Web Development

Session 2 - Backend

BY GDSC@KTH

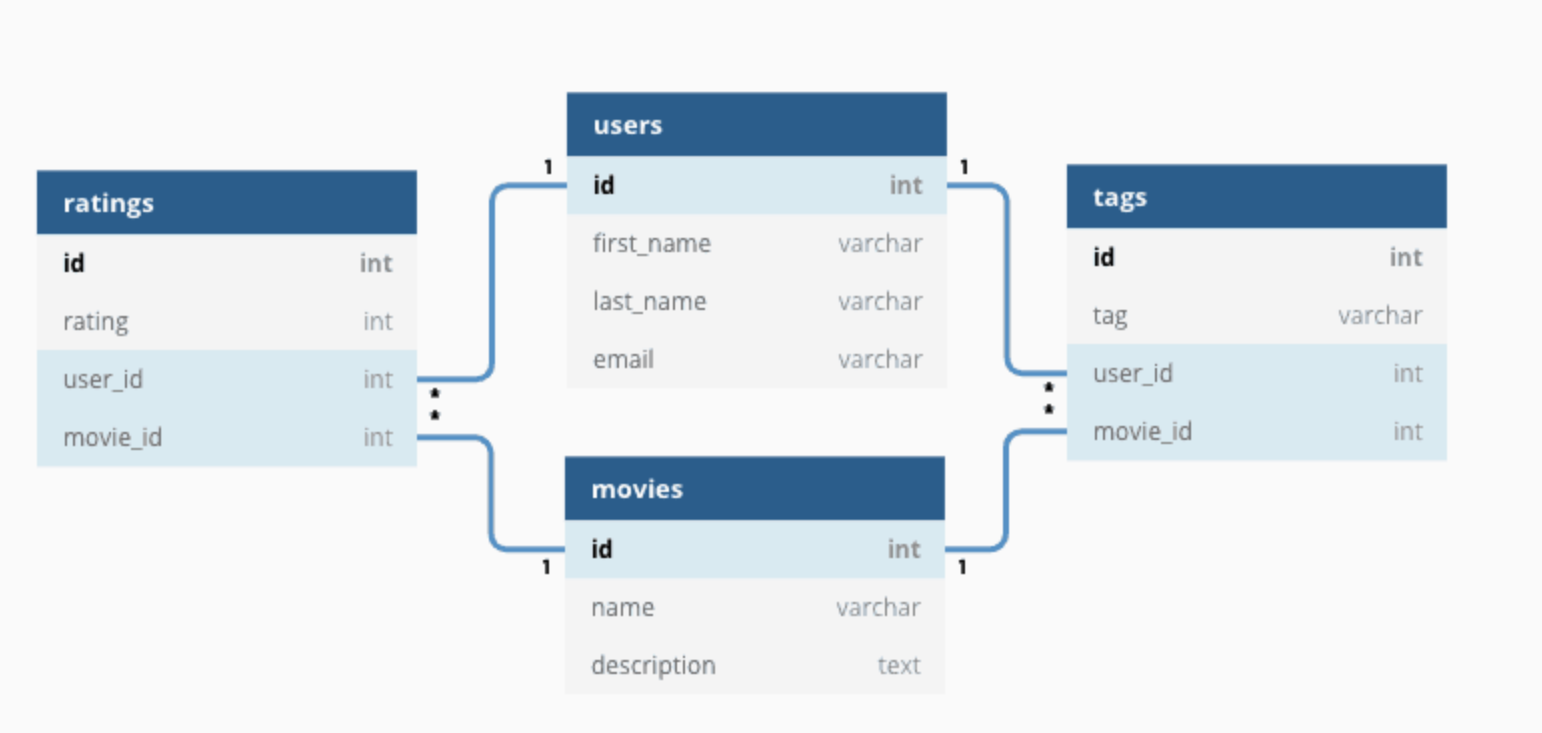
Zezhe Huang

What Can Be Expected Today?

- Manage blog data in database
- Build up APIs for a Blog app
  - Create, edit, view and delete blog data
- Access control
  - Only the author can access his/her blogs

