SERVICE-PRÄSENTATION

TEAM 1, SERVICE 7 (PERSONS)

AGENDA

- Service
- Product Backlog
- ER-Modell
  - Person
  - Person Controller
  - Validierung des Builders
- Lessons Learned
- Weiteres Vorgehen

TEAMMITGLIEDER

- Jonas Liehmann (DevLead)
- Tobias Kärst (PO)
- Niklas Schumann
- Justin Noske

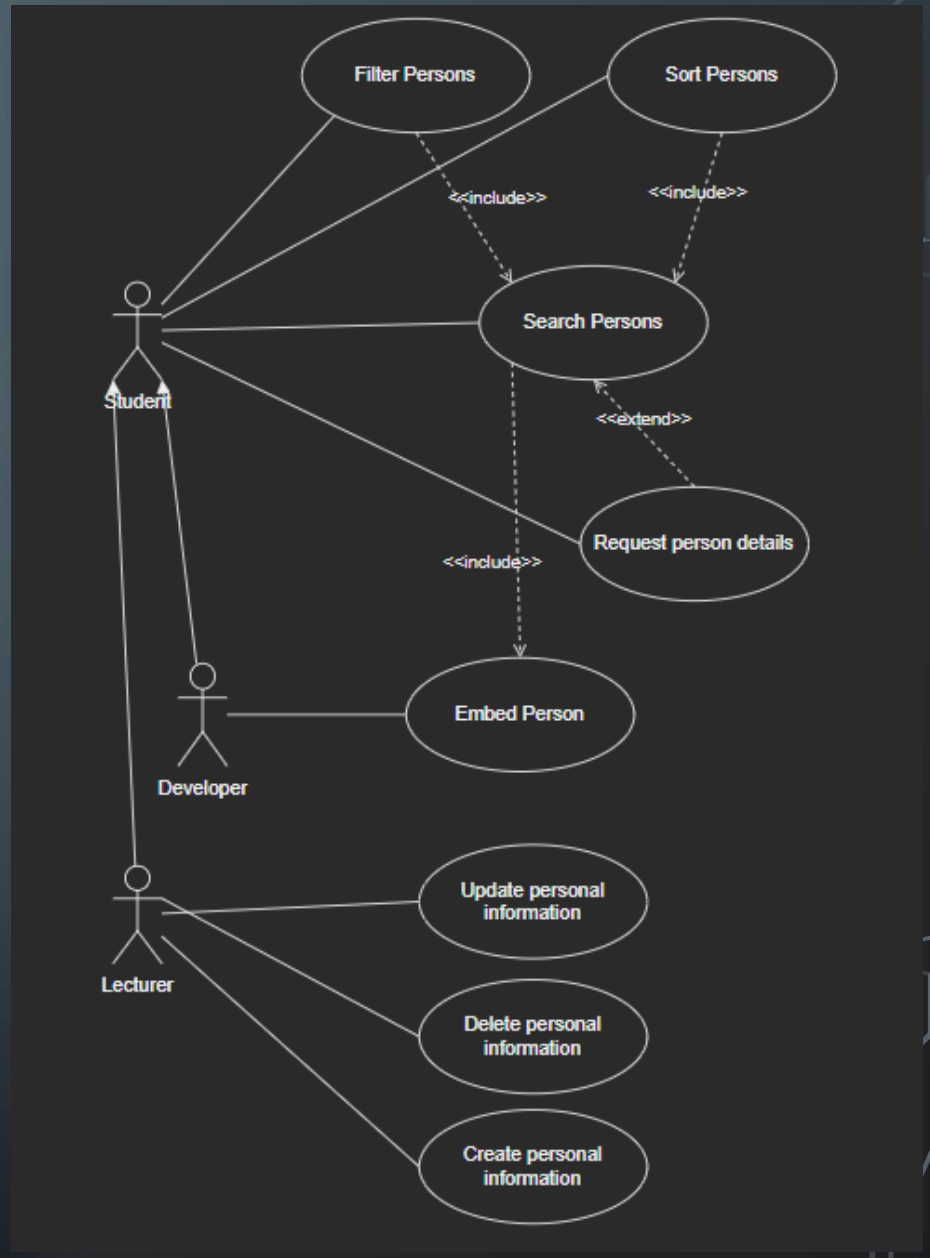
SERVICE

- Beinhaltet alle Informationen und Funktionen zur Kontaktaufnahme mit den Lehrenden an der FHE
- Ermöglicht eine intuitive Suche nach Personen nach bestimmten Suchkriterien
- Bietet eine Übersicht über die für Studierende relevanten Personen

