Cassandra Corrodi, Dennis Schäppi

TBZ  28.01.2017

M151

Inhaltsverzeichnis

[Ziel dieses Moduls & Anforderungen 2](#_Toc504498939)

[Adminseite 2](#_Toc504498940)

[Userseite 2](#_Toc504498941)

[Projekt Lagerverwaltung 2](#_Toc504498942)

[Realisation Admin- und Userpage 2](#_Toc504498943)

[Tier 1 2](#_Toc504498944)

[Tier 2 3](#_Toc504498945)

[Menge aktualisieren 4](#_Toc504498946)

[Datenbank/Funktionen 4](#_Toc504498947)

[Tier 3 4](#_Toc504498948)

[Tier 4 4](#_Toc504498949)

[ERM 5](#_Toc504498950)

[Version 1 5](#_Toc504498951)

[Architektur 6](#_Toc504498952)

[Testprotokoll 6](#_Toc504498953)

[Reflexion 7](#_Toc504498954)

[Dennis Schäppi 7](#_Toc504498955)

[Cassandra Corrodi 7](#_Toc504498956)

Modul 151

# Ziel dieses Moduls & Anforderungen

Das Ziel dieses Moduls ist es eine Web-Applikation zu implementieren, bei der eine Datenbank eingebunden wurde. In dieser Datenbank sollte unter anderem auch die verschiedenen Berechtigungen implementiert sein.

Folgende Punkte sollten bei der Implementation sollten beachtet werden:

## Adminseite

* Verschlüsseltes Login mit Session-Überprüfung
* Benutzerverwaltung mit verschiedenen Rechten
* Online-Administration der Websiteinhalte
* Formularüberprüfung

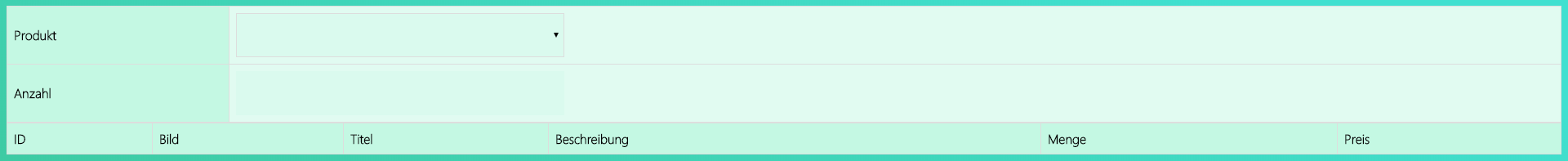
## Userseite

* Navigation funktionsgerecht
* Sortieren möglich
* Usability

# Projekt Lagerverwaltung

Die Idee welche in diesem Modul umgesetzt wird ist eine Lagerverwaltung. Diese sollte dazu dienen, dass Mitarbeitende in einem Geschäft die Übersicht über alle Produkte haben. Es sollte ersichtlich sein was für Produkte vorhanden sind, wie diese heissen und wie gross der Bestand im Lager ist.

Zudem sollten berechtigte Mitarbeiter die Attribute der verschiedenen Waren ändern können.



Hier kann man ein Produkt hinzufügen

# Realisation Admin- und Userpage

## Tier 1

Bevor wir die Funktionen in die Seiten implementiert haben, haben wir die einzelnen Seiten im HTML-Format erstellt. Dies sind unsere Presentation Layer:

|  |  |  |
| --- | --- | --- |
| **Name File** | **Inhalt** | **Nutzen** |
| Login.html | Feld Benutzername und Passwort und Login-Button. | Diese Seite bezweckt das Login des Benutzers in sein eigenes Konto. |
| List.html | Liste mit allen Bestellungen des Benutzers, Button für das Erfassen einer neuen Bestellung und Logout-Button. | Diese Seite ist für die Verwaltung, Aktualisierung und Bestellungsprozess zuständig |
| Addpurchase.html | Liste mit allen Produkten inklusive Auswahl-Button, Anzeige des ausgewählten Produktes inklusive Änderung der Anzahl und Erfassen-Button. | Diese Seite erfasst eine neue Bestellung eines Benutzers. |

## Tier 2

Als nächstes haben wir den Business Layer implementiert. Dies sind hauptsächlich die Funktionen, welche für die Eingabeüberprüfung und das Aufrufen des ODBCs zuständig ist. Dies sind unsere Business Layer:

|  |  |
| --- | --- |
| **Name File** | **Nutzen** |
| addpurchasebl.php | Ruft die Funktion «addpurchase» auf und fügt eine neue Bestellung in die Datenbank. |
| check\_session.php | Überprüft, ob eine Session vorhanden ist, bzw. der Benutzer eingeloggt ist. |
| checklogin.php | Überprüft die Login-Daten des Benutzers, existierender Benutername und Passwort, und leitet den Benutzer dementsprechend weiter. |
| db.php | Stellt die ODBC-Verbindung her. |
| getcurrentstatus.php | Ruft die Funktion «getcurrentstatus» auf und liefert den Status einer bestimmten Bestellung zurück. |
| getlager.php | Ruft die Funktion «getlager» auf und liefert die Anzahl verfügbare Menge eines bestimmten Produktes zurück. |
| getproductbyid.php | Ruft die Funktion «getproductbyid» auf und liefert die Informationen eines bestimmten Produktes zurück. |
| getproducts.php | Ruft die Funktion «getproducts» auf und liefert die Informationen aller Produkte zurück. |
| getpurchases.php | Ruft die Funktion «getpurchases» auf und liefert die Informationen aller Bestellungen eines bestimmten Benutzers zurück. |
| getstatus.php | Ruft die Funktion «getstatus» auf und liefert die Informationen aller Status zurück. |
| logout.php | Löscht alle Session-Daten und Loggt den Benutzer somit aus, welcher zurück auf die Login-Seite weitergeleitet wird. |
| updatepurchase.php | Ruft die Funktion «updatepurchases» auf und aktualisiert die Daten einer bestimmten Bestellung eines bestimmten Benutzers. |
| updatequantity.php | Ruft die Funktion «updatequantity» auf und aktualisiert die Menge eines bestimmten Produktes. Diese Aktualisierung erfolgt durch eine mehrstufige Überprüfung. |

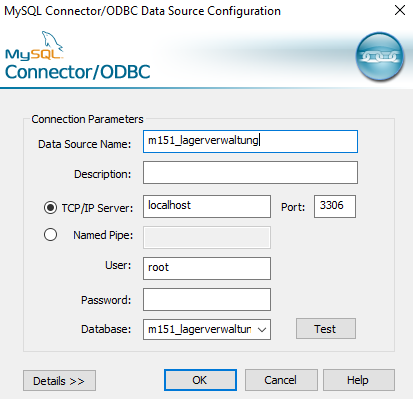
### Menge aktualisieren

Der Lagerbestand eines Produktes wird nur dann aktualisiert, wenn die Anzahl der Produkte subtrahiert von der Menge des Produktes keine negative Zahl ergibt. Wenn der Status auf «bezahlt» gesetzt wurde, dann kann der Benutzer den Status und die Anzahl nicht mehr ändern.

# Datenbank/Funktionen

## Tier 3

Als dritter Layer kommt die Konfiguration für die ODBC-Datenbankverbindung. Diese haben wir so konfiguriert:



## Tier 4

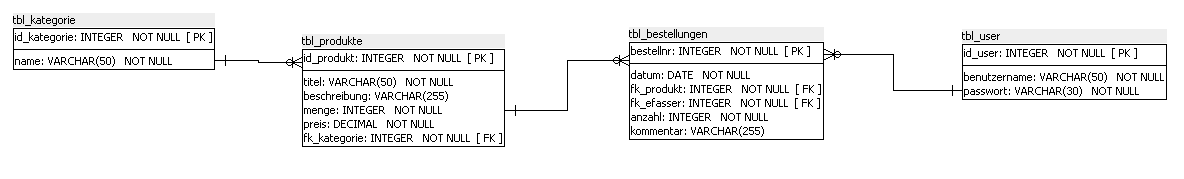
Beim zweitletzten Layer haben wir alle Funktionsaufrufe für alle Datenbankabfragen erstellt. Diese heissen Stored Procedures. Dies sind unsere Funktionen:

|  |  |
| --- | --- |
| **Aufruf** | **Abfrage** |
| addpurchase (IN `$datum` DATE, IN `$fk\_produkt` INT, IN `$fk\_erfasser` INT, IN `$anzahl` INT) | INSERT INTO tbl\_bestellungen(datum, fk\_produkt, fk\_erfasser, anzahl, fk\_status, kommentar)  VALUES ($datum, $fk\_produkt, $fk\_erfasser, $anzahl, 1, '-') |
| getcurrentstatus (IN `$id` INT) | SELECT fk\_status FROM tbl\_bestellungen WHERE bestellnr = $id |
| getlager (IN `$id` INT) | SELECT menge FROM tbl\_produkte WHERE id\_produkt = $id |
| getproductbyid (IN `$id` INT) | SELECT id\_produkt, titel, beschreibung, menge, preis, link FROM tbl\_produkte WHERE id\_produkt = $id |
| getproducts () | SELECT id\_produkt, titel, beschreibung, menge, preis, link FROM tbl\_produkte |
| getpurchases (IN `$userid` INT) | SELECT b.bestellnr, b.datum, b.fk\_produkt, b.fk\_erfasser, b.anzahl, b.kommentar, p.titel, b.fk\_status FROM tbl\_bestellungen b INNER JOIN tbl\_produkte p ON p.id\_produkt = b.fk\_produkt WHERE fk\_erfasser = $userid  ORDER BY b.datum DESC |
| getstatus () | SELECT id\_status, status FROM tbl\_status ORDER BY id\_status |
| login (IN `$username` VARCHAR(50)) | SELECT id\_user, benutzername, passwort FROM tbl\_user WHERE benutzername = $username |
| updatepurchases (IN `$id` INT, IN `$comment` VARCHAR(255), IN `$anzahl` INT, IN `$status` INT) | UPDATE tbl\_bestellungen SET anzahl = $anzahl, kommentar = $comment, fk\_status = $status WHERE bestellnr = $id |
| updatequantity (IN `$id` INT, IN `$menge` INT) | UPDATE tbl\_produkte SET menge = $menge WHERE id\_produkt = $id |

## Tier 5 (ERM)

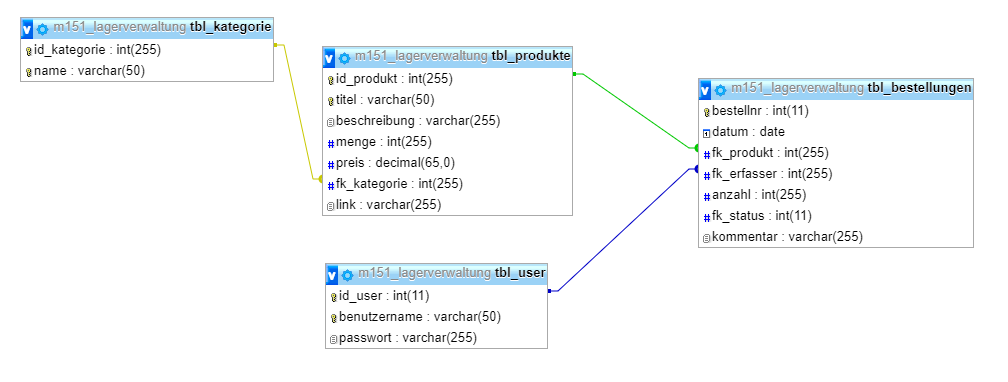
Unser Datenbankdesign hat einige Versionen durchlebt. Zuerst hatten wir einige Dinge, welche wir nicht miteingeplant haben, wie z.B. der Status.

### Version 1



### Version 2

In der ersten Version hatten wir noch keinen Status, weil sich erst im Verlauf des Projekts herausgestellt hat, dass der Status wichtig ist, um darzustellen, ob eine Bestellung abgeschlossen wurde oder nicht. Somit sieht nun unser ERM so aus:

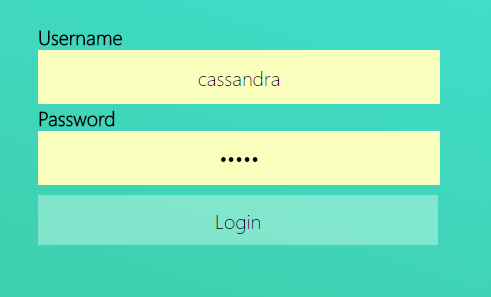


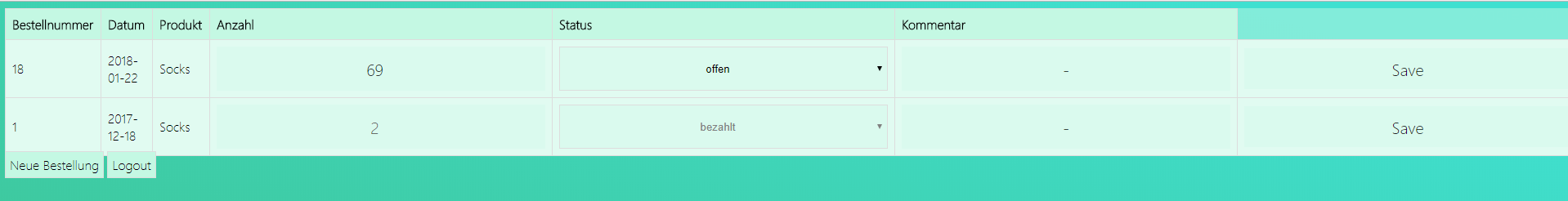
# Architektur

Wir haben unser Projekt mit einer 5-Tier-Architektur realisiert. Oben beim Realisieren haben wir die einzelnen Layer schon ausführlich dargestellt, jedoch haben wir nochmals einen Überblick zur Kontrolle, ob es sich auch wirklich um eine 5-Tier-Architektur handelt.

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer Nr.** | **Bezeichnung** | **Beschreibung** | **Status** |
| 1 | Presentation | HTML-Dateien welche alles grafisch ausgeben | Vorhanden |
| 2 | Business | Datenüberprüfung und Funktionsaufruf | Vorhanden |
| 3 | ODBC | ODBC-Verbindung zur DB | Vorhanden |
| 4 | Stored Procedures | Alle Funktionen der Datenbankabfragen | Vorhanden |
| 5 | Data |  |  |

# Ergebnis







# Nicht implementierte Funktionen

# Testprotokoll

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name des Tests** | **Eingabe** | **Erwartetes Ergebnis** | **Status** |
| N1 | Login | Benutzername: cassandra Passwort: panda | Login erfolgreich 🡺 Bestellliste wird aufgerufen | OK |
| N2 | Kommentar ändern | test | Kommentar wird in der DB aktualisiert | OK |
| N3 | Anzahl ändern | 3 | Anzahl wird in der DB aktualisiert | OK |
| N4 | Status auf bezahlt | Select Auswahl auf «bezahlt» | Felder Anzahl & Status werden deaktiviert und es wird die Menge vom Produkt in der DB abgezogen | OK |
| N5 | Produkt bei Erfassung auswählen | Auf «Auswählen» klicken | Produkt wird ausgewählt: Oben erscheint die ID, Name und Preis in der Abhängigkeit der Anzahl | OK |
| N6 | Anzahl bei Erfassung ändern | 2 | Preis bei Produkt wird neu berechnet | OK |
| N7 | Produkt erfassen | Auf «Produkt erfassen» klicken | Produkt wird erfasst und Benutzer wird zur Bestellliste weitergeleitet | OK |
| N8 | Logout | Auf «Logout» klicken | Benutzer wird ausgeloggt | OK |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name des Tests** | **Eingabe** | **Erwartetes Ergebnis** | **Status** |
| E1 | Falsches Login | Benutzername: cassandra Passwort: 1234 | Benutzer wird nicht weitergeleitet | OK |
| E2 | Kommentar 256 Zeichen | Text mit 256 Zeichen | Kommentarfeld grenzt Text auf 255 Zeichen | OK |
| E3 | Anzahl 0 und negativ | -2 | Feld lässt 0 und negativ nicht zu | OK |
| E4 | Anzahl bei Erfassung 0 und negativ | -2 | Feld lässt 0 und negativ nicht zu | OK |
| E5 | Ausgeloggt Liste aufrufen | http://localhost/m151/ Lagerverwaltung/src/ app/list.html | Benutzer wird wieder zur Login-Seite weitergeleitet | OK |
| E6 | Ausgeloggt Erfassung aufrufen | http://localhost/m151/ Lagerverwaltung/src/ app/addpurchase.html | Benutzer wird wieder zur Login-Seite weitergeleitet | OK |
| E7 | Status ändern bei zu wenig Lagerbestand | Eingabe: 69; Lagerbestand: 4 | Applikation lässt den Status nicht ändern | OK |

# Reflexion

## Dennis Schäppi

Bla Bla

## Cassandra Corrodi

Ich fand dieses Modul gut, denn neben dem eigentlichen Ziel habe ich auch noch wieder meine PHP Kenntnissen auffrischen können. Zudem habe ich auch nach langer Zeit wieder mit XAMPP und PHPMyAdmin gearbeitet. Im Geschäft arbeite ich nicht wirklich mit dem, aber es ist sicher gut, dass ich es wieder mal gemacht habe, ehe ich es komplett verlernt hätte.

Zudem habe ich gelernt, wie man eine Datenbank in ein Webprojekt einbindet, ich habe schon einmal eine Datenbankeinbindung gemacht, aber mit C#, das ist etwas ganz anderes als mit PHP und einer SQL Datenbank. Aber ich finde es selbstverständlich gut wenn ich in so vielen Bereichen wie möglich Erfahrungen sammeln kann.

Im grossen Ganzen fand ich das Modul sehr lehrreich, vor allem weil es so ziemlich das Gegenteil davon ist, was ich im Geschäft machen muss.