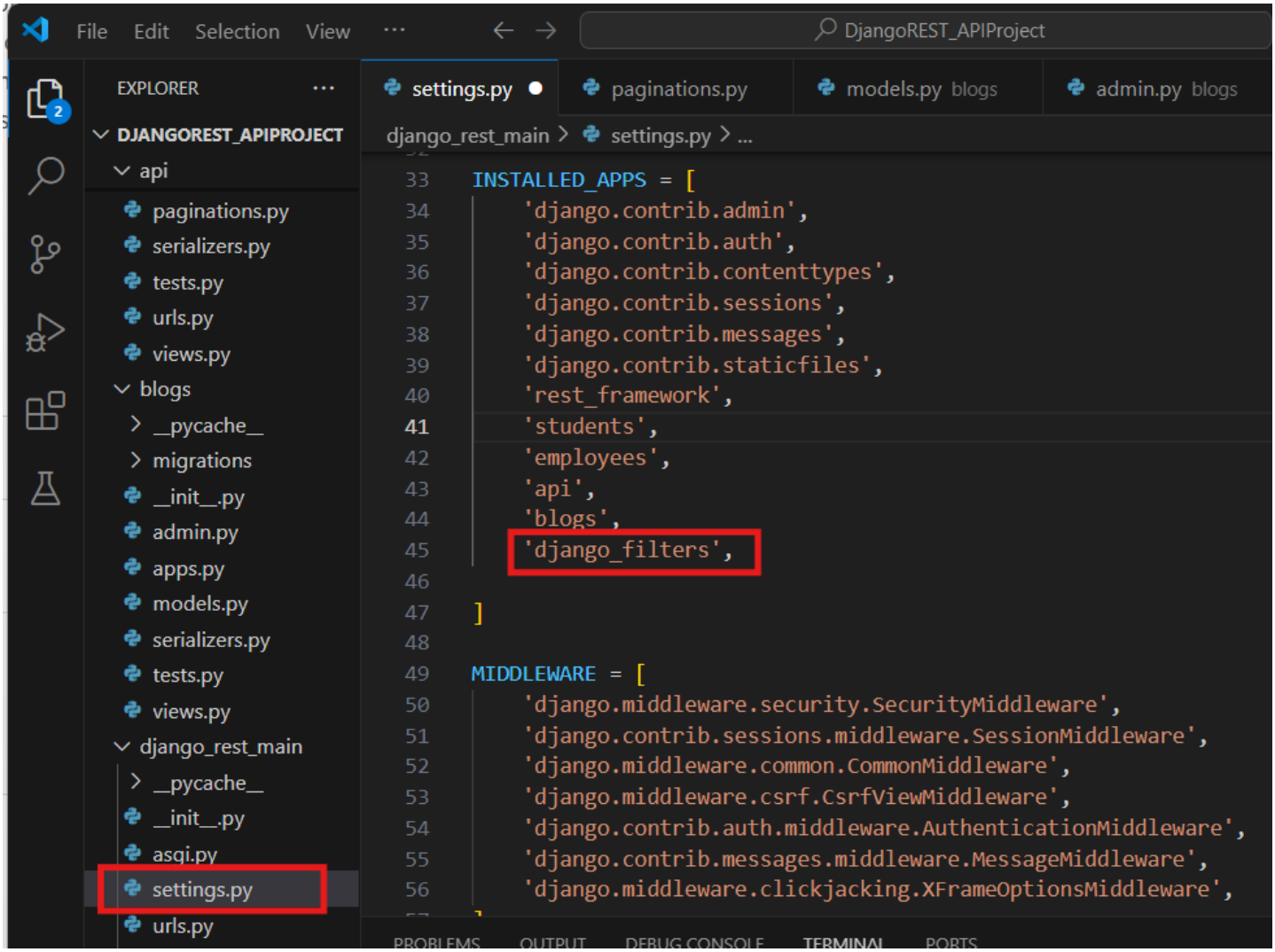
Overriding this method allows you to customize the queryset returned by the view in a number of different ways.