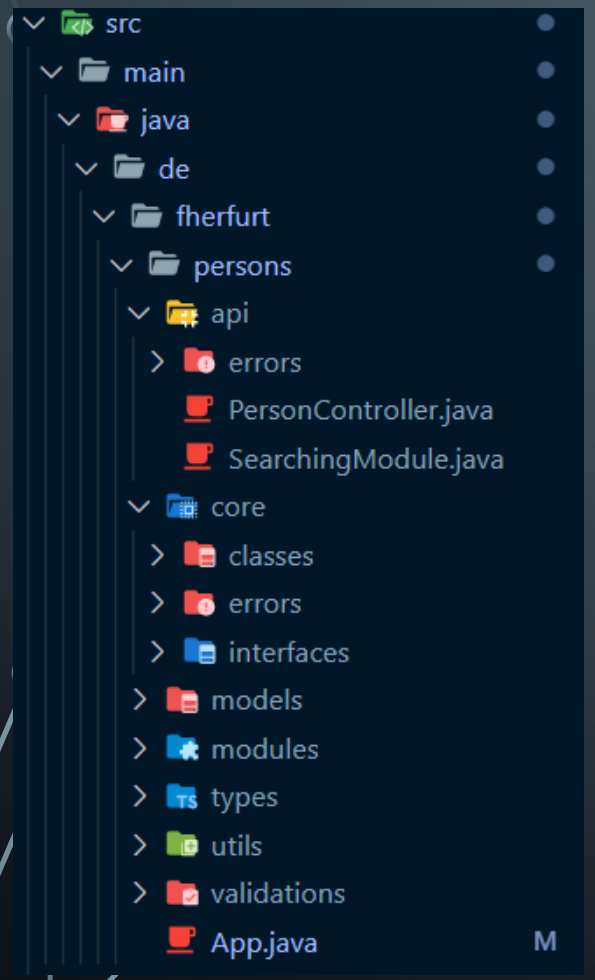
PRODUCT BACKLOG

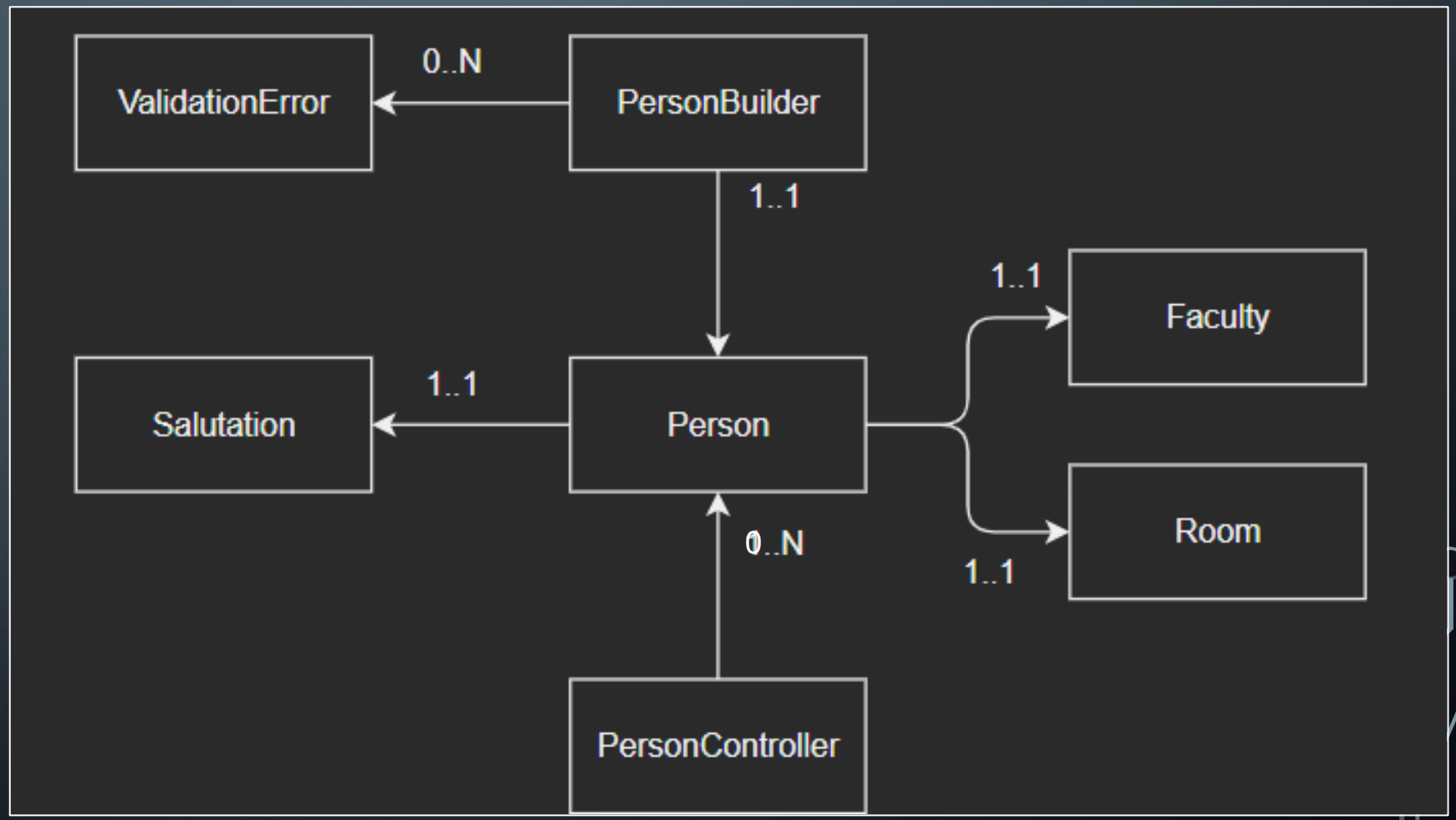
- Lehrende sollen in der Lage sein, ihre Daten im System zu erstellen
- Lehrende sollen in der Lage sein, ihre Daten zu verändern
- Lehrende sollen in der Lage sein, ihre Daten zu löschen

