Hochschule für Telekommunikation Leipzig (HfTL)

Profilierung Netzbasierte Anwendungen

Profilierung Mobile Applikationen

Softwaredokumentation

TaskY - Cache und Notifications in mobilen Webanwendungen

David Howon (147102)

Michael Müller (147105)

Wintersemester 2016/17

Hochschule für Telekommunikation Leipzig
University of Applied Sciences

Inhaltsverzeichnis

1 Einleitung

Diese Dokumentation entstand im Rahmen der Profilierungen „Mobile Applikationen“ und „Netzbasierte Anwendungen“ im Wintersemester 2016/17 an der Hochschule für Telekommunikation Leipzig (HfTL).

... Einleitung moderne webtechnologien –> webapps statt nativen Apps ...
... Beschreibung der Aufgabe/des Problems ...
... Versuch der Lösungsfindung/Kurzbeschreibung Projekt ...

Entwicklung

GitHub		https://github.com/task-notification
Demo	WebApp	https://tasky.atria.uberspace.de/
	Business-Server	TBA

2 Anforderungen

2.1 allgemeine Beschreibung der Applikation

Nach erfolgreicher Registrierung und Anmeldung kann der Benutzer Aufgaben anlegen, bearbeiten, anzeigen und löschen. Weiterhin gibt es eine Kontaktliste, in welcher alle Kontakte angezeigt werden, die ebenfalls für die Anwendung registriert sind und zu persönlichen Kontakten hinzugefügt wurden. Aufgaben können mit persönlichen Kontakten geteilt werden. Ebenso ist es möglich Gruppen anzulegen, dieser Kontakte hinzuzufügen und Aufgabe mit der Gruppe zu teilen.

Über Änderungen an Gruppen oder Aufgaben wird der Benutzer über PUSH-Benachrichtigungen informiert. Wenn einer Aufgabe ein Benachrichtigungszeitpunkt angegeben wurde, wird ebenfalls eine PUSH-Notification angezeigt sobald die Aufgabe terminiert.

2.2 Web Applikation

2.2.1 funktionale Anforderungen

Die WebApp soll für den **Mehrbenutzerbetrieb** ausgelegt werden. Ein Benutzer soll sich für die Nutzung **Registrieren** und anschließend am Portal **Anmelden** können. Es können eigene **Aufgabenlisten** angelegt, bearbeitet oder gelöscht werden. Weiterhin können Aufgabenlisten mit mehreren Benutzern (Kontakte bzw. Gruppen) geteilt werden.

Eine Aufgabe muss mindestens aus einem Titel bestehen und kann mit einem Ort, einer Beschreibung, einer hinterlegten Checkliste, einem Zeitraum sowie einer Fälligkeit erweitert werden.

Der Benutzer soll mittels PUSH-Benachrichtigungen über aktuelle Änderungen an den Aufgabenlisten und Gruppen informiert werden.

2.2.2 nicht-funktionale Anforderungen

- Look and Feel einer nativen Android App
- Single Page Application (SPA)

2.3 Serverkomponente

2.3.1 funktionale Anforderungen

Der Zugriff auf die API ist nur für authentifizierte Benutzer möglich. Die API wird mit HTTP Basic Authentication geschützt.
Der API Server unterstützt folgende Anforderungen um die Funktionalitäten einer RESTful-Schnittstelle zu erfüllen:

- Bereitstellung von CRUD[1]-Funktionalität für Entities
- Aufruf von Ressourcen über eindeutige und einfache URLs (z.B. https://example.de/api/task/ und https://example.de/api/task/:taskId)
- Verwendung der standardisierten HTTP-Methoden (GET, POST, PUT und DELETE)
- Rückgabe im JSON-Format
- alle Requests werden auf der Konsole ausgegeben

2.3.2 nicht-funktionale Anforderungen

[1] *CRUD: create, read, update, delete*

3 Betrachtung aktueller Technologien

3.1 Client Storage Technologien

3.1.1 AppCache

3.1.2 Local Storage/Session Storage

3.1.3 IndexedDB

3.1.4 WebSQL

3.2 Benutzeroberfläche

3.2.1 Onsen UI

... Beschreibung Onsen UI und warum es nicht in Frage kommt ...

3.2.2 Bootstrap 3

... Beschreibung Bootstrap und warum es nicht in Frage kommt ...

3.2.3 JQuery Mobile

... Beschreibung JQuery Mobile ...

3.2.4 nativeDroid 2

... Was ist nativeDroid2 ...

3.3 Entscheidung

Um das „Look and Feel“ einer nativen App zu erreichen wird das UI-Framework **nativeDroid2** verwendet.

4 Realisierung

... Wie wurden die beiden Ideen umgesetzt ...

4.1 Architekturbeschreibung

Die Anwendung beruht auf dem Client-Server-Prinzip. Dabei stellt der Client lediglich die Oberfläche zur Interaktion mit dem Anwender dar. Außer der notwendigen UI Logik ist die gesamte Geschäftslogik auf einen dedizierten (Business-)Server ausgelagert. Dieser stellt ebenfalls die Datenbank bereit. Mittels AJAX und REST werden dynamische Daten vom Client beim Server angefragt.

... Beschreibung (mit Schema) der Softwarearchitektur ... ... Tier2/Tier3 ...

4.2 Clientkomponente

4.2.1 Umsetzung ServiceWorker

... Umsetzung mittels ServiceWorker ...

4.2.2 Umsetzung mittels nativem Android Service

... Umsetzung mittels eigenem nativen Service ...

4.3 Serverkomponente

4.3.1 API Beschreibung

Route	HTTP-Methode	Beschreibung
/api/signup	POST	Registriert einen neuen Benutzer
/api/authenticate	POST	Authentifiziert einen Benutzer
/api/tasks	GET	Gibt alle Aufgaben zurück
/api/tasks	POST	Legt eine neue Aufgabe an
/api/tasks/:taskId	GET	Gibt eine einzelne Aufgabe zurück
/api/tasks/:taskId	PUT	Aktualisiert eine einzelne Aufgabe
/api/tasks/:taskId	DELETE	Löscht eine einzelne Aufgabe

Tabelle 1: Übersicht API Routen

5 Abgrenzung

... Was kann nicht geleistet werden? ...

... Was ist eventuell zukünftig möglich ? ...