Django

TomerBu

נושאים להיום:

utils לפרוייקטים
העלאת תמונת פרופיל
Users Permissions Groups
Authorization
Authentication
Token Authentication

שימוש במודול לפעולות עזר:

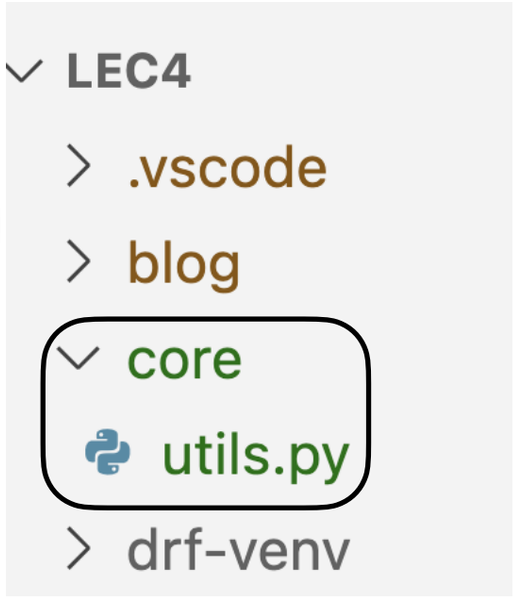
views.py

```python
from core.utils import try_parse_int
from .serializers import *
from .models import Comment, Post, UserProfile, PostUserLikes
from rest_framework.viewsets import ModelViewSet
from rest_framework.response import Response


class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer

    def create(self, request, *args, **kwargs):
        data = request.data
        reply_to = data.get('reply_to')
        post_id = try_parse_int(data.get('post'))

        if reply_to:
            replied = Comment.objects.get(id=reply_to)
            print(replied)
            if (
                replied and replied.post.id != post_id
            ):
                return Response(
                    {"error": "Reply must be on the same post"},
                    status=400
                )

        return super().create(request, *args, **kwargs)
```

בשיעור הקודם - בדקנו בדיקה שלילית - ולא עלינו על הצורך בparse

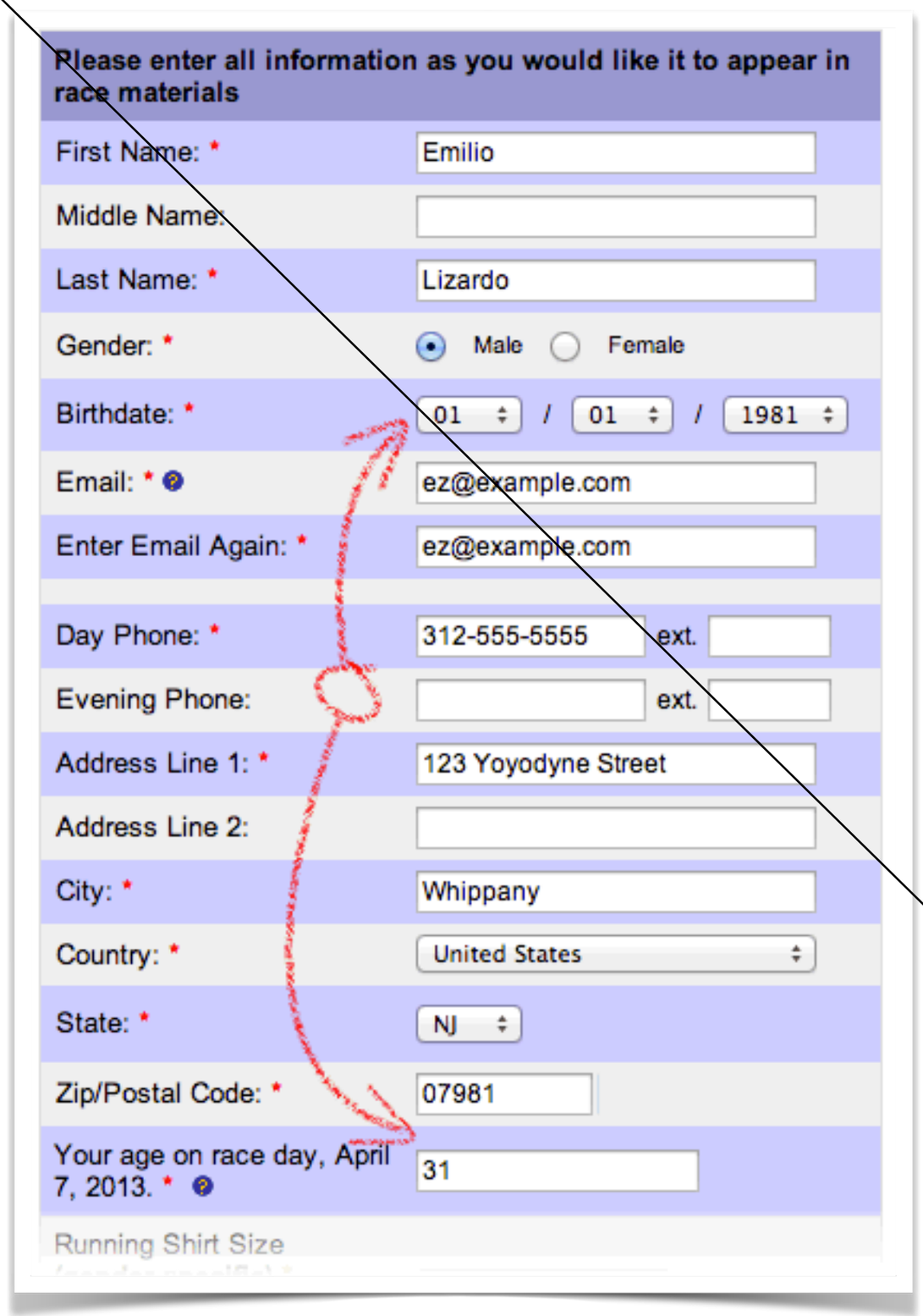
זה ממחיש את הצורך לבצע בדיקה שלילית ובדיקה חיובית

