Database table

id title description goal image ...

1 Fidofund Fido needs \$ \$5000 dog.jpg ...

2 RexFactor Rex is sick \$9000 dog1.jpg ...

The database stores information as a set of tables. Each table has rows and columns. Each column relates to one piece of information, and each row relates to one instance.

Here we have a table of projects.
The "title" column stores information about what each instance is called, and there are two rows - one for a project instance called "FidoFund", and one for a project instance called "RexFactor".

models.py

class Project(models.Model):

/ title = models.CharField(max_length=200)
description = models.TextField()
goal = models.IntegerField()
image = models.URLField()
is_open = models.BooleanField()
date_created = models.DateTimeField()
owner = models.CharField(max_length=200)

The models.py files contain code that defines our models. Each field in a model represents a column in the database table that stores information about instances of that model.

For instance, this code says that every instance of a project needs to have a name, a description, a goal, etc...

serializers.py

class ProjectSerializer(serializers.Serializer):

id = serializers.ReadOnlyField()
title = serializers.CharField(max_length = 200)
description = serializers.CharField(max_length=None)
goal = serializers.IntegerField()
image = serializers.URLField()
is_open = serializers.BooleanField()
date_created = serializers.DateTimeField()
owner = serializers.CharField(max_length=200)

def create(self, validated_data):
return Project.objects.create(**validated_data)

The **serializers.py** files contain code that handles two things:

- 1. **converting** data from **requests** into **model instances** for the database and...
- 2. **converting model instances** from the database into data for **responses**.

This example defines a list of fields that we expect to find in data relating to Project models.

views.py

class ProjectList(APIView):

def get(self, request):
 projects = Project.objects.all()
 serializer = ProjectSerializer(projects, many=True)
 return Response(serializer.data)

def post(self, request):
 serializer = ProjectSerializer(data=request.data)
 if serializer.is_valid():
 serializer.save()
 return Response(
 serializer.data,
 status=status.HTTP_201_GREATED
)
 return Response(
 serializer.errors,
 status=status.HTTP_400_BAD_REQUEST
)

The views.py files contain code that tells each endpoint what do to when it receives a given type of request, and what to return in response.

Very often, requests contain data about models that need to be processed in some way, and responses contain data about models from the database.

urls.py

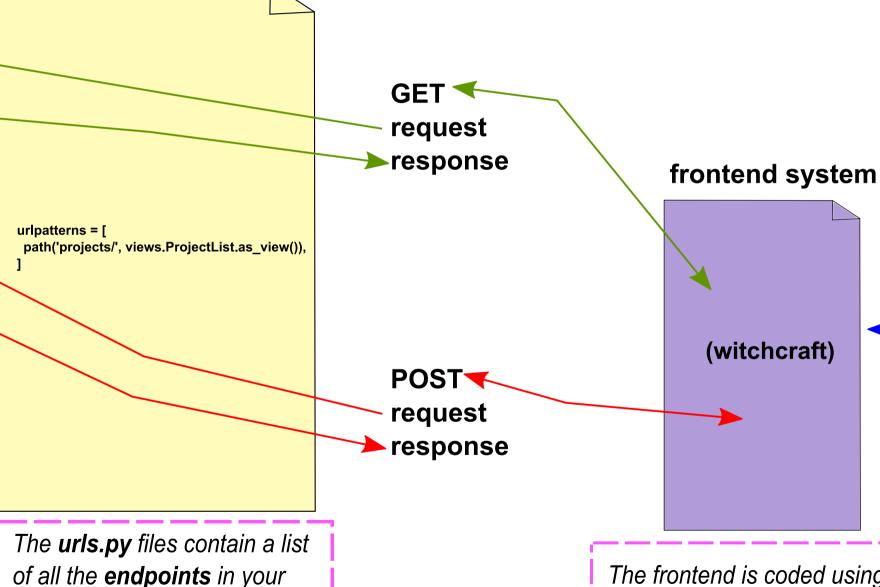
backend system. Each

endpoint is an address that

the frontend can access, which

takes **requests** as input and

returns **responses** as output.



The frontend is coded using

React.js. It submits requests

to the backend, and uses the
data in responses the
backend provides to build a
beautiful, functional website.





