Fragen Backend

* **ID UUID oder Integer**

Bbesser der Datenbank überlassen @GeneratedValue (strategy = GenerationType.UUID oder IDENTITY (autoincrement) oder SEQUENCE - (kann die Sequenz selbst definieren) oder TABLE (Zuordnung über eine Tabelle, schlecht scalierbar)

<https://www.baeldung.com/hibernate-identifiers>

Hibernate / JPA bekommt es hin, UUID auf String oder UUID umzuwandeln.

* **primitive Datentypen oder Wapper Klasse**?

„einfaches“ Rechnen natürlich immer mit primitiven Datentypen

Persistente Speicherung:

Wrapper-Klassen brauch mehr Speicher

Brauchen Wrapper Klassen aber bei Liste. Parsing etc.

ACHTUNG Wrapper haben der Vorteil, dass sie **Nullable** sind

* **LocalDateTime or Instant** for Created / updated

LocalDateTime hat den Nachteil, dass es keine Zeitzonen speichert

**Instant speichert alles in UTC** (für Internationalisierung) – wenn man es in lokaler Zeit anzeigen will, muss man es im Programmcode umrechnen.

ZonedDateTime zoned = lokaleZeit.atZone(ZoneId.systemDefault());

return zoned.toInstant();

<https://www.baeldung.com/hibernate-creationtimestamp-updatetimestamp>

<https://www.baeldung.com/java-instant-jackson-format-object-mapper>

[*https://www.baeldung.com/hibernate-one-to-many*](https://www.baeldung.com/hibernate-one-to-many)

* Wie gehen wir mit Gastorders um (Dummyuser? – müssen wir uns noch anschauen)
* **Auflösung von (many to many) Beziehungen.**

@Jointable @Joincolumn

<https://www.baeldung.com/jpa-many-to-many>

Oft sind eigene Zwischentabellen sehr sinnvoll – mehr Kontrolle 🡪 man muss sie eventuell mit der Businesslogik befüllen.

* **Arraylist oder List bei Entitäten** -> List ich will es nicht der Datenbank vorgeben
* **Optional:**

ifPresent() 🡪 true or flase

.*isEmpty()* 🡪 true or false

.orElse

return userRepository.findById(id).orElseThrow( () -> new ObjectNotFoundException("Kein Benutzer mit der ID " + id + " vorhanden"));

return userRepository.findById(id).ifPresentOrElse((String s) -> Anweisung; () -> Anweisung);

Lambda Funktion 🡪 anonyme Funktionen - Code der irgendwann ausgeführt werden kann, Callback;

Controller advice , Controller acception Handling

* Muss diese **Validation Dependency** eingebaut werden:

Lt. Diesen Artikel schon:

<https://www.baeldung.com/spring-boot-bean-validation>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency

Eigene Validierung schrieben

<https://www.baeldung.com/java-bean-validation-constraint-composition>

------- bis hierher wurden die Fragen beantwortet

* Festlegung (falls es ok ist) Warenkorb 🡪 wird nur im Frontend behandelt

**Allergene 🡪 Enum** oder Klasse mit den Werten in der Datenbank

Was ist bei dem **Enums besser** – Direkt im Feld die Bezeichnung dazu oder über eine Map (mich nervt der Konstruktor bei dieser Variante:

public enum Country {

AT("Österreich"),

DE("Deutschland"),

CH("Schweiz"),

FR("Frankreich");

private final String displayName;

Country(String displayName) {

this.displayName = displayName;

}

public String getDisplayName() {

return displayName;

}

}

* Ist es schlecht Programmierstil, Variablen als Enums zu definieren, die Zugriffe aber mittels Strings (oder Integer) zu machen
* Macht es wirklich Sinn für alle Beziehungen die Elemente in beiden Tabellen zu kennzeichnen / einzufügen (zB updatedby, Produkte Richtung Hauptkategorien;
* Wie bildet man Hauptkategorien / Unterkategorien am besten ab? 🡪 ev. Tabelle einfach Tabelle in der Datenbank
* Deutsch Englisch (z.B bei den ENUMS) -🡪 ich will ja gleich den Wert der ENUMS verwenden
* Hierarchische Kategorien oder Tags?
* Ist das gefährlich? cascade = CascadeType.*ALL/PERSISTENT*, orphanRemoval = true (ZB geht Order dann über OderItem zu den Produkten und dann zu den Allergenen)
* Lt. ChatGpt sind Listen zu initialisieren (Nachteil ich muss den Listtyp vorgeben), weil es sonst beim Laden durch JPA aus der Datenbank zu NullPointer Exceptions kommen kann (es wird kein Konstruktor verwendet)
* Verhindern von mehrfachspeichern von Orders durch drücken eines Buttons
* OrderItem wirklich bei Produkten?
* Created /Upated mit oder / Ohne Schema hidden

@ManyToOne  
@Schema(hidden = true)  
private User createdBy;  
@ManyToOne  
@Schema(hidden = true)  
private User lastUpdatedBy;

Bei User ist es ohne , sonst mit –Siehe Swagger

* Generell @JasonIgnore, @@Schema(hidden = true)

Speicherung Klassen - OderItem Tabelle / Objekte – wie funktioniert da genau der Zusammenhang ID in der Datenbank, Objekt in Java – wie werden die Daten im Jason übergeben?

* Naming-Convention für Service?? Methoden