لاگ بقا

این پروژه یک سیستم API برای بازماندگان یک دنیای زامبی‌زده است. هدف اصلی اینه که بازماندگان بتونن اطلاعات حیاتی رو به صورت امن با هم به اشتراک بذارن.

می‌خواهیم یک API بسازیم که در آن:

- کاربران بتوانند به عنوان «**بازمانده**» (Survivor) در سیستم ثبت‌نام کرده و هویت خود را تایید کنند.
- بازماندگان تایید شده بتوانند «**لاگ بقا**» ثبت کنند و بگویند در کدام «**منطقه امن**» (Safe Zone) هستند و چه پیامی برای دیگران دارند.
- هر کسی (حتی کسانی که وارد سیستم نشده‌اند) بتواند لاگ‌های ثبت شده را بخواند تا از وضعیت دیگران با خبر شود.
- یک بازمانده فقط بتواند لاگ‌های خودش را ویرایش یا حذف کند.
- فقط بازماندگانی که هویتشان توسط سیستم به عنوان «**تایید شده**» (Verified) مشخص
- شده، اجازه ثبت لاگ را داشته باشند. این کار جلوی زامبی‌های باهوش را می‌گیرد که ممکن است یاد گرفته باشند از سیستم استفاده کنند و مناطق آلوده را به عنوان امن جا بزنند!
- «**مناطق امن**» از قبل در دیتابیس تعریف شده‌اند و کاربران نمی‌توانند منطقه جدیدی اضافه کنند، فقط می‌توانند از لیست موجود انتخاب کنند.

Authentication

اینجا هویت افراد را مشخص می‌کنیم. هر کاربر یک «بازمانده» است.

برای ثبت‌نام یک بازمانده جدید: POST /api/survivors/register/.

برای ورود و گرفتن توکن مجوز دسترسی: POST /api/survivors/token/

وقتی به اندپوینت توکن ریکوئست ارسال کردید یه توکن بهتون میده. چطور درخواست‌های بعدی رو ارسال می‌کنیم؟ توکن رو ورمیداریم و به اینصورت میذاریمش داخل هدر ریکوئست‌های بعدیمون:

Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b

اندپوینت‌های لاگ

GET /api/logs/: (نمایش لیست تمام لاگ‌های بقا برای همه. (نیازی به لاگین ندارد

GET /api/logs/<id>/: نمایش جزئیات یک لاگ خاص.

POST /api/logs/: (ثبت یک لاگ جدید. (نیاز به **لاگین** و **تایید شده بودن** بازمانده دارد

PUT /api/logs/<id>/: (ویرایش یک لاگ. (فقط توسط **صاحب** آن لاگ امکان‌پذیر است

DELETE /api/logs/<id>/: (حذف یک لاگ. (فقط توسط **صاحب** آن لاگ امکان‌پذیر است

چالش آخرالزمانی

هر بار که کسی لاگ جدید می‌فرسته، یه عدد تصادفی بین ۱ تا ۱۰۰ تولید کن.

- اگر اون عدد **زوج** بود، لاگ با تاخیر ۱۰ ثانیه ذخیره بشه
- اگر فرد ۳ بار متوالی تاخیر خورد، باید یک **معما** بهش نشون بدی و حلش کنه تا اجازه ثبت لاگ بعدی رو بگیره.

چالش بعدی: ساختن یه سیستم اطلاعاتی

تا الان، بازماندگان فقط می‌گفتن: «من در منطقه امن A **هستم**». این خوبه، ولی کافی نیست.

ایده جدید اینه که بازماندگان بتونن گزارش بدن: «من **از** منطقه امن A **به** منطقه امن B رفتم و در طول مسیر این چیزها رو دیدم.»

با جمع‌آوری این گزارش‌ها، ما یک **نقشه هوشمند از مسیرهای ارتباطی** می‌سازیم. بعد از مدتی می‌تونیم به سوال‌های حیاتی جواب بدیم:

- کدام مسیر بین دو منطقه امن، بیشترین رفت و آمد رو داشته (احتمالاً امن‌تره)؟
- در کدام مسیرها زامبی دیده شده؟
- کدام مناطق کاملاً ایزوله شدن و هیچ راهی بهشون ثبت نشده؟

چی قراره برگردونید؟

```
[
  {
    "from_zone__name": "مدرسه",
    "to_zone__name": "بیمارستان",
    "total_reports": 8,
    "zombie_alerts": 3
  },
  ...
]
```

:/GET /api/connections

نمایش لیست تمام مسیرهای ثبت‌شده بین مناطق (برای همه کاربران قابل مشاهده است).

:/POST /api/connections

ثبت یک مسیر جدید بین دو منطقه. (فقط توسط بازمانده‌های لاگین‌شده و تایید شده)

:/GET /api/connections/graph

نمایش خلاصه تحلیلی مسیرها شامل تعداد گزارش و هشدار زامبی بین هر دو منطقه (قابل استفاده برای رسم گراف و تحلیل رفت‌وآمد) – هنوز باید اضافه شود.

Step 0: Set Up the Environment

```

```
python -m venv .venv
source env/bin/activate   # or .\env\Scripts\activate on Windows
pip install django djangorestframework
django-admin startproject apocalypse_api .
python manage.py startapp survivors
python manage.py startapp logs
```
```

```

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'survivors',
    'logs',
    'rest_framework.authtoken',]
```

## Step 1: Set Up Authentication

```

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
    ]
}
```

```

## Step 3: Models

```

```
class ZoneConnection(models.Model):
    from_zone = models.ForeignKey(Zone, related_name='connections_from',
on_delete=models.CASCADE)
    to_zone = models.ForeignKey(Zone, related_name='connections_to',
on_delete=models.CASCADE)
    survivor = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    note = models.TextField(blank=True)
    seen_zombies = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.from_zone.name} → {self.to_zone.name} by
```

```
{self.survivor.username}"
```
```

## Step 3: Serializers

```
class ZoneSerializer(serializers.ModelSerializer):
    class Meta:
        model = Zone
        fields = ['id', 'name', 'description']

class SurvivalLogSerializer(serializers.ModelSerializer):
    class Meta:
        model = SurvivalLog
        fields = '__all__'
        read_only_fields = ['owner', 'created_at']
```

Step 4: Views

```
``
@api_view(['GET'])
def zone_list(request):
    zones = Zone.objects.all()
    serializer = ZoneSerializer(zones, many=True)
    return Response(serializer.data)

@api_view(['GET', 'POST'])
@permission_classes([permissions.IsAuthenticatedOrReadOnly,
IsVerifiedSurvivor])
def survival_log_list(request):
    if request.method == 'GET':
        logs = SurvivalLog.objects.select_related('owner', 'zone')
        if request.user.is_authenticated:
            logs = logs.filter(models.Q(is_public=True) |
models.Q(owner=request.user))
        else:
            logs = logs.filter(is_public=True)
        serializer = SurvivalLogSerializer(logs, many=True)
```
```

```
        return Response(serializer.data)


@api_view(['GET', 'PUT', 'DELETE'])
@permission_classes([permissions.IsAuthenticatedOrReadOnly])
def survival_log_detail(request, pk):
    log = get_object_or_404(SurvivalLog, pk=pk)
    if request.method == 'GET':
        if log.is_public or (request.user.is_authenticated and log.owner
== request.user):
            serializer = SurvivalLogSerializer(log)
            return Response(serializer.data)
        return Response({"detail": "دسترسی ندارید."},
status=status.HTTP_403_FORBIDDEN)

    if log.owner != request.user:
        return Response({"error": "Not your log."},
status=status.HTTP_403_FORBIDDEN)
```