מודול לפעולות עזר:




```python
def try_parse_int(value: str, default=None):
    try:
        return int(value)
    except (ValueError, TypeError):
        return default
```

UX Anti Pattern



user

userprofile

תמונת פרופיל:

Django REST framework!!!

User Profile List

GET /api/v1/user-profile/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept


```
[
    {
        "id": 1,
        "bio": "blogger blogs",
        "profile_pic": null,
        "birth_date": "1990-10-10",
        "created_at": "2025-02-23T08:26:02.924244Z",
        "updated_at": "2025-02-23T08:26:02.924263Z",
        "user": 1
    },
    {
        "id": 2,
        "bio": "site mod",
        "profile_pic": null,
        "birth_date": "1990-10-10",
        "created_at": "2025-02-23T08:28:40.183062Z",
        "updated_at": "2025-02-23T08:28:40.183073Z",
        "user": 2
    },
    {
        "id": 3,
        "bio": "likes to read",
        "profile_pic": null,
        "birth_date": "1989-02-02",
        "created_at": "2025-02-23T08:28:59.003396Z",
        "updated_at": "2025-02-23T08:28:59.003407Z",
        "user": 3
    }
]
```


Bio

Profile pic Choose File No file chosen

Birth date dd/mm/yyyy

User moderator

תמונת פרופיל:

settings.py

```
import os

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

lec4/urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.conf.urls.static import static
from django.conf import settings
from lec4.settings import MEDIA_ROOT, MEDIA_URL

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/v1/', include('blog.urls')),
]

# https://docs.djangoproject.com/en/5.1/howto/static-files/#serving-static-files-during-development
urlpatterns += static(
        settings.MEDIA_URL,
        document_root=settings.MEDIA_ROOT
)
```

תמונת פרופיל:

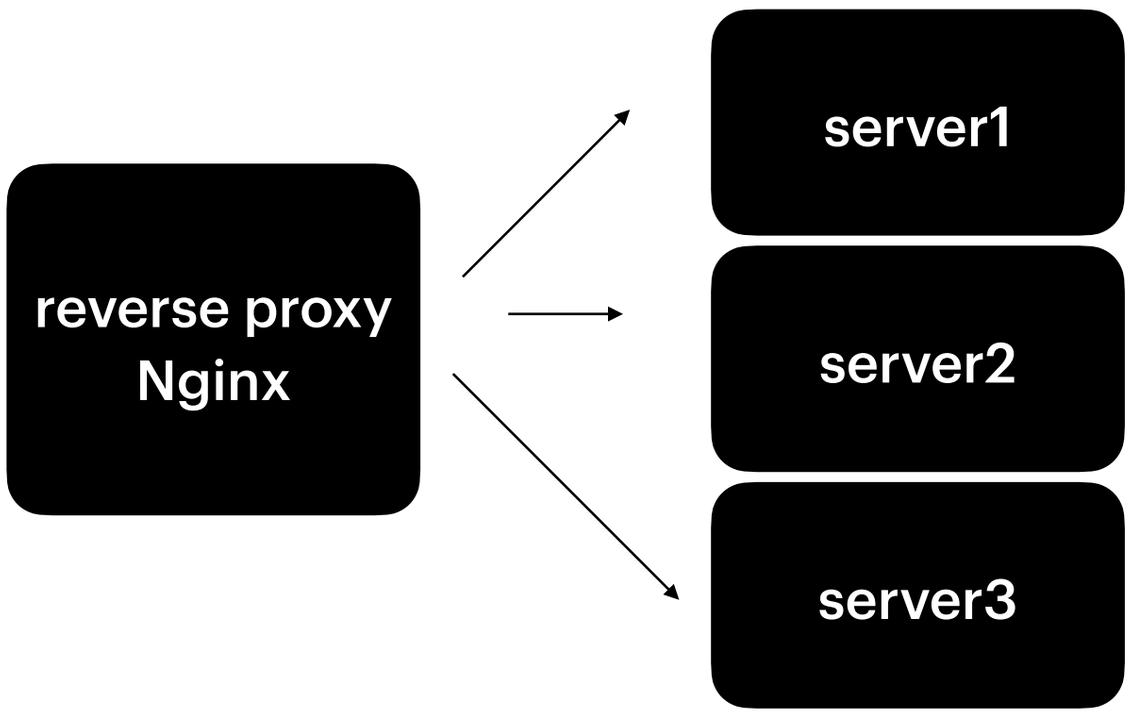
בפרויקטים גדולים:

משתמשים בשרת קבצים ייעודי
משתמשים בCDN

Content Delivery Network

s3 buckets:

שירות שמציעות חברות ענן - לדוגמא אמזון
שמאחסנים קבצים עבור הפרוייקט ברשת CDN



שליחת קובץ בJavascript

https://developer.mozilla.org/en-US/docs/Web/API/FormData/FormData

שולח בקשת Multipart שולח את הקובץ בחלקים
שהשרת של Django יודע לפענח

תצטרכו לבנות טופס רגיל

מהטופס לבנות אובייקט של FormData(form)

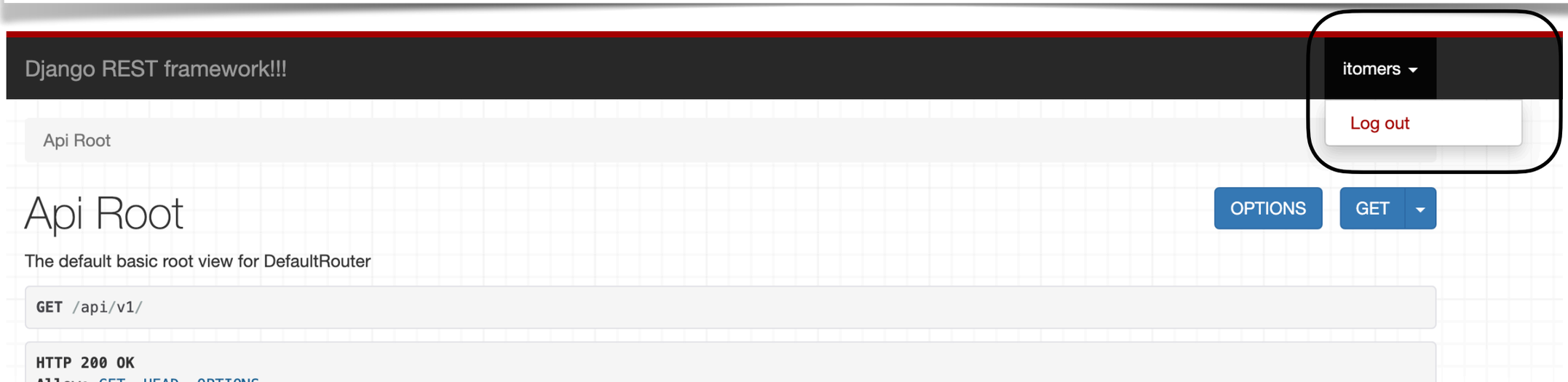
את זה תצרפו בfetch או בaxios

Django Permissions

הוספת כפתורי Login/Logout

blog/urls.py

```python
from django.urls import path, include
from .views import CommentsViewSet, PostsViewSet, UserProfileViewSet, LikesViewSet

from rest_framework.routers import DefaultRouter

router = DefaultRouter()

router.register('comments', CommentsViewSet, basename='comment')
router.register('posts', PostsViewSet, basename='posts')
router.register('user-profile', UserProfileViewSet, basename='user-profile')
router.register('likes', LikesViewSet, basename='likes')

urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls'))
]
```



Django Permissions

המודלים החשובים ביותר הם:

User

User: יישות שמתארת את המשתמש: שם משתמש, סיסמא אימייל ועוד.

Permission

Permission: יישות שמתארת יכולת ספציפית:
add_post, change_post, delete_post,
add_comment, change_comment, delete_comment
(ההרשאות הללו נוצרות אוטומטית ע"י Django כשאנו מוסיפים למשק הניהול).