Databases

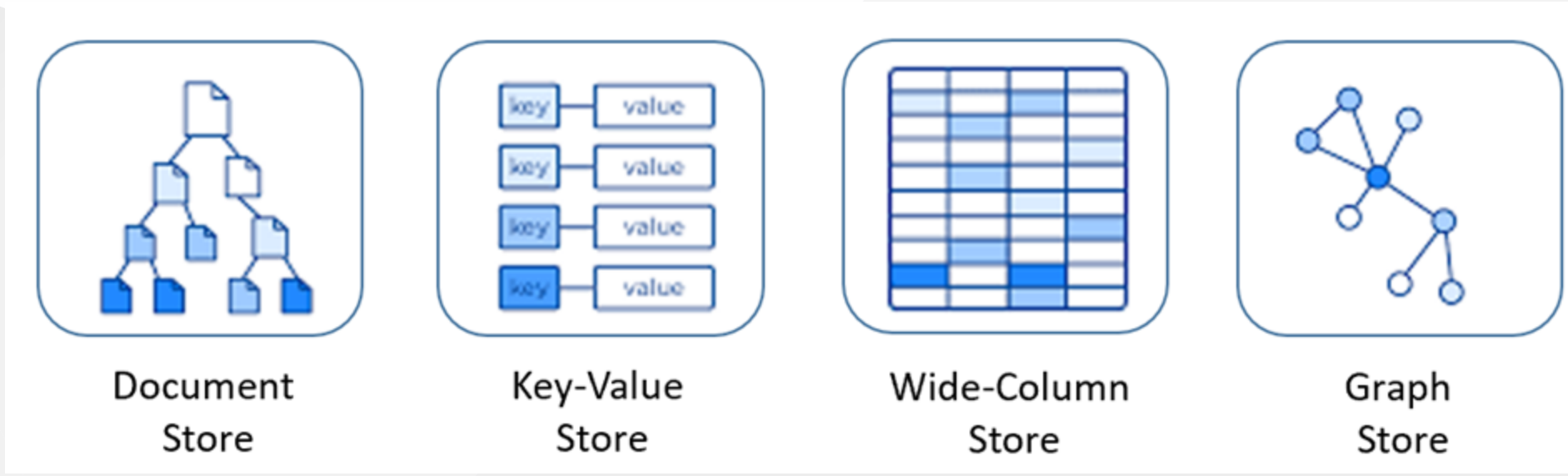
Relation Databases (eg. PostgreSQL)



https://www.omnisci.com/technical-glossary/relational-database

Databases

NonSQL Databases (eg. MongoDB)



https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data

NonSQL Databases - Document

```
{
    "FirstName": "Bob",
    "Address": "5 Oak St.",
    "Hobby": "sailing"
}
```

```
<contact>
    <firstname>Bob</firstname>
    <hobby>sailing</phone>
    <address>
      <street1>5 Oak St.</street1>
    </address>
</contact>
```

https://en.wikipedia.org/wiki/Document-oriented_database

NonSQL Databases - Key-Value

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

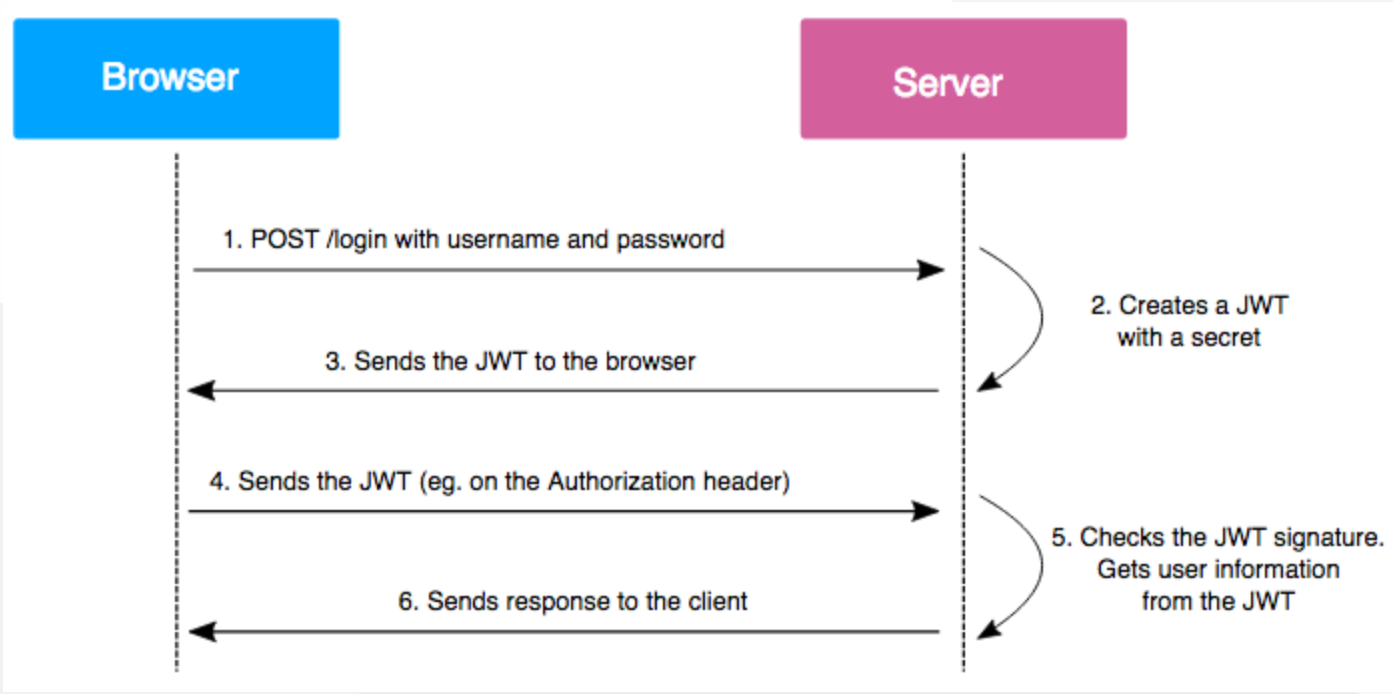
https://upload.wikimedia.org/wikipedia/commons/5/5b/KeyValue.PNG

