Preparations

Conda

```
conda create -n presenv

conda activate presenv

conda install pip
```

Keycloak

- [ ] Import Realms
- [ ] Regenerate Client secrets
- [ ] Update Code for backend and frontend
- [ ] Add Users

Pycharm

- [ ] Presentation Mode
- [ ] Distraction Free Mode
- [ ] Select in Project View Mode
- [ ] Git File Add ausgeschalten
- [ ] Conda Env eingestellt

Console

- [ ] Conda env aktiviert
- [ ] Zoom in
- [ ] Httpie installiert

Firefox

- [ ] Tab auf localhost:8000
- [ ] Rangezoomed

Quotes

- I bin glei finito, owa gewaltig - Renate Willmann
- Herr Kuchinka, Sie san leicht asozial! - Franz Jakob
- Polymer… Teddybär!! - Maximilian Wahl
- Da Schaß steht jo meistns eha vorm Rechner! - Gerald Köck
- Damma a weng schummeln. Des schodt ned. - Thomas Stütz
- Verschicken heißt in dem Fall afoch nur Schreddern - Jan Neuburger
- Des is eh ka Torpedo… Des is a Schokobon - Franz Jakob
- Richtige Antwort, aber falsch! - Franz Jakob
- Wonn wir donn Übungen mochn wirst du sterben… ALLEINE! - Johannes Tumfart

Flask Quote

Basic App

Setup

```
pip install flask
pip install flask-restful
```

Hello World

main.py

```python
from flask import Flask
from flask_restful import Resource, Api, marshal_with, fields

app = Flask(__name__)
api = Api(app)


class HelloWorld(Resource):
    def get(self):
        return {"Hello": "world"}


api.add_resource(HelloWorld, '/')

if __name__ == '__main__':
    app.run(debug=True)
```

Run the App

```
export FLASK_APP=main.py
flask run
```

Model

```
class Person:
    def __init__(self, name, idnum=None):
        self.id = idnum
        self.name = name


class Quote:
    def __init__(self, text, person, idnum):
        self.id = idnum
        self.text = text
        self.person = person


persons = [
    Person("Erik Mayrhofer", 1),
    Person("Test Person", 2)
]

quotes = [
    Quote("Hello this is a quote", persons[1], 1),
    Quote("Hello this is a quote2", persons[0], 2),
    Quote("Hello this is a quote3", persons[1], 3),
]
```

Resource

```
persons_fields = {
    'id': fields.Integer,
    'name': fields.String
}
quote_fields = {
    'id': fields.Integer,
    'text': fields.String,
    'person': fields.Nested(nested=persons_fields)
}


class QuotesResource(Resource):
    @marshal_with(quote_fields)
    def get(self):
        return quotes


api.add_resource(QuotesResource, '/quotes/')
```

Ktor "quotemanager"

Kotlin Web-Framework

Erklärungen

- Client und Serverseitig
- "Quick and Dirty"
- Datenbankanbindung über JPA selber machen
- Ktor Backends (Netty, Jetty, Tomcat)

Todos

Basic Project

Hello World

```kotlin
fun Application.module(testing: Boolean = false) {
    routing {
        get("/") {
            call.respondText("Hello Worlds")
        }
    }
}
```

Model

src/model/Model.kt

```kotlin
data class Quote(
    var id: Int?
) {
    lateinit var text: String
    lateinit var person: Person
}

data class Person(
    var id: Int?
){
    lateinit var name: String
}
```

src/Application.kt

```
val persons = mutableListOf(
    Person(1).apply { name="Erik Mayrhofer" },
    Person(2).apply { name="Test" }
)

val quotes = mutableListOf(
    Quote(1).apply{text="SomeQuoteTest"; person=persons[0]},
    Quote(2).apply{text="SomeQuoteTest"; person=persons[0]}
)
```

Quotes Query

build.gradle

```
compile "org.jetbrains.kotlin:kotlin-stdlib-jdk8:$kotlin_version"
```

src/Application.kt

```
install(ContentNegotiation){
    jackson {}
}
routing {
    get("/quotes") {
        call.respond(quotes)
    }
}
```

Test Queries

```
http localhost:8080/quotes/
```

Post Request

```
fun insertQuote(quote: Quote): Quote{
    quote.person = persons.single { it.id == quote.person.id }
    quote.id = (quotes.mapNotNull { it.id }.max() ?: 0) + 1
    quotes += quote
    return quote
}
...
post("/quotes"){
    val quote = call.receive<Quote>()
    insertQuote(quote)
    call.respond(quote)
}
```

Http Templating

```
implementation "io.ktor:ktor-html-builder:$ktor_version"
```

```
get("/quoteui") {
    call.respondHtml {
        body {
            h1 {
                +"Quotes"
            }
            quotes.forEach {
                ul {
                    p {
                    +"Quote: ${it.text}"
                    }
                }
            }
        }
    }
}
```

Django "quotemanager"

Erik Mayrhofer

Quarkus zu Spring = Flask zu Django

NOTE Homepage: https://www.django-rest-framework.org/

WARNING Trailing slash

Erklärungen

1. Project vs App (Like a Application Server)
2. View, Serializer, Model
3. HATEOAS
4. WSGI

Slide Todos

1. Basic Project
    a. Projekt und App erstellen
    b. Model erstellen (Quote)
    c. Admin panel
    d. Serializer erstellen
    e. View erstellen
    f. URLs eintragen und App installen
    g. Testen
2. Modell verbessern
    a. ModellKlasse
    b. Serializer
    c. View
    d. Url
3. Hateoas
4. Keycloak

