JAVA I PROJEKT SERVICE NEWS

Benjamin Ehnes (Dev Lead)

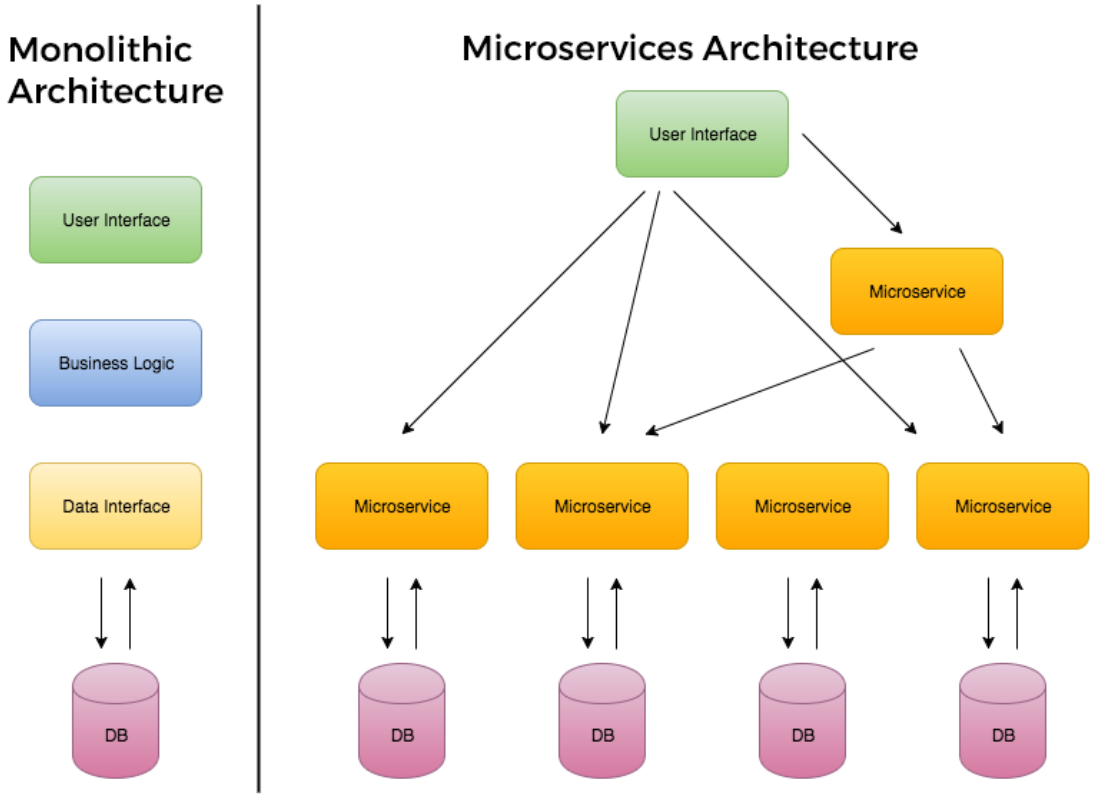
Antonia Geschke (PO)

Lucian Gerasch – Celina Ludwigs – Lisa Sluka

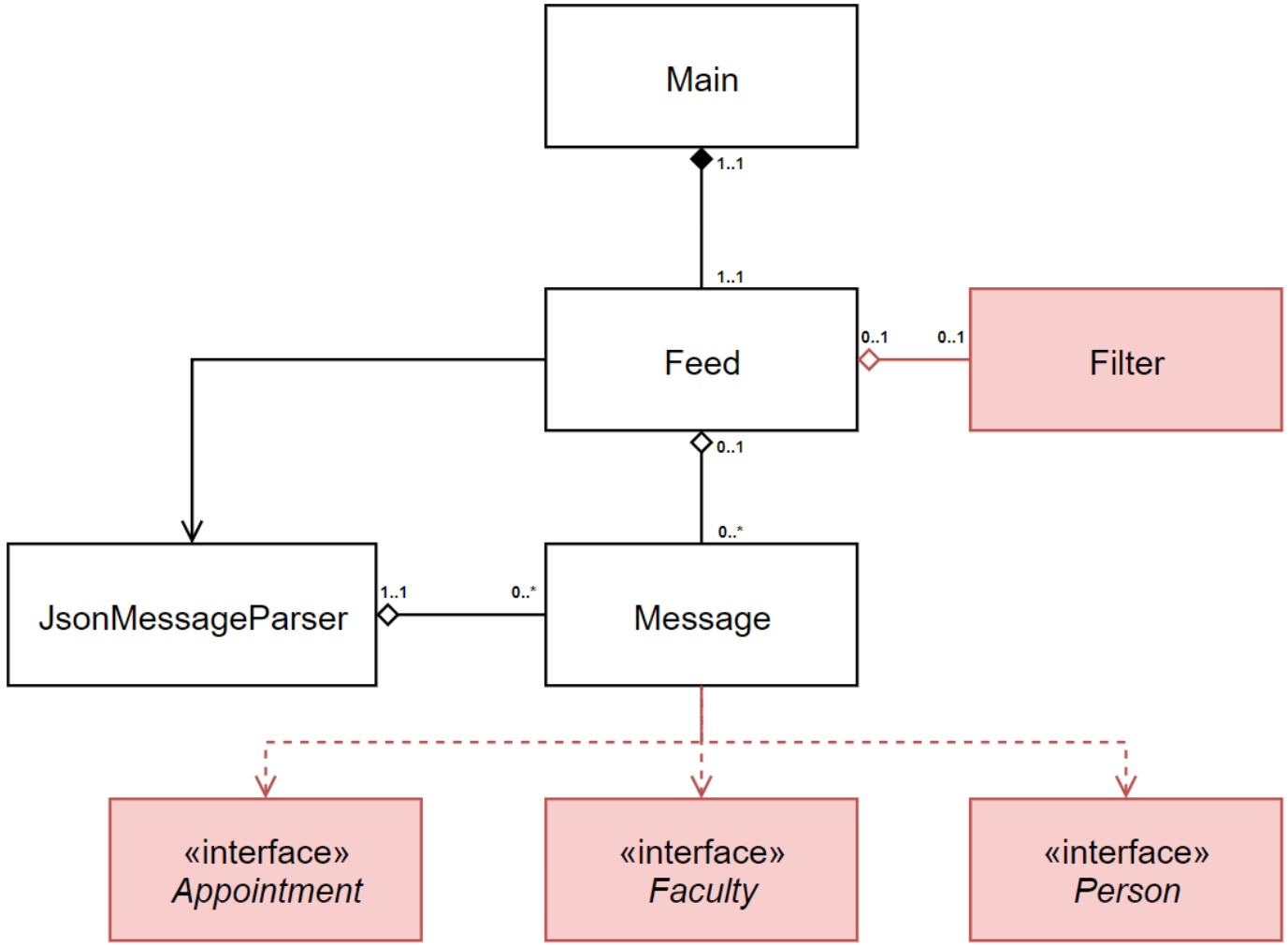
Agenda

1. Verstehen der Architektur
2. Altes Klassendiagramm
3. Verbessertes Klassendiagramm
4. Von uns umzusetzen
5. Klassen: Message, Image, UserPreferences
6. Filter
7. Vom Feed zum Service
8. Ziele bis zur Abgabe
9. Probleme, Anmerkungen & Lessons learned

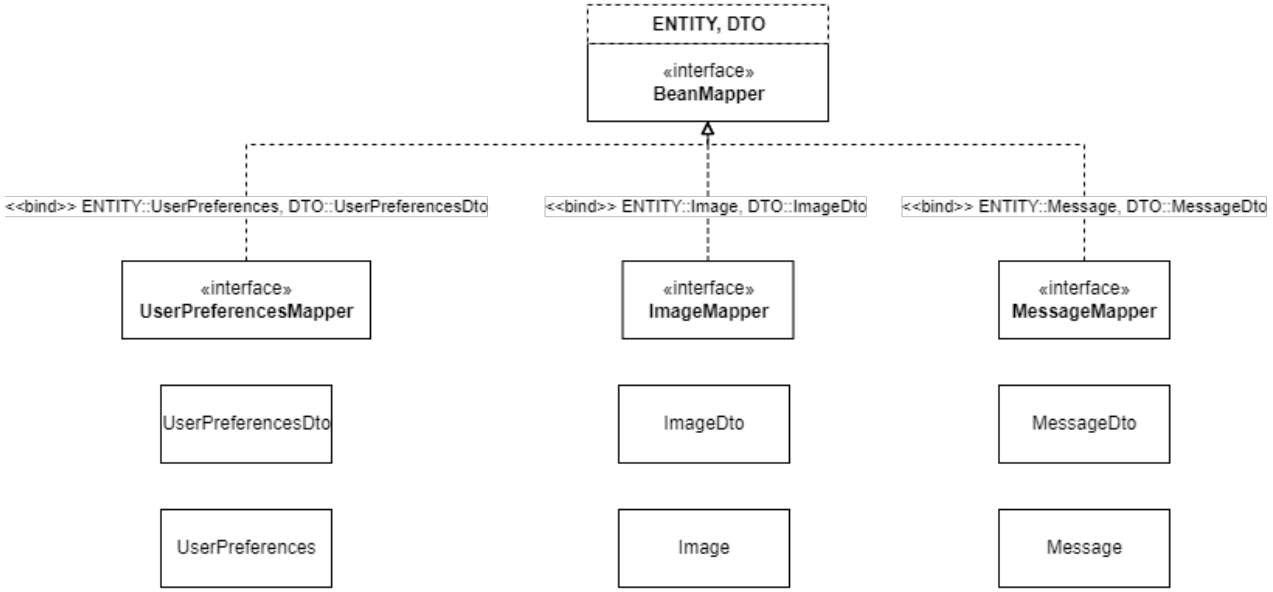
Verstehen der Architektur

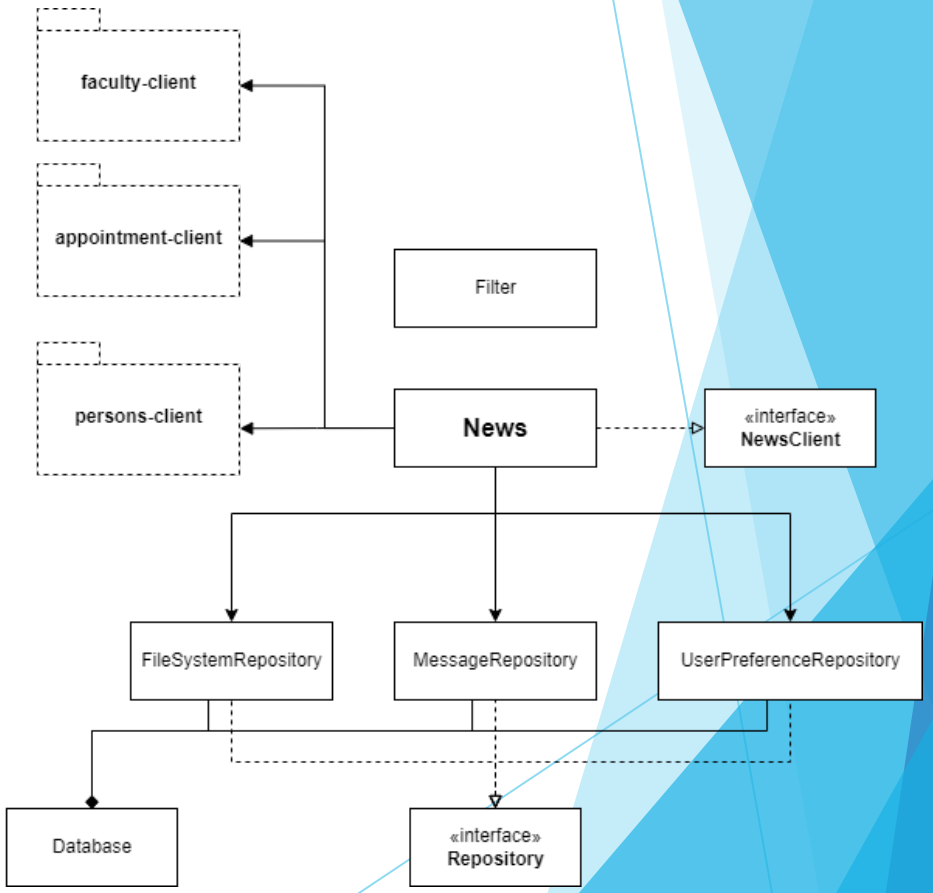


Altes Klassendiagramm



Verbessertes Klassendiagramm





Von uns umzusetzen

- Datenstrukturen (Nachricht, Bilder, Userpreferences)
- Logik im Service
- Anlehnung des Repositories an Beispielprojekt "Mensa"
- Übernahme der Datenbank (Abstraktion) von Herrn Rhöse
- statt des ModelMappers für BeanMapper: Verwendung von Herr Rhöses Lösung

Klassen - Message – Image - UserPreferences

Message

- DataHolder
- typische News-Attribute wie: Autor, Titel, Fakultät,...

Image

- simple Klasse
- beinhaltet das Bild in Bytes
- String zum Pfad

```java
public class Image extends BaseBusinessEntity {
    private String path;

    @Builder (setterPrefix = "with")

    public Image(int id, String path) {
        super(id);
        this.path = path;
    }
}
```

UserPreferences

- soll Daten beinhalten, die zu groß wären um sie bei jeder Anfrage zu übermitteln z.B. ignorierte Autoren
- bei Anfragen sollen diese mit in das Message-Predicate übernommen werden → Erweiterung der Filterbedingung

Filter

- Abänderung von eigentlicher Idee
- beinhaltet keine weitere "Logik"
- reine Utility
- baut Predicates
- API noch nicht genau ausdefiniert

```
public class Filter {
    public static Predicate<MessageDto> containingDateBefore(LocalDateTime localDateTime) {...}

    public static Predicate<MessageDto> containingAuthor(Integer... authors) {...}

    public static Predicate<MessageDto> containingTopic(String... topics) {...}
```

```java
public class News implements NewsClient {
    private final MessageRepository messageRepository = MessageRepository.of();
    private final FilesystemRepository filesystemRepository = FilesystemRepository.of();
    private final UserPreferenceRepository userPreferenceRepository = UserPreferenceRepository.of();

    @Override
    public int save(MessageDto messageDto) {…
    }

    @Override
    public Optional<MessageDto> findBy(int id) {…
    }

    @Override
    public void delete(int id) {}
    }

    @Override
    public List<ImageDto> loadImagesBy(int messageId) {}

    @Override
    public List<MessageDto> findBy(Predicate<MessageDto> predicate) {}

    @Override
    public void saveAuthorToUserPreferences(int author, int userId) {}

    @Override
    public void deleteAuthorFromUserPreferences(int author, int userId) {}
}
```

Vom Feed zum Service

- große Änderungen
- kein Feed mehr an sich
- soll Voreinstellungen, Messages und Bilder verwalten
- primäre Funktionalität: Nachrichten erstellen und abfragen
- implementiert außerdem den Client

Was bis zur Abgabe noch implementiert werden soll

- Fertigstellung der Repositories
- Ausbau des News-Service
- Erweiterung der Unit-Tests
  - nach der Umstrukturierung sind die meisten Tests entfallen
- letzter Feinschliff für die anderen Services

Probleme, Anmerkungen & Lessons learned

- Fehleinschätzung bezüglich Projektumfang
- Hohe Komplexität

- Zeitmanagement
- Kommunikation mit anderen Services
- Verstehen und Umsetzen von Microservices

Vielen Dank!

Noch Fragen?