Authorization

- Basic authentication
  - Base64 encoding of ID and password joined by a single colon `:`
- OAuth 2.0
- **Json Web Token**
  - Less requests
  - Stateless
- ...

https://en.wikipedia.org/wiki/Basic_access_authentication

Json Web Token



https://www.vaadata.com/blog/jwt-tokens-and-security-working-principles-and-use-cases/

Restful API

REpresentational State Transfer (REST) architectural constraints:

1. Uniform interface
2. Client-server
3. Stateless
4. Cacheable
5. Layered system
6. Code on demand

https://restfulapi.net/

Setup of Django Project

```
pip install django
django-admin startproject myblog
cd myblog
```

Now have a look at the settings! What we'll use today:

- `INSTALLED_APPS`
- `DATABASES`

https://www.djangoproject.com/start/

PostgreSQL Configuration

- Run docker instance: `docker run --name my-postgres -p 5432:5432 -e POSTGRES_PASSWORD=mypassword -d postgres`
- Access bash in docker instance: `docker exec -it DOCKER_ID bash`

```
su postgres
psql
CREATE DATABASE myblog;
```

Project Settings

Install database engine: `pip install psycopg2-binary`

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'myblog',
        'USER': 'postgres',
        'PASSWORD': 'mypassword',
        'HOST': 'localhost',
        'PORT': 5432,
    }
}
```

Migrate the database by `python manage.py migrate`

and create a user: `python manage.py createsuperuser`

Blog Model

- Create an app: `python manage.py startapp blogs`
- Add the app to settings:

```
INSTALLED_APPS = [
    ...
    'blogs',
]
```

Model Class

Define the model in `models.py`

```
class Blog(models.Model):
    author = models.ForeignKey(
        settings.AUTH_USER_MODEL,on_delete=models.CASCADE,
    )
    created_at = models.DateTimeField(auto_now_add=True)
    title = models.TextField()
    content = models.TextField()
```

Migrate to database:

```
python manage.py makemigrations
python manage.py migrate
```

Admin Page

Register to admin page in `admin.py`

```
from django.contrib import admin
from blogs.models import Blog

class BlogAdmin(admin.ModelAdmin):
    pass
admin.site.register(Blog, BlogAdmin)
```

Run the server by

`python manage.py runserver 0.0.0.0:8000`

and check out the admin page:

http://localhost:8000/admin

Json Web Token

Install Django REST framework JWT:

```
pip install djangorestframework-simplejwt
```

Follow the guide to configure

https://django-rest-framework-simplejwt.readthedocs.io/en/latest/getting_started.html#installation

Restful API

Install Django REST framework:

```
pip install djangorestframework
```

Add `'rest_framework'` to your `INSTALLED_APPS` setting.

```
INSTALLED_APPS = [
    ...
    'rest_framework',
]
```

Serializer and ViewSet

Create `Serializer` and `Viewset` in `views.py`

```
class BlogSerializer(serializers.ModelSerializer):
    author = serializers.PrimaryKeyRelatedField(read_only=True)
    class Meta:
        model = Blog
        fields = '__all__'


class BlogViewSet(viewsets.ModelViewSet):
    queryset = Blog.objects.all()
    serializer_class = BlogSerializer
```

Router

Add urls in `urls.py`

```
router = routers.SimpleRouter()
router.register(r'blogs', BlogViewSet)

urlpatterns = [
    ...
    path('', include(router.urls)),
]
```

Router document: https://www.django-rest-framework.org/api-guide/routers/

Access Control

Only allow the author to change his/her blogs:

```
class AuthorPermission(permissions.BasePermission):
    def has_object_permission(self, request, view, obj):
        if request.method in permissions.SAFE_METHODS:
            return True
        return obj.author == request.user
```

Assign it to the `permission_classes` of `BlogViewSet`:

```
permission_classes = [AuthorPermission, ]
```

Check Out Your Results

Online demo: https://gdsc-web.herokuapp.com/

Available users:

`admin:admin`

`testuser:testpassword`

Useful Links

- Authorization: https://learning.postman.com/docs/sending-requests/authorization/