Group

Group:

קבוצה של אנשים
עם הרשאות מוגדרות

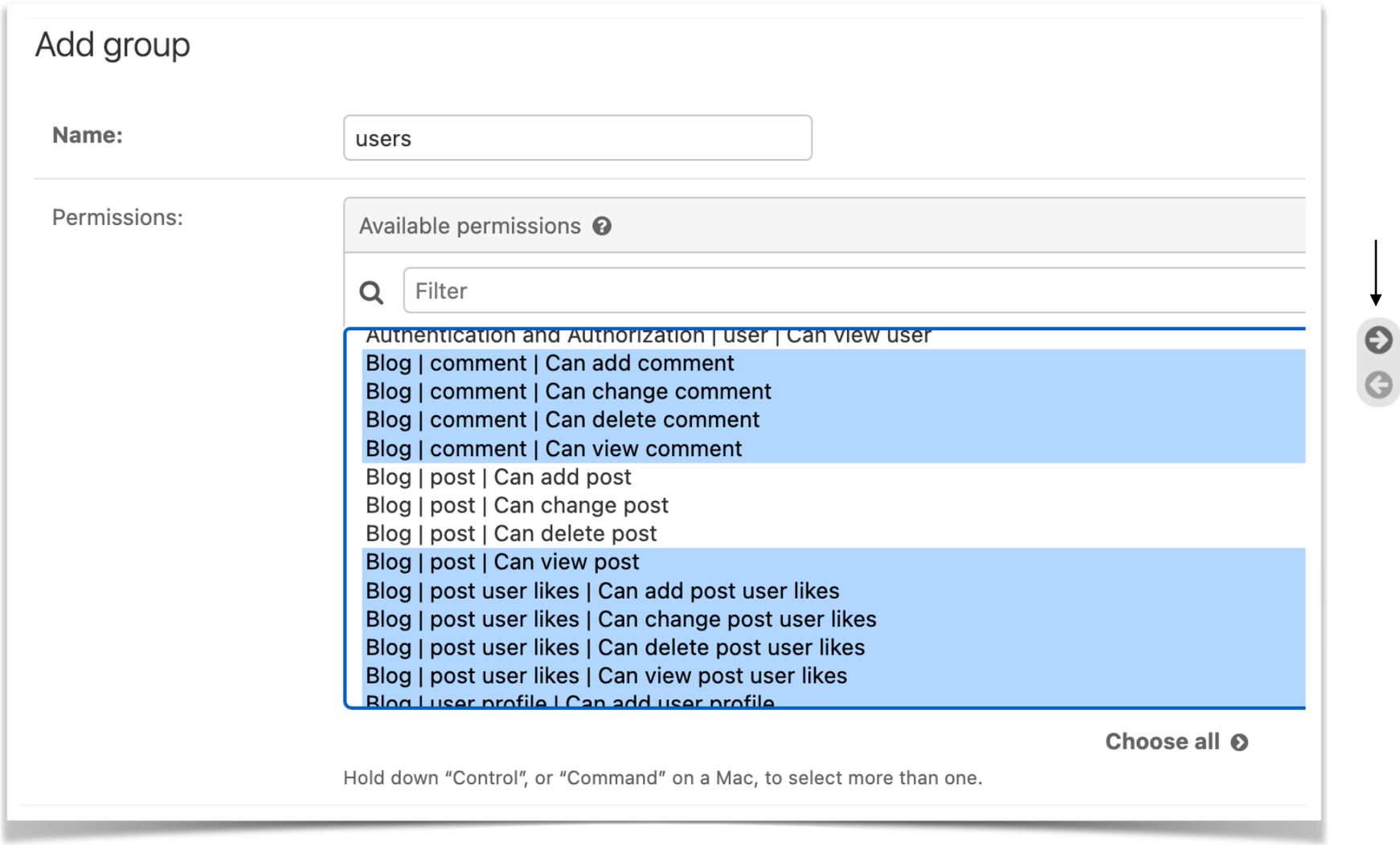
לדוגמא: ניצור קבוצת users
ונעניק לקבוצה:

view_post
view_comment
change_comment
delete_comment
add_comment

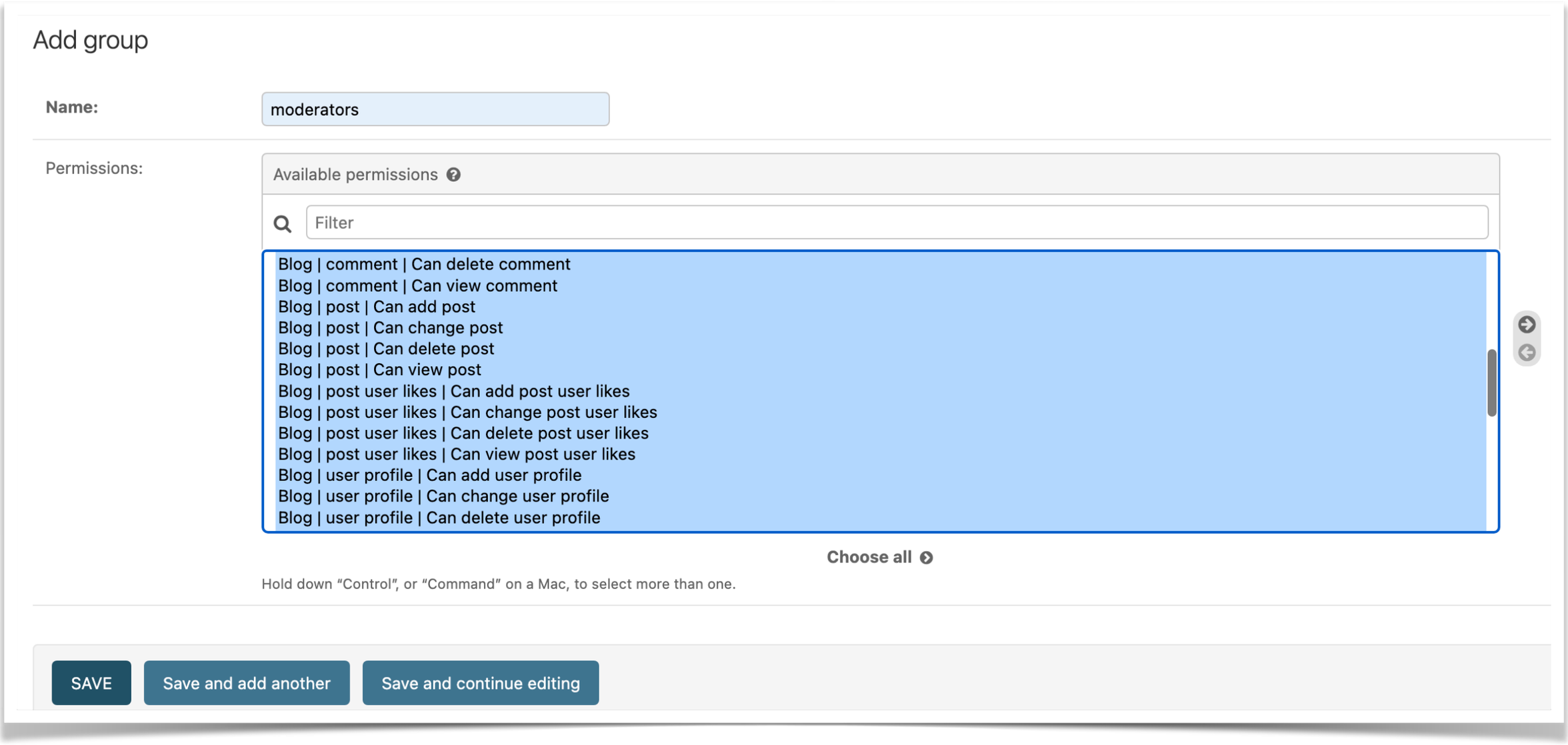
דג'נגו מגדירה את המודלים + יחסי גומלין
+ מערכת לבדיקה

אנחנו נלמד להשתמש בהם

Django Permissions



Django Permissions



Global Level Security

```
REST_FRAMEWORK = {
    'DEFAULT_RENDERER_CLASSES': [
        'rest_framework.renderers.JSONRenderer',
        'rest_framework.renderers.BrowsableAPIRenderer',
    ],

    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ]
}
```

משתמש חייב להיות מחובר כדי לבצע את כל הפעולות

```
'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.AllowAny',
]
```

ברירת מחדל

Authentication VS Authorization

Authentication
זיהוי של המשתמש

קיימת מערכת זיהוי מובנית login כבר קיים

Authorization
האם מותר למשתמש לבצע פעולה

נצטרך להגדיר הגדרות משלנו

Django Permissions

core/permissions.py

```python
from rest_framework import permissions
from rest_framework.permissions import BasePermission

class IsAdminOrModerator(BasePermission):
    # by default has_object_permission returns True
    def has_permission(self, request, view):
        is_admin = (
            request.user and
            request.user.is_authenticated and
            request.user.is_superuser
        )
        if is_admin:
            return True

        in_moderators_group = (
            request.user and
            request.user.is_authenticated and
            request.user.groups.filter(name='moderators').exists()
        )
        return in_moderators_group
```

Django Permissions

blog/views.py

```
from core.utils import try_parse_int
from .serializers import *
from .models import Comment, Post, UserProfile, PostUserLikes
from rest_framework.viewsets import ModelViewSet
from rest_framework.response import Response
from rest_framework.permissions import IsAuthenticated
from core.permissions import IsAdminOrModerator


class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
    permission_classes = [IsAdminOrModerator]
```

Django Permissions

core/permissions.py

```python
from rest_framework.permissions import BasePermission


class IsOwnerOrReadOnly(BasePermission):
    def has_object_permission(self, request, view, obj):
        # GET is allowed
        if request.method in permissions.SAFE_METHODS:
            return True
        # posts and comments has attribute author
        if hasattr(obj, 'author'):
            return obj.author.user == request.user
        return False
```