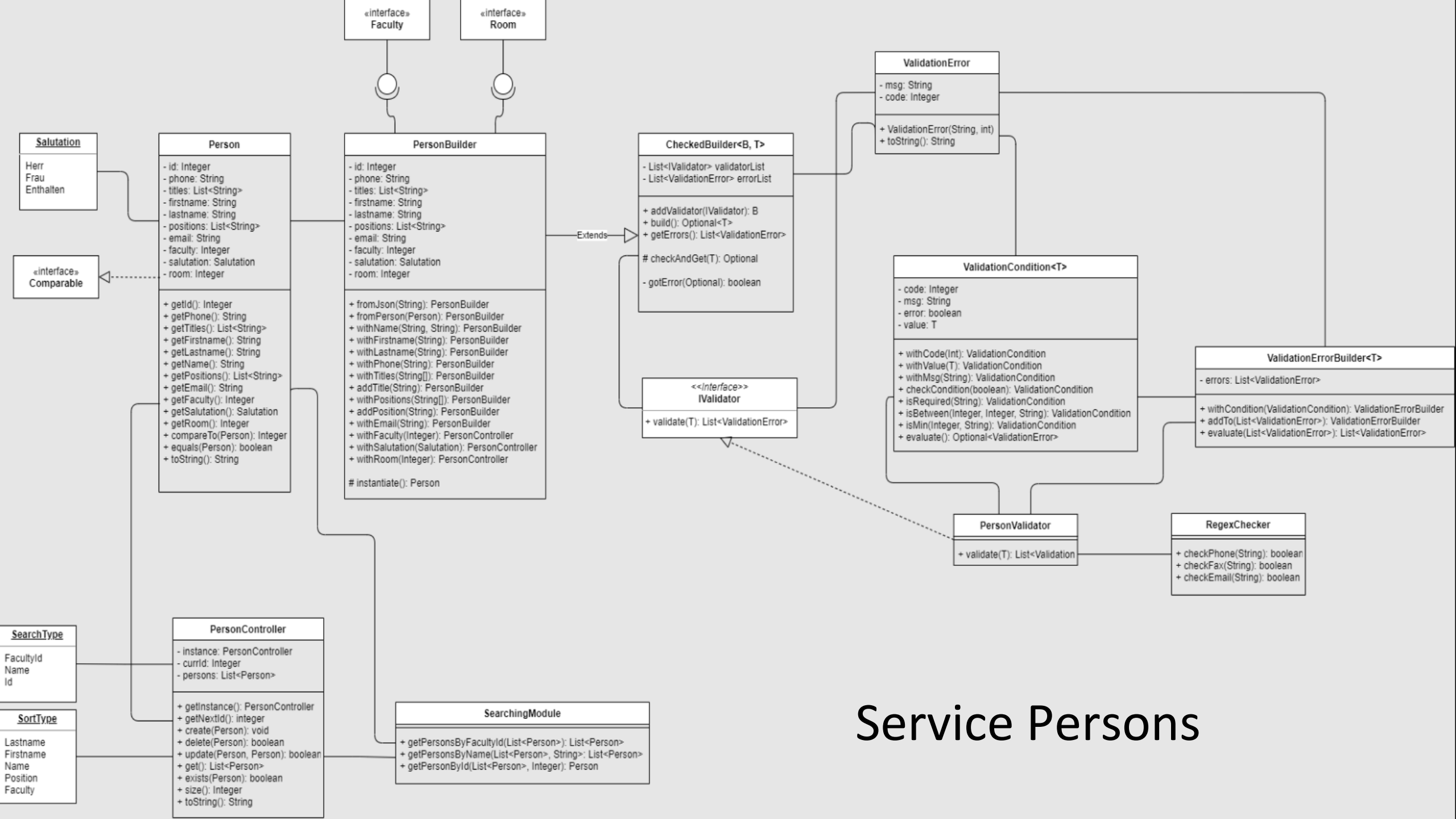
- Studierende sollen in der Lage sein, Lehrende nach Namen zu suchen
- Die Namenssuche, soll keinen direkten Stringvergleich durchführen, sondern nach Namen suchen, die der Eingabe ähnlich sehen
- Studierende sollen in der Lage sein, die Lehrenden nach der Fakultät filtern zu können
- Studierende können die Personenliste nach folgenden Kriterien sortieren: Vorname, Nachname, Name, Position, Fakultät
- Für die Studierenden soll ein Suchverlauf erstellt werden



