Sitzungsprotokoll

Vom 19.05.2011

# Dokumentinformationen

## Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 19.05.2011 | 1.0 | Erste Version des Dokuments | HC |

## Inhaltsverzeichnis

[1 Dokumentinformationen 1](#_Toc287347230)

[1.1 Änderungsgeschichte 1](#_Toc287347231)

[1.2 Inhaltsverzeichnis 1](#_Toc287347232)

[2 Traktanden 1](#_Toc287347233)

[3 Diskussion / Beschlüsse 1](#_Toc287347234)

[4 Aufgabenverteilung 1](#_Toc287347235)

[4.1 Elmer Lukas 1](#_Toc287347236)

[4.2 Heidt Christina 1](#_Toc287347237)

[4.3 Steiner Diego 2](#_Toc287347238)

[4.4 Treichler Delia 2](#_Toc287347239)

[4.5 Waltenspül Remo 2](#_Toc287347240)

# Traktanden

Besprechung weiteres Vorgehen

Reviewbesprechung mit Herrn Rudin

# Diskussion / Beschlüsse

Besprechung weiteres Vorgehen:

* Kein Autoplay!
* Auswertung von Redmine mit Screenshots und Diagrammen und Berichtsgenerierung.
* Zudem Zugriff zu Redmine (Link).
* Keine Usability Tests für Server.

Besprechung mit Herrn Rudin:

* Kleines Dokument zur Prüfung der Software/Tests, Unit Tests, Refactoring, etc. wurde so und so gehandhabt. Im allgemeinen Berichtdokument festhalten.
* Usability Test nicht zwingend nötig.
* Architekturdokument: 3.3.1. Erklärung wieso wird Hashcode benutzt?
* Architekturdokument: 3.3.1. Anstatt „alt“ „opt“ benutzen. „alt“ wird bei mehreren Ablaufvarianten benutzt.
* Architekturdokument: Deployments Diagramm „Wird als Server execution Node…“ kann gestrichen werden, braucht keine Erklärung
* Architekturdokument: Was ist eine Rack-Applikation?
* Architekturdokument: 3.4.2.8.1 – 3.4.2.8.3 – Hat Bezug auf die obrigen Rails Komponenten 🡪 diese angeben
* Architekturdokument: 3.4.3.4 LOKALE Datenbank
* Architekturdokument: 3.4.3.4, 3.4.3.7 – 3.4.3.8 Rolle im Android Framework 🡪 Bezug zu Android Framework
* Architekturdokument: 3.6.1.1.3 Thread-Pool ist nötig, damit mehrere Aussendienstmitarbeiter den Server gleichzeitig beanspruchen können. Trotzdem ist Thread-Pool umgehbar. Leichtes Umschreiben des Abschnitts.
* Architekturdokument: Polymorphismus nicht zwingend aufzuführen.
* Architekturdokument: Kein Null-Object sondern eher eine Convenience Klasse. Adapter müsste abstrakt sein. Analogie zu MouseListenerAdapter und Namen umschreiben.
* Architekturdokument: JSON-Objekt Know-How könnte in separates Sub-Package verschoben werden oder in eine Oberklasse? Oder mittels Reflection? Andere Lösung? 🡪 Refactoring Ideen
* Architekturdokument: In Dokument begründen warum JSON bei Models eingesetzt wird (siehe vorhergegangener Punkt)
* Architekturdokument: DatabaseSpec fehlt in Klassenstrukutr 4.4.2
* Architekturdokument: Iteration genauer beschreiben bei Services. Was ist mit Exportklasse gemeint? Gleiches bei den Activities.
* Architekturdokument: 3.1.2.2 Erklärung warum Synchronisation auf diese Weise funktioniert und wieso nicht mit Benachrichtigungsmechanismen
* Architekturdokument: Anstatt Referenz gleich Bild einfügen.
* API eher eine Protokollspezifizierung für den Server.

# Aufgabenverteilung

## Elmer Lukas

* Deployment und Release erstellen V1.0
* Persönlicher Bericht (1/2 - 3/4 Seite)
* Refactoring
* Präsentation vorbereiten
* Projektplan aktualisieren
* Einsatz der Werkzeuge auch in Projektplan einschreiben.

## Heidt Christina

* Bedienungsanleitung Server
* Bedienungsanleitung Client
* Persönlicher Bericht (1/2 - 3/4 Seite)
* Usability Tests

## Steiner Diego

* Persönlicher Bericht (1/2 - 3/4 Seite)
* Hauptteil Codereview Dokument
  + Übereinstimmung mit Design
  + Codierrichtlinien eingehalten
  + Codestatistik
* Diagramm Stundenaufwand

## Treichler Delia

* Persönlicher Bericht (1/2 - 3/4 Seite)
* Unit Tests
* Performance Tests
* Systemtests

## Waltenspül Remo

* Persönlicher Bericht (1/2 - 3/4 Seite)
* Codereview Dokument Teil Qualität von Code (temporäres Objekt)
* Risikomanagement anpassen