|  |
| --- |
| Elmer Lukas, Heidt Christina, Steiner Diego, Treichler Delia, Waltenspül Remo  12. April 2011 |

|  |
| --- |
| SE2 Projekt MRT |
| Software Architektur |
|  |

****

# Dokumentinformationen

## Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 28.03.2011 | 1.0 | Erste Version des Dokuments | HC |
| 11.04.2011 | 1.1 | Vervollständigung des Dokuments | TD |

## Inhaltsverzeichnis

[1 Dokumentinformationen 1](#_Toc290318876)

[1.1 Änderungsgeschichte 1](#_Toc290318877)

[1.2 Inhaltsverzeichnis 1](#_Toc290318878)

[1.3 Abbildungsverzeichnis 2](#_Toc290318879)

[2 Einführung 3](#_Toc290318880)

[2.1 Zweck 3](#_Toc290318881)

[2.2 Gültigkeitsbereich 3](#_Toc290318882)

[2.3 Definitionen und Abkürzungen 3](#_Toc290318883)

[2.4 Referenzen 3](#_Toc290318884)

[3 Software Architektur 4](#_Toc290318885)

[3.1 Architektonische Ziele & Einschränkungen 4](#_Toc290318886)

[3.1.1 Ziele 4](#_Toc290318887)

[3.1.2 Einschränkungen 4](#_Toc290318888)

[3.2 Architekturübersicht 5](#_Toc290318889)

[3.3 Systemstruktur 6](#_Toc290318890)

[3.3.1 Physische Sicht 8](#_Toc290318891)

[3.3.2 Logische Sicht Rails 8](#_Toc290318892)

[3.3.3 Logische Sicht Android 10](#_Toc290318893)

[3.4 Design Pakete Rails 10](#_Toc290318894)

[3.5 Design Pakete Android 10](#_Toc290318895)

[3.5.1 Package services 10](#_Toc290318896)

[3.5.2 Package gui/gen 11](#_Toc290318897)

[3.5.3 Package activities 11](#_Toc290318898)

[3.5.4 Package persistence 12](#_Toc290318899)

[3.5.5 Package network 14](#_Toc290318900)

[4 Prozesse und Threads 15](#_Toc290318901)

[5 Datenspeicherung 15](#_Toc290318902)

[6 Grössen und Leistung 15](#_Toc290318903)

## Abbildungsverzeichnis

[Abbildung 1 - Deployment Diagram MRT 8](#_Toc290303339)

[Abbildung 2 - Architektur von Ruby on Rails 9](#_Toc290303340)

[Abbildung 3 - Ablauf eines Requests 9](#_Toc290303341)

[Abbildung 4 - Architektur in Ruby on Rails 10](#_Toc290303342)

# Einführung

## Zweck

Dieses Dokument beschreibt die Software Architektur für das Projekt MRT (Mobile Reporting Tool).

## Gültigkeitsbereich

Dieses Dokument ist während der gesamten Projektdauer gültig (21.02 bis 03.06.2011).

## Definitionen und Abkürzungen

Die Definitionen und Abkürzungen befinden sich in der ausgelagerten Datei doc/01\_Projektplan/glossar.docx.

## Referenzen

* doc/03\_Anforderungsspezifikationen/\*
* doc/templates/template.dotx
* doc/templates/java\_formatting.xml
* doc/templates/ruby\_formatting\_settings.zip
* doc/media/logo.png

# Software Architektur

## Architektonische Ziele & Einschränkungen

<Beschreibt die Softwareanforderungen und Objekte, welche einen Einfluss auf die Architektur haben [Bspl: Safety, Security, Privacy, Distribution, …] Beinhaltet auch eine Beschreibung von Design und Implementationsstrategie, Teamstruktur, Entwicklungstools, Zeitplan, etc…>

### Ziele

* Es soll möglichst einfach möglich sein, den Android Client durch einen anderen Client (z.B. iPhone, Windows7 Phone) zu ersetzen. Deshalb darf die Architektur keine Android-spezifischen Konzepte enthalten.
* Die Benutzeroberfläche des Android Clients soll sehr einfach und übersichtlich gestaltet werden. Deshalb stellt der Android Client nur die wichtigsten Funktionalitäten zur Verfügung.
* Mehrere Mitarbeiter müssen gleichzeitig Stundeneinträge machen können. Darum müssen die einzelnen Einträge voneinander getrennt werden.
* Die Daten müssen konsistent bleiben, auch wenn die Datenverbindung zwischen Client und Server unterbrochen wird. Das soll gezielt durch geplante Abläufe erreicht werden.
* Die Schnittstellen zwischen Client und Server sollen genau definiert werden, damit die zwei Teile unabhängig voneinander entwickelt werden könnten.

### Einschränkungen

* Der Android Client wird über den Touchscreen bedient. Deshalb muss darauf geachtet werden, dass beispielsweise die Buttons gross genug sind.
* Da für die Entwicklung des Prototyps nur wenig Zeit zur Verfügung steht und es in diesem Projekt nicht primär um das grafische Erscheinungsbild geht, wird dieses nicht ausgereift sein.

## Architekturübersicht

Wie anhand der nachstehenden Abbildung ersichtlich, besteht die Architektur aus einem Client-Server System. Als Clients werden Computer sowie Smartphones mit dem Android Betriebssystem eingesetzt. Die Android Mobiltelefone verwenden eine eigene kleine Datenbank, welche verwendet wird falls ein Zeiteintrag nicht direkt übermittelt werden kann.

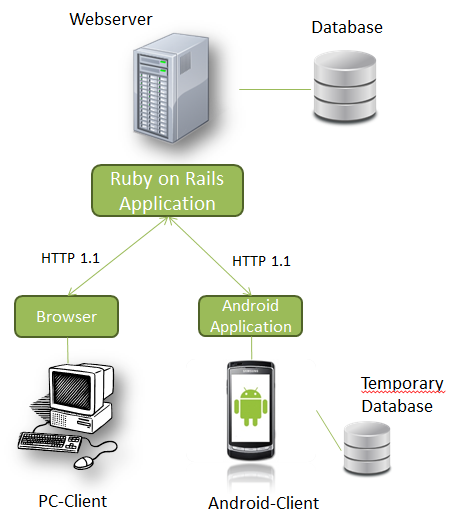


Abbildung 1 - Architekturübersicht

## Kernarchitektur Übertragung Zeiteintrag

### Sequenzdiagramm

Die folgende Abbildung zeigt den Ablauf beim Übertragen eines Zeiteintrags zwischen dem Client (Android) und dem Server.



Abbildung 2 - Sequenzdiagramm Übertragung TimeEntry

### Zustandsdiagramm

Das Zustandsdiagramm zeigt den Ablauf beim Übertragen eines Zeiteintrags mit allen Übergängen und Aktionen.



Abbildung - Zustandsdiagramm Übertragung TimeEntry

## Systemstruktur

### Physische Sicht



Abbildung 4 - Deployment Diagram MRT

Da der Server die Rails Applikation hostet, wird er als Server Execution Node dargestellt. Der Client verbindet sich mit dem Server mit Hilfe des Protokolls HTTP/HTTPS. Deshalb wird der Client als Client Execution Node dargestellt.

Zu beachten ist, dass auf dem Server und dem Client je eine andere Applikation läuft. Auf dem Server ist dies eine Rails Applikation (rails\_app), auf dem Client eine Android Applikation (mrt.apk).

Server und Client kommunizieren über HTTP/HTTPS. Natürlich läuft HTTP/HTTPS über TCP/IP. Eine klare Abgrenzung ist hier wichtig, demzufolge ist für das Projekt die Schicht HTTP/HTTPS die tiefste Schicht.

### Logische Sicht Rails

<Beschreibung mit Text und Diagramm der Architektur. Aufteilung in Packages (zum Beispiel: 3-Layer-Architektur mit GUI, Problem Domain und Datenhaltung)>

Die Architektur der Rails Applikation ist zu einem grossen Teil vorgegeben.

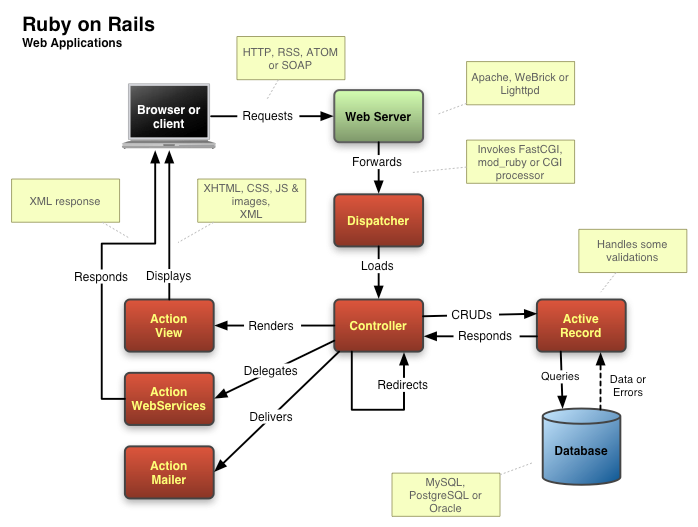


Abbildung 5 - Architektur von Ruby on Rails

Abbildung 6 - Ablauf eines Requests

### Logische Sicht Android

<Beschreibung mit Text und Diagramm der Architektur. Aufteilung in Packages (zum Beispiel: 3-Layer-Architektur mit GUI, Problem Domain und Datenhaltung)>

## Design Pakete Rails

<für jedes definierte Package erfolgt eine Beschreibung mit Diagramm >

Die Architektur in Ruby on Rails ist grundsätzlich bereits vorgegeben. Deshalb ist dieses Kapitel sehr kurz gehalten.

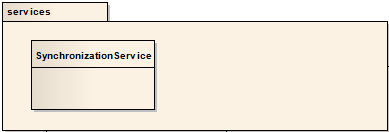
## Design Pakete Android

### Package services

#### Beschreibung des Packages

Das Package Services ist dafür verantwortlich, dass Daten vom Server auf den Client und in die entgegengesetzte Richtung, also vom Client auf den Server, geladen werden.

#### Diagramme



#### Schnittstellen

#### Operationen

##### Interne Operation 1

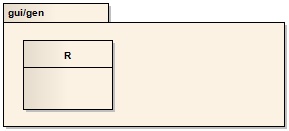
##### Interne Operation 2

### Package gui/gen

#### Beschreibung des Packages

Gui/Gen wird vom Android generiert. Durch dieses Package können die User Interfaces Angaben, welche in einem separaten XML-File gespeichert sind, identifiziert werden.

#### Diagramme



#### Schnittstellen

#### Operationen

##### Interne Operation 1

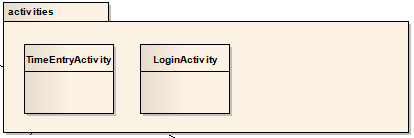
##### Interne Operation 2

### Package activities

#### Beschreibung des Packages

Dieses Package ist die Schnittstelle des Benutzers. In den „activities“ wird die Problem Domain abgehandelt. Activities nehmen Benutzerbefehle entgegen, bearbeiten Models und starten Services.

#### Diagramme



#### Schnittstellen

Wie bereits beschrieben, bearbeiten „activities“ Models und starten Services. Deshalb haben sie eine Abhängigkeit von den Packages Service und Persistence mit dem Unterpackage Models. Zudem besteht auch eine Verbindung zum Package GUI/GEN.

#### Operationen

##### Interne Operation 1

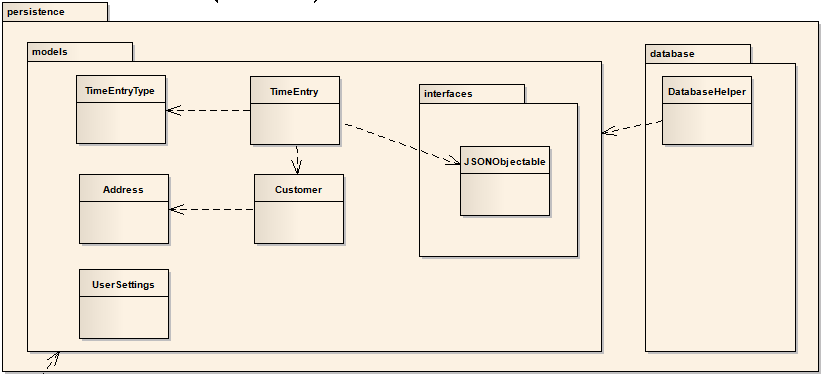
##### Interne Operation 2

### Package persistence

#### Beschreibung des Packages

Das Package „persistence“ ist dafür zuständig, dass Daten persistent gespeichert werden, die Datenbank richtig initialisiert wird und automatisch Tabellen erzeugt werden und Daten korrekt in JSON serialisiert werden.

#### Diagramme



#### Unterpackages

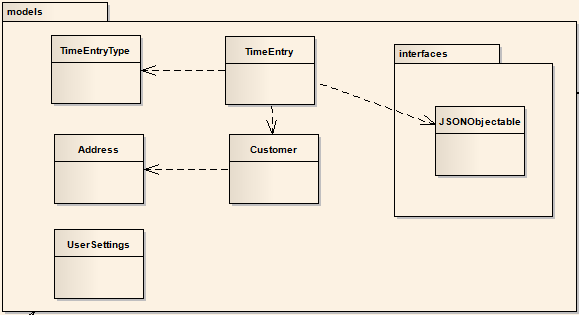
„persistence“ beinhaltet zwei Unterpackages: „models“ und „database“.

##### Package models

###### Beschreibung des Packages

„models“ beinhaltet alle Daten, welche persistent gespeichert werden sollen. Es enthält ein Unterpackage „interfaces“.

###### Diagramme



###### Schnittstellen

###### Operationen

Interne Operation 1

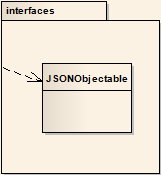
Interne Operation 2

###### Unterpackage interfaces

Beschreibung des Packages

Das Package „interfaces“ beinhaltet die JSON-Serialisierung, welche von den Klassen im Package „models“ benötigt werden.

Diagramme



Schnittstellen

Operationen

Interne Operation1

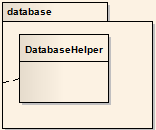
Interne Operation2

##### Package database

###### Beschreibung des Packages

„database“ ist dafür verantwortlich, dass eine Verbindung zur Datenbank hergestellt wird. Ist es zuständig für die Erstellung der Datenbank, mit den einzelnen Tabellen.

###### Diagramme



###### Schnittstellen

###### Operationen

Interne Operation 1

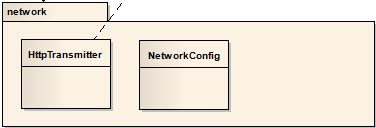
Interne Operation 2

### Package network

#### Beschreibung des Packages

Das Package „network“ ist dafür verantwortlich, dass die Daten übertragen werden.

#### Diagramme



#### Schnittstellen

#### Operationen

##### Interne Operation 1

##### Interne Operation 2

# Prozesse und Threads

<Wenn mehrere Prozesse oder Threads eingesetzt werden wird hier beschrieben, wie diese ablaufen, miteinander funktionieren, Daten austauschen, sich synchronisieren, etc...>

# Datenspeicherung

<Beschreibung mit Diagramm der Datenspeicherung [Data Model]. (zum Beispiel: Datenbank)>

# Grössen und Leistung

<Einschränkungen der Applikation bezüglich Speicher, Leistung, etc…. (zum Beispiel: Verwaltung unterstützt maximal 20'000 Einträge)>