Schlußbewertung

"MOAM"

Mobile Appointment Manager

ein Projekt von:

Autoren:	Oliver Normann	s722074	onormann@gmx.net
	Markus Hampel	s723044	darkdecember@gmx.de
Institution:	TFH Berlin		
Datum:	15.07.2008		
Version:	1.0		
Dozent:	Professor Christoph Knabe		

1. Kurzbeschreibung

1.1 Die Idee

Der Mobile Appointment Manager soll das Leben vereinfachen.
Die Koordination seiner eigenen Aufgaben und Kontakte erleichtern , sowie die Sinmplifizierung oft schwieriger Terminabsprachen mit anderen Menschen. Darüber hinaus soll er als Brücke fremder Menschen mit den gleichen Interessen dienen. Durch den mobilen Klienten soll gewährleistet werden, dass man dies zu jederzeit und an jedem Ort bewerkstelligen kann.

1.2 Die Projektmitglieder

Oliver Normann	s722074	onormann@gmx.net
Markus Hampel	s723044	darkdecember@gmx.de

1.3 Verantwortlichkeiten

Oliver Normann: J2ME, JMS

Markus Hampel: EJB3, Persistenz

1.4. Besonderheiten

Die MOAM-Applikation benutzt alle drei Java Plattformen: J2ME, J2EE und J2SE.

1.5 Eingesetzte Techniken und Ihre Bewertungen

JBoss Application Server:

Der JBoss Server wurde ausgewählt, weil er EJB3.0 unterstüzt, sowie die Nutzung der Java Persistence API erlaubt. Weiterhin dient er als JMS Provider für den iBus//Mobile Gateway.

iBus//mobile Gateway:

Dient als Schnittstelle zwischen J2ME und J2EE. Der Gateway benutzt JMS zur Kommunikation.

Sony Ericsson SDK:

Frei erhältliches Kit, speziell für Sony Ericsson Handys. Zwar bringt Netbeans selbst das WTK2.5.2 mit, doch haben wir uns aufgrund seines guten Emulators für das SE SDK entschieden.

EJB3 mit Annotations:

Die Arbeit mit EJB3.0 hatte für uns den Vorteil keinen großen Bruch von Analysephase zur Umsetzung zu haben. Die Metadaten durch Annotation direkt in den Sourcecode zu schreiben erleichterte die Arbeit ungemein.
EJB3.0 setzt sehr viele vordefinierte Einstellungen, so dass man sich nur noch um die kümmern braucht, die man wirklich benötigt.

J2ME:

Der „kleine Bruder“ der Java Standard Edition erwies sich als eine Art Zwangsjacke was die Verfügbarkeit von Paketen aus J2SE betraf sowie die Darstellung komplexer Inhalte auf einem sehr kleinen Bildschirm. Aktuelle Plattformen, wie Android oder UIQ wären hier aus Gründen ihres Funktionsumfangs interessanter gewesen.

JMS:

Die Wahl der Technologie zur Kommunikation zwischen Client und Server fiel letztendlich auf JMS. Anfangs legten wir uns auf WebServices fest, hier kam es aber zu Konformitätsproblemen auf Client-seite. Die JMS API erwies sich als sehr überschaulich und doch sehr mächtig. Durch den iBus//Mobile Gateway haben wir eine zuverlässige Vermittlungsschicht, allerdings beschränkt auf 5 Verbindungen, da es ein kommerzielles Produkt.

J2ME Polish:

„Poliert“ die Oberfläche für verschiedene Handys auf.

JUnit/JMUnit:

Bekanntes Test-Framework. Es gab Probleme mit der Testsuite, da wir mit einem eingebetteten Server getestet haben, der wesentlich weniger Zeit in Anspruch nimmt als ein richtiger Server.

Clover:

Ein sehr gutes Analysewerkzeug, welches die Testabdeckung eines Projekts prüft und einen ausführlichen Bericht erstellt. Leider ist Clover an eine Lizenz gebunden, und nur als Testversion frei.

2. Persönliche Eindrücke

Markus Hampel:

Ich habe sehr viel gelernt in viel zu kurzer Zeit. Ich hätte gern mehr Zeit gehabt um mich noch besser mit den genutzten Techniken auseinander zu setzen. SWP ist das wichtigste Fach während des Studiums. Alles was wir bisher gelernt haben, musste umgesetzt werden. Weiterhin lernt man was Projektarbeit in einem Team bedeutet, was gut funktioniert hat.

Oliver Normann:

Dieses Projekt, so klein es Anfangs schien wuchs zum Ende hin in seinem Umfang stark an vor allem was die Benutzungsoberfläche und die Kommunikationsschicht betrifft. Der anfängliche Hinweis maximal etwa 5 persistente Klassen zu modellieren erwies sich als sehr wichtig und im Rückblick sehr angemessen.

Durch die Arbeit und die damit verbundenen Vorgaben habe ich sehr viel dazugelernt, sei es in technologischer oder stilistischer Hinsicht.