2. Install the library for Django Filter, go here. Go for the the latest version.

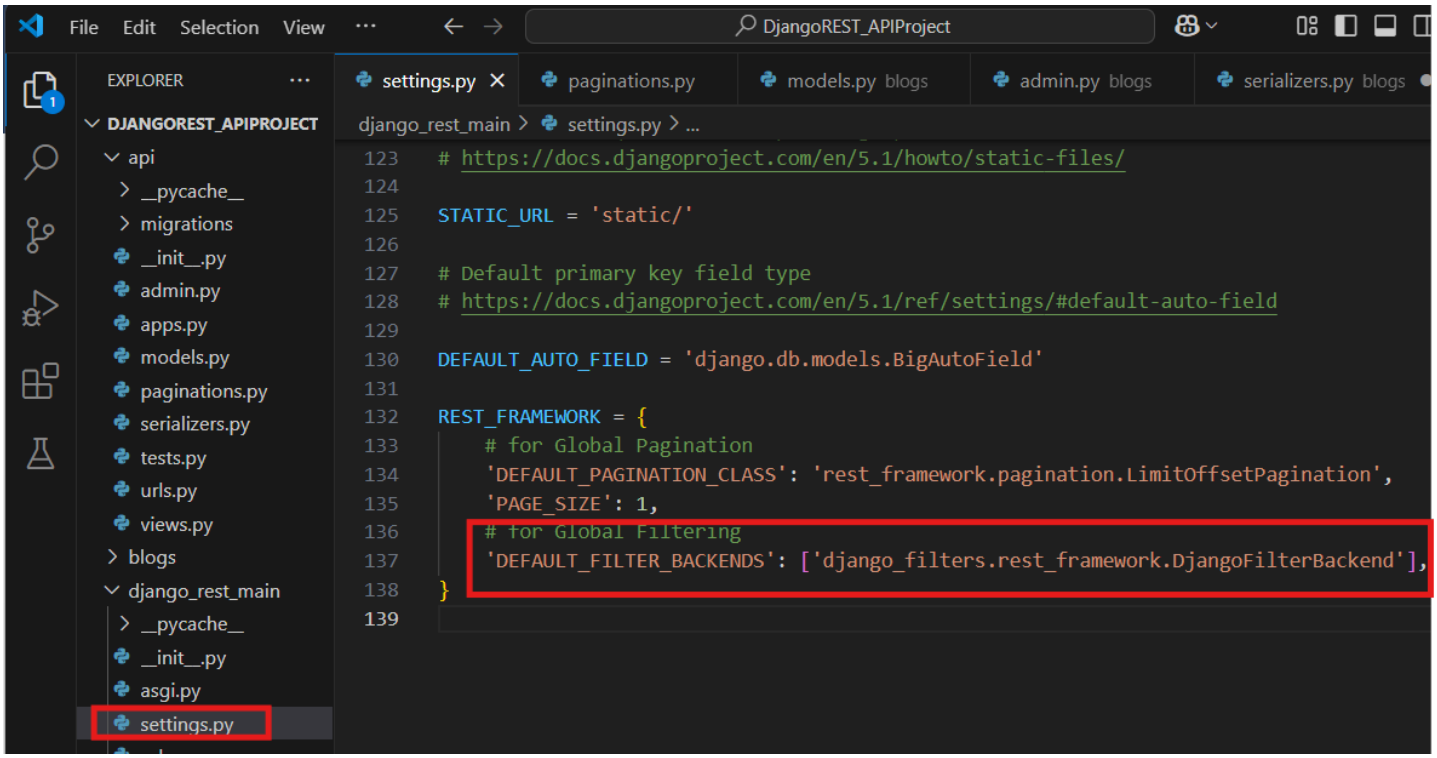




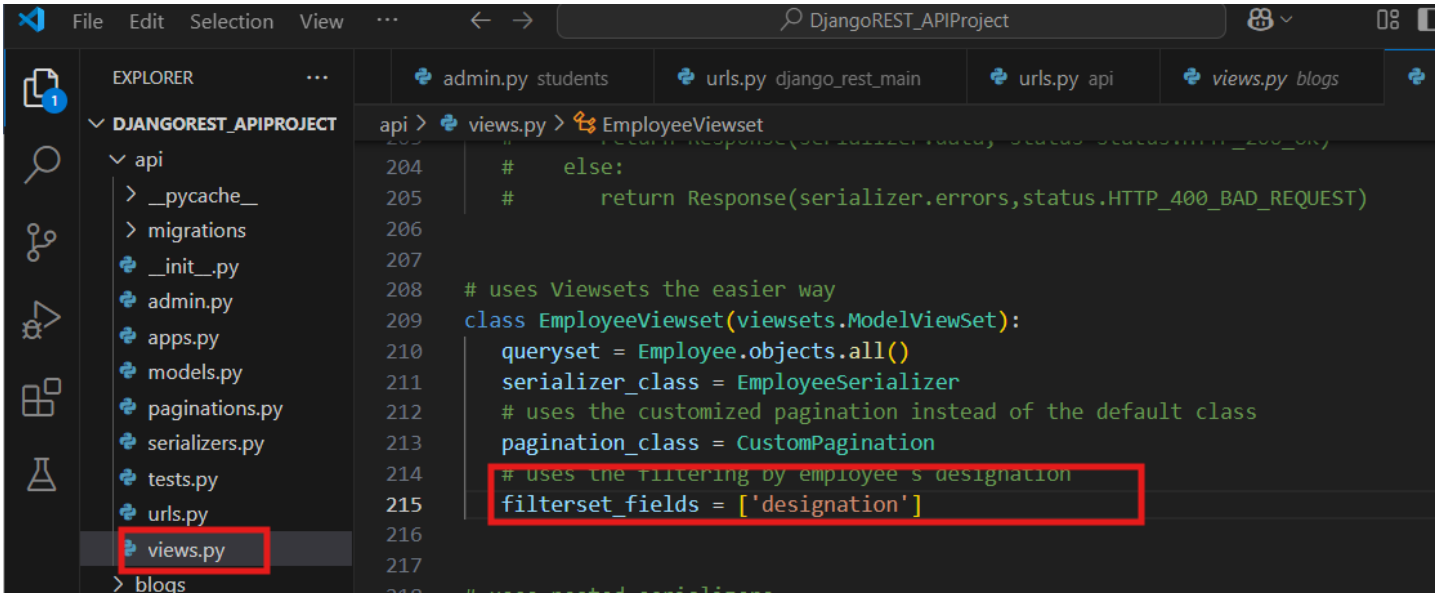
3. Register your Django package in SETTINGS.PY

