**Aufgabe 2 - Konzeptuelle Modellierung**

# **A. Use Case Beschreibungen**

## **A.1 Detaillierte textuelle Beschreibung (2 Punkte)**

#### Suche nach einem Gut

|  |  |
| --- | --- |
|  |  |
| Name | Suche nach einem Gut und bekommen der Informationen |
| Akteure | Kunde |
| Kurzbeschreibung | Ein Kunde kann nach einem bestimmten Gut suchen. Die entsprechenden Ergebnisse werden aufgelistet und die Informationen gesendet. |
| Ziel | Kunde sucht nach einem Gut, findet es und bekommt die Informationen. |
| Trigger | Kunde initiiert die Suche |
| Vorbedingungen | Kunde ist registriert und authentifiziert. |
| Essentielle Schritte | 1. Kunde gibt Suchterme an und initiiert die Suche.  2. Das System empfängt die Suchanfrage.  3. Das System sucht nach passenden Ergebnissen.  4. Das System retourniert die gefunden Ergebnisse inkl. Details wie beispielsweise die Beschreibung. |
| Erweiterungen | 2a. Kunde ist nicht authentifiziert - der Prozess wird abgebrochen. |
| Nachbedingungen | Kunde hat die gefundenen Suchergebnisse empfangen |

#### Durchführung des Kaufs eines Gutes mit entsprechender Geldtransaktion

|  |  |
| --- | --- |
|  |  |
| Name | Kauf eines Gutes |
| Akteure | Kunde |
| Kurzbeschreibung | Ein Kunde kann ein oder mehrere Produkte bestellen, also kaufen und bezahlen. |
| Ziel | Kunde kauft ein oder mehrere Produkte |
| Trigger | Kunde initiiert den Kauf. |
| Vorbedingungen | Kunde ist registriert und authentifiziert. |
| Essentielle Schritte | 1. Kunde gibt die zu bestellenden Produkte samt gewünschter Menge an und initiiert den Kauf.  2. Das System überprüft ob der Kauf bzw. der Käufer authentifiziert ist.  3. System speichert die Bestellung und gibt Rechnungsinformationen zurück.  4. Kunde wickelt die Bezahlung ab. |
| Erweiterungen | 2a. Kunde ist nicht authentifiziert - der Prozess wird abgebrochen. |
| Nachbedingungen | Bestellung wurde aufgegeben und bezahlt. |

#### Änderung/Stornierung einer Bestellung mit entsprechender Geldtransaktion

|  |  |
| --- | --- |
|  |  |
| Name | Änderung/Stornierung einer Bestellung |
| Akteure | Kunde |
| Kurzbeschreibung | Solange eine Bestellung noch nicht Versendet ist (Lieferdatum festgelegt) kann diese geändert oder sogar storniert werden. |
| Ziel | Änderung oder Stornierung einer Bestellung. |
| Trigger | Kunde initiiert Bearbeitung oder Stornierung einer Bestellung. |
| Vorbedingungen | 1. Kunde ist registriert und authentifiziert.  2. Kunde hat etwas bestellt.  3. Bestellung wurde noch nicht versendet. |
| Essentielle Schritte | Änderung  1. Kunde gibt geänderte Bestellung an und initiiert die Änderung.  2. System überprüft ob der Kunde authentifiziert ist.  3. System speichert die Änderungen inkl. geändertem Rechnungsbetrag.  4. System initiiert die Überweisung oder Nachforderung der Differenz.  Stornierung  1. Kunde initiiert die Stornierung einer Bestellung.  2. System überprüft ob der Kunde authentifiziert ist.  3. System überprüft den Bestellungsstatus.  4. System speichert die Änderungen.  5. System initiiert die Überweisung der Differenz. |
| Erweiterungen | 2a. User ist nicht authentifiziert – der Prozess wird abgebrochen. |
| Nachbedingungen | Bestellung wurde geändert gespeichert oder gelöscht. |

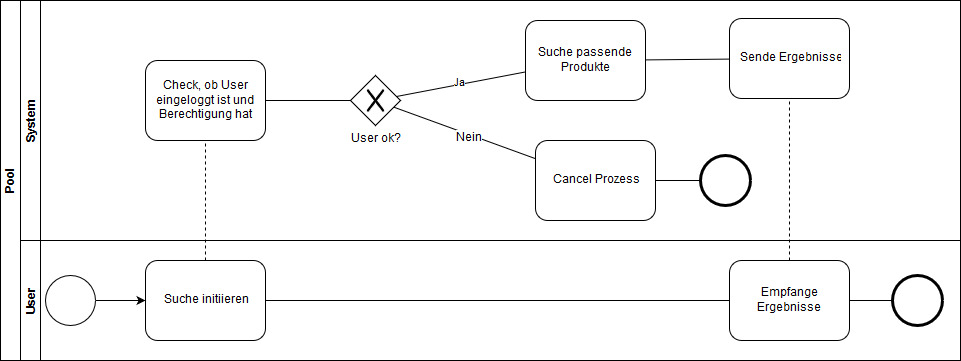
#### Administration der eigenen Kundendaten

