

Zentrale Architekturdokumentation

Software Architektur

Software zur Verwaltung von Laufdaten

Milos Babic & Tim Schmiedl

Übungsblatt Nr. 2

24. April 2013

[illegible]

Gliederung

1. Einführung und Ziele.....	5
4.0. Aufgabestellung.....	5
4.1. Architekturziele.....	5
4.2. Stakeholder.....	6
2. Randbedingungen.....	7
2.1 Programmiersprache.....	7
2.2 Laufzeitumgebung.....	7
2.3 Tools.....	7
3. Kontextsicht.....	8
4. Bausteinsicht.....	9
4.1 Komponentendiagramm.....	9
4.3 Klassendiagramm.....	10
5. Laufzeitsicht.....	11
5.1 Use Case Diagramm.....	11
5.2 Sequenzdiagramm.....	14
6. Verteilungssicht.....	16
7. Entwurfsentscheidungen.....	17
7.1 Architektur Aspekte.....	17
7.2 Implementierungsaspekte.....	17
8. Technische Konzepte.....	18
8.1 Testbarkeit.....	18
9. Risiken.....	19
10. Glossar.....	20

Verzeichnis der Abbildungen

Abb. 1: Kontextdiagramm.....	6
Abb. 2: Komponentendiagramm.....	7
Abb. 3: Use Case-Diagramm - Läufer.....	9
Abb. 4: Use Case-Diagramm - Administrator.....	10
Abb. 5: Use Case-Diagramm – externe Schnittstellen.....	11
Abb. 6: Sequenzdiagramm - Datenimport.....	12
Abb. 7: Sequenzdiagramm – Läufer anmelden.....	13

Verzeichnis der Tabellen

Tab. 1: Stakeholder.....	4
Tab. 2: Glossar und Begriffserklärung.....	18

1. Einführung und Ziele

Im Folgendem wird das zu erstellende Softwaresystem "RunningEasy" als "System" bezeichnet.

4.0. Aufgabestellung

Zur Verwaltung von Laufveranstaltung soll ein neues Softwaresystem – RunningEasy – erstellt werden.

Es soll Veranstaltungen samt ihren Teilnehmern und Ergebnissen erfassen und auswerten.

Der Läufer kann sich an Veranstaltungen anmelden und Ergebnisse anschauen.

Auch soll der Läufer per SMS oder Email benachrichtigt werden, falls die Startgebühr noch nicht überwiesen wurde.

Zudem sollen Schnittstellen zu externen Systemen wie Bankanwendungen, SMS/Email-Service und Datenimportierung vorhanden sein.

4.1. Architekturziele

Anhand des Lastenhefts (Blatt 1 SWA-Praktikum) sind folgende funktionale und nicht funktionale Anforderungen gegeben:

Funktionale Anforderungen:

- Anlegen von Laufveranstaltungen
- Anmelden eines Läufers zu einer Veranstaltung
- Überweisen der Startgebühr
- Anmeldung bei einer Veranstaltung zurückziehen
- Liste der gemeldeten Läufer, die die Startgebühr (noch) nicht überwiesen haben
- Erinnerung per E-Mail oder SMS zur Gebührbezahlung
- Vereinszugehörigkeit eines Läufers ändern
- Erstellen von Startlisten
- Anzahl von Meldungen zu einem bestimmten Tag vor Anmeldeschluss anfordern
- Zuweisen von Startnummern zu den gemeldeten Teilnehmern
- Importieren der Laufzeiten einer Veranstaltung, die von einem externen Zeitmesssystem erfasst werden
- Erstellen von Ergebnislisten
- Liste der Starter, die aufgegeben haben und nicht im Ziel angekommen sind
- Disqualifikation eines Läufers
- Zeitkorrektur: eine erfasste Laufzeit manuell korrigieren können
- Versand der persönlichen Laufzeit und Platzierung per SMS
- Liste aller Ergebnisse für einen bei mehreren Veranstaltungen gestarteten Läufer
- Zahlungseingänge von einer Bankanwendung für ein Konto anfordern
- Zwischenzeiten aus einem externen Laufzeitsystem anfordern
- Datenimport von Vorgängersystemen

Nicht funktionale Anforderungen:

- Erinnerung per E-Mail oder SMS zur Gebührbezahlung nach 5 Tagen
- Datenimport von Vorgängersystemen in serialisierter Form