Django Permissions

blog/views.py

```python
from core.permissions import IsAdminOrModerator, IsOwnerOrReadOnly
from rest_framework.viewsets import ModelViewSet

class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
    permission_classes = [IsOwnerOrReadOnly]
```

Django Permissions

אפשר להגדיר Permission Classes
פר מתודה:

blog/views.py

```python
from rest_framework.permissions import IsAuthenticated
from core.permissions import IsAdminOrModerator, IsOwnerOrReadOnly
from rest_framework.viewsets import ModelViewSet
class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
    permission_classes = [IsOwnerOrReadOnly]

    def get_permissions(self):
        if self.actions in ["create", "update", "partial_update", "destroy"]:
            return [IsAdminOrModerator()]
        return [IsAuthenticated()]
```

Django Model Permissions

מגדירים בממשק הניהול את ההרשאות פר-קבוצה
(ראינו בממשק הניהול איך מגדירים והרשאות)

```python
from rest_framework.viewsets import ModelViewSet
from rest_framework.permissions import DjangoModelPermissions

class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
    permission_classes = [DjangoModelPermissions]
```

נכון לעכשיו:

מנהלים יכולים להוסיף ולערוך כתבות.
אדמין יכול לערוך הרשאות למשתמשים.
משתמשים יכולים להגיב לערוך תגובות ולמחוק תגובות
משתמשים יכולים לצפות בposts
משתמשים יכולים לערוך ולצפות בפרופילים
משתמשים יכולים לערוך ולצפות בלייקים

Django Model Permissions

core/permissions.py

```python
from rest_framework import permissions
from rest_framework.permissions import DjangoModelPermissions, BasePermission


class IsOwnerOrModelPermissions(DjangoModelPermissions):
    # we already have has_permission method in DjangoModelPermissions
    # so we need to override has_object_permission

    def has_object_permission(self, request, view, obj):
        if (
            request.method in permissions.SAFE_METHODS
            and super().has_permission(request, view)
        ):
            return True

        return (
            obj.author.user == request.user
            or request.user.is_superuser
            or request.user.groups.filter(name='moderators').exists()
        )
```

Django Model Permissions

views.py

```python
from core.permissions import IsOwnerOrModelPermissions

class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
    permission_classes = [IsOwnerOrModelPermissions]
```

תוכנית עבודה:

1) כל משתמש שנוצר -> יש להוסיף אוטומטית לקבוצת המשתמשים

2) משתמש לא יכול לשלוח בבקשה user_id
המזהה של המשתמש חייב להלקח מפרטי הזיהוי

3) Authentication:

ללמוד מה ברירת המחדל Base Auth
לעבור לשימוש בTokens
לעבור לשימוש בJWTTokens

Register/Login/Logout Endpoints

תוכנית עבודה:

כל משתמש שנוצר -> יש להוסיף אוטומטית לקבוצת המשתמשים

האתגר:
משתמש יכול להיווצר בModelViewSet
או בAdmin Panel

אסטרטגיה לפתרון:
Django Signals

האזנה לאירועים:

https://docs.djangoproject.com/en/5.1/topics/signals/#connecting-to-signals-sent-by-specific-senders

https://docs.djangoproject.com/en/5.1/ref/signals/#post-save

```python
from django.apps import AppConfig
from django.db.models.signals import post_save
from django.dispatch import receiver

# settings for our blog app
class BlogConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'blog'

    @receiver(post_save, sender='auth.User')
    def perform_add_user_to_users_group(sender, instance, created, **kwargs):
        # the import is nested inside the function
        from django.contrib.auth.models import Group, User
        from blog.models import UserProfile

        if not created:
            return
        group, _ = Group.objects.get_or_create(name='users')
        instance.groups.add(group)
        instance.save()

        UserProfile.objects.get_or_create(user=instance)
        print(f'User {instance.username} added to group {group.name}')
```

Authentication

ברירת המחדל של DRF עבור Authentication

היא Basic Authentication

הסיבה - מנגנון פשוט שלא דורש הגדרות.

נכיר קודם Basic Authentication

Basic Authentication

Home Workspaces API Network

Overview | GET http://127.0.0.1:8000/a | GET http://127.0.0.1:8000/c | GET http://127

HTTP **http://127.0.0.1:8000/api/v1/posts/**

GET http://127.0.0.1:8000/api/v1/posts/

Params Authorization Headers (9) Body Scripts Settings

Headers 8 hidden

	Key	Value
☑	Authorization	basic bW9zaGU6YXBhc3N3b3JkYUE=
	Key	Value

Body Cookies Headers (10) Test Results

{} JSON Preview Visualize

```
1  [
2      {
3          "id": 1,
```

חסרון של Basic
שולח את שם המשתמש והסיסמא!

כלומר בצד לקוח יצטרכו לשמור
את שם המשתמש והסיסמא

ב Local Storage
ולכן חובה להשתמש בToken

Token Authentication

```
'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.SessionAuthentication',
        'rest_framework.authentication.BasicAuthentication',
        'rest_framework.authentication.TokenAuthentication',
],
```

1

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'rest_framework.authtoken',
    'taggit',
    'hw',
    'blog.apps.BlogConfig',
]
```

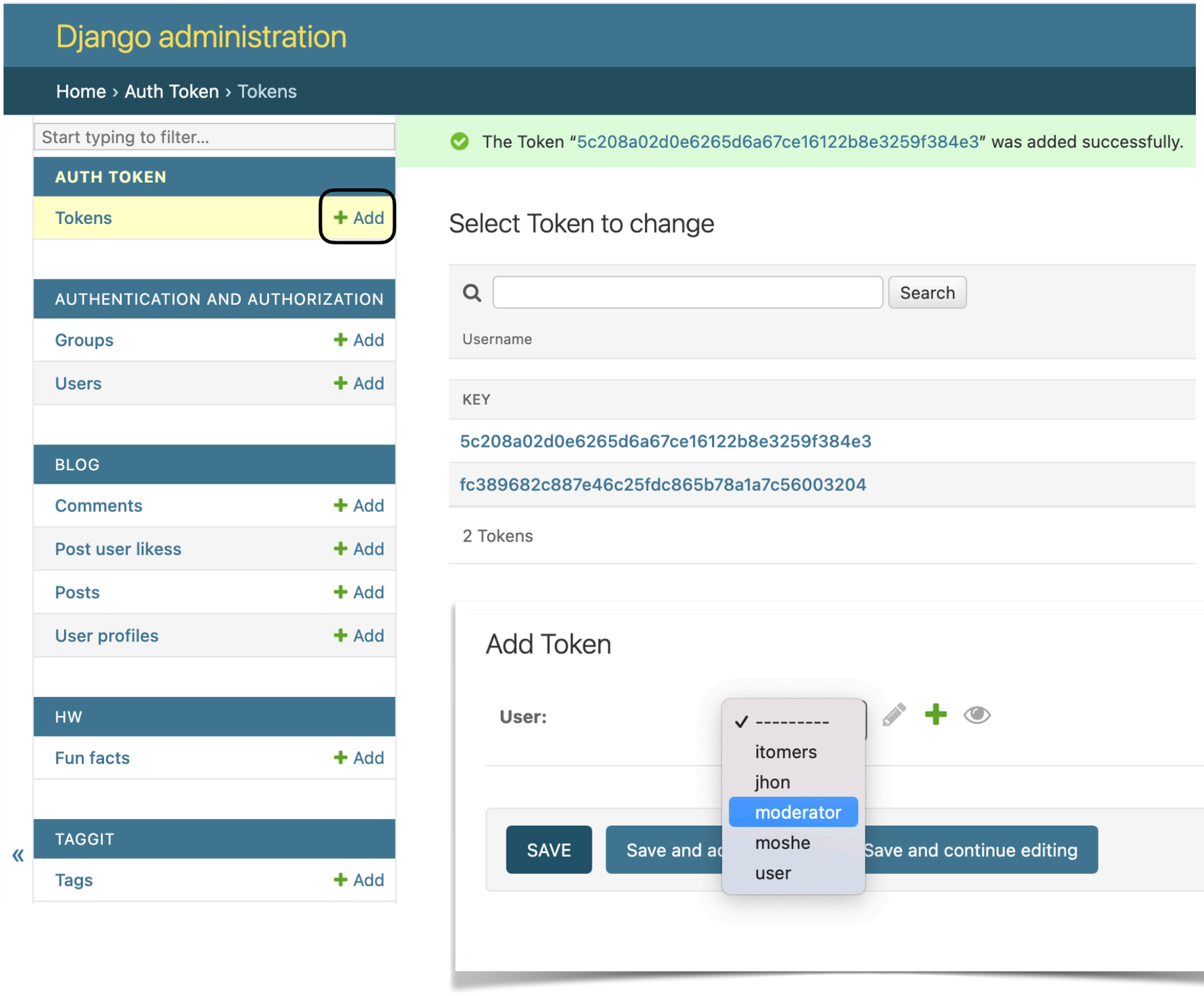
2

```
python manage.py migrate
```

3

Token Authentication

python manage.py runserver



Token Authentication

הוספת Token לכל המשתמשים שכבר קיימים:

```
python manage.py shell
```

```
from rest_framework.authtoken.models import Token
from django.contrib.auth.models import User

for user in User.objects.all():
    Token.objects.get_or_create(user=user)
```

```
exit()
```

Token Authentication

הוספת Token לכל המשתמשים שבדרך :-)

apps.py

blog > apps.py > BlogConfig > perform_add_user_to_users_group

```python
from django.apps import AppConfig
from django.db.models.signals import post_save
from django.dispatch import receiver