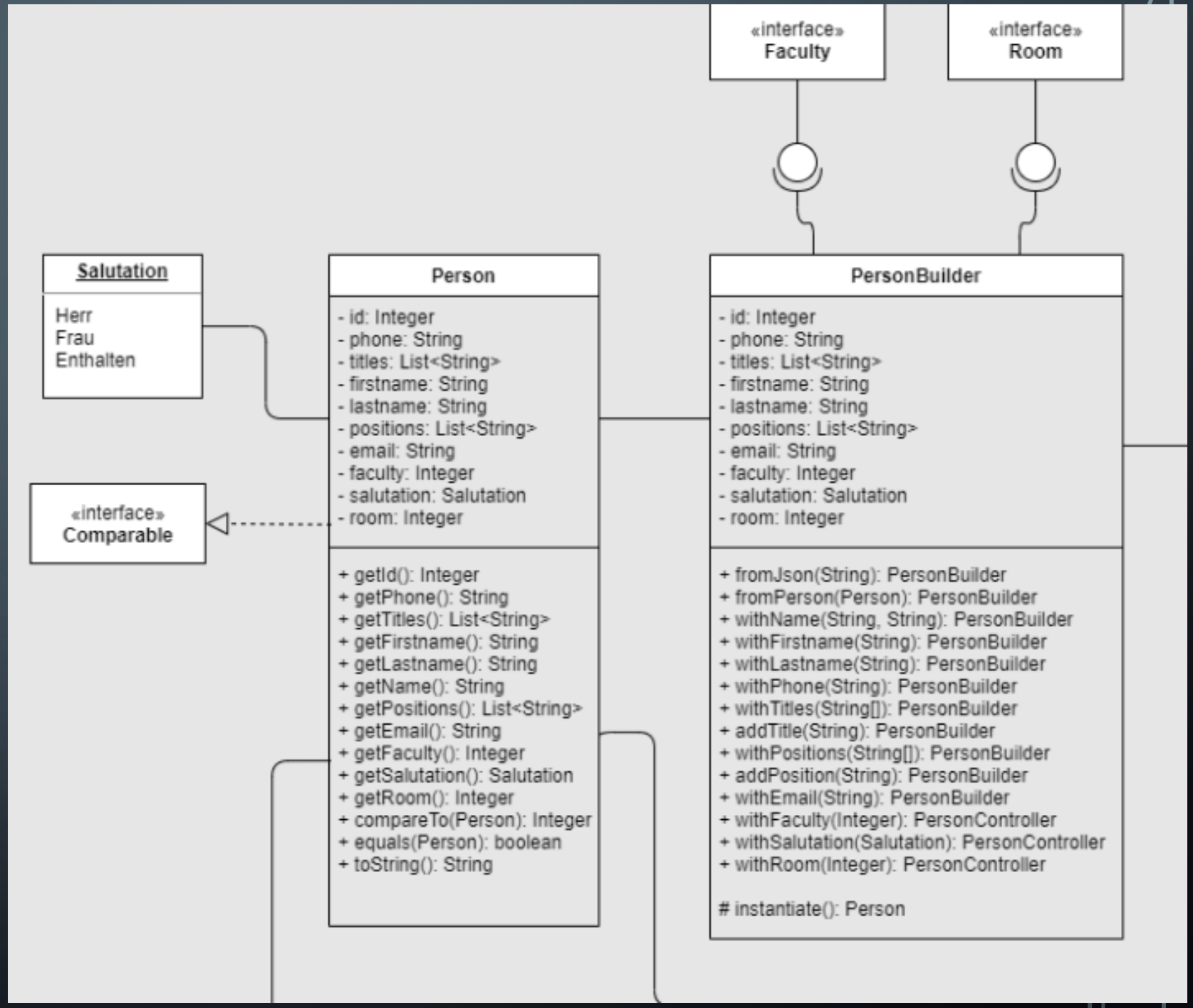
Übersicht Service Persons



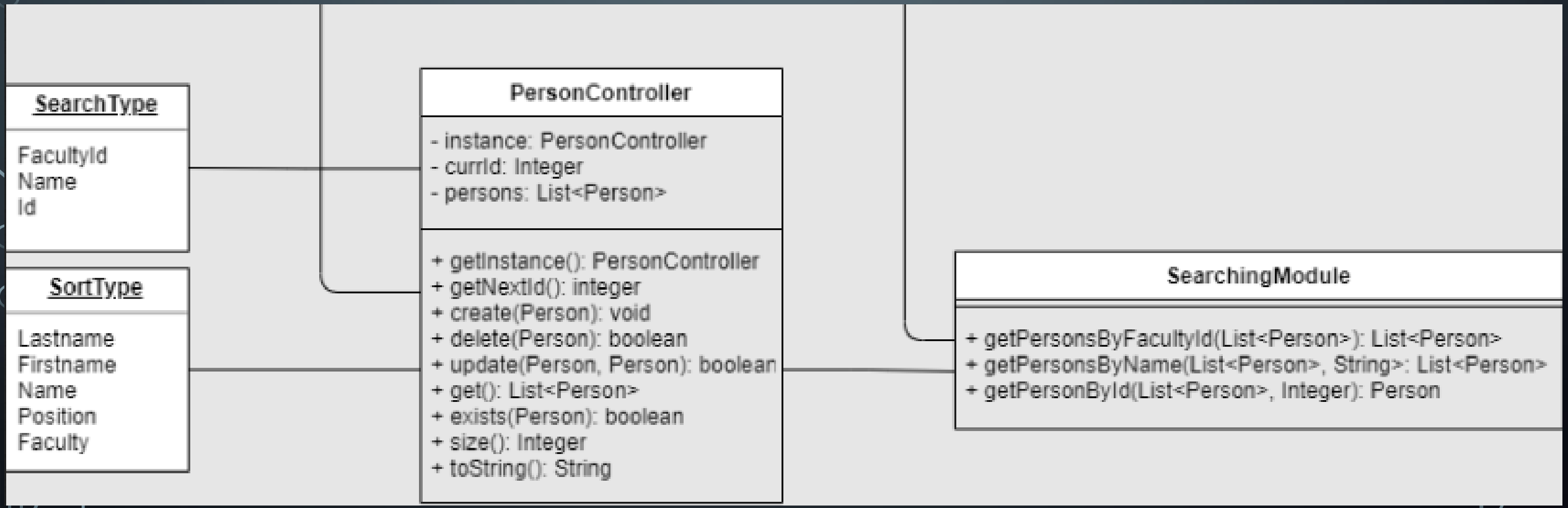



«interface»
Faculty
«interface»
Room
Salutation
Herr
Frau
Enthalten
Person
- id: Integer
- phone: String
- titles: List<String>
- firstname: String
- lastname: String
- positions: List<String>
- email: String
- faculty: Integer
- salutation: Salutation
- room: Integer
+ getId(): Integer
+ getPhone(): String
+ getTitles(): List<String>
+ getFirstname(): String
+ getLastname(): String
+ getName(): String
+ getPositions(): List<String>
+ getEmail(): String
+ getFaculty(): Integer
+ getSalutation(): Salutation
+ getRoom(): Integer
+ compareTo(Person): Integer
+ equals(Person): boolean
+ toString(): String
«interface»
Comparable
PersonBuilder
- id: Integer
- phone: String
- titles: List<String>
- firstname: String
- lastname: String
- positions: List<String>
- email: String
- faculty: Integer
- salutation: Salutation
- room: Integer
+ fromJson(String): PersonBuilder
+ fromPerson(Person): PersonBuilder
+ withName(String, String): PersonBuilder
+ withFirstname(String): PersonBuilder
+ withLastname(String): PersonBuilder
+ withPhone(String): PersonBuilder
+ withTitles(String[]): PersonBuilder
+ addTitle(String): PersonBuilder
+ withPositions(String[]): PersonBuilder
+ addPosition(String): PersonBuilder
+ withEmail(String): PersonBuilder
+ withFaculty(Integer): PersonController
+ withSalutation(Salutation): PersonController
+ withRoom(Integer): PersonController
# instantiate(): Person
Extends
CheckedBuilder<B, T>
- List<IValidator> validatorList
- List<ValidationError> errorList
+ addValidator(IValidator): B
+ build(): Optional<T>
+ getErrors(): List<ValidationError>
# checkAndGet(T): Optional
- gotError(Optional): boolean
ValidationError
- msg: String
- code: Integer
+ ValidationError(String, int)
+ toString(): String
<<Interface>>
IValidator
+ validate(T): List<ValidationError>
ValidationCondition<T>
- code: Integer
- msg: String
- error: boolean
- value: T
+ withCode(Int): ValidationCondition
+ withValue(T): ValidationCondition
+ withMsg(String): ValidationCondition
+ checkCondition(boolean): ValidationCondition
+ isRequired(String): ValidationCondition
+ isBetween(Integer, Integer, String): ValidationCondition
+ isMin(Integer, String): ValidationCondition
+ evaluate(): Optional<ValidationError>
ValidationErrorBuilder<T>
- errors: List<ValidationError>
+ withCondition(ValidationCondition): ValidationErrorBuilder
+ addTo(List<ValidationError>): ValidationErrorBuilder
+ evaluate(List<ValidationError>): List<ValidationError>
PersonValidator
+ validate(T): List<Validation
RegexChecker
+ checkPhone(String): boolean
+ checkFax(String): boolean
+ checkEmail(String): boolean
SearchType
FacultyId
Name
Id
SortType
Lastname
Firstname
Name
Position
Faculty
PersonController
- instance: PersonController
- currId: Integer
- persons: List<Person>
+ getInstance(): PersonController
+ getNextId(): integer
+ create(Person): void
+ delete(Person): boolean
+ update(Person, Person): boolean
+ get(): List<Person>
+ exists(Person): boolean
+ size(): Integer
+ toString(): String
SearchingModule
+ getPersonsByFacultyId(List<Person>): List<Person>
+ getPersonsByName(List<Person>, String>: List<Person>
+ getPersonById(List<Person>, Integer): Person
Service Persons

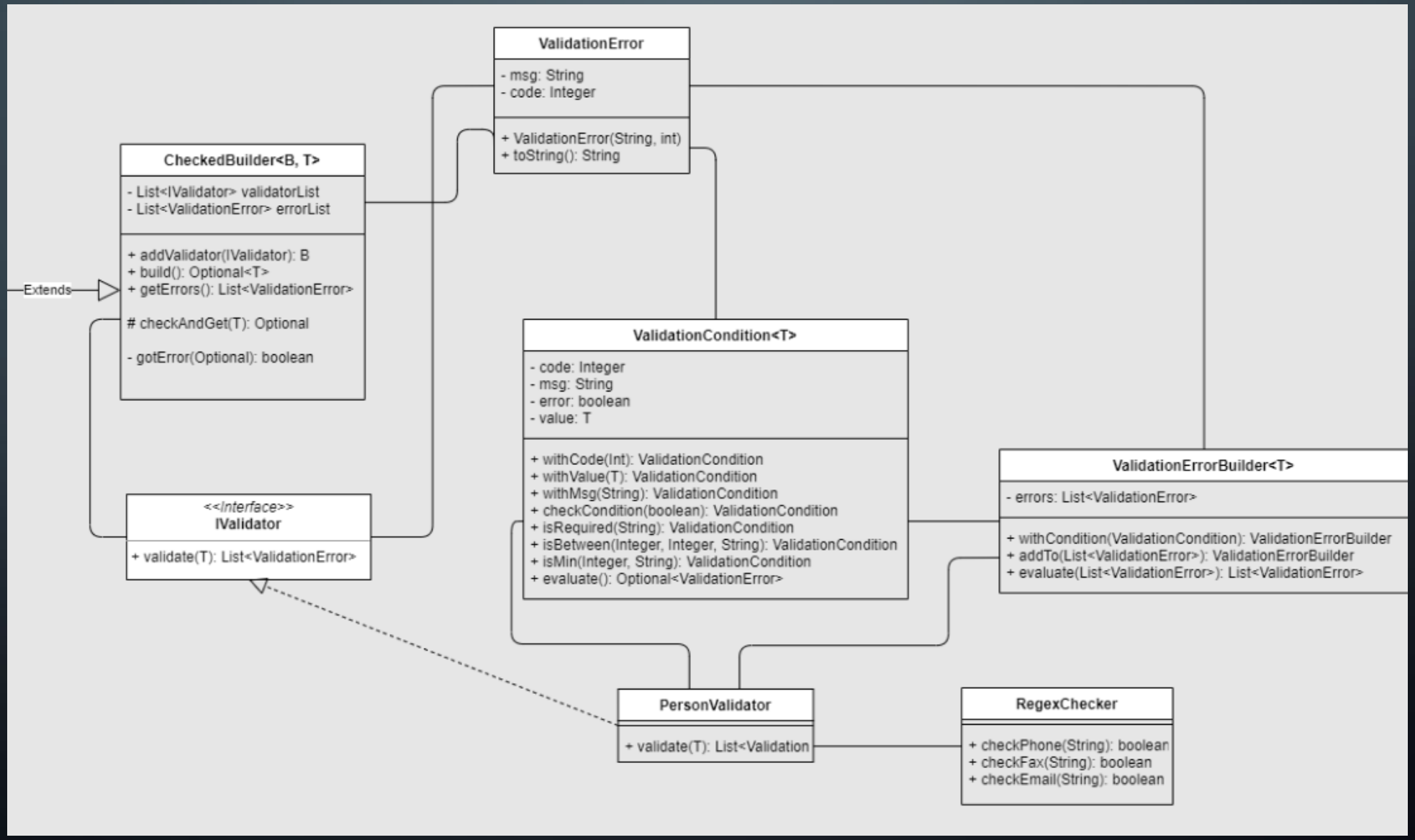
PERSON MODULE



PERSON CONTROLLER



VALIDIERUNG DES BUILDERS



```
String json = """{
        "firstname": "Max",
        "lastname": "Mustermann",
        "faculty": 3,
    }
""";

Optional<Person> person = builder
        .addValidator(new PersonFilledValidator())
        .fromJSON(json)
        .build();

if (person.isPresent()) {
    System.out.println(person.get());
}
```

```
Optional<Person> person = builder
    .addValidator(new PersonFilledValidator())
    .withName("Max", "Mustermann")
    .build();

if (person.isPresent()) {
    System.out.println(person.get());
}
```

```
new ValidationErrorBuilder<String>()
    .withCondition(new ValidationCondition<String>()
        .withValue(person.getFirstname())
        .isRequired("firstname"))
    .withCondition(new ValidationCondition<String>()
        .withValue(person.getFirstname()
        .isBetween(2, 30, "firstname"))
    .addTo(errors);
```

```
Optional<Person> person = builder
    .addValidator(new PersonFilledValidator())
    .fromPerson(oldPerson)
    .withName("Paul", "Mustermann")
    .build();
}
```

LESSONS LEARNED

- Abkapselung in Module besonders für Teamaufteilung und Tests sinnvoll
- Durch Abstraktion der API kann man sich voll auf die Logik konzentrieren, sodass am Ende eine viel saubere Architektur entsteht

WEITERES VORGEHEN

- Restliche Unit-Tests fertig implementieren
- Damit verbundene Fehlersuche
- Projektdokumentation fertig schreiben

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT