1. Gedanken und Überlegungen zu der Importerfunktionalität

1.1 Designentscheidung

Die Aufgabenstellung umfasst die Anforderung an eine Schnittstelle um Aktivitäten aus Fremdsystemen zu importieren und nutzbar zu machen. Da es keine festen Angaben gab, wie das Format auszusehen hat, haben wir uns für das XML Format entschieden. Die Vorteile von XMLs liegen auf der Hand.

* Standardisierung
* Offenheit
* Sehr stark verbreitet
* Sehr gute Erweiterbarkeit der Attribute (Flexibilität)
* Einfachheit

Durch die Baumstruktur (DOM) lässt sich ein XML Dokument verhältnismäßig einfach parsen, da der Standard von XML definierte Start und Endpunkte vorgibt in Form von öffnenden und schließenden Tags.

XML ist kein proprietäres Format, sondern wird durch das W3C entwickelt. Es kostet nichts und ist nicht an ein bestimmtes Unternehmen gebunden.

Neben JSON und CSV ist XML quasi der de facto Standard bei der Strukturierung von Daten.

In Abgrenzung zu den beiden zuvor genannten Formaten bietet XML allerdings die Möglichkeit der Selbstbeschreibung durch die Attributtags.

XML hat auch einige Nachteile, die nicht unerwähnt bleiben sollen:

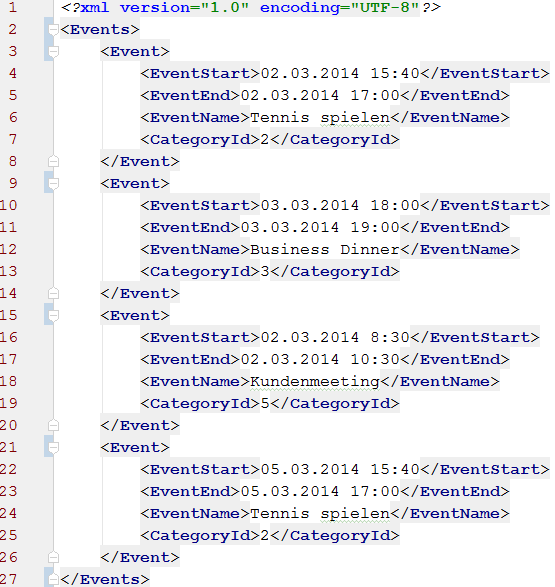
* fehlende Skalierbarkeit
* Großer Daten-Overhead
* Beschränkte Modellierungsfähigkeiten

Der Nachteil der fehlenden Skalierbarkeit und des großen Daten-Overheads durch die Tags ist aufgrund der Aufgabenstellung und der geringen Daten, die übertragen bzw importiert werden sollen, nicht relevant in Bezug auf die Aufgabenstellung. Da wir außerdem mit Strings und Integern arbeiten ist die Notwendigkeit komplexe Datentypen wie zum Beispiel Matrizen oder Tabellen abzubilden nicht gegeben.

Zusammengefasst überwiegen die Vorteile deutlich die Nachteile. Weiterhin gibt es durch die starke Verbreitung von XML viele Bibliotheken, die sich hervorragend um das Parsen von XML Dateien kümmern können.

Die XML Struktur soll dabei der Objekt Struktur der Events nachempfunden werden um die Daten direkt in das Model parsen zu können.

Eine Beispiel XML könnte so aussehen:



Es ist angedacht die XML durch Schleifen-Konstrukte pro Knotenpunkt nach Daten suchen zu lassen und bei gefundenen Daten, diese in das Eventobjekt zu schreiben. Bei falschen Datentypen oder falscher Syntax und fehlenden Daten soll eine Fehlerbehandlung eingreifen und den Benutzer informieren, dass der entsprechende Datensatz nicht importiert werden konnte.

Es macht beispielsweise keinen Sinn einer Aktivität einen Startzeitpunkt aber keinen Endzeitpunkt zuzuweisen, da ansonsten keine sinnvolle Kollisionsabfrage implementiert werden könnte.

Alle Datensätze sind Pflichtfelder, die korrekt mit Daten gefüllt sein müssen, damit das System funktionieren kann.

Bevor die Daten aus dem Eventobjekt in der Datenbank persistiert werden, soll eine Überprüfung stattfinden, ob es möglichweise Kollisionen der Zeiträume gibt. Auch in diesem Fall soll der entsprechende Datensatz nicht persistiert werden und der Benutzer über die Kollision benachrichtigt werden.

Ein Datensatz soll mit dem Knotenpunkt <Event> starten und </Event> enden und die Attribute

* EventStart
* EventName
* EventEnd
* CategoryId

beinhalten. Es soll weiterhin möglich sein mehrere Events in einer XML zusammenzufassen und importieren zu können um dem Benutzer dadurch mehr Usability zu bieten. Alternativ wäre es auch denkbar für jede Aktivität eine eigene XML bereit zu stellen, welche wir aber aus Gründen der Benutzerfreundlichkeit nicht betrachten wollen.

**Was soll die Importschnittstelle explizit nicht bieten?**

* Die importierten XMLs sollen nicht gespeichert werden
* Es sollen nicht gleichzeitig mehrere XMLs importiert werden können
* Die importierten XMLs sollen keine individuelle Dateinamen haben können