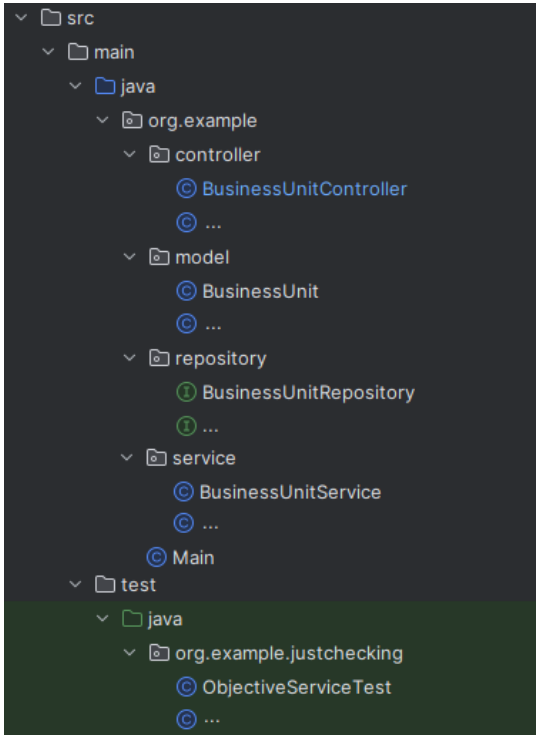
TH BINGEN
University of Applied Sciences

Projekt: EPRO

Backend für OKR-Framework

Grundstruktur



- @RestController + entsprechendem mapping
- @Entity
- @Repository
- @Service
- Main.java
- Test-Klasses

Entity - BusinessUnit

- id, name, size und company Variablen/Verweis
- List<Objective>, List<Unit>
- Konstruktor, Getter, Setter
- toString Methode

Entity - Company

- id, name und size Variablen
- List<BusinessUnit>, List<Objective>
- Konstruktor, Getter, Setter
- toString Methode

Entity - HistoricalDataEntry

- id, date, value (KeyResult.current), comment Variablen
- Verweis auf KeyResult
- Konstruktor, Getter, Setter
- setComment Setter stellt sicher, dass Kommentare nicht leer sind

Entity - KeyResults

- id, name, progress, current, goal, confidenceLevel und buObjective Variablen/Verweis
- List<HistoricalDataEntry>
- Konstruktor, Getter, Setter
- setCurrent Setter stellt sicher, dass Kommentare nicht leer sind
- setGoal Setter stellt Goal > 0.0 sicher
- progress wird dynamisch anhand von current und goal berechnet

Entity - Objective

- id, name Variablen
- List<KeyResult>
- Konstruktor, Getter, Setter
- toString Methode

Entity - Unit

- id, name, size und businessUnit Variablen/Verweis
- List<Objective>
- Konstruktor, Getter, Setter
- toString Methode

Entity - User

- id, username und password Variablen
- List<Role> für alle Zugriffsrechte des Users
- Konstruktor, Getter, Setter
- toString Methode

Service Klassen

- Service Klassen für alle Entities
- getAll, getById, create, update und delete
- weiteres Sicherstellen der Geschäftslogik
- DashboardService

Service Klassen

- getAllEntity Methoden über die JpaRepository findAll() Methode

```
public List<Company> getAllCompanies() {
    return companyRepository.findAll();
}
```

```
public List<Objective> getAllObjectives() {
    return objectiveRepository.findAll();
}
```

Service Klassen

- getEntityById Methoden über die JpaRepository getById() Methode

```
public Company getCompanyById(Long id) {
    Optional<Company> company = companyRepository.findById(id);
    return company.orElse( other: null);
}
```

```
public Objective getObjectiveById(Long id) {
    Optional<Objective> objectiveOptional = objectiveRepository.findById(id);
    return objectiveOptional.orElse( other: null);
}
```

Service Klassen

- createEntity Methoden über die save Methode von CrudRepository

```java
public Objective createObjective(Objective objective) {
    if (objective.getKeyResults().size() < 3) {
        throw new IllegalArgumentException("Objectives must have at least three Key Results.");
    }
    if (objective.getKeyResults().size() > 5) {
        throw new IllegalArgumentException("Objectives can only have up to five Key Results.");
    }
    return objectiveRepository.save(objective);
}
```

```java
public Company createCompany(Company company) {
    return companyRepository.save(company);
}
```

Service Klassen

- updateEntity Methoden über die save Methode von CrudRepository

```java
public Company updateCompany(Long id, Company company) {
    if (companyRepository.existsById(id)) {
        company.setId(id);
        return companyRepository.save(company);
    }
    return null;
}
```

```java
public BusinessUnit updateBusinessUnit(Long id, BusinessUnit businessUnit) {
    if (businessUnitRepository.existsById(id)) {
        businessUnit.setId(id);
        return businessUnitRepository.save(businessUnit);
    }
    return null;
}
```

Service Klassen

- deleteEntity Methoden über deleteById von CrudRepository

```java
public boolean deleteKeyResult(Long id) {
    try {
        keyResultRepository.deleteById(id);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

```java
public boolean deleteHistoricalDataEntry(Long id) {
    if (historicalDataEntryRepository.existsById(id)) {
        historicalDataEntryRepository.deleteById(id);
        return true;
    }
    return false;
}
```

DashboardService

- Print für alle Objectives inklusive assoziierte KeyResults

- Print für explizites Objective inklusive assoziierte KeyResults

Controller Klassen

- Endpunkte für HTTP – Anfragen
- GET
  - greift auf [Entity]Service.get[Entity]ById() zurück
- POST
  - greift auf [Entity]Service.create[Entity]() zurück
- PUT
  - greift auf [Entity]Service.update[Entity]() zurück
- DELETE
  - greift auf [Entity]Service.delete[Entity]() zurück

Controller Klassen

```java
@GetMapping("/{id}")
public Objective getObjectById(@PathVariable Long id) {
    return objectiveService.getObjectiveById(id);
}
@PostMapping
public Objective createObjective(@RequestBody Objective objective) {
    return objectiveService.createObjective(objective);
}
@PutMapping("/{id}")
public Objective updateObjective(@PathVariable Long id, @RequestBody Objective objective) {
    return objectiveService.updateObjective(id, objective);
}
@DeleteMapping("/{id}")
public void deleteObjective(@PathVariable Long id) {
    objectiveService.deleteObjective(id);
}
```

Danke für Ihre Aufmerksamkeit