|  |  |
| --- | --- |
|  |  |
| Name | Administration der eigenen Kundendaten |
| Akteure | Kunde |
| Kurzbeschreibung | Der Kunden kann bestimmte Daten, die ihn betreffen, bearbeiten und anzeigen lassen. Außerdem kann er seinen Account entfernen, wenn alle Bedingungen erfüllt sind. |
| Ziel | Administration der eigenen Daten (Anzeigen, Ändern, Löschen) |
| Trigger | Kunde initiiert die Bearbeitung seiner Daten |
| Vorbedingungen | Kunde ist registriert und authentifiziert. |
| Essentielle Schritte | Anzeigen  1. Kunde initiiert die Anzeige seiner persönlichen Daten.  2. System überprüft ob er dafür authentifiziert ist.  3. System retourniert die entsprechenden Daten.  Löschen  1. Kunde initiiert das Löschen seines Accounts.  2. System überprüft ob der User dafür authentifiziert ist.  3. System überprüft, ob der User noch offene Rechnungen ausstehen hat. 3. System löscht den Account des Users.  Bearbeiten  1. Kunde gibt die gewünschten Änderungen an und initiiert das Bearbeiten der persönlichen Daten.  2. System überprüft ob der User dafür authentifiziert ist.  3. System speichert die neuen Änderungen.  4. System retourniert die neuen Daten. |
| Erweiterungen | 2a. Kunde ist nicht authentifiziert – Prozess wird abgebrochen. |
| Nachbedingungen | Kundendaten sind geändert/werden angezeigt. / Account wurde gelöscht. |

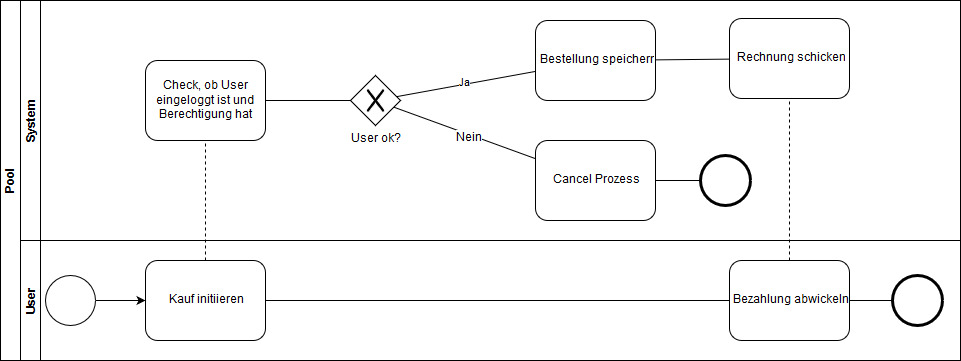
## **A.2 Grafische Darstellung des dynamischen Ablaufs jedes Use Cases (z.B. BPMN, UML Sequenzdiagramme) (4 Punkte)**

Anmerkung: leider ging mit dem Tool etwas schief, natürlich sollten es Pfeile sein und nicht einfach nur Striche.

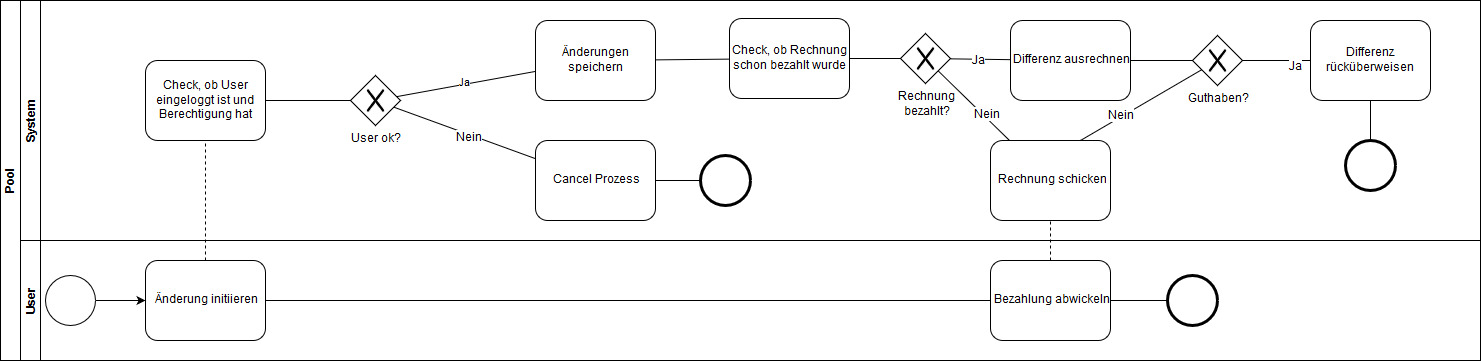
#### Suche nach einem Gut und Feststellung der Verfügbarkeit

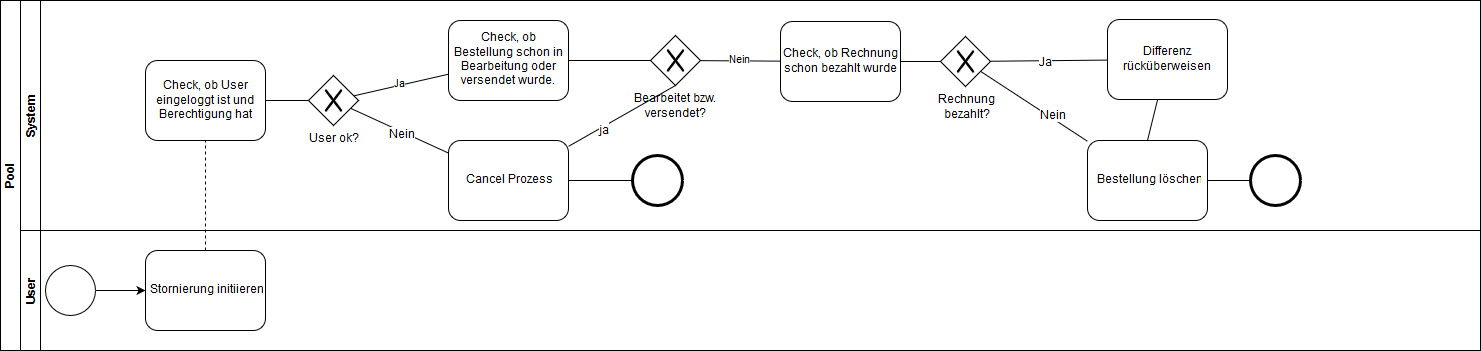


#### Durchführung des Kaufs eines Gutes mit entsprechender Geldtransaktion

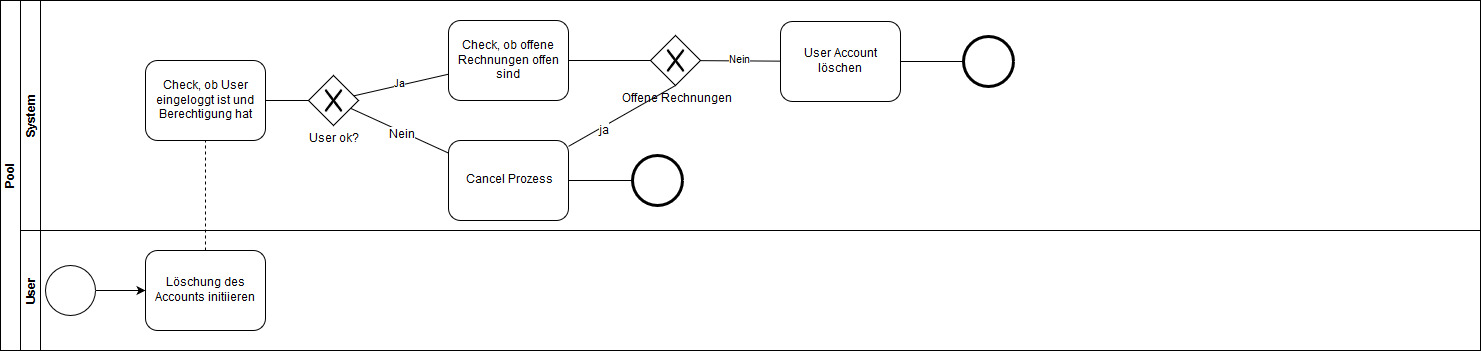


#### Änderung/Stornierung einer Bestellung mit entsprechender Geldtransaktion





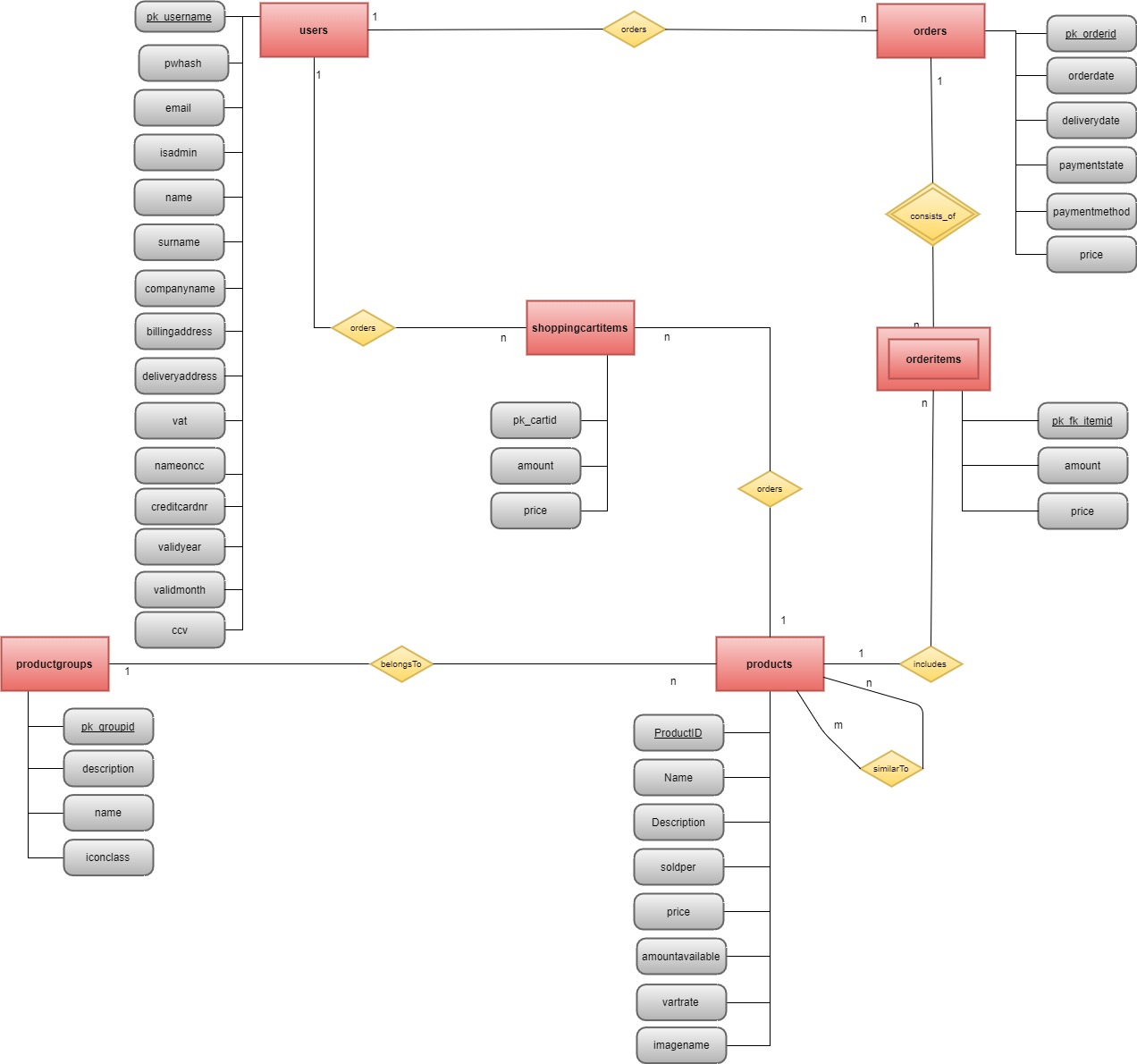
#### Administration der eigenen Kundendaten



## 

## 

# **B. Datenmodell**



# **C. Definition der EDI Sprache für B2B Use Case Prozesse**

## **Definieren Sie die Grammatik für die EDI Sprachkonstrukte und geben Sie für Konstrukt ein Beispiel seiner Verwendung beim Datenaustausch (5 Punkte)**

Es werden Json Nachrichten verwendet, um Informationen zwischen API und Client auszutauschen.

## Json Schemas betreffend Orders

### POST ORDER - call

"order": {

"id": "order",

"description": "one order",

"required": [ "orderdate", "paymentstate", "item", "price"],

"properties": {

"orderdate":{

"type": "date",

"description": "date when order was issued"

}

"deliverydate":{

"type": "date",

"description": "date when order will be delivered"

}

"paymentstate":{

"type": "string",

"description": "state of payment. Can be open or payed"

}

"paymentmethod":{

"type": "string",

"description": "method chosen for payment. Can be cc or bill"

}

"price":{

"type": "integer",

"description": "total bill amount"

}

"item":{

"pk\_fk\_itemid": {

"type": "integer",

"description": "the item id"

}

"price": {

"type": "integer",

"description": "price of item"