# settings for our blog app
class BlogConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'blog'

    @receiver(post_save, sender='auth.User')
    def perform_add_user_to_users_group(sender, instance, created, **kwargs):
        # the import is nested inside the function
        from django.contrib.auth.models import Group, User
        from blog.models import UserProfile
        from rest_framework.authtoken.models import Token

        if not created:
            return
        group, _ = Group.objects.get_or_create(name='users')
        instance.groups.add(group)
        instance.save()

        UserProfile.objects.get_or_create(user=instance)
        Token.objects.get_or_create(user=instance)
        print(f'User {instance.username} added to group {group.name}')
```

Token Authentication

הוספת נקודת קצה לLogin:

blog/urls.py

```
from rest_framework.authtoken import views as auth_views


urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls')),

    path('login/', auth_views.obtain_auth_token)
]
```

Token Authentication

הוספת נקודת קצה לLogin:

http://localhost:8000/api/v1/login/

POST http://localhost:8000/api/v1/login/

Params Authorization Headers (10) Body Scripts Settings

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

```
{
    "username":"itomers",
    "password":"123456"
}
```

Body Cookies Headers (10) Test Results

JSON Preview Visualize

```
{
    "token": "fc389682c887e46c25fdc865b78a1a7c56003204"
}
```

Token Authentication

http://127.0.0.1:8000/api/v1/posts/

Overview | GET http://127.0.0.1:8000/a | GET http://127.0.0.1:8000/c | GET http://127.0.0.1:8000/a

HTTP **http://127.0.0.1:8000/api/v1/posts/**

GET http://127.0.0.1:8000/api/v1/posts/

Params Authorization Headers (9) Body Scripts Settings

Headers 8 hidden

	Key	Value	Description
☑	Authorization	token 0eb5d7243a0d1ff7f08501f00e76d5...	
	Key	Value	Description

Body Cookies Headers (10) Test Results 200 OK

המחבר של Post או Comment ילקח מפרטי הזיהוי

https://www.django-rest-framework.org/api-guide/fields/#hiddenfield

https://www.django-rest-framework.org/api-guide/validators/#currentuserdefault

HiddenField

A field class that does not take a value based on user input, but instead takes its value from a default value or callable.

Signature: `HiddenField()`

For example, to include a field that always provides the current time as part of the serializer validated data, you would use the following:

```
modified = serializers.HiddenField(default=timezone.now)
```

```
from rest_framework.serializers import CurrentUserDefault
from rest_framework.serializers import HiddenField


class PostSerializer(TaggitSerializer, ModelSerializer):
    tags = TagField(style={'base_template': 'textarea.html'})
    author = HiddenField(default = CurrentUserDefault())
```

המחבר של Post או Comment ילקח מפרטי הזיהוי

https://www.django-rest-framework.org/api-guide/fields/#hiddenfield

https://www.django-rest-framework.org/api-guide/validators/#currentuserdefault

```python
from rest_framework.serializers import HiddenField

class CurrentUserDefault():
    requires_context = True

    def __call__(self, serializer_field):
        request =  serializer_field.context['request']
        return request.user.userprofile


class PostSerializer(TaggitSerializer, ModelSerializer):
    tags = TagField(style={'base_template': 'textarea.html'})
    author = HiddenField(default = CurrentUserDefault())

    class Meta:
        model = Post
        fields = '__all__'
```

אותו דבר בCommentSerializer:

```python
class CommentSerializer(ModelSerializer):
    author = HiddenField(default = CurrentUserDefault())
    author_id = SerializerMethodField()

    class Meta:
        model = Comment
        fields = '__all__'

    def get_author_id(self, obj):
        return obj.author.id
```

הערה:
אפשר ליצור מחלקה שיורשת מModeSerializer

עם התכונות:

```python
author = HiddenField(default = CurrentUserDefault())
author_id = SerializerMethodField()
```

והמתודה

ואז לרשת מהמחלקה ב-2 הSerializers שיצרנו

יצירת משתמשים:

רישום משתמש חדש -> פעולת דטה-בייס

בדיקה שהמידע תקין

הצפנת הסיסמא

Serializer
View
Urls

יצירת משתמשים:

serializers.py

```python
from rest_framework.serializers import ModelSerializer
from django.contrib.auth.models import User

class UserSerializer(ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username', 'email', 'password']
        extra_kwargs = {
            'password': {'write_only': False, 'required': True},
            'id': {'read_only': True},
            'email': {'required': True},
            'username': {'required': True, 'min_length':3},
        }

    def create(self, validated_data):
        user = User.objects.create_user(**validated_data)
        return user
```

views.py

```python
class UsersViewSet(ModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer
```

נסיים בשיעור הבא :-)

urls.py

```python
from .views import UsersViewSet
router.register('auth', UsersViewSet, basename='auth')
```