



Projektbezeichnung

## planet.shop

**Mitglieder:** Robin Gugel | Daniel Mazanek | Patrick Mistlberger | David Pointner | Oscar Yim SPRINT BACKLOG

**Datum:** 20.11.2017

**Zeit:** 9:00 – 9:20 Uhr

# USER STORIES:

## 1. Waren anzeigen

### Spezifikation:

Die Waren werden in einer Liste mit Bild und Preis angezeigt.  
 Die Produkte sollen mittels Rest von einem Server angefragt werden.  
 Ein Updaten der View wird nicht durchgeführt, wenn ein Produkt am Server entfernt wird.  
 Es werden aus Programmiertechnischen Gründen schon zum Start der App alle Produkte vom Server abgefragt

Name: max 20 Zeichen

Bild: 265\*265 pixel

Maximaler Preis: 2^31 €

### Priorität:

Must have

### nicht funktuinale anforderungen

-

### definition of done:

Es soll auf einem Handy die App PlanetShop gestartet werden können, man sieht dann auf dem Home Screen alle verfügbaren Produkte in einem Grid angezeigt werden.

### Tasks:

Main UI erstellen done

## 2. Warendetails anzeigen:

### Spezifikation:

Wenn auf ein Produkt geklickt wird, soll ein weiteres Fenster geöffnet werden,

in dem zusätzliche Informationen (Beschreibung, weitere Bilder)

zu dem Produkt angezeigt werden sollen.

### Priorität:

Must have

#### Nicht funktionale Anforderungen:

-

#### Definition-of-Done:

Ich als Käufer möchte auf eine Ware in der Warenliste klicken, und möchte daraufhin die Details sehen.

#### Tasks:

Datenbank anpassen – 1 std done

Neue Request hinzufügen, Server und App Logic anpassen– 2 std done

Main UI anpassen – 1 std done

UI Fenster „ProductDetails“ implementieren – 4 std done

## 3. Ware vermerken

### Spezifikation:

Es soll ein zusätzlicher Menüpunkt "Warenkorb" angelegt werden. Es sollen Produkte in einem eigenen Fenster

aus dem Warenkorb anzeigbar und entfernbar sein.

Auf der Detailsseite von einer Ware soll diese zum Warenkorb hinzufügbar sein.

Alle Benutzer teilen sich den Selben Warenkorb (Da Login noch nicht implementiert ist). Der Warenkorb wird lokal gespeichert.

### Priorität:

Important to have

### Nicht-funktionale Anforderungen:

Easy to use

### Definition-of-Done:

Ich als Käufer möchte mit einem Klick Artikel in einen Warenkorb ablegen.

Außerdem möchte ich mir meinen Warenkorb anzeigen lassen und, falls von mir erwünscht, Waren aus dem Warenkorb entfernen.

### Tasks:

Main UI anpassen und Warendetails UI anpassen – 1 std

UI Warenkorb Fenster implementieren - 4 std

Warenkorb File verwalten – 1,5 std

App Logic anpassen – 1 std

# task spezifikationen

### US1 MAIN UI Erstellen

Es muss ein User Interface erstellet werden, auf welchem die Produkte in einer Liste mit Bild, Namen und Preis angezeigt werden.

//TODO: Layout einfügen

//TODO: Implementierung genauer Spezifizieren

### us2 datenbank anpassen

Es müssen folgende neue Spalten in Tabelle PLANETCARD angelegt werden: „Description“  
 Die Entity PlanetCard muss um das Attribut „Description“ erweitert werden

### US2 Neue Request hinzufügen, Server und App Logic anpassen

Es soll ein einzelnes Produkt über eine get-request mit der URL [http://localhost:8080/planetshop/rs/shop/getproductdetails/{id}](http://localhost:8080/planetshop/rs/shop/getproductdetails/%7bid%7d)   
 angefordert werden können

Die derzeitige request Get-All-Products soll umgebaut werden auf die URL <http://localhost:8080/planetshop/rs/shop/getallproducts> und nur mehr {Bild, Preis, Name} zurücksenden

CardFacade muss derzeitig noch nicht geändert werden.

Die Klassen Request, RequestType, RequestHandler, ClientRequestHandler, WebShopEndpoint sollen entfernt werden, um weniger Abstraktion in dem Projekt zu haben. Es muss ein neuer Endpoint angelegt werden, der die oben stehenden Anforderungen erfüllt, hierfür soll die CardFacade verwendet werden, um die Logik auszulagern

Klasse DataService in der App überarbeiten. Die Methode getAllProducts (äquivalent zu getJson) soll implementiert werden. Die Methode getProductDetails soll implementiert werden, sie soll mittels get request zu der URL [http://10.0.2.2:8080/planetshop/rs/rs/getproductdetails/{id}](http://10.0.2.2:8080/planetshop/rs/rs/getproductdetails/%7bid%7d) die Produkt Details abfragen.

### US2 MAIN UI anpassen

Wenn auf ein Produkt geklickt wird (Bild oder Text) soll auf die Seite „Produkt Details“ weitergeleitet werden.

//TODO: Implementierung genauer spezifizieren

### US2 UI FENSTER „Productdetails“ IMPLEMENTIEREN

Hier soll das Produkt nach vorgegebenem Layout mit allen Details angezeigt werden.

//TODO: Layout einfügen

//TODO: Implementierung genauer spezifizieren

### Us3 MAIN UI anpassen und Warendetails UI anpassen

Ein Warenkorb Icon soll in der oberen rechten Ecke angezeigt werden. Wenn auf dieses Icon geklickt wird, soll die neue Seite „Warenkorb bearbeiten“ angezeigt werden.

Die Seite „Warendetails Anzeigen“ soll um das Icon „Zu Warenkorb hinzufügen“ erweitert werden.

//TODO: Implementierung genauer spezifizieren

### US3 UI WARENKORB FENSTER implementieren

Hier soll der Warenkorb nach vorgegebenem Layout angezeigt werden.

//TODO: Layout einfügen

//TODO: Implementierung genauer spezifizieren

### US3 warenkorb file verwalten

Auf der App muss zuerst überprüft werden, ob das File „ShoppingCart.csv existiert, dazu muss es noch auf semantische Korrektheit (dazu gehört auch: überprüfen, ob das Produkt noch verfügbar ist, wenn bereits verkauft: User informieren) überprüft werden. Bei Start der App wird der Warenkorb mit den im File korrekt vorhandenen Produkten befüllt. Bei Änderung des Warenkorbes wird das File automatisch upgedatet.

### US3 App Logic anpassen

Funktionen hinzufügen: //TODO: genauen Ort spezifizieren

void addToShoppingCart(int id) – die Warenkorb-file-Verwaltung muss benachrichtigt werden

void removeFromShoppingCart(int id) - die Warenkorb-file-Verwaltung muss benachrichtigt werden

ShoppingCartCard getShoppingCart() – der Warenkorb wird aus dem File ausgelesen und zurückgegeben

Entity ShoppingCartCard { List<ShoppingCartCardEntry { PlanetCard, NumberOfPieces }> } hinzufügen //weitere attribute sind offen