}

"amount": {

"type": "integer",

"description": "amount to buy for the chosen item"

}

"fk\_productid": {

"type": "integer",

"description": "product id"

}

}

}

}

### PUT ORDER - call

"order": {

"id": "order",

"description": "one order",

"properties": {

"orderdate":{

"type": "date",

"description": "date when order was issued"

}

"deliverydate":{

"type": "date",

"description": "date when order will be delivered"

}

"paymentstate":{

"type": "string",

"description": "state of payment. Can be open or payed"

}

"paymentmethod":{

"type": "string",

"description": "method chosen for payment. Can be cc or bill"

}

"price":{

"type": "integer",

"description": "total bill amount"

}

"item":{

"pk\_fk\_itemid": {

"type": "integer",

"description": "the item id"

}

"price": {

"type": "integer",

"description": "price of item"

}

"amount": {

"type": "integer",

"description": "amount to buy for the chosen item"

}

"fk\_productid": {

"type": "integer",

"description": "product id"

}

}

}

}

## Beispiele für Json Files

### GET Orders - Antwort

[

{

"pk\_orderid": 1,

"orderdate": "2017-12-16T23:00:00.000Z",

"deliverydate": null,

"paymentstate": "open",

"paymentmethod": "bill",

"price": "1.0",

"items": [

{

"pk\_fk\_itemid": 1,

"amount": 2,

"fk\_pk\_orderid": 1,

"price": "1.0",

"fk\_productid": 1

},

{

"pk\_fk\_itemid": 2,

"amount": 1,

"fk\_pk\_orderid": 1,

"price": "0.4",

"fk\_productid": 2

}

]

