Fachhochschule Salzburg

Softwareentwicklung
Architektur

vom 10. Mai 2014

Daniela Pointinger
Michael Haslauer
Stefan Winkler
Johannes Briewasser | itsb-m2013

Version 1.0 | 17. Mai 2014

Inhaltsverzeichnis

1 Einführung und Ziele

Das folgende Dokument beschreibt die Softwarearchitektur des Projekts "Bahn Zahlsystem", dass in der Übung Software-Engineering entwickelt wurde. Bei der Applikation handelt es sich um eine Neu-Entwicklung auf der Basis eines Online-Shops. Da es sich hierbei um eine Online-Applikation handelt die über unterschiedliche Wege erreicht werden soll muss ein hierfür geeignetes Architekturkonzept gewählt werden.

Das Projekt wird mit der Programmiersprache Java (Version 8) entwickelt und wird auf einem Applikationsserver (Oracle Glassfish) deployed.

2 Architekturkonzept

Das Projekt „Bahn Zahlsystem“ ist eine Online-Applikation die mehrere Möglichkeiten der Nutzung bietet. Außerdem müssen externe Drittapplikationen in die Applikation integriert werden können. Das Grundkonzept das hier verwendet werden soll ist die Schichtenarchitektur. Diese bietet den Vorteil, dass Änderungen in bestimmten Bereichen nicht die komplette Applikation beeinflussen sondern nur in der jeweiligen Schicht angepasst werden müssen. Außerdem lassen sich in diesem Fall die Komponenten der Schichten auf logischer Ebene trennen und jeder Schicht einen speziellen Aufgabenbereich zuordnen. Die Architektur umfasst dabei die 4 Schichten für:

- Domänenobjekte und Peristenz
- Business-Logik
- Benutzeroberfläche
- Anbindung Drittsysteme

Im folgenden werden die einzelnen Schichten und deren Schnittstellen genauer beschrieben.

3 Schichtenbeschreibung

Bei einem Schichtenmodell gilt, dass die einzelnen Layer des Modells nur mit den direkt an einen Layer angrenzende Schichten kommunizieren kann.

3.1 Domänenobjekte und Persistenz

In dieser Schicht sind die einzelnen Objekte und deren Metadaten definiert, die für den Einsatz in dieser Domäne nötig sind und in der Applikation verwendet werden. Die Objekte bilden die Schnittstelle zu den konkreten Daten für die Verarbeitung und Durchführung von Prozessen. Der Zugriff von der darüberliegenden Schicht auf Daten erfolgt somit nicht direkt, sondern wird über diese Objekte gekapselt. Sie bietet demnach eine reine Objektschnittstelle. Außerdem befindet sich in dieser Schicht keine weitere Logik, außer derjenigen, die für die Erstellung und Speicherung der Objekte nötig ist.
Eine Bereitstellung der Objekte bzw. der Persistierung von Änderungen in der Datenbank erfolgt nur auf Anfrage. Die darüberliegende Schicht fragt über diese Persistenzschicht an, das Objekt mit den Daten aus der Datenbank zu befüllen. Um geänderte Daten in die Datenbank zu speichern, wird von der darüberliegenden Schicht eine Methode dieser Persistenzschicht

aufgerufen.
Die Menge der Domänenobjekte und deren Metadaten ist eine nahezu vollständig definierte Menge. Änderungen in dieser Schicht ergeben weitreichendere Folgen für alle darüber angesiedelten Schichten. Außerdem ist in dieser Schicht die technische Schnittstelle zur Datenbank angesiedelt, sowie die Datenbank selbst.

3.1.1 Dienste

Folgende Aufzählung zeigt die Entitäten, die in der Persistenz-Schicht abgebildet sind:

- User
- Produkt
- Warenkorb
- Einkauf
- Log

Funktionen die diese Dienste bereitstellen sind:

- Suchen - gibt Liste<Objekt> zurück
  `public List<Objekt> search(String searchterm);`
- Erhalten - gibt Objekt zurück
  `public Objekt get(long id);`
- Erstellen eines Objekts
  `public Objekt new();`
- Speichern eines Objekts
  `public void store(Objekt obj);`
- Löschen eines Objekts
  `public void delete(Objekt obj);`

3.2 Business-Logik

Diese Schicht beinhaltet die komplette Logik die für die Abwicklung der Business-Prozesse nötig ist. Sie bildet somit die Use-Cases der Requirement-Engineering Phase ab. Die Logik greift auf die Objekte der unteren Schicht zu und führt Operationen darauf aus. Außerdem erfolgt die komplette Fehlerbehandlung dieser und der darunterliegenden Schichten in dieser Schicht. Lediglich die Benutzeroberfläche beinhaltet ein eigenes Fehlermanagement, da dieses potentiell nicht Teil der selbstentwickelten Lösung sein muss. Die Dienste der Business-Logik stehen den darüberliegenden Schichten als REST-Schnittstelle zur Verfügung die eine Datenschnittstelle via JSON/XML bietet.

3.2.1 Dienste

Benutzer registrieren:
URI: /registerUser/
JSON-Request:

```
{
        "vorname": "",
        "nachname": "",
        "gebdat": { "type": "date" },
        "email": "",
        "geschlecht": [ "m", "w" ]
}
```

JSON-Response:

```
{
        "meldungscode": [ "OK", "Fehler" ],
        "meldungstext": ""
}
```

Login:
URI: /login/
JSON-Request:

```
{
        "email": "",
        "passwort": ""
}
```

JSON-Response:
Weiterleitung auf Index-Seite

```
{
        "meldungscode": [ "OK", "Fehler" ],
        "meldungstext": ""
}
```

Suche:
URI: /search/
JSON-Request:

```
{
        "searchString": ""
}
```

JSON-Response:

```
{
        "produkte": { "type": "ArrayList<Produkt>" }
}
```

Produkt zu Warenkorb hinzufügen/entfernen:
URI: /shoppingcart/
JSON-Request:

```
{
        "warenkorb": "{ "type": "Warenkorb" }"
}
```

JSON-Response:

```
{
        "warenkorb": "{ "type": "Warenkorb" }",
        "meldungscode": [ "OK", "Fehler" ],
        "meldungstext": ""
}
```

Bezahlen:
URI: /shoppingcart/
JSON-Request:

```
{
        "warenkorb": "{ "type": "Warenkorb" }"
}
```

JSON-Response:

```
{
        "einkauf": "{ "type": "Einkauf" }",
        "meldungscode": [ "OK", "Fehler" ],
        "meldungstext": ""
}
```

Logout:
URI: /logout/
JSON-Request: -
JSON-Response:
Weiterleitung auf Login-Seite

```
{
        "meldungscode": [ "OK", "Fehler" ],
        "meldungstext": ""
}
```

3.3 Benutzeroberfläche

Die Benutzeroberfläche ist in einer eigenen Schicht gekapselt und ist somit getrennt von der Bunsiness-Logik. Sie beinhaltet alle nötigen Steuerelemente die der Benutzer braucht

um Prozesse durchzuführen. Durch die Trennung der Oberfläche können somit Anpassungen vorgenommen werden, ohne das Änderungen in den Schichten darunter nötig sind. Ob es sich bei der Benutzeroberfläche um ein Webinterface oder eine Mobile-App handelt ist damit unerheblich. Außerdem können mehrere unterschiedliche Benutzeroberflächen parallel existieren.

3.4 Anbindung Drittsysteme

Die Anbindung von Drittsystemen wird in einer eigenen Schicht gekapselt. Diese Schicht beinhaltet alle nötige Logik um ein Drittsystem nutzen zu können. Die Schnittstellen der Drittsysteme werden somit in einen eigene Schnittstelle umgewandelt die letztendlich von der Business-Logik genutzt werden. Die Business-Logik bekommt somit nichts vom Drittsystem mit, was ein Austauschen leichter möglich macht, sofern die Schnittstelle unverändert bleibt. Die Besonderheit dieser Schicht liegt darin, dass sie sich parallel zur Business-Logik schicht befindet und nicht darunter bzw. darüber. Sie kann somit ebenfalls die Domänenobjekte nutzen, allerdings werden ihre Dienste nur von der Business Logik genutzt und über eine Funktionsschnittstelle bereitgestellt.

3.5 Schichtendiagramm

In Abbildung 1 ist das Schema der einzelnen Schichten in der Architektur visualisiert.

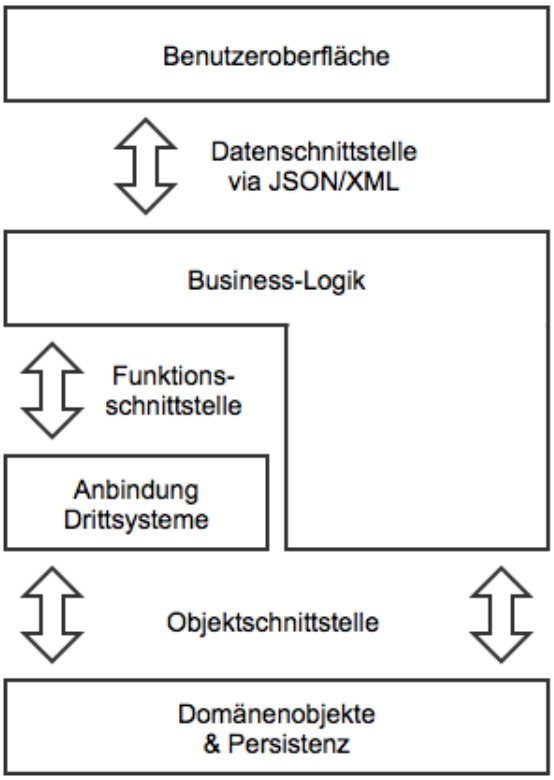

Abbildung 1: Schema der Schichten

Zu erkennen sind außerdem die einzelnen Schnittstellen und um welchen Typ von Schnittstelle es sich hierbei handelt.

4 Architekturumsetzung

Abbildungsverzeichnis