Basic Project

Setup

```
pip install django
pip install djangorestframework
```

Create Project

```
django-admin startproject quotemanager

cd quotemanager

python manage.py migrate

python manage.py createsuperuser

charm

python manage.py startapp quotes
```

Modell erstellen

quotes/models.py

```python
from django.db import models


class Quote(models.Model):
    text = models.CharField(max_length=250)
    person = models.CharField(max_length=50)
```

Admin Panel herzeigen

`http://localhost:8000/admin/` im Browser

Serializer erstellen

quotes/serializers.py

```
from rest_framework import serializers

from .models import Quote


class QuoteSerializer(serializers.ModelSerializer):
    class Meta:
        model = Quote
        fields = ('id', 'text', 'person')
```

View erstellen

quotes/views.py

```
from django.shortcuts import render
from rest_framework import viewsets

from .models import Quote
from .serializers import QuoteSerializer


class QuoteView(viewsets.ModelViewSet):
    queryset = Quote.objects.all()
    serializer_class = QuoteSerializer
```

URLs und App eintragen

quotemanager/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('quotes.urls'))
]
```

quotes/urls.py

```
from django.urls import path, include
from rest_framework import routers

from .views import QuoteView

router = routers.DefaultRouter()
router.register('quotes', QuoteView)

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

migrations

```
python manage.py makemigrations

python manage.py migrate

python manage.py runserver
```

Testen

1. `localhost:8000/quotes` im Browser
2. Commandline:

```
http -v POST localhost:8000/quotes/ text="Polymer Teddibär" person="Max Wahl"
```

3. Bei der Console auch Fehlenden Parameter prüfen (validation), und PUT DELETE, zu langer Name (max 50 Zeichen)
4. Admin Gui im Browser `localhost:8000/admin` ⇒ Nichts wird angezeigt

Admin

quotes/admin.py

```
from django.contrib import admin

from quotes.models import Quote

admin.site.register(Quote)
```

1. Admin Gui im Browser `localhost:8000/admin` ⇒ Anzeige Hässlich

quotes/models.py

```
class Quote(models.Model):
    ...
    def __str__(self):
        return f"{self.text} - {self.person}"
```

Modell Verbessern

Klassen anpassen

Wir fügen eine Person hinzu

quotes/models.py

```
class Person(models.Model): ①
    name = models.CharField(max_length=50)

    def __str__(self):
        return self.name


class Quote(models.Model):
    text = models.CharField(max_length=250)
    person = models.ForeignKey(Person, on_delete=models.CASCADE) ②

    def __str__(self):
        return f"{self.text}" ③
```

① Person erstellen

② Foreign key Setzen

③ Str anpassen

Migrieren

```
python manage.py makemigrations
```

SQLite-DB hat Daten drinnen: Einfach das File löschen

```
python manage.py migrate
```

Serializer

quotes/serializers.py

```
class PersonSerializer(serializers.ModelSerializer):
    class Meta:
        model = Person
        fields = ('id', 'name')
```

View

quotes/views.py

```
class PersonView(viewsets.ModelViewSet):
    queryset = Person.objects.all()
    serializer_class = PersonSerializer
```

Urls

qutoes/urls.py

```
router.register('persons', PersonView)
```

Testen

```
python manage.py runserver
```

1. Web Interface Browsen

Hateoas

quotes/serializers.py

```
class ...Serializer(serializers.HyperlinkedModelSerializer):
    fields = (..., 'url', ....)
```

Keycloak

Install

```
pip install django-oauth-toolkit
```

quotemanager/settings.py

```
INSTALLED_APPS = [
    ...
    'oauth2_provider',
    ...
]

...

OAUTH2_PROVIDER = {
    'SCOPES': {'read': 'Read scope', 'write': 'Write scope', 'groups': 'Access to your
groups'},
    'RESOURCE_SERVER_INTROSPECTION_URL':
        'http://localhost:8080/auth/realms/master/protocol/openid-
connect/token/introspect',
    'RESOURCE_SERVER_INTROSPECTION_CREDENTIALS': ('django-backend', '7031ca56-87dc-
4f2b-aa93-52fb79eb5a86')
}

REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.IsAuthenticated',
    ),
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'oauth2_provider.contrib.rest_framework.OAuth2Authentication',
    ]
}
```

Protecting the API

```
from rest_framework import viewsets, permissions
```

```
class QuoteView(viewsets.ModelViewSet):
    ...
    permission_classes = [permissions.IsAuthenticated]
```

Testing out the Endpoitn

Obtain Bob’s Token

```
http -p b --form \
  -a frontend:460d1a14-b774-482e-b03e-a3830874d9c1 \
  POST localhost:8080/auth/realms/master/protocol/openid-connect/token \
  username=bob password=bob grant_type=password | jq -r ".access_token"

TOKEN=$(http -p b --form \
  -a frontend:460d1a14-b774-482e-b03e-a3830874d9c1 \
  POST localhost:8080/auth/realms/master/protocol/openid-connect/token \
  username=bob password=bob grant_type=password | jq -r ".access_token")
```

Query Protected URL

```
http -v localhost:8000/quotes/ Authorization:"Bearer $TOKEN"
```