}]

### GET Order - Antwort

{

"pk\_orderid": 2,

"orderdate": "2017-12-31T23:00:00.000Z",

"deliverydate": null,

"paymentstate": "payed",

"paymentmethod": "cc",

"price": "3.19",

"fk\_username": "paul.meier"

}

### POST Order - Call

{

"orderdate": "2017-12-16T23:00:00.000Z",

"deliverydate": null,

"paymentstate": "openopen",

"paymentmethod": "bill",

"price": "1.4",

"item":[{"pk\_fk\_itemid":1,

"price":2.5,

"amount":2,

"fk\_productid":2

}

]

}

### PUT Order

{

"deliverydate ": "2017-12-16T23:00:00.000Z",

}

# Json Schema betreffend Users

### GET Users - Antwort

[

{

"pk\_username": "string",

"pwhash": "string",

"email": "string",

"isadmin": true,

"name": "string",

"surname": "string",

"companyname": "string",

"billingaddress": "string",

"deliveryaddress": "string",

"vat": "string",

"nameoncc": "string",

"creditcardnr": "string",

"validyear": 0,

"validmonth": 0,

"ccv": 0

}

]

### PUT User - call

"user": {

"id": "user",

"description": "One user",

"required": ["email", "name", "surname", "billingaddress", "deliveryaddress"],

"properties": {

"pk\_username": {

"type": "string",

"description": "Uniqe username"

},

"email": {

"type": "string",

"description": "email address of the user"

},

"isadmin": {

"type": "boolean",

"description": "only set for administrators"

},

"name": {

"type": "string",

"description": "name of the user"

},

"surname": {

"type": "string",

"description": "surname of the user"

},

"companyname": {

"type": "string",

"description": "in case of commercial users; name of the company the user is part of"

},

"billingaddress": {

"type": "string",

"description": "address for the bill"

},

"deliveryaddress": {

"type": "string",

"description": "address for the delivery"

},

"vat": {

"type": "string",

"description": "vat number of the company"

},

"nameoncc": {

"type": "string",

"description": "name of the credit card of the user"

},

"creditcardnr": {

"type": "string",

"description": "number of the credit card of the user"

},

"validyear": {

"type": "integer",

"description": "year till that the credit card of the user is valid"

},

"validmonth": {

"type": "integer",

"description": "month till that the credit card of the user is valid"

},

"ccv": {

"type": "integer",

"description": "ccv of the users credit card"

}

}

}

### POST User (Register) – Call

"user": {

"id": "user",

"description": "One user",

"required": ["username", "pwhash", "email"],

"properties": {

"pk\_username": {

"type": "string",

"description": "Uniqe username"

},

" pwhash ": {

"type": "string",

"description": "hash for PW"

},

"email": {

"type": "string",

"description": "email address of the user"

},

"isadmin": {

"type": "boolean",

"description": "only set for administrators"

},

"name": {

"type": "string",

"description": "name of the user"

},

"surname": {

"type": "string",

"description": "surname of the user"

},

"companyname": {

"type": "string",

"description": "in case of commercial users; name of the company the user is part of"

},

"billingaddress": {

"type": "string",

"description": "address for the bill"

},

"deliveryaddress": {

"type": "string",

"description": "address for the delivery"

},

"vat": {

"type": "string",

"description": "vat number of the company"

},

"nameoncc": {

"type": "string",

"description": "name of the credit card of the user"

},

"creditcardnr": {

"type": "string",

"description": "number of the credit card of the user"

},

"validyear": {

"type": "integer",

"description": "year till that the credit card of the user is valid"

},

"validmonth": {

"type": "integer",

"description": "month till that the credit card of the user is valid"

},

"ccv": {

"type": "integer",

"description": "ccv of the users credit card"

}

}

}

## Beispiele für Json Files dazu

### GET Users

[

{

"pk\_username": "paul.meier",

"pwhash": "jGl25bVBBBW96Qi9Te4V37Fnqchz/Eu4qB9vKrRIqRg=",

"email": "paul.meier@gmx.at",

"isadmin": false,

"name": "Paul",

"surname": "Meier",

"companyname": null,

"billingaddress": "Sackgasse 25/1 1010 Wien",

"deliveryaddress": "Sackgasse 20 1010 Wien",

"vat": null,

"nameoncc": "Paul Arthur Meier",

"creditcardnr": "204038810305038",

"validyear": 2020,

"validmonth": 6,

"ccv": 421

},

{

"pk\_username": "admin",

"pwhash": "jGl25bVBBBW96Qi9Te4V37Fnqchz/Eu4qB9vKrRIqRg=",

"email": "admin@lovefoods.at",

"isadmin": true,

"name": "Admin",

"surname": "Istrator",

"companyname": null,

"billingaddress": null,

"deliveryaddress": null,

"vat": null,

"nameoncc": null,

"creditcardnr": null,

"validyear": null,

"validmonth": null,

"ccv": null

}]

