Ann-Kathrin Meyerhof
Katja Schneider
Maik Bartels
Monique Schopper

Aufgabe 1 Event Storming

- **Bürger registriert sich**: Ein neuer Bürger wird in das System aufgenommen.
- **Beschwerde erstellen**: Ein Bürger erstellt eine Beschwerde.
- **Beschwerde zurückziehen**: Ein Bürger zieht die Beschwerde wieder zurück.
- **Beschwerde zuweisen**: Die Beschwerde wird einem Mitarbeiter zugewiesen.
- **Status der Beschwerde**: Ein Mitarbeiter ändert den Status der Beschwerde.
- **Beschwerde bearbeiten**: Ein Mitarbeiter bearbeitet eine Beschwerde.
- **Benachrichtigung zu Beschwerde**: Ein Mitarbeiter schickt eine Benachrichtigung zu einer Beschwerde an den Bürger.
- **Beschwerde schließen**: Ein Mitarbeiter schließt eine fertig bearbeitete Beschwerde.
- **Beschwerde archivieren**: Die Beschwerde wird im System archiviert.
- **Bürger löscht Account**: Der Bürger wird samt Daten aus dem System entfernt.
- **Bürger aktualisiert Daten**: Die Systemdaten des Bürgers werden aktualisiert.
- **Mitarbeiter registriert sich**: Ein neuer Mitarbeiter wird in das System aufgenommen.
- **Mitarbeiter aktualisiert seine Daten**: Die Systemdaten des Mitarbeiters werden aktualisiert.
- **Mitarbeiter löschen**: Ein Mitarbeiter wird aus dem System entfernt.

Ann-Kathrin Meyerhof
Katja Schneider
Maik Bartels
Monique Schopper

Aufgabe 2 Domänenmodell

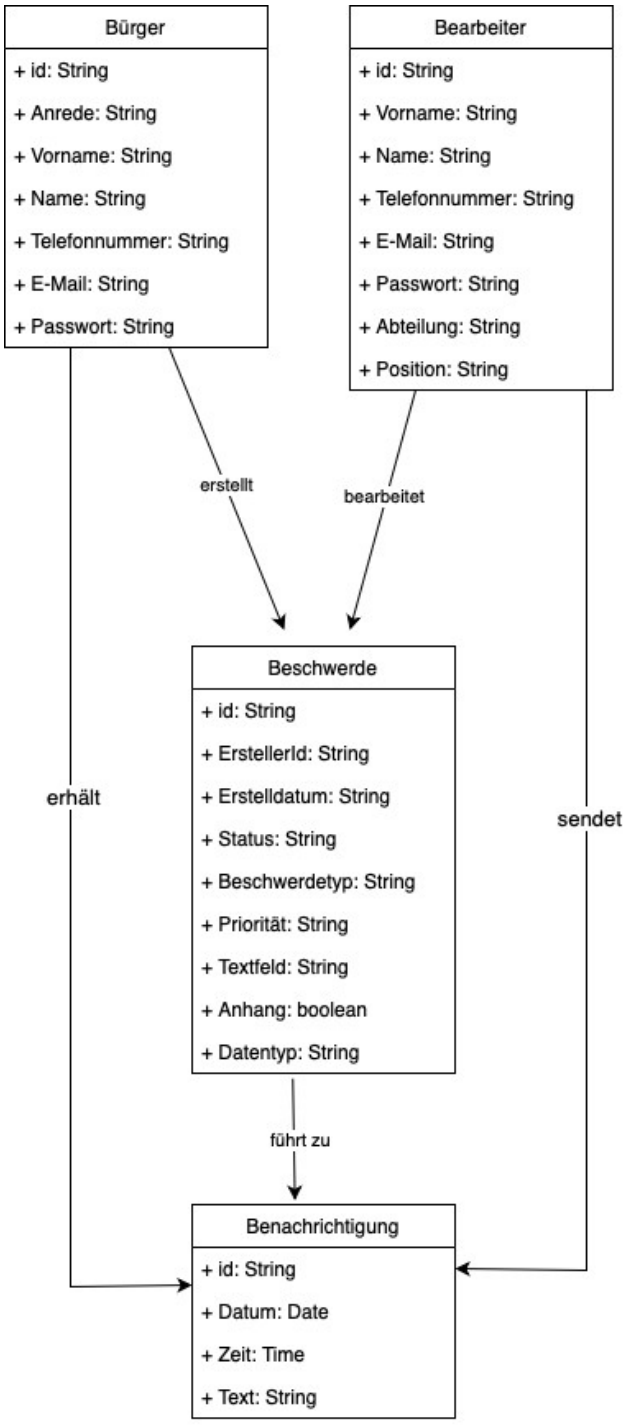

Abbildung 1: Domänenmodell

Ann-Kathrin Meyerhof
Katja Schneider
Maik Bartels
Monique Schopper

Aufgabe 3 Bounded Contexts

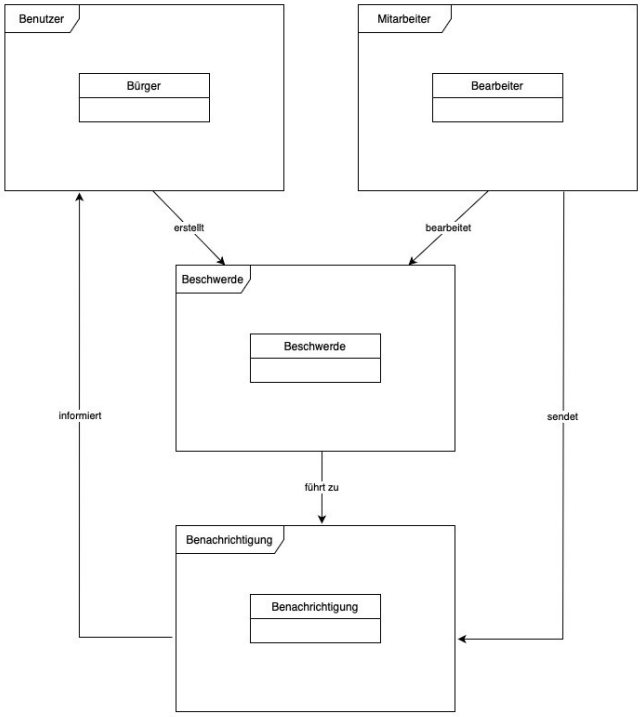

Abbildung 2: Bounded Context

Aufgabe 4 Entitäten und Aggregate definieren

Bürgermanagement

- **Entität**: Bürger
- **Aggregate**: Bürger (alle Daten des Bürgers, z. B. Name, E-Mail, Telefonnummer)

Beschwerdemanagement

- **Entität**: Beschwerde
- **Aggregate**: Beschwerde (alle Daten einer Beschwerde, z. B. Erstelldatum, Status, Anhang)

Mitarbeitermanagement

- **Entität**: Mitarbeiter
- **Aggregate**: Mitarbeiter (alle Daten eines Mitarbeiters, z. B. Name, Abteilung, E-Mail)

Benachrichtigungsmanagement

- **Entität**: Benachrichtigung
- **Aggregate**: Benachrichtigung (alle Daten einer Benachrichtigung, z. B. Datum, Zeit, Text)

Ann-Kathrin Meyerhof
Katja Schneider
Maik Bartels
Monique Schopper

Aufgabe 5 Domain Services und Repositories

Domain Services

BürgerService

- **Beschreibung:** verwaltet die Interaktionen mit den Bürgern, einschließlich der Erstellung eines neuen Bürgerkontos und der Verwaltung der Beschwerden des Bürgers.

MitarbeiterService

- **Beschreibung:** ist verantwortlich für die Verwaltung der Mitarbeiter und deren Zuweisung zu der Beschwerde.

BeschwerdeService

- **Beschreibung:** ist verantwortlich für die Verwaltung und Bearbeitung der Anliegen, einschließlich der Zuordnung von Mitarbeitern, dem Ändern des Status und dem Schließen von Beschwerden.

BenachrichtigungsService

- **Beschreibung:** verwaltet die Benachrichtigungen an Bürger, wenn sich der Status einer Beschwerde ändert.

Repositories

BürgerRepository

- **Beschreibung:** Verwaltet die Persistenz von Bürgern.
- **Methode:** findBuergerById(String id)

MitarbeiterRepository

- **Beschreibung:** Verwaltet die Persistenz von Mitarbeitern.
- **Methode:** findMitarbeiterById(String id)

BeschwerdeRepository

- **Beschreibung:** Verwaltet die Persistenz von Beschwerden.
- **Methode:** findBeschwerdeById(String id)

BenachrichtigungRepository

- **Beschreibung:** Verwaltet die Persistenz von Benachrichtigungen.
- **Methode:** findBenachrichtigungById(String id)

Aufgabe 6 Implementierungsstrategie

Die folgenden Implementierungsschritte möchten wir vornehmen:

a) **Entitäten und Aggregate:** Für jede zentrale Entität (Bürger, Mitarbeiter, Beschwerde, Benachrichtigung) wird eine eigene Java-Klasse erstellt. Diese Klassen repräsentieren die jeweiligen Aggregate und beinhalten alle relevanten Attribute und Methoden, die notwendig sind, um die Geschäftslogik zu kapseln und Beziehungen abzubilden.

b) **Domain Services:** Für jeden Bereich wird ein spezifischer Service (z. B. BürgerService, MitarbeiterService) implementiert. Die Service-Klassen übernehmen die Geschäftslogik und arbeiten eng mit den Repositories zusammen, um Operationen wie das Erstellen, Verwalten und Zuweisen von Beschwerden zu realisieren. Beispielhafte Services umfassen den BürgerService (für Bürgerinteraktionen und Kontoverwaltung), den BeschwerdeService (für die Bearbeitung und Statusänderungen von Beschwerden) sowie den BenachrichtigungsService (zum Senden von Statusbenachrichtigungen an Bürger).

c) **Repositories:** Für die Persistenz werden Repositories (z. B. BürgerRepository, MitarbeiterRepository, BeschwerdeRepository) als Schnittstellen implementiert. Sie definieren Methoden zum Abrufen der Entitäten. Eine geeignete Persistenzlösung erleichtert dabei die Datenbankintegration und gewährleistet eine strukturierte Verwaltung aller Daten.

Beispiele

```java
public class BuergerService {

    public Buerger findBuergerById(String buergerId) {
        return buergerRepository.findById(buergerId)
            .orElseThrow(() -> new RuntimeException(
               BUERGER_EXISTIERT_NICHT + buergerId));
    }
}
```

```java
public class BeschwerdeService {

    public Beschwerde findBeschwerdeById(String beschwerdeId) {
        return beschwerdeRepository.findById(beschwerdeId)
            .orElseThrow(() -> new RuntimeException(
               BESCHWERDE_EXISTIERT_NICHT + beschwerdeId));
    }

    public Beschwerde updateStatusBeschwerde(String beschwerdeId,
       Beschwerde beschwerde) {
        Beschwerde aktuelleBeschwerde = findBeschwerdeById(
           beschwerdeId);

        if (beschwerde.getStatus() != null) {
            // ...
        } else if (/* Bedingung */) {
            // ...
        }

        beschwerdeRepository.save(aktuelleBeschwerde);
    }
}
```