4.2. Stakeholder***Tab. 1: Stakeholder***

Stakeholder	Anforderungen
Läufer/Teilnehmer	<ul style="list-style-type: none">• fordern unkomplizierte Anmeldungen an Veranstaltungen .• fordern eine übersichtliche Statistik der Veranstaltungen und Läufer bzw. Teilnehmer.
Administrator	<ul style="list-style-type: none">• Einfache Bedienung zur Änderung/Verwaltung von Informationen zu Läufer, Laufzeiten, Vereinen, Teilnahmen und Ergebnissen.• Schnittstellen zu externen System leicht wartbar und erweiterbar
Veranstalter	<ul style="list-style-type: none">• fordern Erstellung und Konfigurierung von Veranstaltungen, und Zahlungskontrolle der Teilnehmer.
Externe Schnittstellen	<ul style="list-style-type: none">• müssen kompatibel zu dem System sein• Dazu gehören u.A.<ul style="list-style-type: none">• Bankanwendung• Laufzeitsystem• Benachrichtigungsdienst
Altdaten	<ul style="list-style-type: none">• müssen im vorhanden Format importiert werden können

2. Randbedingungen

2.1 Programmiersprache

- Es wird ausschließlich in Java programmiert.

2.2 Laufzeitumgebung

- pure Java (kein JavaEE mit EJB etc.) in Java virtual machine JVM
- JPA 2.0 mit EclipseLink als Implementierung

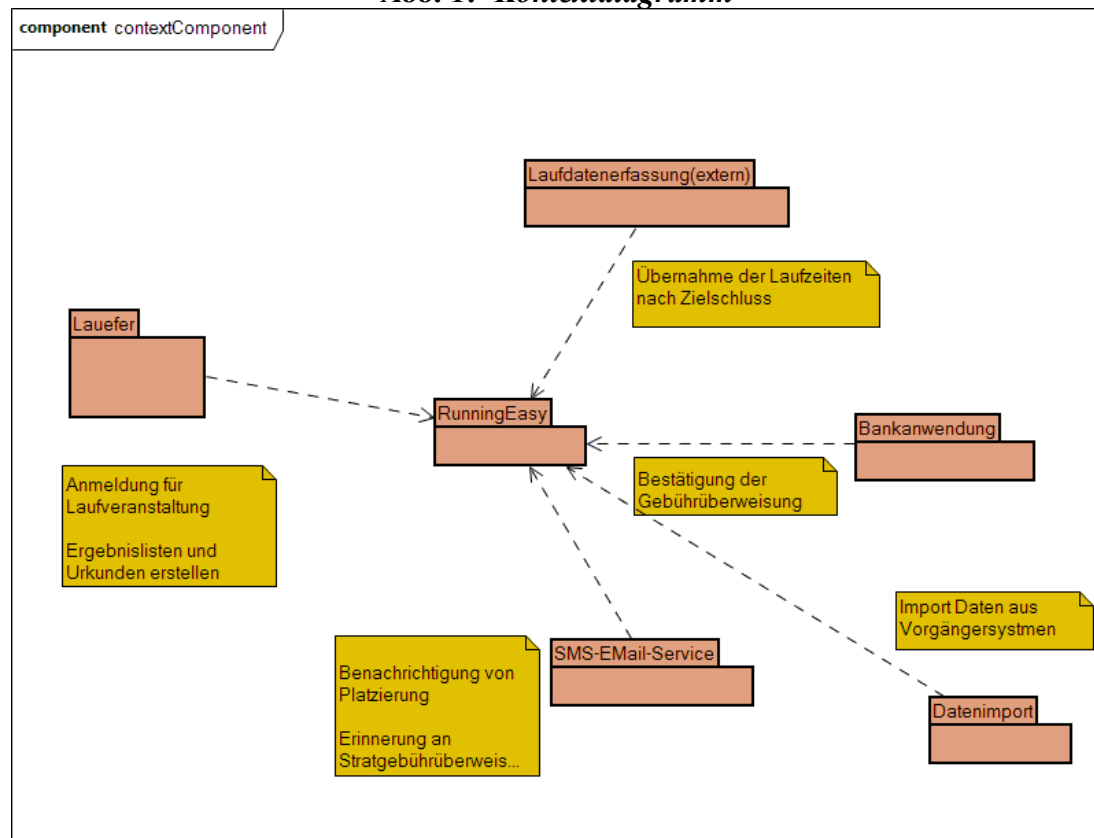
2.3 Tools

- Eclipse Java EE zur Java-Programmierung
- TopCased für UML-Diagramme
- OpenOffice/Libreoffice als Officeprogramm
- Git als Versionsverwaltungstool
- Github als Issuemanagement-Tool/Bugtracker

3. Kontextsicht

Im folgenden Diagramm ist das System im Kontext mit angrenzenden Systemen und Akteuren dargestellt.

Abb. 1: Kontextdiagramm



Der Läufer benutzt RunningEasy um sich für eine Laufveranstaltung anzumelden/stornieren, seine Ergebnisse anzufordern.

Das SMS-EMail-Service versendet Emails/SMSs, die vom RunningEasy erstellt wurden an die Läufer oder Teilnehmer gerichtet sind.

Der Datenimport besitzt Daten aus Vorgängersystem bzw. alten Verwaltungssystem für Laufveranstaltungen. Diese sollen in das RunningEasy import werden können.

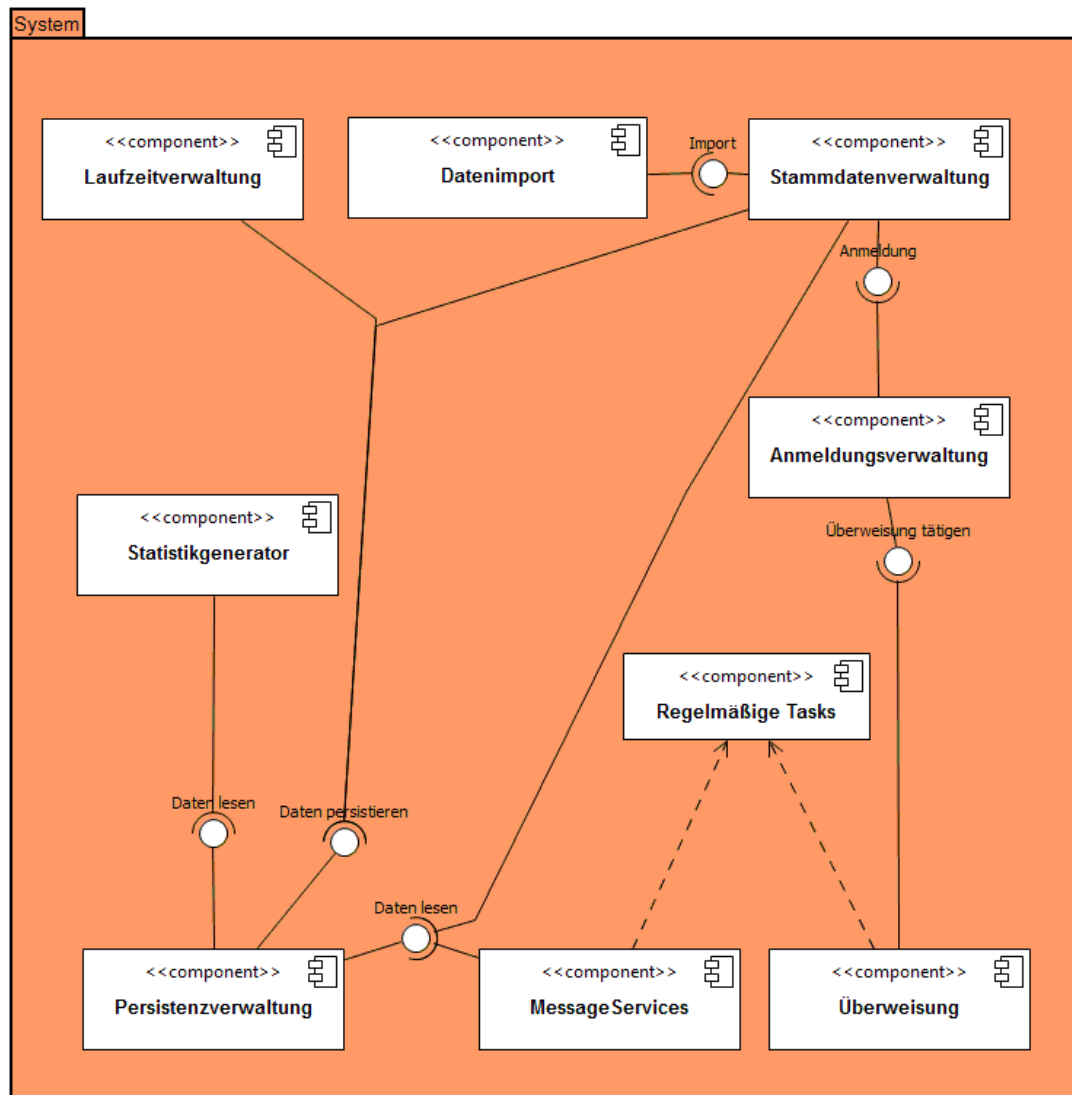
Von der Bankanwendung kann RunningEasy sich Bestätigung der Gebührüberweisung melden lassen. Somit behält das System den Überblick über bezahlte und unbezahlte Gebühren.

Das Laufdatenerfassungssystem enthält Laufdaten von Teilnehmern einer Veranstaltung. Diese benutzt RunningEasy um danach die Ergebnisse der Veranstaltung und Teilnehmer zu erstellen.

4. Bausteinsicht

4.1 Komponentendiagramm

Abb. 2: Komponentendiagramm



Im obigen Diagramm sind die wichtigsten Komponenten mitsamt ihren Schnittstellen und Verbindungen angezeigt.

Besondere Erklärung bedarf die Komponente „Regelmäßige Tasks“, welche hier als abstrakte „Oberkomponente“ von „MessageServices“ und „Überweisung“ dargestellt ist. Nach unserem Lösungsansatz war dies notwendig, da diese Teile der Anwendung nicht direkt durch Benutzereingabe sondern in irgendeiner Form zeitgesteuert getriggert werden müssen.

Beispielsweise muss die Komponente MessageService automatisch nach Ablauf bestimmter Zeiten Mitteilungen an außenstehende Systeme verschicken, ohne dass es einen Input eines menschlichen Akteurs bedarf.

Dies unterscheidet diese Teile der Anwendung vom Restsystem und ist daher wie dargestellt modelliert.

Die Stammdatenverwaltung kümmert sich um jegliche Änderung des Datenbestandes welche durch Eingaben im Client durchgeführt werden.

Die Datenimport Komponente liest die Altdaten ein und legt über eine Schnittstelle mit der Stammdatenverwaltung diese importierten Daten auch im aktuellen System wieder an.

Alle Komponenten mit schreibenden oder lesenden Zugriff auch persistente Daten kommunizieren über vorgegebene Schnittstellen mit einer Persistenzverwaltungs-Komponente welche die Implementierung der Persistenzschicht kapselt.

4.2 Klassendiagramm

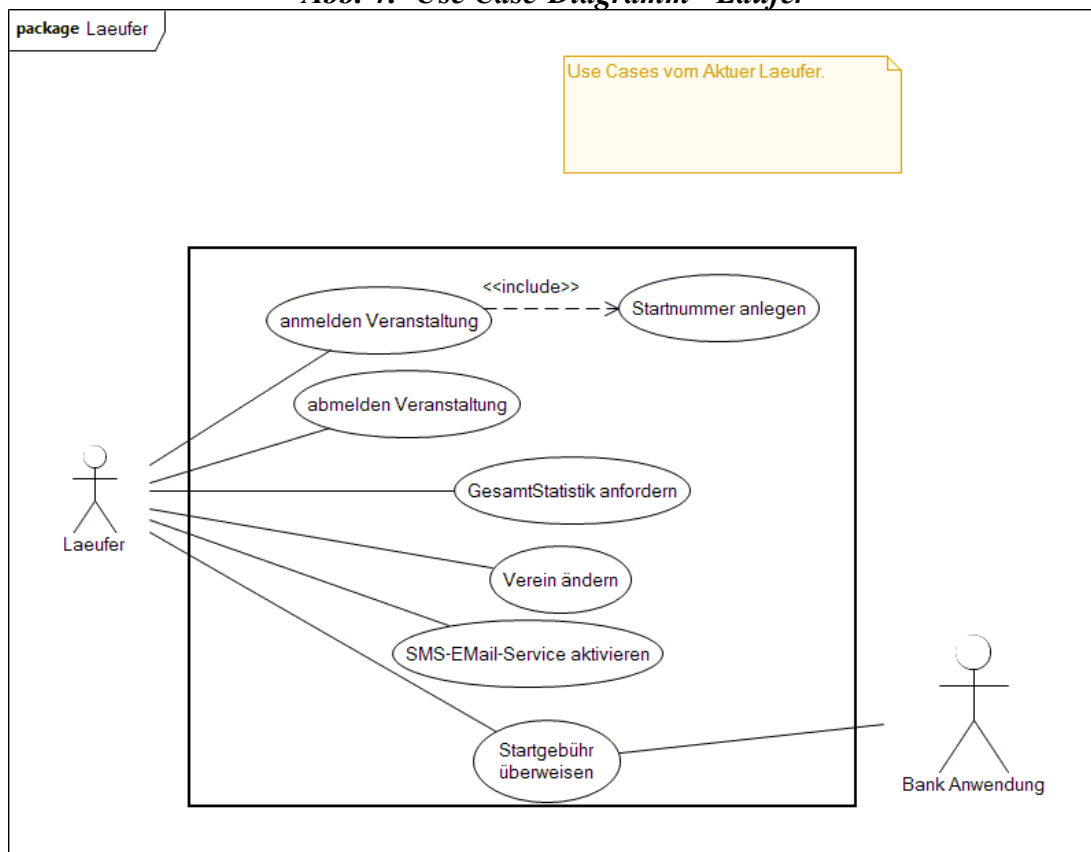
Abb. 3: Klassendiagramm

5. Laufzeitsicht

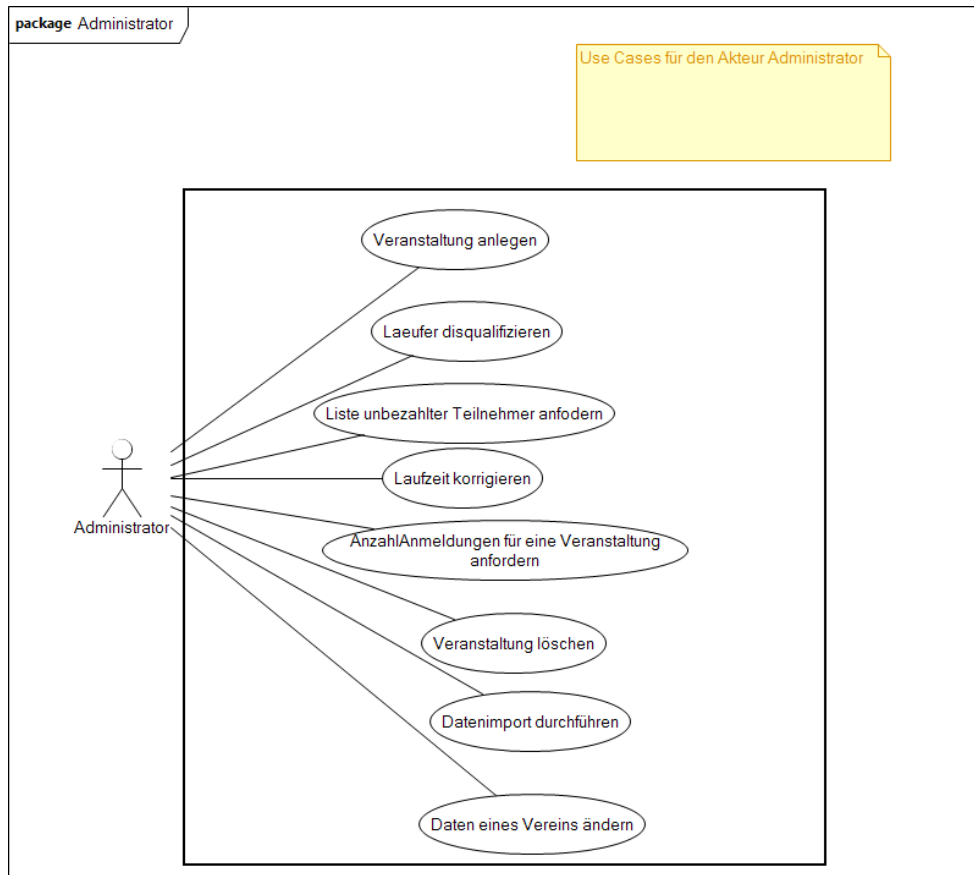
5.1 Use Case Diagramm

Für eine bessere Übersicht wurden das Use Case-Diagramm in drei separate Diagramme gespalten, welche jeweils um einen zentralen Akteur angeordnet sind. Dies sind einmal der Läufer bzw. Teilnehmer, ein Administrator/Systemverwalter sowie die externen Schnittstellen.

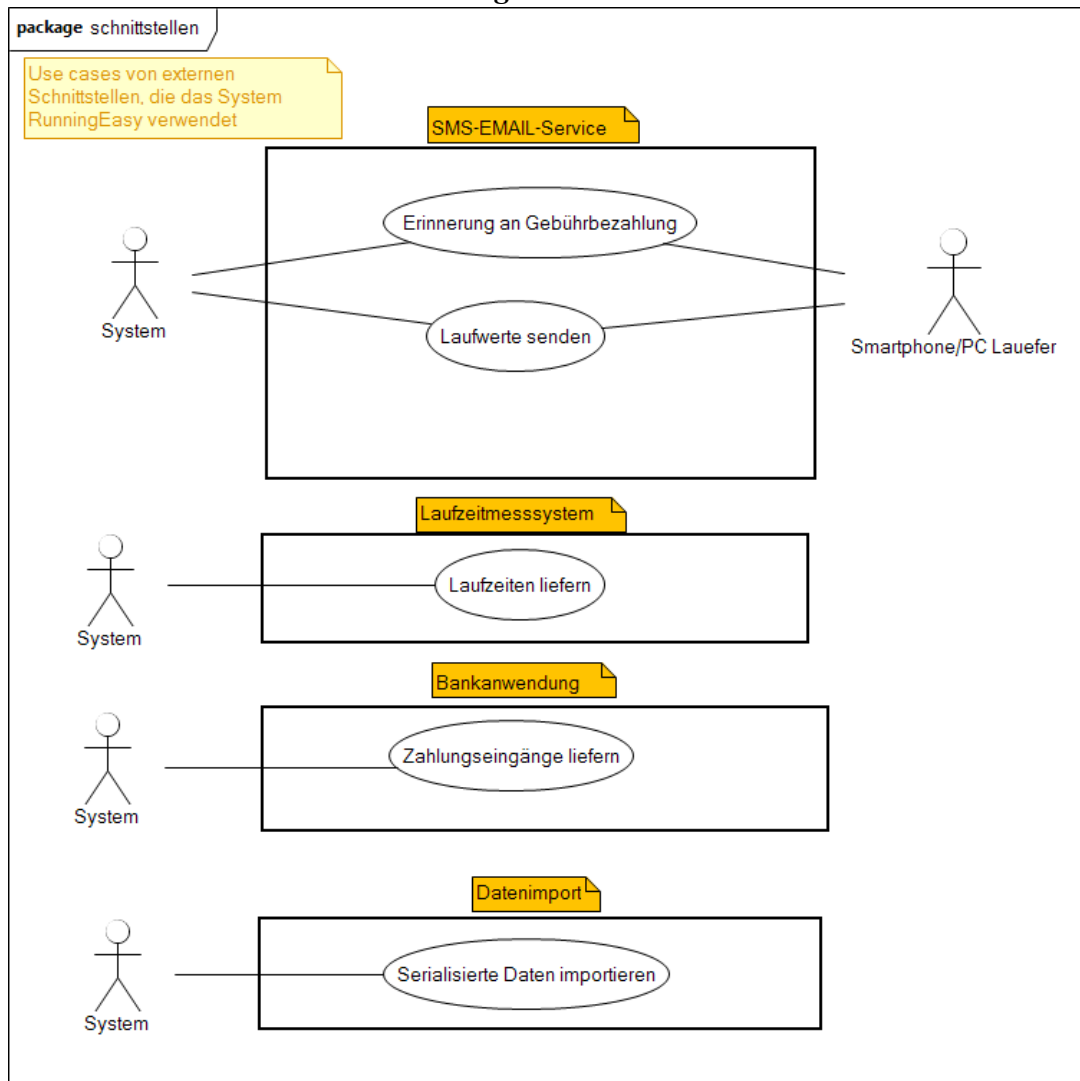
Abb. 4: Use Case-Diagramm - Läufer



Im obigen Diagramm sind die alle bedeutenden Use Cases des Akteurs "Läufer" dargestellt.

Abb. 5: Use Case-Diagramm - Administrator

Im obigen Diagramm sind die alle bedeutenden Use Cases des Akteurs "Administrator" dargestellt.

Abb. 6: Use Case-Diagramm – externe Schnittstellen

Im obigen Diagramm sind die alle bedeutenden Use Cases der externen Schnittstellen dargestellt. Hierbei ist jeweils das System "Runningeasy" der Akteur welches mit externen Programmen/Komponenten interagiert.

5.2 Sequenzdiagramm

Um eine grobe Idee von der Abarbeitungskette des Systems von den einzelnen Use Cases zu geben sind hier zwei Sequenzdiagramme modelliert.

Abb. 7: Sequenzdiagramm - Datenimport

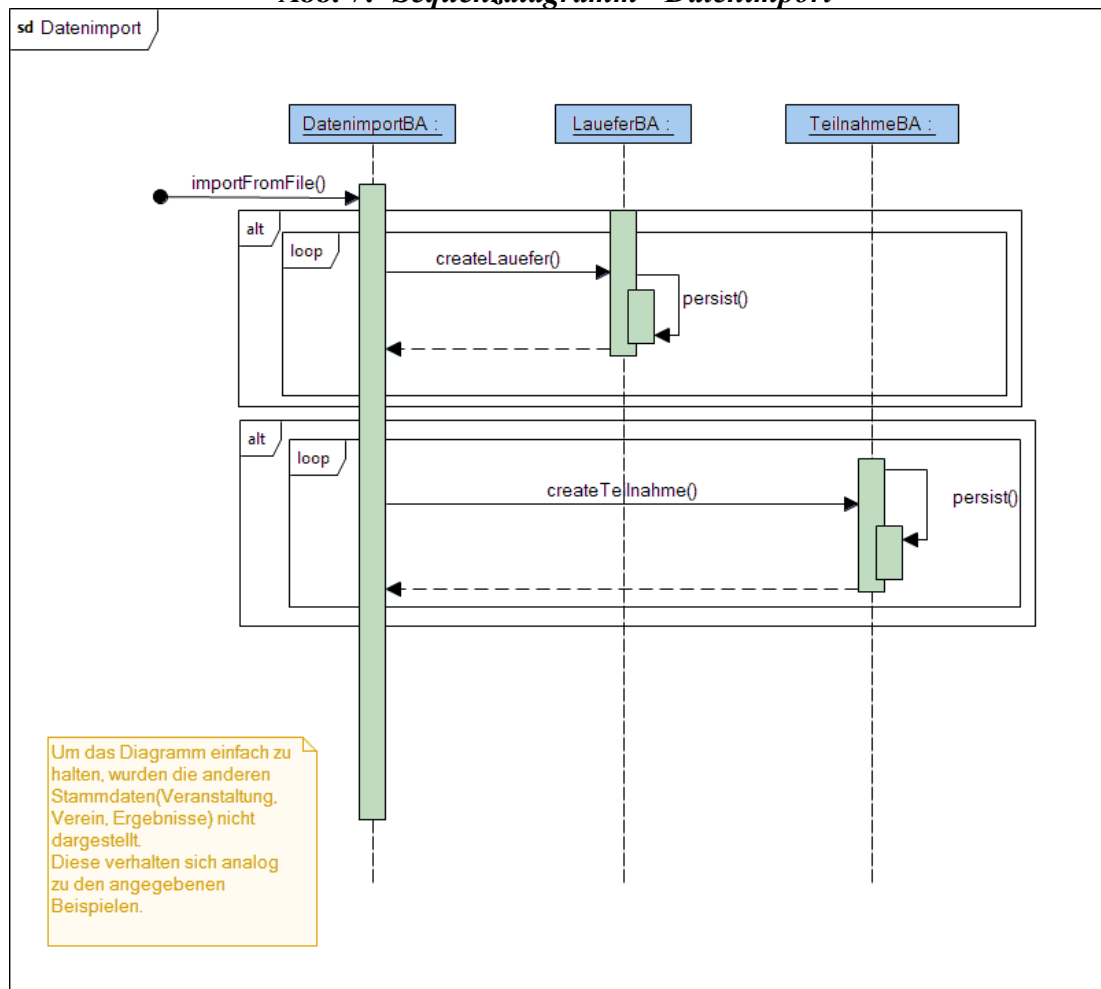
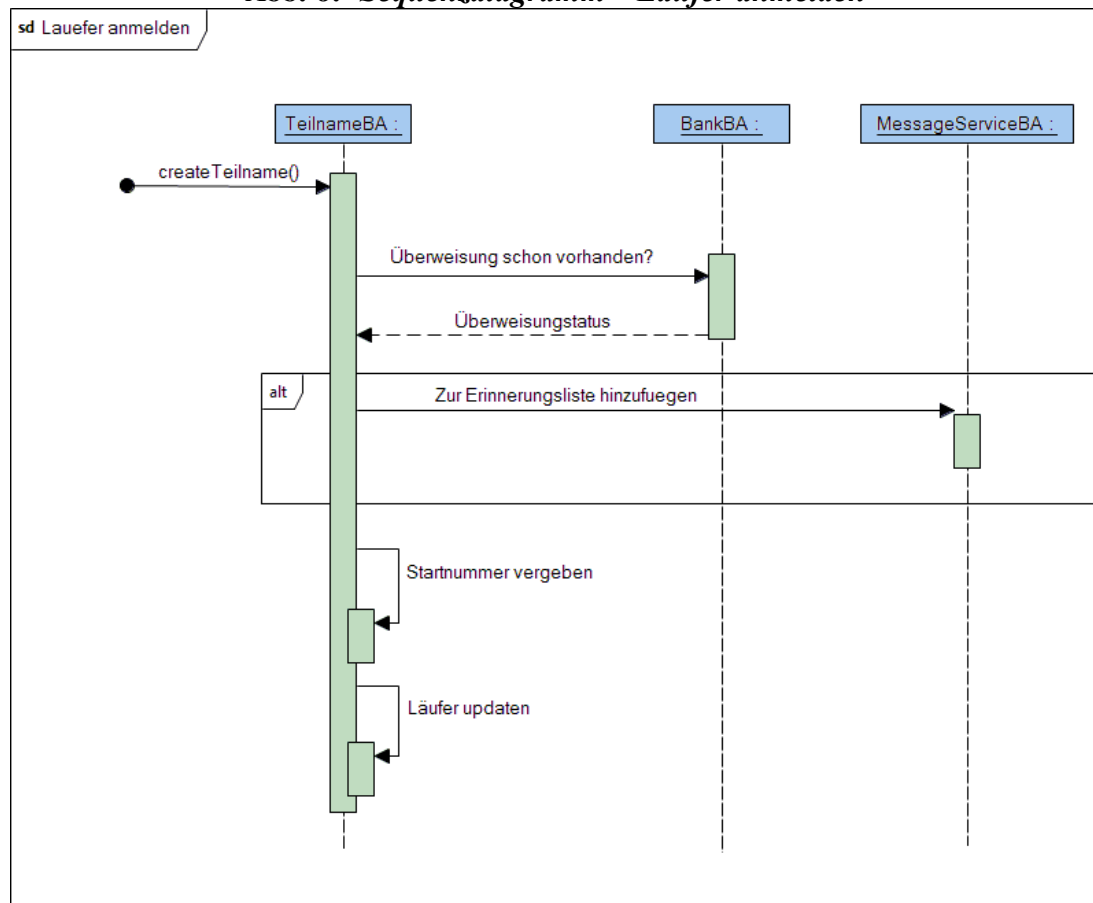


Abb. 8: Sequenzdiagramm – Läufer anmelden

6. Verteilungssicht

Keine nähere Beschreibung.

7. Entwurfsentscheidungen

7.1 Architekturaspekte

- klare Trennung von Logik und Daten (BA-Klassen - Business Logik, BE-Klassen Datenobjekte)
- einfaches, einheitliches und verständliches Design der einzelnen Schichten des Systems
- klare Hierarchien und Abarbeitungsverläufe bei Vermeidung von Zyklen in der Aufrufhierarchie

7.2 Implementierungsaspekte

- guter, eleganter und an "Java Code Conventions" (<http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html>) angepasster Sourcecode
- übersichtlicher und verständlicher Code (gut dokumentiert, aber auch nur dort wo es notwendig ist)
- "Codesprache" Englisch, Fachbegriffe werden jedoch nichtübersetzt
- gute Testbarkeit, Testfunktionen zu den implementierten Füllen

7.3 Managementsicht

- alle relevanten Aufgaben werden mithilfe des Issue-System von Github dokumentiert (mit festem Prozessablauf)
- 4-Augenprinzip bei jeder Issue: Review der Arbeit des Partners
- Selbständiges Arbeiten aber durch regelmäßiges Besprechungen synchronisiert

8. Technische Konzepte

8.1 Testbarkeit

- JUnit-TestCases werden verwendet.

9. Risiken

Keine nähere Beschreibung.

10. Glossar

Tab. 2: Glossar und Begriffserklärung

Stammdaten	Grundinformationen, meist statische Daten welche nur selten geändert werden, meistens keinen Zeitbezug
BA	Business Activity, „stateless“, Klasse mit Business Logik
BE	Business Entity, keinerlei Logik, persistierbar, serialisierbar Attribute