### PUT User

[

{

"email": "paul.meier@gmx.at",

"name": "Paul",

"surname": "Meier",

"billingaddress": "Sackgasse 25/1 1010 Wien",

"deliveryaddress": "Sackgasse 20 1010 Wien",

}

]

### POST User (Register) -Call

{

"pk\_username": "bu.meier",

"pwhash": "jGl25bVBBBW96Qi9Te4V37Fnqchz/Eu4qB9vKrRIqRg=",

"email": "paul.meier@gmx.at",

"isadmin": false,

"name": "Paul",

"surname": "Meier",

"companyname": null,

"billingaddress": "Sackgasse 25/1 1010 Wien",

"deliveryaddress": "Sackgasse 20 1010 Wien",

"vat": null,

"nameoncc": "Paul Arthur Meier",

"creditcardnr": "204038810305038",

"validyear": 2020,

"validmonth": 6,

"ccv": 421

}

# Json Schema betreffend Products

### GET Products - Antwort

"product": {

"id": "product",

"description": "One product",

"properties": {

"pk\_productid": {

"type": "integer",

"description": "Uniqe productid"

},

"name": {

"type": "string",

"description": "name of the product"

},

"description": {

"type": "string",

"description": "description about the product"

},

"soldper": {

"type": "string",

"description": "gives the amount/grams/whatever per which that product is sold"

},

"price": {

"type": "double",

"description": "gives the price of the product"

},

"amountavailable": {

"type": "integer",

"description": "gives the nuber of available items of that product"

},

"vatrate": {

"type": "integer",

"description": "gives the vat rate for that product"

},

"imagename": {

"type": "string",

"description": "gives the name of the image for that product (used for GUI)"

}

"fk\_groupid": {

"type": "integer",

"description": "group id to which this product belongs ( used for filtering in GUI)"

}

}

}

### POST Products - Call

"product": {

"id": "product",

"description": "One product",

"required": ["name", "soldper", "price", "amountavailable", "vatrate", "fk\_groupid"],

"properties": {

"pk\_productid": {

"type": "integer",

"description": "Uniqe productid"

},

"name": {

"type": "string",

"description": "name of the product"

},

"description": {

"type": "string",

"description": "description about the product"

},

"soldper": {

"type": "string",

"description": "gives the amount/grams/whatever per which that product is sold"

},

"price": {

"type": "double",

"description": "gives the price of the product"

},

"amountavailable": {

"type": "integer",

"description": "gives the nuber of available items of that product"

},

"vatrate": {

"type": "integer",

"description": "gives the vat rate for that product"

},

"imagename": {

"type": "string",

"description": "gives the name of the image for that product (used for GUI)"

}

"fk\_groupid": {

"type": "integer",

"description": "group id to which this product belongs ( used for filtering in GUI)"

}

}

}

### GET Products - Antwort

## Beispiele für Json Files dazu

### GET Products - Antwort

[

{

"pk\_productid": 1,

"name": "Apple",

"description": "Apples \"Pink Lady\". Great sweet taste. Country of origin: Austria.",

"soldper": "piece",

"price": "0.5",

"amountavailable": 1,

"vatrate": 20,

"imagename": "apple.jpg",

"fk\_groupid": 1

},

{

"pk\_productid": 2,

"name": "Banana",

"description": "SanLucar Bananas. SanLucar gourmet bananas are harvested in Ecuador all around the year.",

"soldper": "piece",

"price": "0.4",

"amountavailable": 2,

"vatrate": 20,

"imagename": "banana.jpg",

"fk\_groupid": 1

}]

### POST Product

{

"name": "Apple",

"description": "Apples \"Pink Lady\". Great sweet taste. Country of origin: Austria.",

"soldper": "piece",

"price": "0.5",

"amountavailable": 1,

"vatrate": 20,

"imagename": "apple.jpg",

"fk\_groupid": 1

}

# **D. B2B Schnittstellenbeschreibung**

## **D.1 Geben Sie eine Definition der B2B Service Schnittstelle, die es erlaubt, die Use Cases zu realisieren (2 Punkte)**

Die Schnittstelle für B2B Kunden wird eine REST Schnittstelle sein. Auch vom Web Frontend aus wird ­auf die REST-API zugegriffen werden.

Es wird HTTPS und Basic Authentication verwendet, also Username und Passwort base64-encoded im Header mitgegeben. Alle Requests, ausgenommen Registrierung, setzen solch eine Authentication voraus.

#### Registrieren

**POST /register**

Ein POST auf /register ermöglicht die Registrierung eines Users. Dabei müssen alle nötigen Informationen, wie in der entsprechenden Schema-Definition aus Punkt C, im Body übertragen werden.