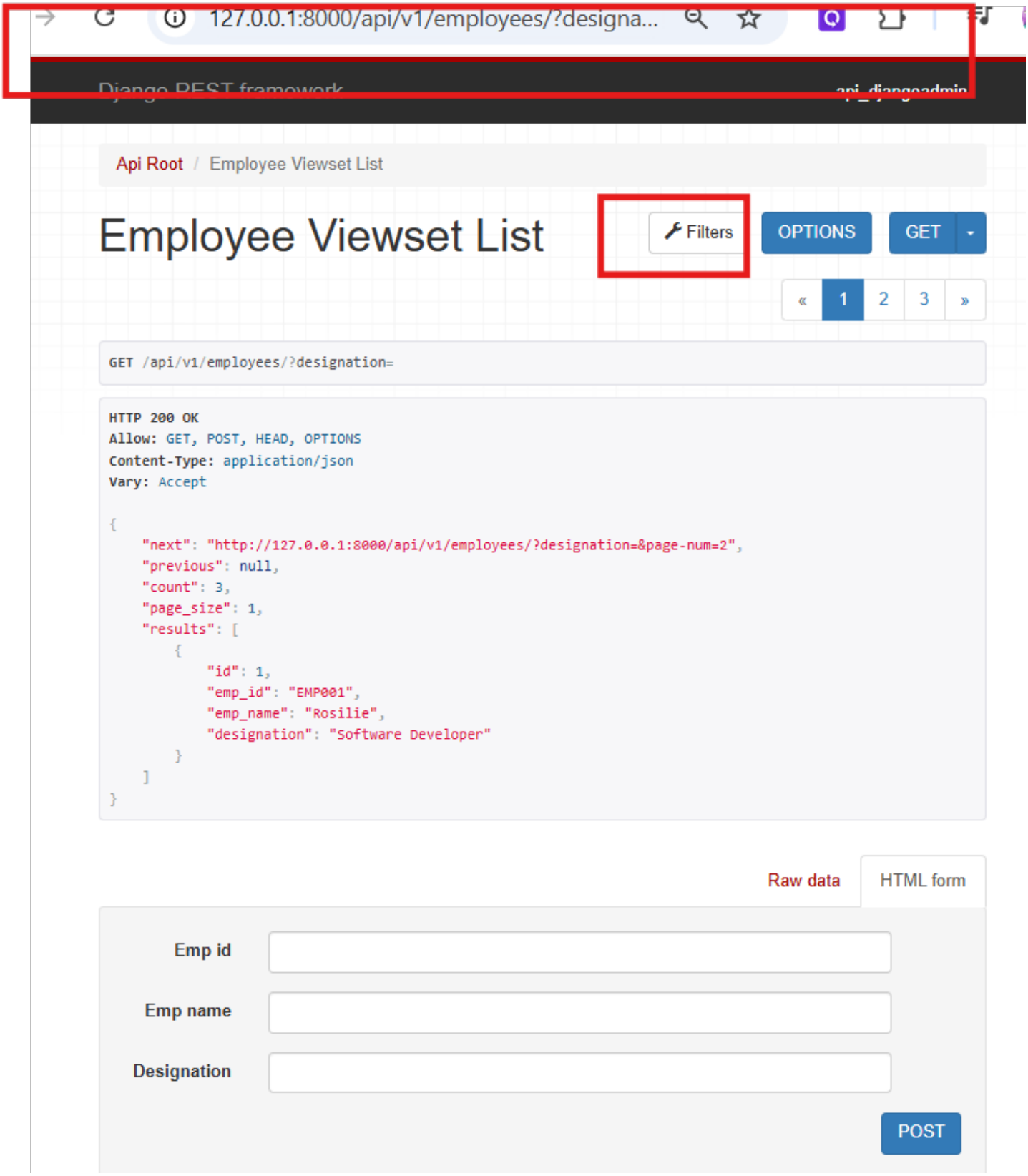


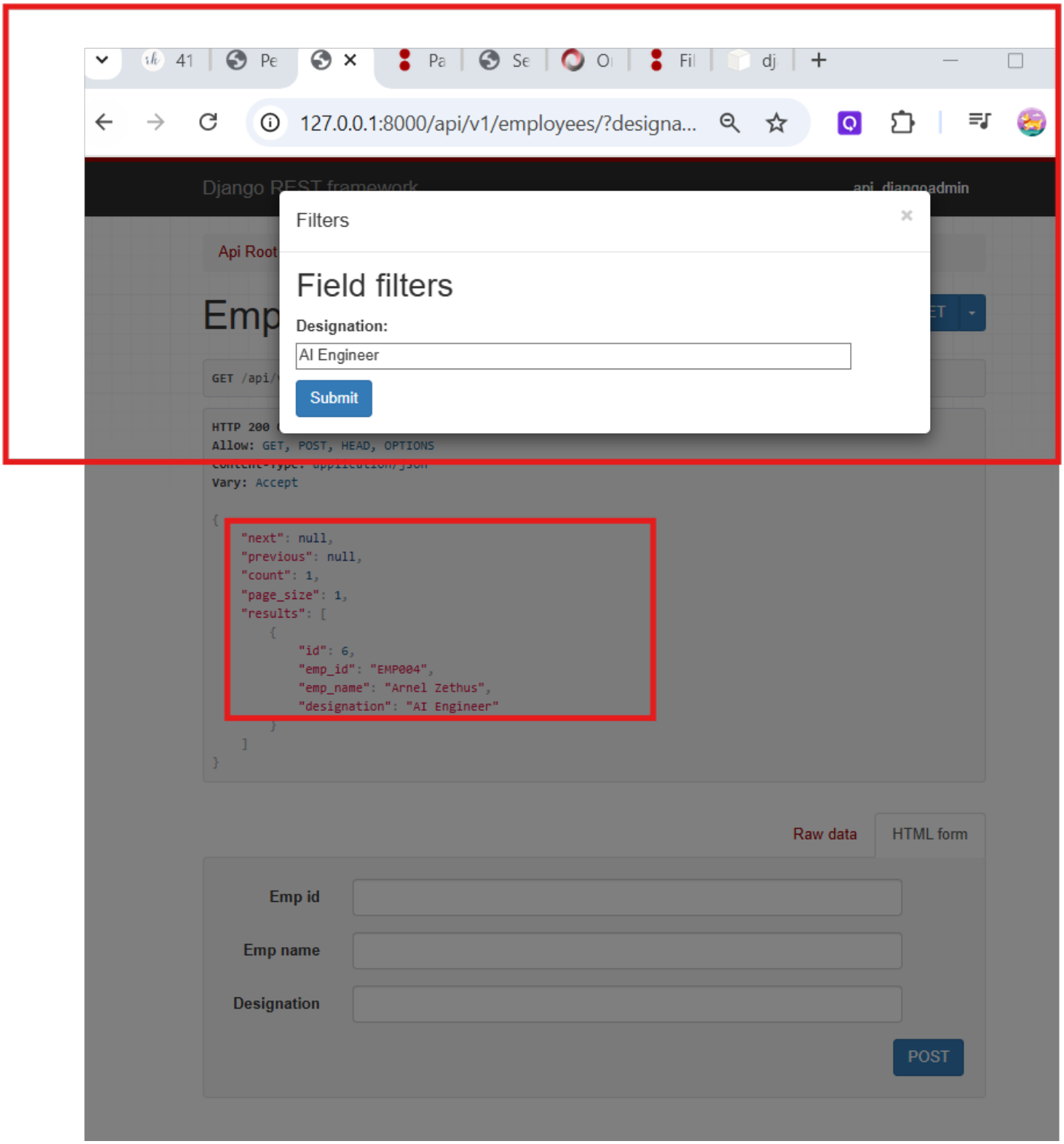
4. To set the GLOBAL FILTER, do this just like how you did the GLOBAL PAGINATION.



5. Update our API\VIEWS.PY:

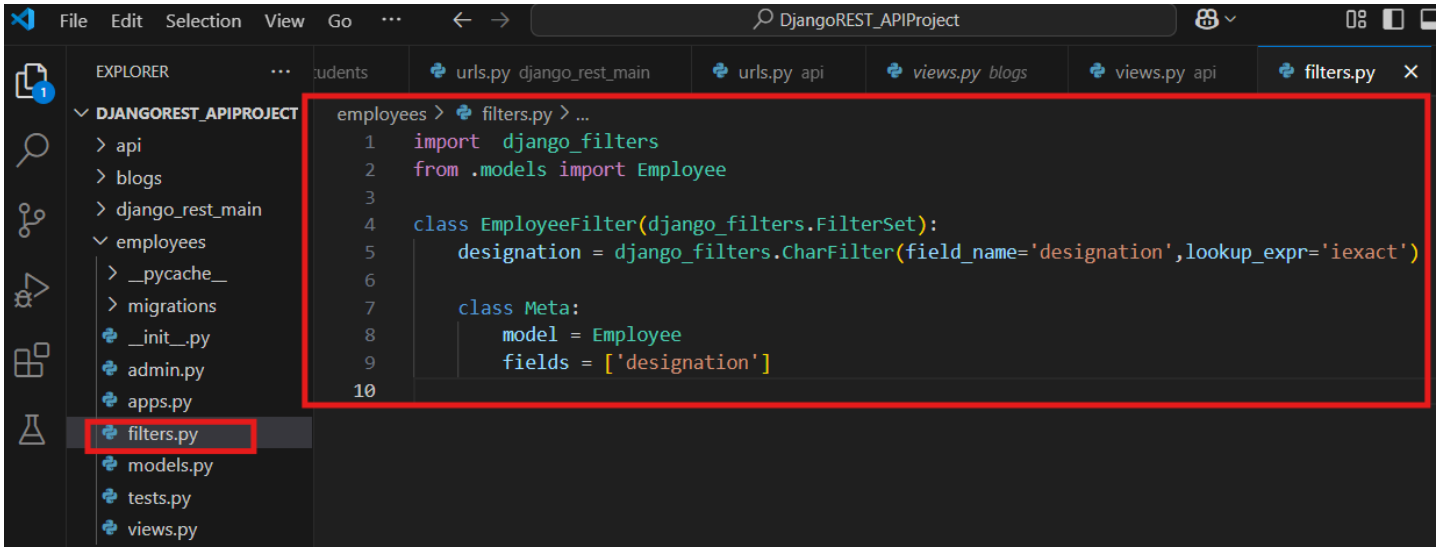

File Edit Selection View
DjangoREST_APIProject
EXPLORER
admin.py students
urls.py django_rest_main
urls.py api
views.py blogs
DJANGOREST_APIPROJECT
api
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
paginations.py
serializers.py
tests.py
urls.py
views.py
blogs
api > views.py > EmployeeViewset
204 # else:
205 # return Response(serializer.errors,status.HTTP_400_BAD_REQUEST)
206
207
208 # uses Viewsets the easier way
209 class EmployeeViewset(viewsets.ModelViewSet):
210 queryset = Employee.objects.all()
211 serializer_class = EmployeeSerializer
212 # uses the customized pagination instead of the default class
213 pagination_class = CustomPagination
214 # uses the filtering by employee's designation
215 filterset_fields = ['designation']
216
217


