**Web**

**Technologie 3  
Case Study**

**Autor**Angela Scherer  
Olivier Walther  
Arash Afshar

**Veröffentlicht**   
22. März 2017

Inhalt

[Aufgabenstellung 4](#_Toc477866614)

[Situationsanalyse 4](#_Toc477866615)

[Ziele 5](#_Toc477866616)

[Mussziele 5](#_Toc477866617)

[Wunschziele 6](#_Toc477866618)

[Use Case Diagrams 6](#_Toc477866619)

[Lösungsvariante 6](#_Toc477866620)

[Twitter Bootstrap 6](#_Toc477866621)

[Programmiersprache 6](#_Toc477866622)

[Framework 6](#_Toc477866623)

[Datenbank 7](#_Toc477866624)

[Versionsverwaltungsprogramm 7](#_Toc477866625)

[Integrated Development Environment 7](#_Toc477866626)

[Übersicht Architektur und Komponenten 8](#_Toc477866627)

[Architektur der Applikation 9](#_Toc477866628)

[UML 10](#_Toc477866629)

[Backend-Controller 10](#_Toc477866630)

[Backend-Services 11](#_Toc477866631)

[Frontend-Controller 12](#_Toc477866632)

[Frontend-Services 13](#_Toc477866633)

[ERM 14](#_Toc477866634)

[Meilensteine 15](#_Toc477866635)

[Projektplanung 16](#_Toc477866636)

[Architektur & Komponenten bestimmen 16](#_Toc477866637)

[GIT Hub Projekt einrichten 17](#_Toc477866638)

[IDE einrichten 17](#_Toc477866639)

[DB Design 17](#_Toc477866640)

[Realisierungsphase 1 18](#_Toc477866641)

[Realisierungsphase 2 18](#_Toc477866642)

[Realisierungsphase 3 19](#_Toc477866643)

[Realisierungsphase 4 19](#_Toc477866644)

[Realisierungsphase 5 20](#_Toc477866645)

[Testen 21](#_Toc477866646)

[Bugfixing 22](#_Toc477866647)

[Zwischenstand 22](#_Toc477866648)

[PowerPoint-Präsentation erstellen 22](#_Toc477866649)

[Dokumentation 23](#_Toc477866650)

[Abgabe & Präsentation 23](#_Toc477866651)

[Total 24](#_Toc477866652)

[Positive und negative Aspekte der Arbeit 24](#_Toc477866653)

[Lessons Learned 24](#_Toc477866654)

[Arash Afshar 24](#_Toc477866655)

[Angela... 24](#_Toc477866656)

[Oliver… 24](#_Toc477866657)

[Quellen 24](#_Toc477866658)

[Projekt (Web Case Study) 24](#_Toc477866659)

# Aufgabenstellung

## Situationsanalyse

## Ziele

### Mussziele

|  |  |  |
| --- | --- | --- |
| ID | Prio | Beschreibung |
| FA\_1.1 | 1 | Im Web-Shop sollen Artikel über das Internet angeboten und verkauft werden. Die Aspekte der Bezahlung und der Auslieferung werden der Einfachheit halber nicht berücksichtigt. |
| FA\_1.2 | 1 | Die Artikel im Web-Shop sind in Produktkategorien in 2 Levels strukturiert |
| FA\_1.3 | 1 | Die Produktkategorien und die Zuordnung der Produkte werden in der DB konfiguriert, dadurch ist der Web-Shop generisch für die verschiedensten Produkte verwendbar |
| FA\_1.4 | 1 | Produkte werden in der DB konfiguriert. Sie haben minimal die folgenden Attribute Produkt-ID (eindeutig), Bezeichnung (max. 200 Zeichen), Beschreibung (max. 2000 Zeichen), 0-5 Bilder, Status (active, hidden) |
| FA\_1.5 | 2 | Zusätzlich werden zu jedem Produkt noch die folgenden Angaben verwaltet: Lagerbestand, Nachbestellung (Lieferdatum und Bestellmenge) |
| FA\_1.6 | 1 | Mittels geeigneter Navigation kann der Kunde durch Produktkategorien und Produkte navigieren |
| FA\_1.7 | 1 | Der Kunde kann Produkte seinem Warenkorb hinzufügen. |
| FA\_1.8 | 1 | Der Kunde kann seinen Warenkorb anzeigen lassen (Auflistung der Produkte inkl. Preis und Menge) und editieren (Menge ändern, Produkte herausnehmen) |
| FA\_1.9 | 2 | Sobald ein Kunde Produkte im Warenkorb hat, kann er "zur Kasse gehen". Dort wird ihm die Bestellung nochmals angezeigt, er muss seine Adresse und Email eingeben und kann die Bestellung auslösen. |
| FA\_1.10 | 2 | Alle Angaben einer Bestellung (Warenkorb, Kundendaten, Bestelldatum) werden in der DB gespeichert |
| FA\_1.11 | 2 | Anzeigen einer Liste aller Produkte. Durch Anklicken des Produktes gelangt man in die Detailansicht welche alle spezifizierten Attr. des ausgewählten Produktes darstellt (Admin Funktion) |
| FA\_1.12 | 2 | Anzeigen einer Liste aller Bestellung. Durch Anklicken einer Bestellung gelangt man in die Detailansicht (Admin Funktion) |
| FA\_2.1 | 1 | Die Administration von Produkten und Produktkategorien (anlegen, editieren, löschen) soll auf einer entsprechenden Admin Seite möglich sein. |
| FA\_2.2 | 1 | Ein Kunde kann sich im Web-Shop registrieren. Nach dem Login mit der Email und Passwort muss er bei einer Bestellung seine Adresse nicht mehr eingeben. |
| FA\_2.3 | 2 | Das Passwort darf nicht als Klartext in der DB abgelegt werden. |
| FA\_2.4 | 2 | Ein registrierter Kunde kann seine Daten einsehen und editieren. |
| FA\_3.1 | 1 | Der Shop wird in der Basiswährung CHF betrieben. Die Shop Besucher sollen auf Wunsch (Popup oder ähnliches) eine andere Anzeigewährung auswählen können. (US$, € und JPY) |
| FA\_3.2 | 1 | Der Kurs soll jeweils (mindestens) Tagesaktuell von der Webseite <http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml> sein. Alternativ ist auch ein anderer Onlinekurslieferant erlaubt. |
| FA\_3.3 | 2 | Die Bestellung wird in Basswährung ausgeführt, es soll aber auch die Ausgewählte Währung des Kunden, sowie der Kurs zum Zeitpunkt der Bestellung ersichtlich sein. |
| NFA\_1.1 | 1 | Ein User mit minimalen Internetkenntnissen muss den Web-Shop intuitiv bedienen können. (Benutzbarkeit) |
| NFA\_1.2 | 2 | Der Web-Shop soll unter Linux und Windows lauffähig sein (Übertragbarkeit) |
| NFA\_1.3 | 2 | Die Antwortzeit des Web-Shops soll unter 3 Sek. liegen (Zeitverhalten) |
| NFA\_1.4 | 2 | Der Shop muss gegen SQL Injections immun sein |
| NFA\_1.5 | 3 | Der Shop muss auch gegen XSS Angriffe abgesichert sein |

### Wunschziele

## Use Case Diagrams

# Lösungsvariante

## Twitter Bootstrap

Eine Weblösung kommt heut zu Tage ohne Bootstrap CSS nicht mehr aus, die meisten Firmen verwenden den Bootstrap. Da auf das Design der Seite nicht viel Wert gesetzt wird, werden wir Bootstrap Plain einsetzen ohne eigene Customization vorzunehmen.

## Programmiersprache

Für die Umsetzung des Auftrags haben wir drei Entwicklungswerkzeuge PHP, JAVA, SCALA näher in Betracht gezogen und miteinander verglichen.

|  |  |  |  |
| --- | --- | --- | --- |
|  | PHP | JAVA | SCALA |
| Einfache Programmiersprache | 5 | 1 | 0 |
| Einfache Webserver Konfiguration | 2 | 3 | 3 |
| Einfacher Debugging-Support | 2 | 5 | 5 |
| OOP-Support | 3 | 5 | 5 |
| Type-Safe | 0 | 5 | 5 |
| IDEA Compiler-Support | 3 | 5 | 5 |
| Saubere Trennung zwischen GUI und Logik | 5 | 5 | 5 |
| Robuste Frameworks vorhanden | 5 | 5 | 5 |
| Potenzielle Entwickler auf dem Markt | 5 | 3 | 0 |
| Geringe Serverauslastung | 5 | 2 | 2 |
| Geringe Hosting Kosten | 5 | 1 | 1 |
| Total | **40** | **40** | **36** |

Für die Implementierung werden wir PHP verwenden, weil die meisten Entwickler sich mit PHP mehr oder weniger auskennen. Zudem gibt es viele Web Hoster welche PHP Out of the Box unterstützen.

## Framework

|  |  |  |  |
| --- | --- | --- | --- |
|  | Symfony | Laravel | CodeIgniter |
| Ausführliche Dokumentation | 5 | 2 | 3 |
| Bereite Community | 5 | 3 | 2 |
| Vorkenntnisse im Team vorhanden | 3 | 0 | 0 |
| IDE Support | 4 | 3 | 2 |
| Total | **17** | **8** | **7** |

Da für die Umsetzung die Sprache PHP gewählt wurde kommen nicht mehr viele Frameworks zu Frage. Symfony ist eines der bekanntesten und hat eine bereite und ausgereifte Community was sehr wichtig ist für neu Einsteiger.

Direkter Link Symfony Framework: <https://symfony.com/>

## Datenbank

|  |  |  |  |
| --- | --- | --- | --- |
|  | MS-SQL | MySQL | Postgre-SQL |
| Opensource | 0 | 5 | 5 |
| User-Defined Data-Ttypes | 1 | 2 | 5 |
| User-Defined Domains | 2 | 0 | 3 |
| Plattform unabhängig | 0 | 3 | 5 |
| Geringe Aufwand für Initialisierung | 1 | 5 | 1 |
| Vorkenntnisse im Team vorhanden | 2 | 5 | 1 |
| GUI Oberfläche von Haus aus unterstützt | 0 | 5 | 0 |
| Total | **6** | **25** | **20** |

Als Datenbank werden wir MySQL verwenden mit PHP-My-Adamin.

## Versionsverwaltungsprogramm

|  |  |  |  |
| --- | --- | --- | --- |
|  | GIT | TSF | SVN |
| Intelligentes System | 5 | 5 | 2 |
| Verbreitet | 5 | 2 | 5 |
| Bereite Community | 5 | 3 | 3 |
| Plattform unabhängig | 5 | 3 | 5 |
| Vorkenntnisse im Team vorhanden | 5 | 0 | 3 |
| Total | **25** | **13** | **18** |

Für die Realisierung des Projektes werden wir GIT für ein verteiltes Version Verwaltung einsetzen. Als GIT Repository werden wir GitHub verwenden, das Projekt wird unter dem Namen **ibz-it5-casestudy-webshop** veröffentlicht.

Direkter Link: <https://github.com/AruPersia/ibz-it5-casestudy-webshop>

## Integrated Development Environment

Als IDE werden wir IntelliJ als Entwicklungsumgebung verwenden welche eine gute Basis für PHP und Symfony Framework anbietet.

Direkter Link IntelliJ: <https://www.jetbrains.com/idea/>

# Übersicht Architektur und Komponenten

|  |  |  |  |
| --- | --- | --- | --- |
| **Bildergebnis für php**  **PHP 7.1.1** | **Bildergebnis für symfony**  **Symfony 3.1.8** | **Bildergebnis für MySql**  **MySQL 5.0.12** | **Bildergebnis für doctrine logo**  **Doctrine 2.5** |
| **Bildergebnis für phpunit test logo**  **PHP Unit 5.5.7** | **Bildergebnis für twig template logo**  **TWIG 2.0** | **Bildergebnis für git logo**  **GIT 2.12.1** | **Bildergebnis für xampp**  **XAMPP 5.6.30** |
| **Bildergebnis für intellij phpstorm logo**  **DIE 2016.3** | **Bildergebnis für json logo**  **-** | **Bildergebnis für github**  **-** | **Bildergebnis für apache**  **Apache 2.4.5** |
| **Bildergebnis für html5**  **HTML 5** | **Bildergebnis für css3**  **CSS 3** | **Bildergebnis für jquery**  **jQuery 3.1.1** | **Bildergebnis für bootstrap**  **Bootstrap 4.0** |

## Architektur der Applikation

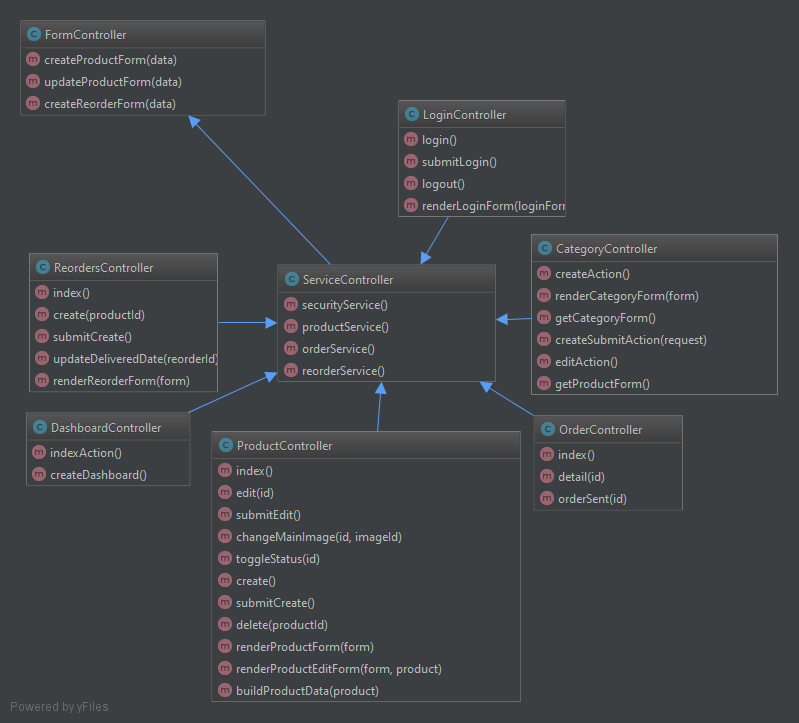
Die Architektur der Software basiert auf MVC Modell (Model, View, Controller) und wird mit Hilfe des Symfony Frameworks (symfony.com) Version 3 umgesetzt. Symfony ist ein in PHP geschriebenes Webframework welches für den Zugriff auf die Datenbank das Framework Doctrine (doctrine-project.org) unterstützt. Das Doctrine Project (doctrine-project.org) ist ein Framework, dass die Möglichkeit der objektrelationalen Abbildung (ORM) sowie eine Datenbankabstraktionsschicht (DBAL) für PHP ab Version 5.3 bereitstellt. Dadurch ist ein vereinfachter Zugriff auf verschiedene Datenbanktypen (z.B. MySQL oder Microsoft SQL Server) möglich, als dies mit reinem PHP möglich wäre. Doctrine orientiert sich an Hibernate, einem ähnlichen Projekt für Java.

Zwischen Controller und Repository, bzw. DBAL stehen Services zur Verfügung welche das Businesslogik der Applikation umsetzen. Ein Beispiel zeigt die untenstehende Grafik:

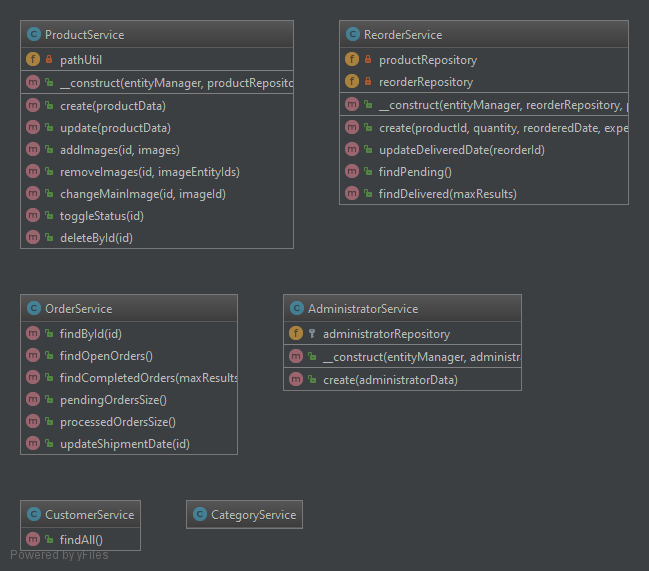


## UML

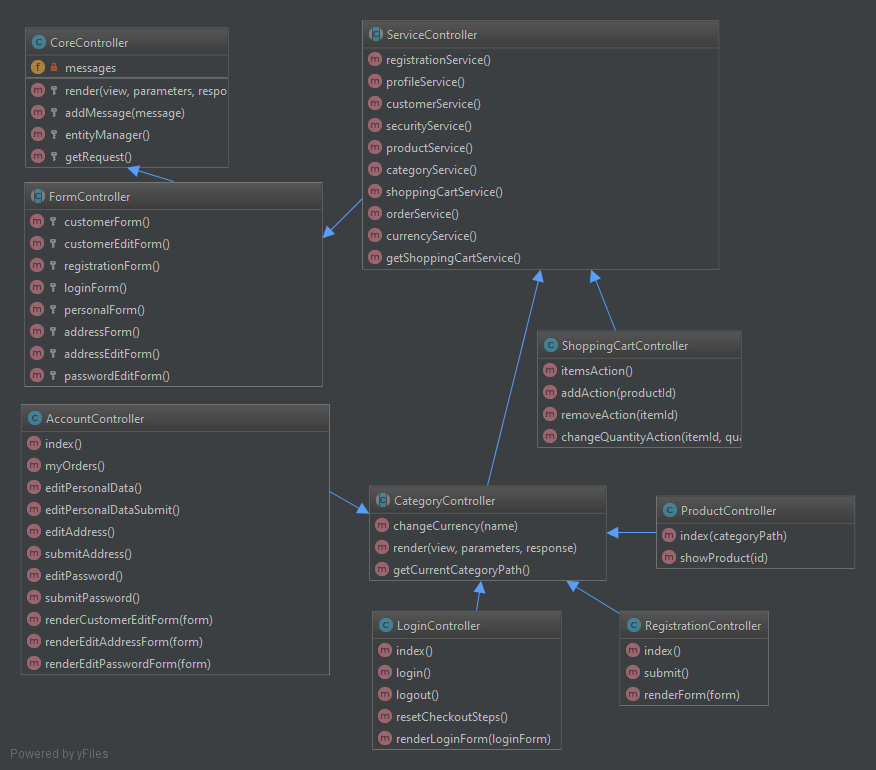
### Backend-Controller



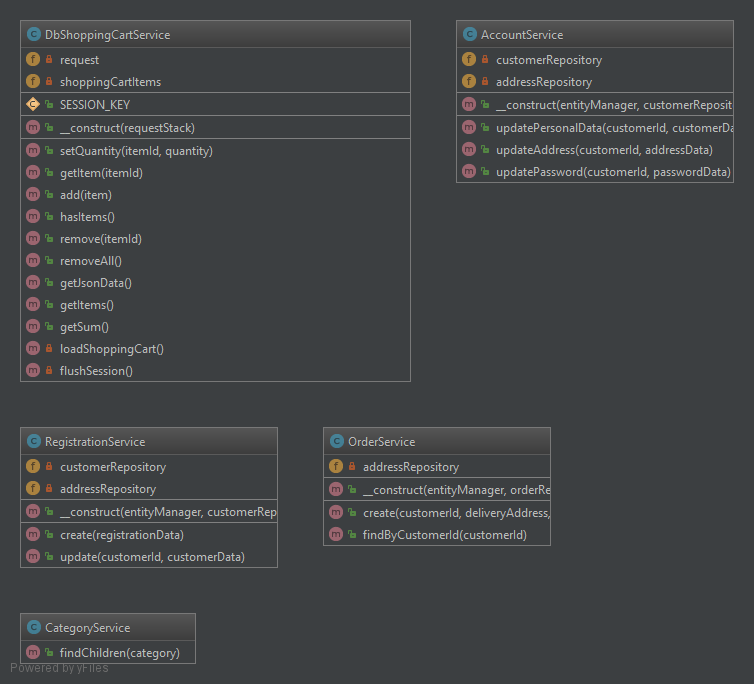
### Backend-Services



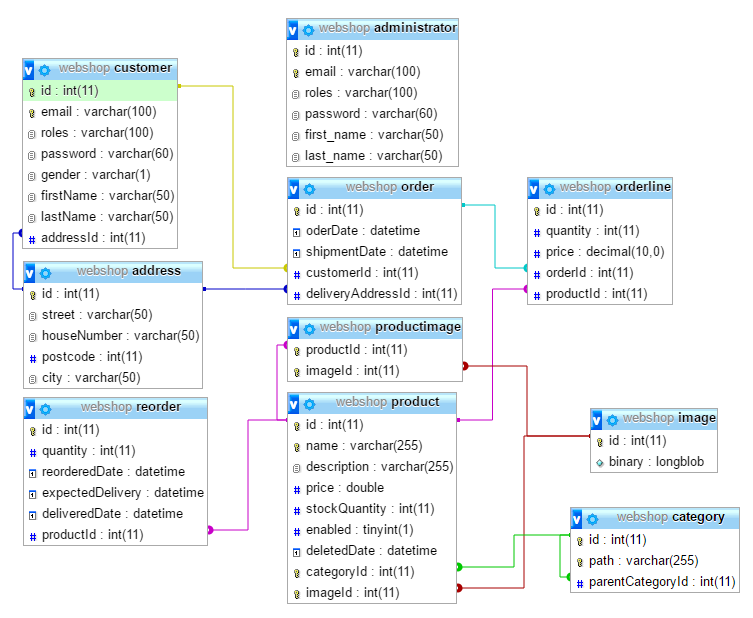
### Frontend-Controller



### Frontend-Services



## ERM



# Meilensteine



### Projektplanung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| * Vorgehensmodell * Definition der Arbeitspakete (ID, Beschreibung, Zeitaufwand, Abhängigkeiten) * Zeitplanung, es soll für jede Woche ersichtlich sein, wer wie lange an welchen Arbeitspaketen arbeitet * Zu erreichende Milestones | AAF | 8 | 6 | 2 |  |

### Architektur & Komponenten bestimmen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| * Beschreibung der SW Architektur des Web-Shops * Welche Technologien und Komponenten eingesetzt werden (Datenbank, Programmiersprache, Webserver, CMS ...) inkl. Version | AAF | 4 | 2 | 2 |  |

### GIT Hub Projekt einrichten

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| * GIT Hub Projekt eröffnen | AAF | 2 | 2 | 0 |  |

### IDE einrichten

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| * IntelliJ einrichten, GIT Hub Zugriff testen | AAF | 4 | 8 | -4 | Es gab schwierigtkeiten X-Debug zu aktivieren um die Applikation debuggen zu können. |
| * IntelliJ einrichten, GIT Hub Zugriff testen | ASC | 4 | 3 | 1 |  |
| * IntelliJ einrichten, GIT Hub Zugriff testen | OWA | 4 | 3 | 1 |  |

### DB Design

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| Datenbank Schema erstellen | AAF | 8 | 4 | 4 | Aufgrund bisherigen Erfahrungen ging das Schema erstellen recht schnell |

### Realisierungsphase 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| DTOs erstellen   * FA\_1.4 * FA\_1.5 * FA\_3.1 (berücksichtigen) * FA\_3.2 (berücksichtigen) | AAF | 8 | 8 |  |  |
| DAOs erstellen | AAF | 8 | 12 |  | Hat doch mehr Arbeit gegeben als gedacht. |
| Forms erstellen | AAF | 8 | 8 |  |  |

### Realisierungsphase 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| Routings definieren | AAF | 4 | 4 |  |  |
| Controllers definieren (Vorbereitung) | AAF | 8 | 8 |  |  |
| Produkt erfassen (Vorbereitung)   * FA\_1.2 * FA\_1.3 | AAF | 8 | 8 |  |  |
| Registration & Login   * FA\_2.2 * FA\_2.3 | AAF | 8 | 8 |  |  |

### Realisierungsphase 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| Navigation erstellen   * FA\_1.6 |  | 4 |  |  |  |
| Warenkorb implementieren   * FA\_1.7 * FA\_1.8 |  | 8 |  |  |  |
| Zahlungsabschluss (Kasse/Bestellung) implementieren   * FA\_1.9 * FA\_1.10 |  | 8 |  |  |  |

### Realisierungsphase 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| Registrierter Kunde kann sein Profil bearbeiten   * FA\_2.4 |  | 8 |  |  |  |
| Admin Funktion: Produkten (anlegen, editieren, löschen)   * FA\_2.1 |  | 8 |  |  |  |

### Realisierungsphase 5

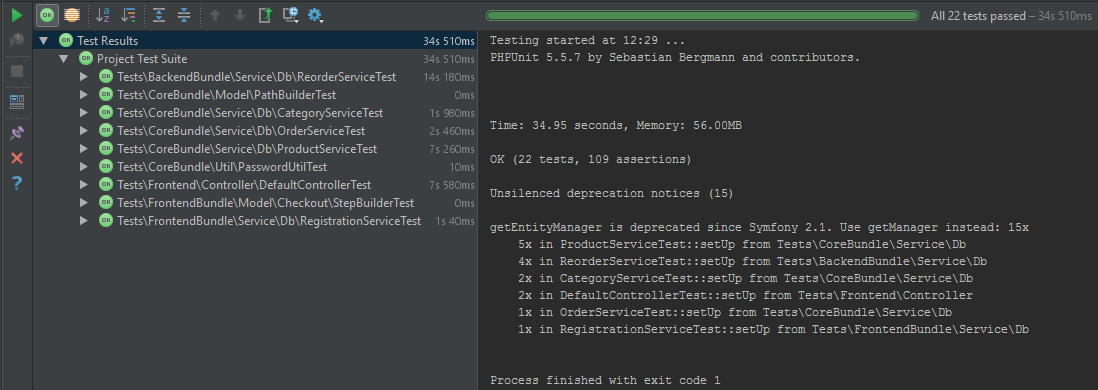
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| Preis in andere Währung anzeigen   * FA\_3.1 * FA\_3.2 * FA\_3.3 |  | 8 |  |  |  |
| Admin Funktion: Anzeigen alle Produkte und Bestellungen   * FA\_1.11 * FA\_1.12 |  | 8 |  |  |  |

### Testen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| Webshop testen und prüfen ob alle Ziele erreicht wurden | AAF | 4 | 12 | 8 | Immer wieder wurden Fehler entdeckt welche beseitigt werden müssten, für das nächste Projekt werden wir mehr Zeit für das Testen einberechnen |

#### PHP Unit Tests

Die automatische PHP Unit Tests waren sehr hilfreich. Bei jeder Applikationsänderung wurden die Tests durchgeführt um sicherzustellen, dass eine Änderung nicht ungewollte gewisse Applikationsteile kaputt macht. Je früher Fehler entdeckt werden desto einfach lassen sie sich beheben.



#### Nicht umgesetzte Ziele

Die untenstehenden Ziele konnten aus Zeitgründen nicht fertig implementiert werden:

* FA\_3.3

### Bugfixing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Gefundene Bugs | Wer | Geplant | Geleistet | Delta | Erklärung |
|  | AAF ASC OWA | 80 | 90 | 10 | Siehe vorherige Abschnitt «Testen» |

### Zwischenstand

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| * Aufzeigen des Soll/Ist Vergleiches von Zeitplanung und Milestones |  | 2 |  |  |  |

### PowerPoint-Präsentation erstellen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| * Vorbereitung einer PowerPoint-Präsentation * Planung & Einteilung wer präsentiert welche Folien |  | 4 |  |  |  |

### Dokumentation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| * Projektplanung (siehe oben) inkl. Soll/Ist Vergleiche von Zeitplanung und Milestones * Zusammenfassung der erreichten Resultate * SW Design (UML Diagramme, Beschreibung der wichtigsten Funktionen) * Datenbank Design (ER - Diagramme) * Positive und negative Aspekte der Arbeit * Lessons Learned | AAF  ASC  OWA | 24 | 18 | 6 | Die Schätzung für die Dokumentation war ein bisschen zu hoch für dieser Art von Projekt, da genug Vorkenntnisse und Erfahrungen vorhanden waren. |

### Abgabe & Präsentation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| * Applikation im Quelltext, inklusive Buildframework als ZIP Datei * Binärversion + Datenbankdump der Applikation * URL zum lauffähigen Projekt, so dass der Shop getestet werden kann. * Der Shop muss mindestens noch für eine Woche nach der Präsentation in Betrieb sein |  | 4 |  |  |  |

### Total

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aktivität | Wer | Geplant | Geleistet | Delta | Erklärung |
| Total | AAF  ASC  OWA | 256h |  |  |  |

# Positive und negative Aspekte der Arbeit

Die Aufteilung hat super geklappte dank der Aufteilung in MVC-Modell, Services und Repository Prinzipien.

# Lessons Learned

## Arash Afshar

Alle Technologien waren mir mehr oder weniger bekannt. Am meisten konnte ich Erfahrungen im Bereich Projektleitung, Aufbau von Architektur und Design sammeln.

## Angela Scherer

………………………………

## Olivier Walther

………………………………

# Quellen

## Projekt (Web Case Study)

Das Projekt ist im GitHub unter dem folgenden Link «https://github.com/AruPersia/ibz-it5-casestudy-webshop» erreichbar.