#### Suche nach einem Gut und Feststellung der Verfügbarkeit

**GET /products**

Gibt alle Produkte zurück. (bzw. die ersten 100, falls es sonst zu viele wären)

Bzw. mit möglichen queryParams: groupname, groupid, name, status, keyword, limit, offset

Gibt alle Produkte, die den angegebenen query-Parametern entsprechen, zurück. Es kann Produkt-Gruppe, Name oder Verfügbarkeitsstatus oder eine beliebige Kombination davon angegeben werden.

**GET /products/{id}**

Gibt das Produkt mit angegebener ID zurück.

#### Durchführung des Kaufs eines Gutes mit entsprechender Geldtransaktion

**POST /orders**

Ein POST auf /orders ermöglicht es, eine neue Bestellung zu tätigen. Dabei müssen alle nötigen Informationen, unter Zuhilfenahme des im Punkt C entsprechend definierten Schema, als XML-Dokument im body übertragen werden.

#### Änderung/Stornierung eines Auftrags mit entsprechender Geldtransaktion

**GET /orders**

Gibt alle Bestellungen des authentifizierten Users zurück.

**GET /orders/{id}**

Gibt die Bestellung mit der entsprechenden ID zurück, sofern der authentifizierte User diese Bestellung getätigt hat.

**PUT /orders/{id}**

Ein PUT auf /orders/{orderid} ermöglicht das Verändern einer Order mit der entsprechenden OrderID. Dabei wird die Order durch die neu, dem entsprechendem Schema aus C, angegebene Order ersetzt, sofern alle Bedingungen (Authentication, Status der Bestellung – siehe UC Beschreibung) erfüllt sind.

**DELETE /orders/{id}**

Um eine bestimmte Bestellung mit einer bestimmten ID zu stornieren, ist ein DELETE auf /orders/{id} nötig.

#### Administration der Kundendaten

**GET /users**

Gibt die persönlichen Daten des angemeldeten User zurück.

**PUT /users/{username}**

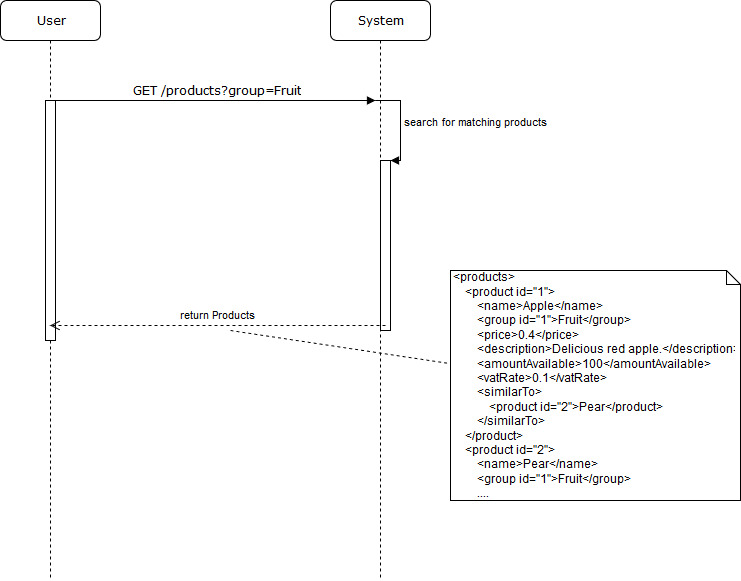
Ein PUT auf /user ermöglicht es, spezifische Kundendaten des angemeldeten Users zu ändern. Im body muss hierfür ein XML Dokument, welches dem entsprechenden Schema aus Punkt C entspricht, übertragen werden.

**DELETE /users/{username}**

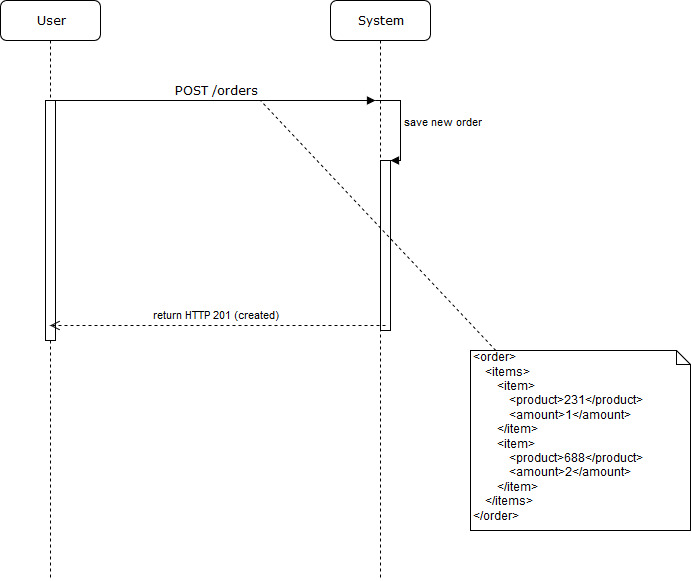
Ein DELETE auf /user ermöglicht es, den Account des angemeldeten Users zu löschen, sofern alle Bedingungen dafür erfüllt sind.