127.0.0.1:8000/api/v1/employees/?designa...
Api Root / Employee Viewset List
Employee Viewset List
Filters
OPTIONS
GET
« 1 2 3 »
GET /api/v1/employees/?designation=
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"next": "http://127.0.0.1:8000/api/v1/employees/?designation=&page-num=2",
"previous": null,
"count": 3,
"page_size": 1,
"results": [
{
"id": 1,
"emp_id": "EMP001",
"emp_name": "Rosilie",
"designation": "Software Developer"
}
]
}
Raw data
HTML form
Emp id
Emp name
Designation
POST

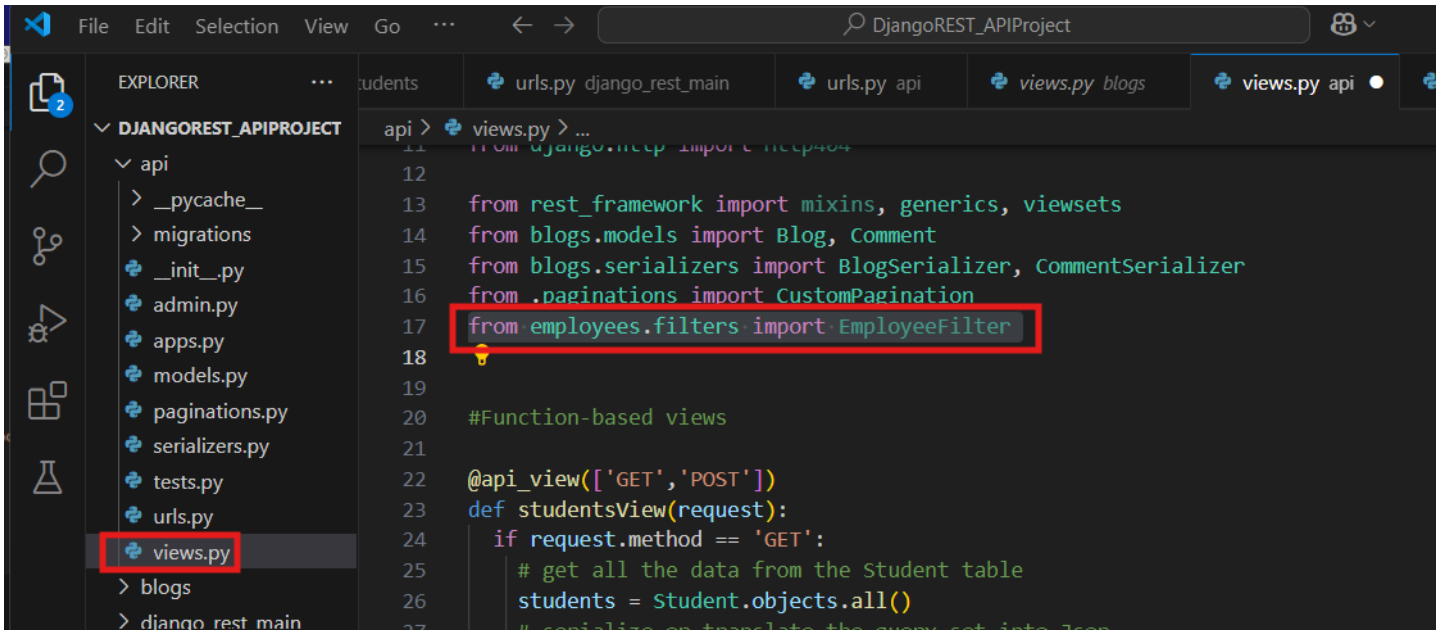


* You need to type the exact keyword to find an exact match, so this is a case-sensitive filter. To solve this, we use custom filters instead.

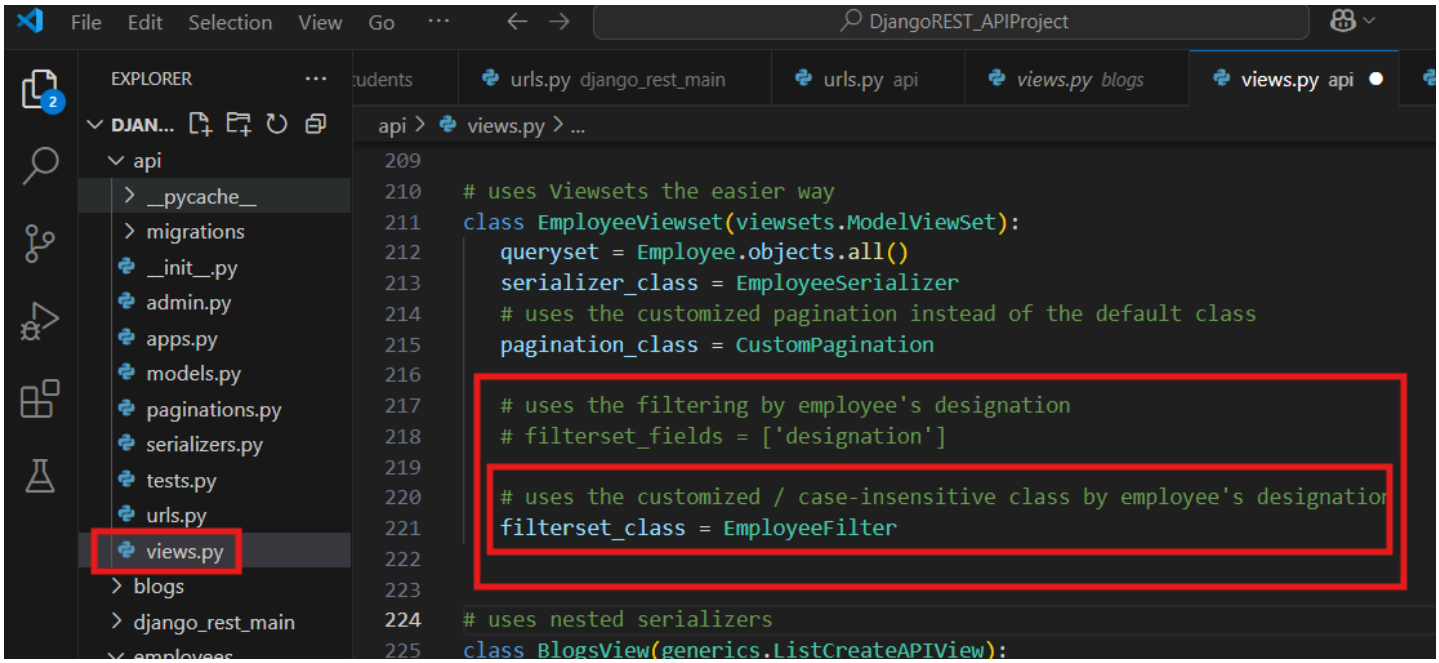
6. To allow for case-insensitive filter, create a new file FILTERS.PY in the EMPLOYEES app or you can have this in API folder.



