

Online-Juwelierladen

Meilenstein 1: Anforderungsanalyse & Konzeptioneller Entwurf

Dieses Diagramm stellt einen Online-Juwelierladen dar, in dem jeder Artikel nach Kundenwunsch handgefertigt wird. Jeder Kunde hat außerdem die Möglichkeit, den Designer vorab per E-Mail zu kontaktieren, um zu prüfen, ob der Designer die Anforderungen des Kunden erfüllen kann. Kunden, die sich per E-Mail auf der Website registrieren, können zwischen Ohrringen oder Halsketten wählen. Diese Produkte werden von Designern mit Hilfe ihrer Assistenten erstellt. Mehrere Designer können gemeinsam an einem Auftrag arbeiten, aber jeder Designer benötigt seinen eigenen Assistenten. Alle Bestellungen müssen persönlich in der Filiale abgeholt werden. Jede Bestellung wird mit einem Abholdatum und einer Rechnung versehen.

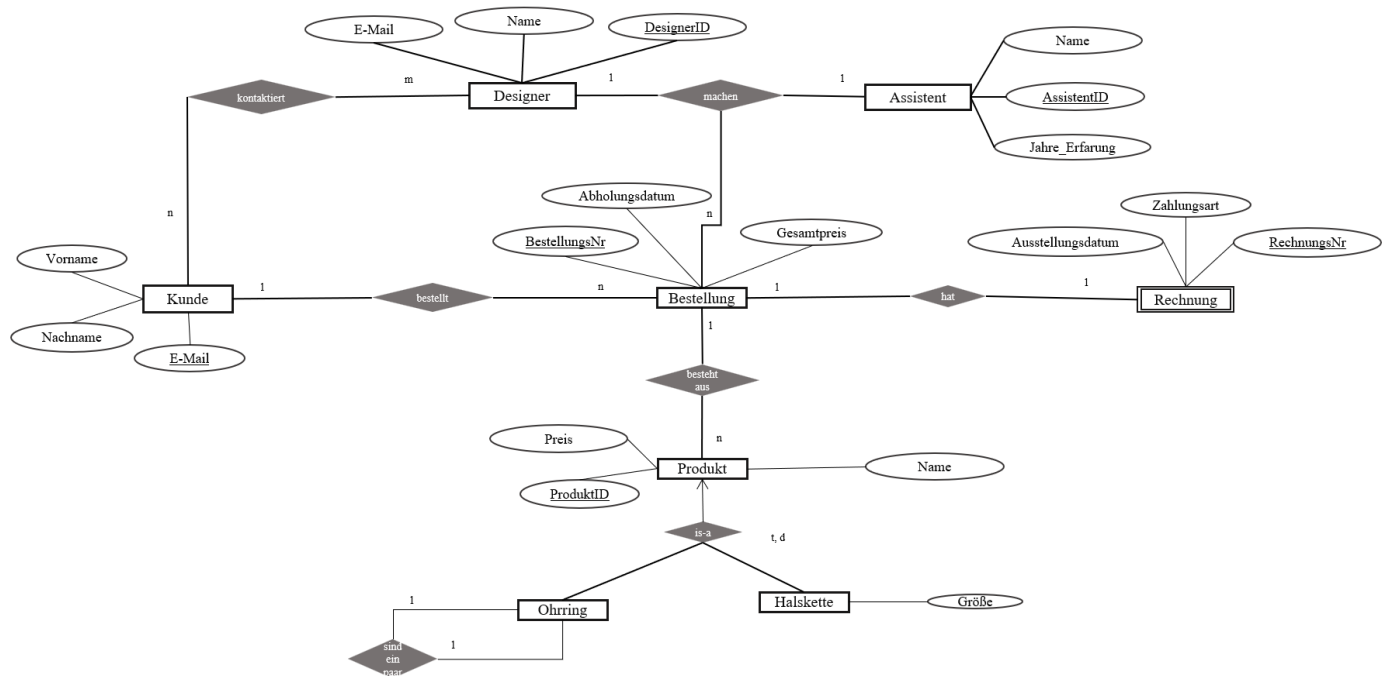


Abbildung 1: Entity-Relationship-Diagramm

Meilenstein 2: Logischer Entwurf

Kunde (E-Mail, Vorname, Nachname)

PK: E-Mail

FK: /

Schlüsselkandidaten: E-Mail

Bestellung (BestellungsNummer, Anholungsdatum, Gesamtpreis, Kunde_E-Mail)

PK: BestellungsNummer

FK: Kunde_E-Mail ◊ E-Mail.Kunde

Schlüsselkandidaten: BestellungsNummer

Ohrring (ProduktID, Preis, Name, *BestellungsNummer*)

PK: ProduktID

FK: BestellungsNummer ◊ BestellungsNummer.Bestellung

Schlüsselkandidaten: ProduktID

Halskette (ProduktID, Preis, Name, Groese, *BestellungsNummer*)

PK: ProduktID

FK: BestellungsNummer ◊ BestellungsNummer.Bestellung

Schlüsselkandidaten: ProduktID

Ohrring_Paar (Ohrring1_ProduktID, Ohrring2_ProduktID)

PK: Ohrring1_ProduktID, Ohrring2_ProduktID

FK: Ohrring1_ProduktID ◊ ProduktID.Ohrring, Ohrring2_ProduktID ◊ ProduktID.Ohrring

Schlüsselkandidaten: Ohrring1_ProduktID, Ohrring2_ProduktID

Designer (DesignerID, E-Mail, Name)

PK: DesignerID

FK: /

Schlüsselkandidaten: DesignerID, E-Mail

Assistent (AssistentID, Name, Jahre_Erfahrung)

PK: AssistentID

FK: /

Schlüsselkandidaten: AssistentID

KreierenBestellung (BestellungsNummer, DesignerID, AssistentID)

PK: BestellungsNummer, DesignerID

FK: BestellungsNummer ◊ BestellungsNummer.Bestellung, DesignerID ◊ DesignerID.Designer,
AssistentID ◊ AssistentID.Assistent

Schlüsselkandidaten: BestellungsNummer, DesignerID, AssistentID

Rechnung (RechnungsNummer, BestellungsNummer, Ausstellungsdatum, Zahlungsart)

PK: RechnungsNummer, BestellungsNummer

FK: BestellungsNummer ◊ BestellungsNummer.Bestellung

Schlüsselkandidaten: BestellungsNummer, Rechnungsnummer

HabenKontakt (DesignerID, Kunde-E-Mail)

PK: DesignerID, Kunde_E-Mail

FK: DesignerID ◊ DesignerID.Designer, Kunde_E-Mail ◊ E-Mail.Kunde

Schlüsselkandidaten: DesignerID, Kunde_E-Mail

Meilenstein 4: Beginn Implementierung

4.1 Java

Bei Data Generator(Java) habe ich für jedes insert, Arrays mit passenden Values gemacht und mit Hilfe von `Random()` mehrere Kombinationen gemacht. Die Tabelle Kunde hat 2000, Ohrring 2250 und Halskette 2300 Einträge, deshalb dauern diese länger. Das ausführen von `Tester.java` dauert nicht länger als eine Minute; im Terminal, mit Hilfe von `System.out.println()`, ist man immer informiert welche Inserts laufen und/oder fertig sind.

Mit diesem Teil des Projekts hatte ich nicht viele Probleme, da ich das Tutorium gefolgt habe und auch nur 2 Java Klassen habe, damit alles einfacher für mich ist. Die Insert-Values Arrays im `Tester.java` machen den gesamten Code weniger lesbar.

Jedes insert hat auch einen Count um zu überprüfen ob dieser erfolgreich war.

```
void insertIntoHalskette(String name, String groesse, Integer bestNr) {  
    try {  
        String sql = "INSERT INTO HALSKETTE(NAME, GROESSE, BESTELLUNGSNUMMER) VALUES( ?, ?,?)";  
        PreparedStatement statement = con.prepareStatement(sql);  
        statement.setString(1, name);  
        statement.setString(2, groesse);  
        statement.setInt(3, bestNr);  
        statement.executeUpdate();  
        statement.close();  
    } catch (Exception e) {  
        System.err.println("Error with insertIntoHalskette(): " + e.getMessage());  
    }  
}
```

Beispiel: `DatabaseHelper.java` -> definition von `insertIntoHalskette(...)`

```
// * * * * * INSERT INTO HALSKETTE * * * * *  
System.out.println("Insert into Halskette dauert ca ~30 sekunden.");  
List<String> groessen = new ArrayList<>(Arrays.asList("XS", "S", "M", "L", "XL"));  
  
for (int h = 0; h < 2300; h++) {  
    int hColor = new Random().nextInt(produktFarbe.size());  
    int hShape = new Random().nextInt(produktForm.size());  
    int hSize = new Random().nextInt(groessen.size());  
    int hBest = new Random().nextInt(bestellungenNummern.size());  
  
    Integer hBestellung = bestellungenNummern.get(hBest);  
    String hName = produktFarbe.get(hColor) + " " + produktForm.get(hShape);  
    String hGroesse = groessen.get(hSize);  
  
    dbHelper.insertIntoHalskette(hName, hGroesse, hBestellung);  
}  
int halsres = dbHelper.countFromHalskette();  
System.out.println("Insert into Halskette fertig! " + halsres + " von 2300");
```

Beispiel: `Tester.java` -> ausführen von `insertIntoHalskette(...)`

4.2 PHP

Zuerst muss ich die Allgemeine Idee beschreiben, damit auch der Zusammenhang zwischen den .php Files verständlicher ist. Die Idee war eine „realistische“ Web-Seite zu machen; jeder Kunde kann sich anmelden und neue Bestellungen machen. Da ich nicht so viel Erfahrung mit PHP und HTML/CSS habe, ist es nicht Perfekt. Die zwei Tabellen für die, die CRUD gilt sind Kunde und Bestellung. Jede Tabelle kann man entweder Lesen oder in diese einfügen. Ohrring, Halskette haben CREATE, READ und DELETE.

Stored Procedure ist für die ausstellung von Rechnung verantwortlich.

LISTE:

addToBestellung.php, allDesigners.php, bestellungInfo.php, changeVorname.php, changeNachname.php, createBestellung.php, createHalskette.php, createOhrring.php, createOhrringPaar.php, createRechnung.php, DatabaseHelper.php, deleteBestellung.php, deleteHalskette.php, deletkunde.php, deleteOhrring.php, designerKontakt.php, index.php, insertKreirenbestellung.php, insertkunde.php, kundeAnmelden.php, kundeBestellungen.php, kundeProfil.php, readAllBestellungen.php, showMitarbeitDaten.php, updateBestellung.php, updateKunde.php.

INDEX.PHP



Startseite, von der hat man Zugriff auf readAllBestellung.php, allDesigner.php und kundeAnmelden.php.

READALLBESTELLUNGEN.PHP

Alle unsere Produkte			
Ohrringe:		Halsketten:	
Name	Preis	Name	Preis
Pale Peach Elephant	8.23 €	Cream Rainbow	45.12€
Champagne Pineapple	20.45 €	Pearl White Lipstick	51.6€
Dusty Rose Crown	9.4 €	Baby Blue Star	34.96€
Seafoam Green Flip Flop	9.58 €	Pearl White Cat	42.32€
Light Mint Camera	13.94 €	Periwinkle Anchor	60.59€
Pale Blue Music Note	9.43 €	Lavender Diamond	62.39€
Salmon Smiley Face	12.76 €	Lilac Feather	22.8€
Soft Pink Star	15.73 €	Blush Heart	31.3€
Powder Pink Music Note	14.92 €	Pale Salmon Ice Cream Cone	58.58€
Pale Peach Watermelon	20.82 €	Seafoam Green Flip Flop	46.06€
Pearl White Raindrop	8.36 €	Pale Green Cupcake	60.67€
Mauve Raindrop	18.6 €	Pale Peach Mermaid Tail	22.31€
Powder Blue Anchor	21.64 €	Pastel Blue Cupcake	40.71€
Baby Blue Pizza Slice	21.77 €	Dusty Rose Cloud	27.78€
Light Grey Taco	11.7 €	Pale Blue Bow	14.6€
Coral Smiley Face	11.03 €	Powder Pink Cat	39.65€
Powder Blue Diamond	9.76 €	Pale Peach Lipstick	38.19€
Pale Peach Sun	15.98 €	Dusty Rose Anchor	57.26€
Aqua Sunglasses	20.25 €		

Hier sind *alle* Einträge von Ohrring und Halskette aufgelistet.

ALLDESIGNERS.PHP


Designers List	
Du kannst ihre Zusammenarbeit sehen HIER	
Klicke auf eine beliebige E-mail um eine nachricht zu senden	
Name	E-Mail
Hello Kitty	Hello_Kitty@sanrio.com">Hello_Kitty@sanrio.com
My Melody	My_Melody@sanrio.com
Little Twin Stars	Little_Twin_Stars@sanrio.com
Cinnamoroll	Cinnamoroll@sanrio.com
Keroppi	Keroppi@sanrio.com
Badtz-Maru	Badtz-Maru@sanrio.com
Chococat	Chococat@sanrio.com
Tuxedosam	Tuxedosam@sanrio.com
Pochacco	Pochacco@sanrio.com
Hangyodon	Hangyodon@sanrio.com
Kuromi	Kuromi@sanrio.com
Sugarbunnies	Sugarbunnies@sanrio.com

Hier sind alle Designer aufgelistet, wenn man auf die E-Mails klickt, führt dieses zu designerKontakt.php. Hier Wird in die Tabelle HabenKontakt insertet. Bei „HIER“ wird das MitarbeitDaten View gezeigt. Der Kunde kann hier sehen wie viele Produkte welcher Designer mit welchen Assistenten gemacht/kreiert hat (showMitarbeitDaten.php).

SHOWMITARBEITDATEN.PHP

Zusammenarbeit von Designer und Assistenten	
Designer: Kuromi	Assistent: Landry
Anzahl Ohringe: 0 & Halsketten: 19	
Designer: Kuromi	Assistent: Pekkie
Anzahl Ohringe: 0 & Halsketten: 3	
Designer: Kuromi	Assistent: Rascal
Anzahl Ohringe: 0 & Halsketten: 12	
Designer: Kuromi	Assistent: Usahana
Anzahl Ohringe: 0 & Halsketten: 15	
Designer: Kuromi	Assistent: Jewelpet
Anzahl Ohringe: 93 & Halsketten: 17	
Designer: Kuromi	Assistent: Monikichi
Anzahl Ohringe: 0 & Halsketten: 2	
Designer: Kuromi	Assistent: Hana-Marui
Anzahl Ohringe: 0 & Halsketten: 8	
Designer: Kuromi	Assistent: Tora-Marui
Anzahl Ohringe: 42 & Halsketten: 8	
Designer: Kuromi	Assistent: Tuxedo Sam

KUNDEANMELDEN.PHP



Hier kannst du dich mit deiner Registration-Email anmelden:

E-Mail:



Du musst dich zuerst Registrieren, damit du etwas bestellen kannst:

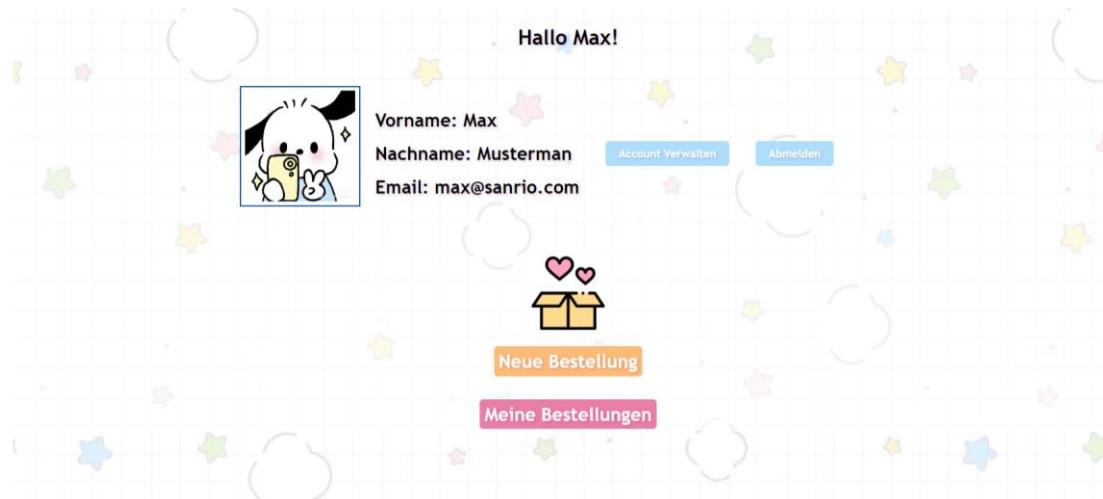
Vorname:

Nachname:

E-Mail:

Hier Kann sich der Kunde anmelden oder registrieren. Dafür braucht braucht man eine E-Mail die der primary key in der Kunde Tabelle ist. Registrieren Knopf führt zu insertKunde.php wo dieser in die Datenbank gespeichert wird (CREATE). Bei Anmelden, wenn die E-Mail in der Datenbank gespeichert ist geht man zu kundeProfil.php(READ).

KUNDEPROFIL.PHP



Dieses PHP ist das READ von Kunde. Bei „Account Verwalten“ kann man den Vornamen und/oder Nachnamen ändern, die Bestätigung für die jeweilige Änderung führt zu changeVorname.php oder changeNachname.php. (Die Namen „updateVorname/Nachname“ würden mehr Sinn machen (UPDATE)). Dort kann man auch sein Profil löschen (DELETE). Bei Abmelden geht man zurück auf index.php. „Neue Bestellung“ führt zu createBestellung.php und „Meine Bestellungen“ zu kundeBestellungen.php.

CREATEBESTELLUNG.PHP



Beim Klicken wird schon eine neue bestellung kreiert(CREATE), weil die Bestellungsnummer für die Anderen Tabellen der foreign key ist. Der Kunde wählt wer seine Produkte kreieren soll, das führt zu insert in KreierenBestellung. Beim Klicken auf den Namen wird die ID gespeichert.

INSERTKREIRENBESTELLUNG.PHP

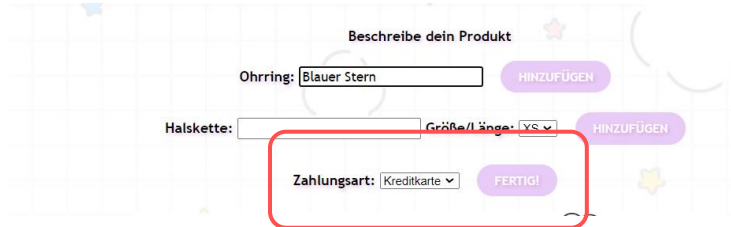


Issue: da schon beim Klicken auf „Neue Bestellung“ eine Bestellung kreiert wird, und wenn diese seite refresht wird, wird immer eine neue Bestellung eingefügt. Deshalb wählt der Kunde seinen Designer aus zuerst und dann die Produkte. Hätte ich die Produkte auswahl auf der selben Seite gemacht, würden die Produkte für unterschiedliche Bestellungen gespeichert. Hier kann der Kunde auch seine Bestellung Abbrechen, dieses führt zum kundeProfil.php. Im DatabaseHelper.php habe ich eine methode die jede Bestellung löscht die einen Gesamtpreis von 0 hat, da diese dann keine Produkte beinhalten, keine valide Bestellung für mein Thema.

ADDTOBESTELLUNG.PHP



Auswahl der Produkte, der Knopf „hinzufügen“ führt zu createOhrring.php oder createHalskette.php. Diese PHP Files führen automatisch zurück auf addToBestellung.php. Erst wenn die Bestellung mindestens ein Produkt enthält, kann der Kunde seine zahlungart auswählen und die Bestellung bestätigen.



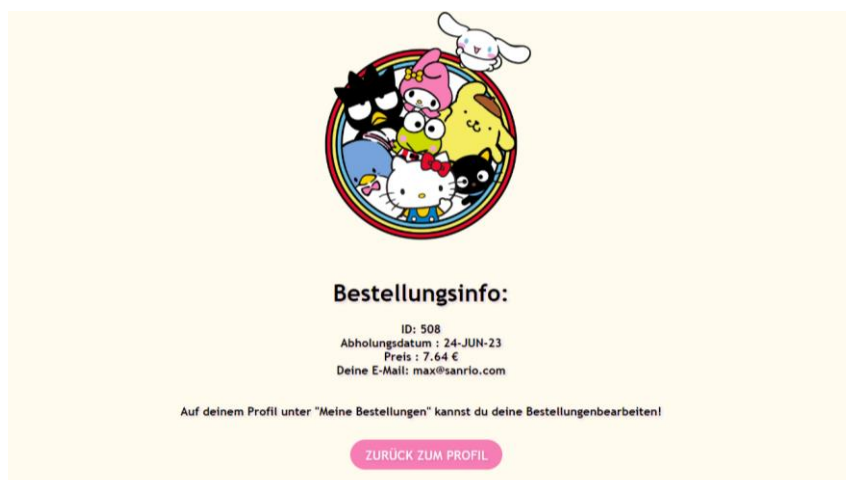
Knopf „Fertig“ führt zu createRechnung.php, dort wird die Stored Procedure ausgeführt, Zahlungsart stellt die eingabe Daten dar und die ausgabe Daten, Rechnungsnummer, ist auf der nächsten PHP Datei dargestellt.

CREATERECHNUNG.PHP



Der Knopf „Gutschein Einlösen“ stellt das UPDATE von Bestellung dar. Diese Verändert den Gesamtpreis. Führt zu updateBestellung.php wo die methode aufgerufen wird.

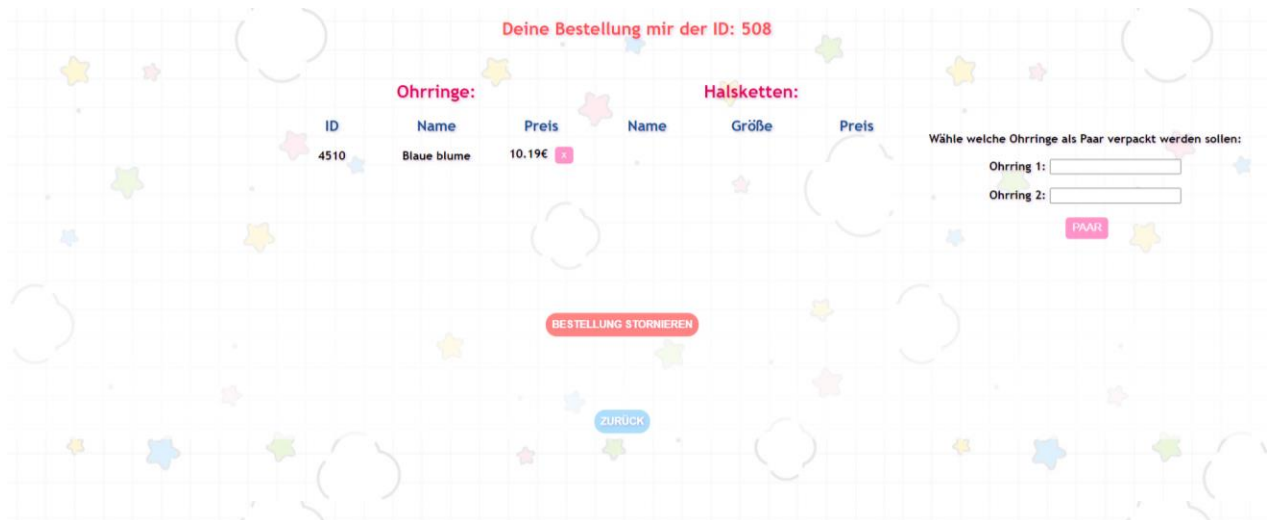
UPDATEBESTELLUNG.PHP



KUNDEBESTELLUNGEN.PHP



BESTELLUNGINFO.PHP



Hier kann der Kunde Produkte entfernen(X) aus der bestellung (deleteOhrring/deleteHalskette.php), oder die ganze Bestellung „Bestellung Stornieren“ (deleteBestellun.php). Oder auswählen welche Ohrringe als Paar verpackt werden sollen, mit der ID eingabe von den Ohrringen.

Trigger welcher nicht ein "Auto-Increment" Verhalten abbildet:

- Gesamtpreis für bestellung
- Zufälliger Preis für Ohrring
- Zufälliger Preis für Halskette

Stored Procedure, die mindestens einen Input-Parameter über die Webseite entgegennimmt, und Output-Parameter zurücksendet

- ErstelleRechnung
- Input: Zahlungsart
- Output: Rechnungsnummer

Bestellung – C R U D

- Create: kundeProfil.php „Neue Bestellung“ -> insertkunde.php
- Read: kundePofil.php „Meine Bestellungen“, bei meiner Web-Seite hat es kein Sinn gemacht alle Bestellungen irgendwo aufzulisten -> kundeBestellungen.php
- Update: createRechnung.php „Gutschein einlösen“ -> updateBestellung.php
- Delete: bestellungInfo.php „Bestellung Stornieren“ -> deleteBestellung.php

Kunde – C R U D

- Create: kundeAnmelden.php „Registrieren“ -> insertkunde.php
- Read: kundeAnmelden.php „Anmelden“ -> kundeProfil.php
- Update: updateKunde.php „Vorname/Nachname ändern“ -> changeVorname.php, changeNachname.php
- Delete: updateKunde.php „Account löschen“ -> deletekunde.php

4.3 Normalformen

1NF

- ✓ jede Tabelle hat ein Primärschlüssel
- ✓ kein Attribut besteht aus strukturierten Werten

2NF

- ✓ jedes nicht primes Attribut hängt von jedem Schlüsselkandidaten ab

3NF

- ✓ kein nicht primes Attribut transitiv vom Primärschlüssel abhängig

Meiner Meinung nach, wenn man ein korrektes ER-Diagramm hat und dieses korrekt in ein Relation-Schema umwandelt, ist immer mindestens die 2NF erfüllt.