## **D.2 Geben Sie für alle Use Cases die B2B Protokollsequenzen für den Einsatz der EDI Sprache (Schreibtischtest) (4 Punkte)**

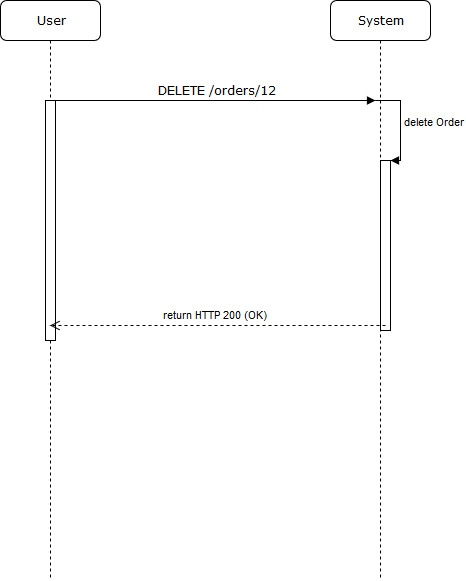
## Suche



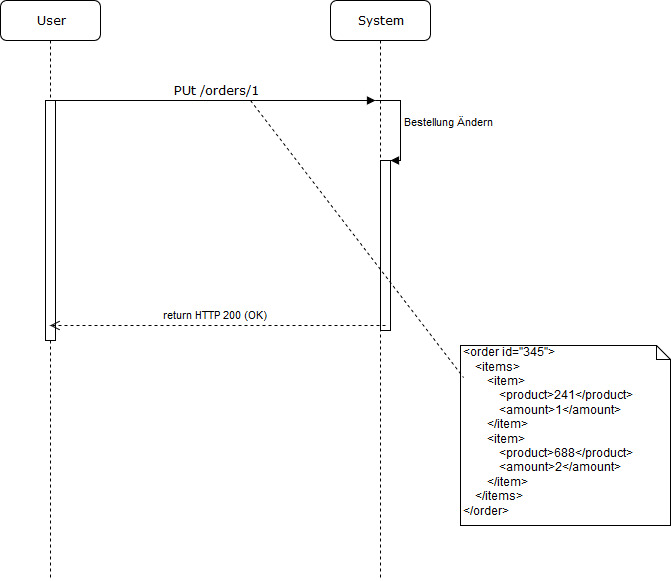
## Bestellen



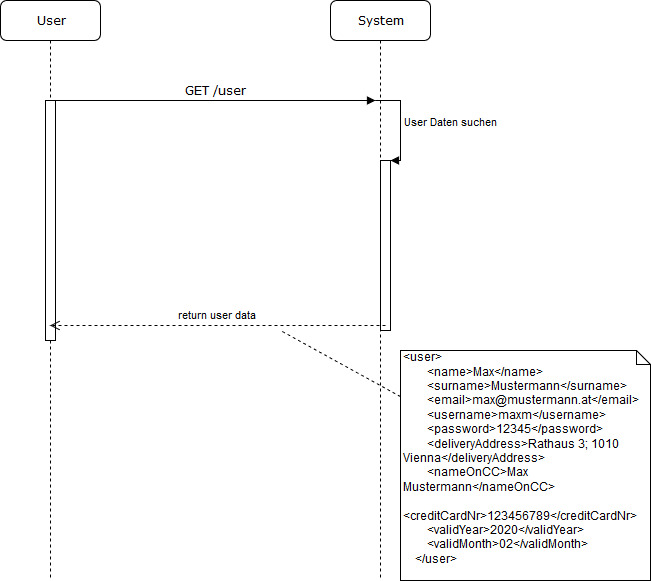
## Stornieren



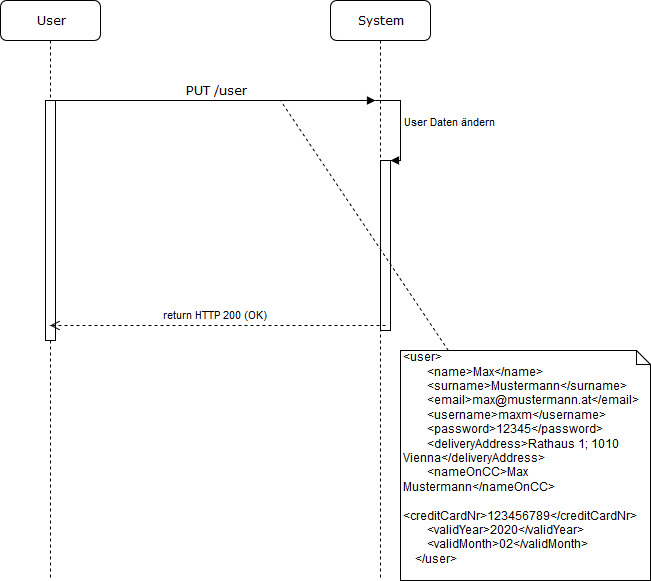
## Ändern



## Userdaten bekommen



## Userdaten ändern



## Useraccount löschen