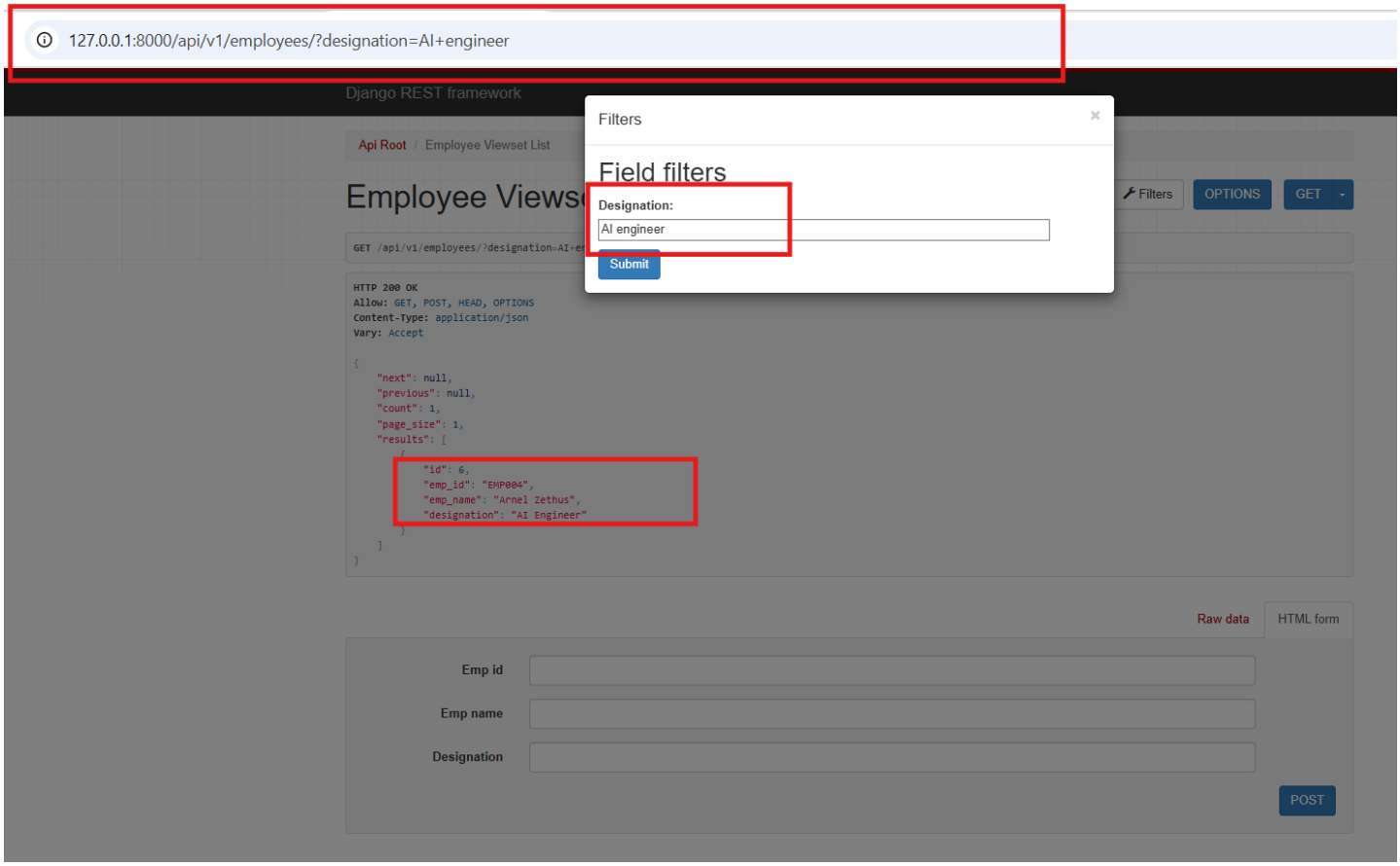
7. Then update our VIEWS.PY. Import the class EmployeeFilter from the Employees\Filters.py



To use the new filter class:

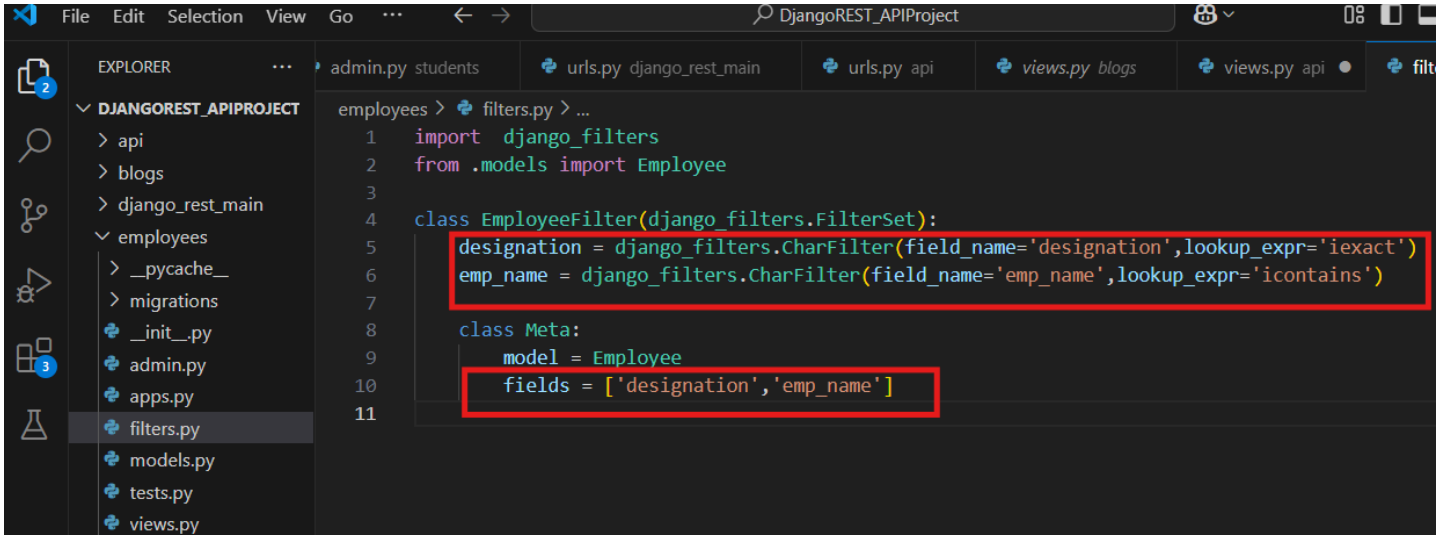


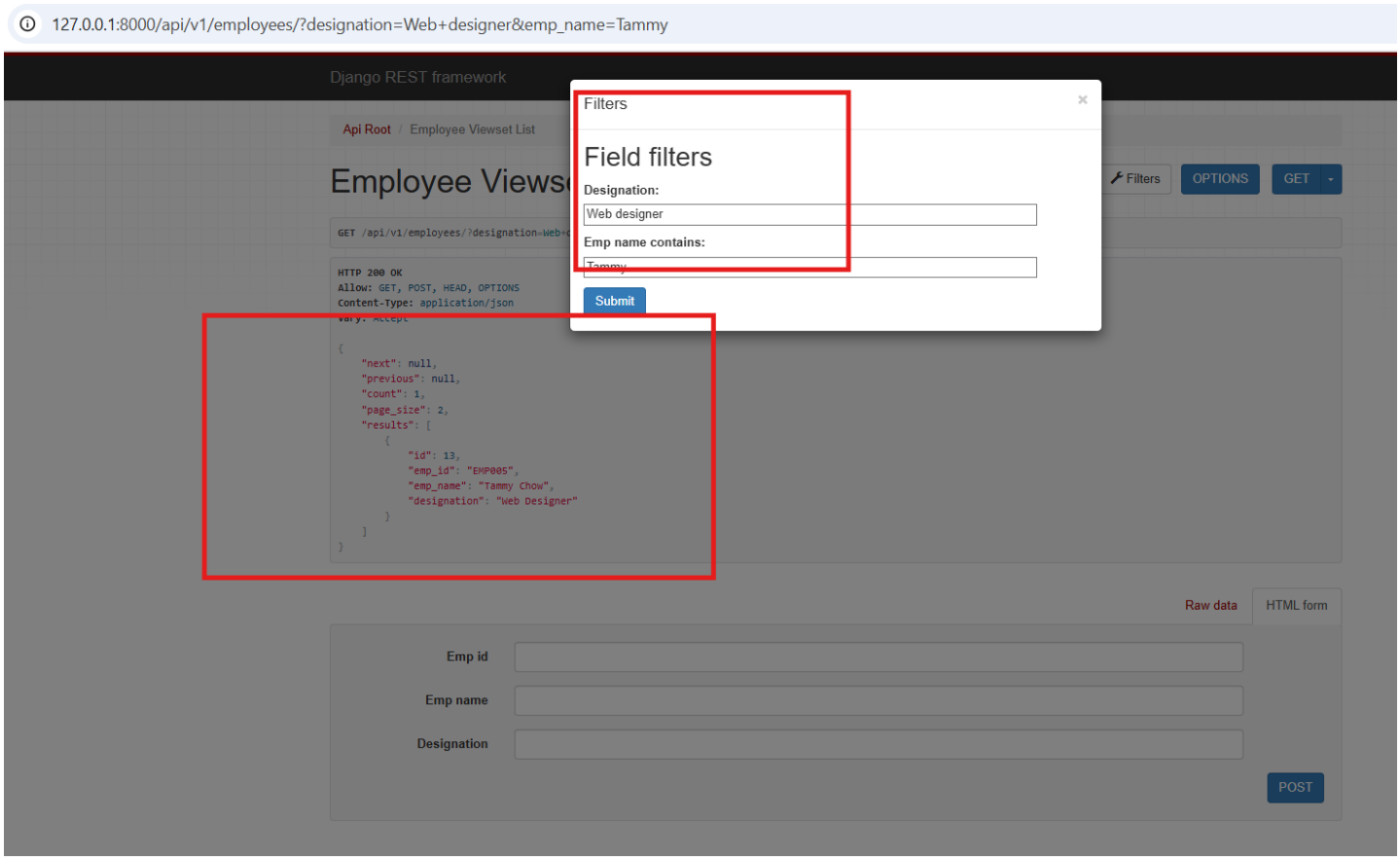
8. Now even if we search without minding the lowercase or uppercase, we can still find the record:

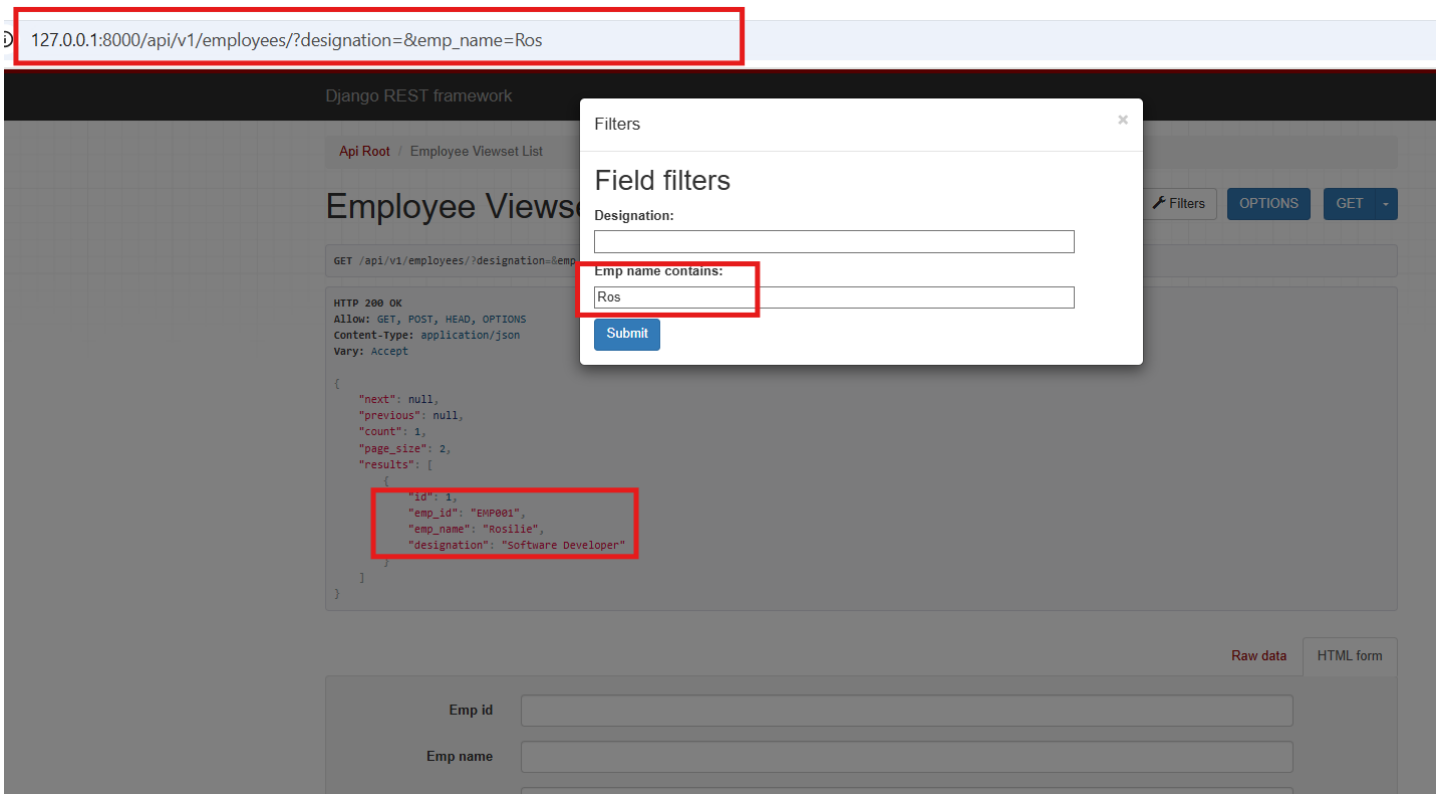


9. To add filters like by name and ID, we update our FILTERS.PY :

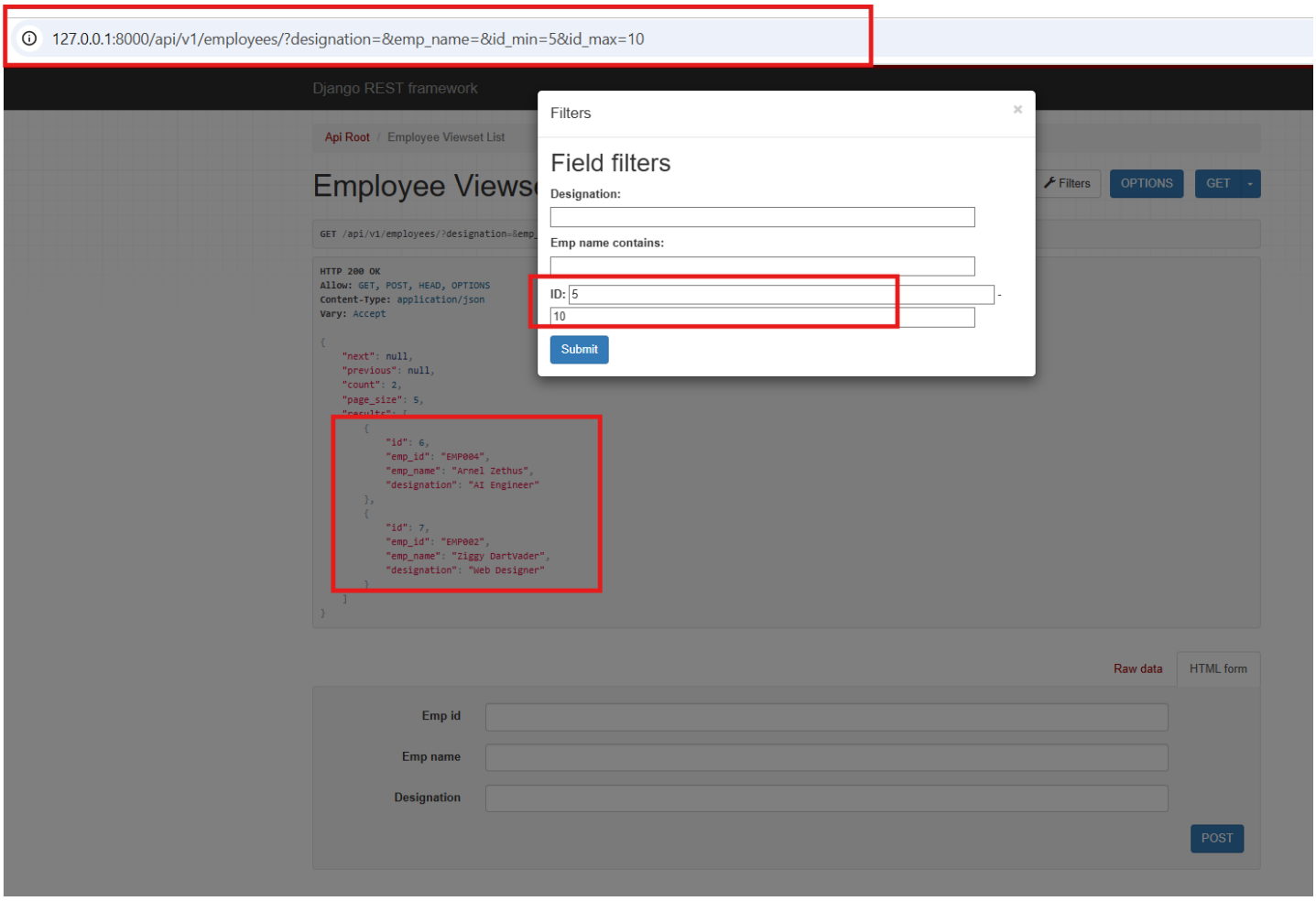
If we add the 2 filter keywords, we search the record for these 2 criteria.





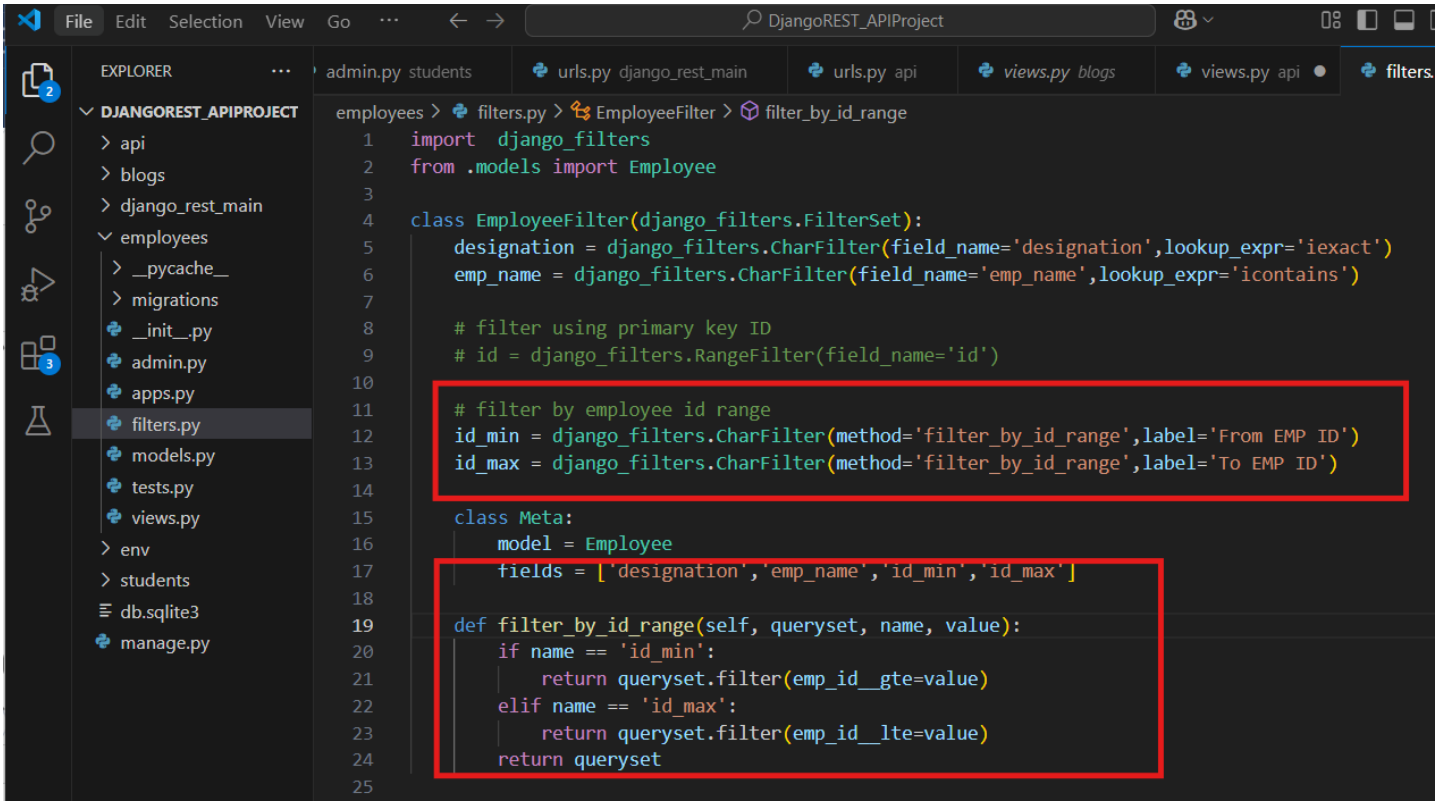


10. To retrieve the employee records within the given ID range for example: PK or ID 5 - 10:

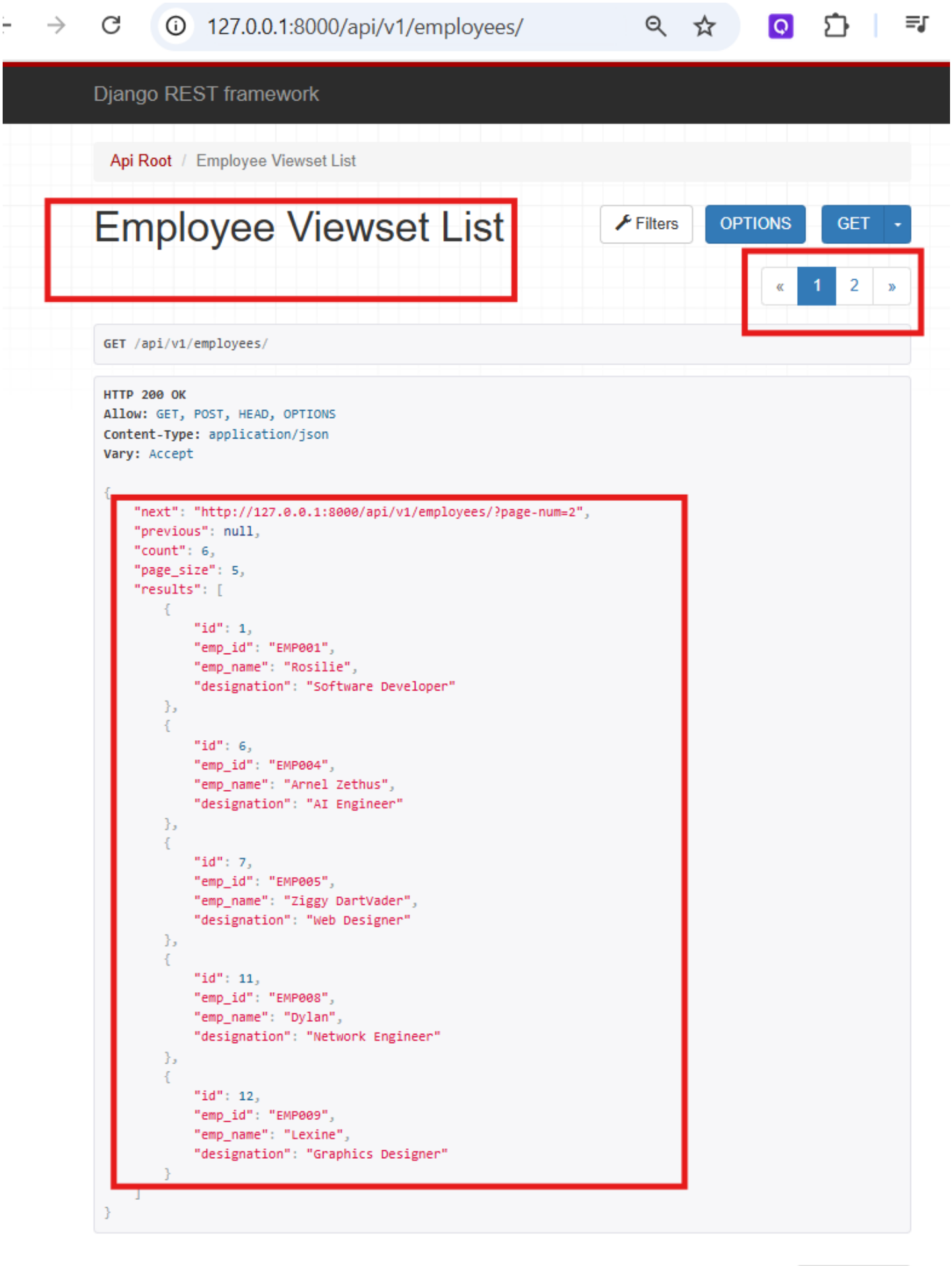


We update our FILTERS.PY as:

11. But to use the EMP_ID (ex. EMP005-EMP008) with CharField as our range filter, we update the FILTERS.PY as:



Before the Filter by Emp_ID:


127.0.0.1:8000/api/v1/employees/
Django REST framework
Api Root / Employee Viewset List
Employee Viewset List
Filters
OPTIONS
GET
« 1 2 »
GET /api/v1/employees/
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"next": "http://127.0.0.1:8000/api/v1/employees/?page-num=2",
"previous": null,
"count": 6,
"page_size": 5,
"results": [
{
"id": 1,
"emp_id": "EMP001",
"emp_name": "Rosilie",
"designation": "Software Developer"
},
{
"id": 6,
"emp_id": "EMP004",
"emp_name": "Arnel Zethus",
"designation": "AI Engineer"
},
{
"id": 7,
"emp_id": "EMP005",
"emp_name": "Ziggy DartVader",
"designation": "Web Designer"
},
{
"id": 11,
"emp_id": "EMP008",
"emp_name": "Dylan",
"designation": "Network Engineer"
},
{
"id": 12,
"emp_id": "EMP009",
"emp_name": "Lexine",
"designation": "Graphics Designer"
}
]
}

After: