HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

INSTITUTE
FOR
SOFTWARE

TabMed

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2009

Autoren:	Fabian Mettler, Rico Suter
Betreuer:	Prof. Dr. Markus Stolze
Projektpartner:	Institut für Medizinische Lehre Bern (IML), Crealogix
Gegenleser:	Prof. Hans Rudin

Dokumente

Folgende Dokumente sind in der angegebenen Reihenfolge in diesem Dokument zu finden:

Abstract

Extended Management Summary

Aufgabenstellung

Projektplan

Domain Analyse

External Design

Software Architecture Document

Testdokumentationen

Benutzeranleitung

Erfahrungsbericht

Glossar

Erklärung

Vereinbarung

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Kurzfassung der Studienarbeit

Abteilung	**Informatik**
Namen der Studierenden	**Fabian Mettler, Rico Suter**
Studienjahr	**HS 2009**
Titel der Studienarbeit	**TabMed - Client-Server Applikation für medizinische Evaluationen**
Betreuer	**Prof. Dr. Markus Stolze**
Themengebiet	**Software**
Projektpartner	**Institut für Medizinische Lehre Bern (IML), Crealogix**
Institut	**IFS**

Kurzfassung der Studienarbeit

Jährlich führt das Institut für Medizinische Lehre viele praktische Prüfungen, sogenannte Observed Structured Clinical Examinations (OSCE), durch. Für das nächste Jahr ist im Zuge der Ausweitung der Prüfungen auf nationaler Ebene ein starker Anstieg an Prüfungsdurchführungen geplant. Dadurch stösst der bisherige papiergebundene Prüfungsdurchführungsprozess an seine Grenzen. Mit diesem Problem vor Augen wurde am Institut für Software das Projekt E-OSCE gegründet, welches Konzepte für eine digitalisierte und effizientere Prüfungsabwicklung hervorbringen soll.

Ein Teil des E-OSCE Projektes stellt diese Studienarbeit „TabMed“ dar. Die Aufgabe war die Entwicklung einer Umgebung, in der diese Prüfungen durchgeführt werden können. Dazu sollte ein auf einem Tablet-PC aufsetzender Prototyp zur Bewertung von Medizinstudenten erstellt werden. Dabei musste darauf geachtet werden, dass die Client-Applikation einfach und intuitiv zu bedienen ist, da sie auch von nicht computer-versierten Experten verwendet wird.

Da mit sensiblen Daten gearbeitet wird, wurde gefordert, dass die Formulare nicht von unbefugten Personen gelesen werden können. Aus diesem Grund wird die gesamte Kommunikation zwischen den Applikationen verschlüsselt. Die Gesetze fordern, dass Prüfungen signiert und nicht mehr geändert werden können. Daher werden alle Prüfungen mit einer Signatur aus einem Zertifikat, das auf einem USB-Stick gespeichert ist, signiert. Wichtig war ausserdem, dass die Applikation von den Experten akzeptiert wird, denn es war bereits eine gute papierbasierte Lösung vorhanden.

Im Laufe des Projekts entwickelten wir eine komplette Prüfungs-Infrastruktur: Dazu gehört ein Server mit Frontend, ein Client und ein Editor um Prüfungsformulare zu erstellen. In einem abschliessenden Usability-Test in Bern wurden diese Prototypen getestet. Dabei wurde festgestellt, dass die Applikation einfach zu verwenden ist und bereits für richtige Prüfungen verwendet werden kann.

Alle Komponenten kommunizieren über eine REST-basierte HTTPS-Schnittstelle. Diese Schnittstelle wird auf dem Server mit dem Jersey-REST-Framework realisiert. Für die grafischen Oberflächen der Applikationen setzten wir auf die neue WPF-Technologie von Microsoft. Die Kommunikation wurde mit XML-Daten realisiert und kann so auch auf anderen Plattformen verwendet werden.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Extended Management Summary

TabMed

Tablet PC Client-Server Applikation für medizinische Evaluationen

Studierende	Fabian Mettler, Rico Suter
Betreuer	Prof. Dr. Markus Stolze
Themengebiet	Software
Industriepartner	Institut für Medizinische Lehre Bern (IML), Crealogix

Einleitung

Jährlich führt das Institut für Medizinische Lehre viele praktische Prüfungen, sogenannte Observed Structured Clinical Examinations (OSCE), durch. Für das nächste Jahr ist im Zuge der Ausweitung der Prüfungen auf nationaler Ebene ein starker Anstieg an Prüfungsdurchführungen geplant. Dadurch stösst der bisherige papiergebundene Prüfungsdurchführungsprozess an seine Grenzen. Mit diesem Problem vor Augen wurde am Institut für Software das Projekt E-OSCE gegründet, welches Konzepte für eine digitalisierte und effizientere Prüfungsabwicklung hervorbringen soll.

Im Frühlingssemester 2009 wurde am IFS im Rahmen der Bachelorarbeit „MoMEA“ eine iPhone-Applikation entwickelt, mit der es möglich ist, diese Prüfungen mit einem iPhone oder iPod Touch durchzuführen. Dieses Gerät erwies sich bei ersten Usability-Tests als zu klein und ineffizient. Basierend auf diesen Erkenntnissen began die Entwicklung einer Applikation für ein Tablet-PCs, welche die iPhone-bezogenen Limitationen nicht aufweisen. Die Aufgabe der Studienarbeit „TabMed“ war die Entwicklung einer kompletten Client/Server-Umgebung, in der diese Prüfungen durchgeführt werden können. Schwerpunkt der Arbeit bilden Usability- und Security-Aspekte, die allgemein mit elektronischen Prüfung in Verbindung gebracht werden.

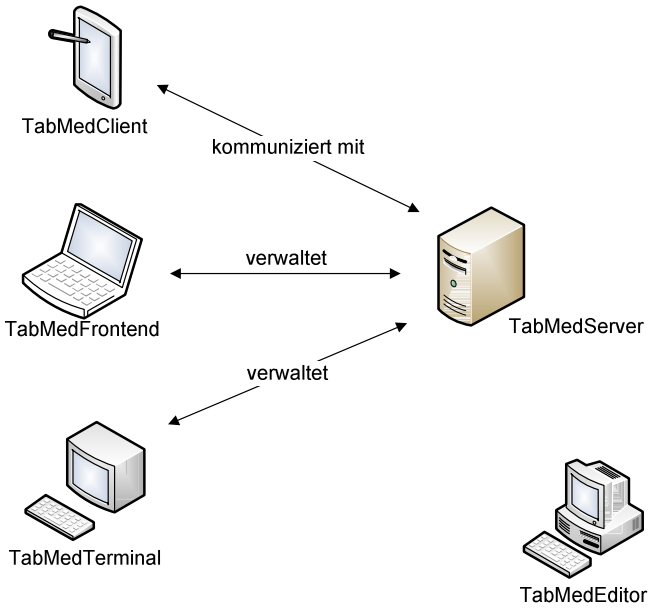
In einer Evaluation von verschiedenen Tablet-PC Geräten sollen ausserdem Aussagen gemacht werden, welcher Gerätetyp sich als Exam-Client am besten eignet.

Anforderungen

Da mit sensiblen Prüfungsresultaten gearbeitet wird, wurde gefordert, dass die Prüfungsformulare nicht von unbefugten Personen gelesen werden können. Aus diesem Grund wird die gesamte Kommunikation zwischen den Applikationen verschlüsselt**.** Alle Zugriffe auf den Server funktionieren nur per HTTPS mit SSL-Verschlüsselung. Die Gesetze fordern, dass Prüfungen einem Experten zugeordnet und nicht mehr geändert werden können. Daher signiert die Client-Applikation alle Prüfungen mit einem Zertifikat, das auf einem USB-Stick gespeichert ist. Dabei wird die XML-Datei mit einem weiteren Block ergänzt, der diese Signatur enthält. Eine weitere essentielle Anforderung stellt die Akzeptanz der Endnutzer gegenüber dem System dar.

Architektur

Im Laufe des Projekts entwickelten wir eine komplette Prüfungs-Infrastruktur, welche die oben erwähnten Anforderungen an eine elektronische Prüfung erfüllen. Dazu gehört ein Server mit Frontend, ein Client und ein Editor um Prüfungsformulare zu erstellen. In einem abschliessenden Usability-Test in Bern wurden diese Prototypen getestet. Dabei wurde festgestellt, dass die Applikation einfach zu verwenden ist und bereits für richtige Prüfungen verwendet werden kann.



Entwickelte Applikationen im Rahmen der TabMed-Studienarbeit

Wir haben uns für eine Kommunikation über eine REST[1]-basierte HTTPS-Schnittstelle entschieden, da sich unsere Abfragen optimal mit REST-Abfragen abbilden lassen. Diese Schnittstelle wird auf dem Server mit dem Jersey-REST-Framework realisiert. Um einfach mit Touch-Features umgehen zu können, haben wir uns für die neue WPF[2]-Technologie von Microsoft entschieden, da dieses Framework diese Funktionlität gut kapselt. Die Daten werden per XML-Serialisierung übermittelt und können so auch auf anderen Plattformen problemlos gesendet und empfangen werden.

[1] Richardson, Leonard; Ruby, Sam (2007), RESTful Web Services, O'Reilly, ISBN 0596529260

[2] http://msdn.microsoft.com/en-us/library/ms754130.aspx

Ergebnisse

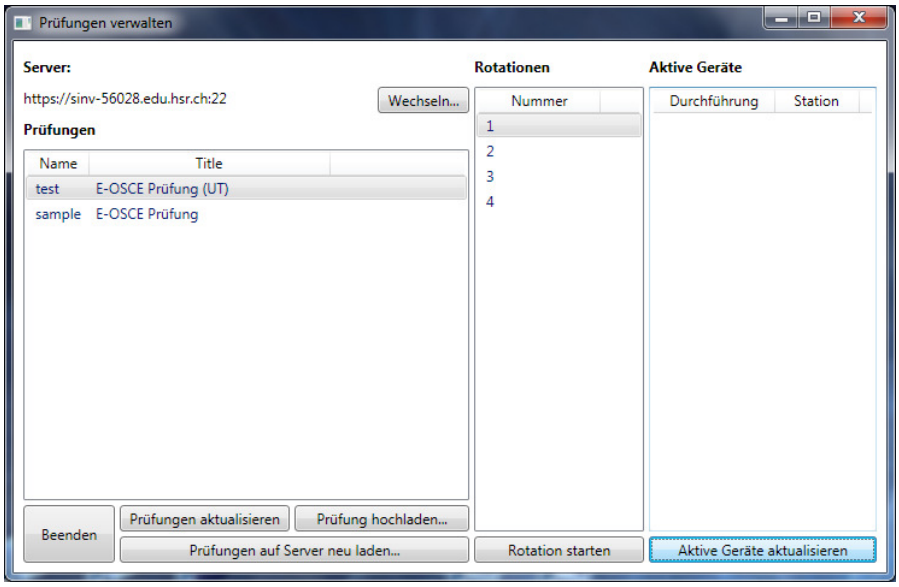
Der Hauptfokus der Arbeit kommt dem TabMedClient zu, mit dem man sich mit dem Server verbinden, Prüfungen herunterladen und ausfüllen kann. Zeitgleich wurden Applikationen zum Verwalten des Servers und Bearbeiten der Prüfungsformulare entwickelt.

TabMedClient: Applikation zum Ausfüllen der Prüfungsbogen.
Links: Ansicht zum Ausfüllen der Prüfungsformulare
Rechts: Übersicht über alle Kandidaten der Prüfung

OSCE-Prüfungen sind mit sehr grossen organisatorischen Aufwänden verbunden, daher ist es eminent wichtig ein Notfallkonzept auszuarbeiten um mit den verschiedenen Fehlerquellen umgehen zu können. Da bei Netzwerkausfällen alles problemlos weiterfunktionieren muss, speichert die Applikation alle Resultate an mehreren Orten gleichzeitig, sodass in jedem Fall ein Backup vorhanden ist. Alle Daten werden verschlüsselt an den Server übermittelt und nur autorisierte Benutzer dürfen sich mit dem Server verbinden.

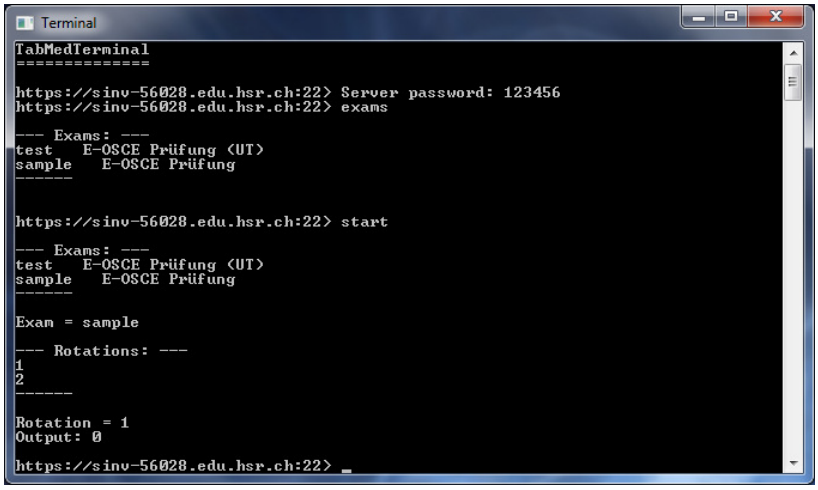
Damit der Server ein Startsignal an die Clients senden kann und dies auch mittels HTTPS-REST-Requests funktioniert, wurde die Long Polling-Technik implementiert. Die Clients fragen dabei beim Server an, ob eine Rotation gestartet wurde und wiederholen dies, bis die Rotation gestartet wurde. Im Unterschied zum traditionellen Polling, blockiert der Server beim Long Polling zustätzlich bis zu einem vorgegebenen Timeout, um die Anfragezyklen zu verlängern und den Verbindungs-Overhead zu minimieren.

Die Prüfungsadministrations-Applikation ermöglicht es, Prüfungen auf dem Server zu verwalten sowie Rotationen zu starten. Für den Support-Techniker ist zudem wichtig, dass überprüft werden kann, welche Geräte mit dem Server verbunden bzw. aktiv sind.



TabMedFrontend zum Verwalten und Starten von Prüfungen

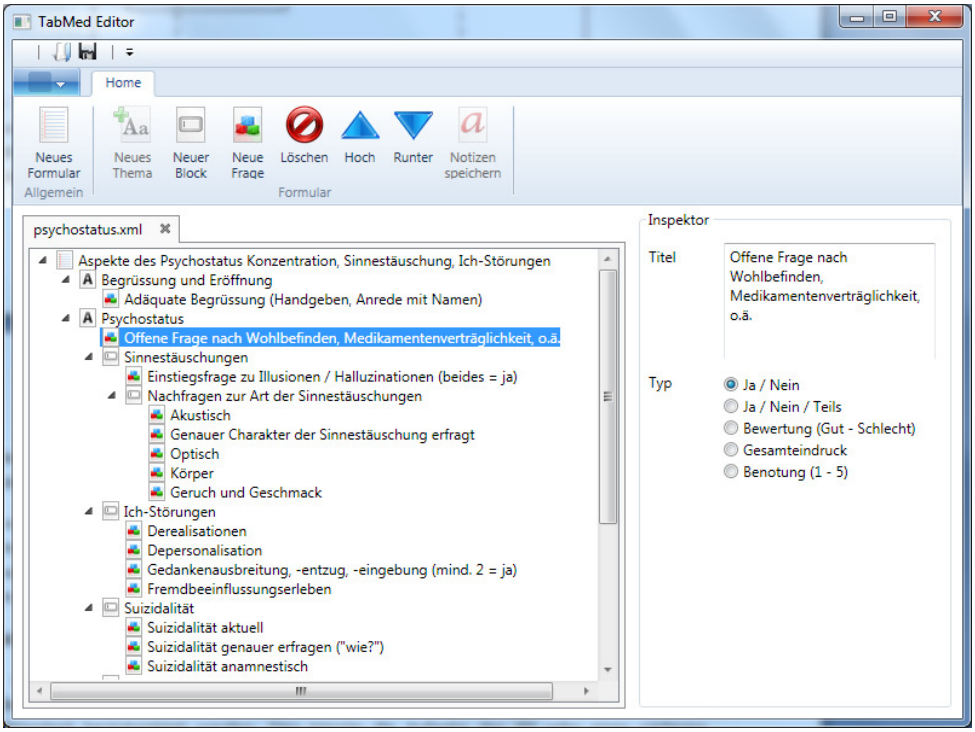
Das Team entwickelte ausserdem eine einfache Konsolenapplikation, genannt TabMedTerminal, um Rotationen zu starten. Diese Applikation funktioniert zusammen mit dem Mono-Framework[3] auch auf Linux- und Mac OS X-Betriebssystemen.



TabMedTerminal zum Starten von Prüfungen per Konsole

Damit die entwickelte Applikation möglichst einfach in den bestehenden Prüfungsprozess integriert werden kann, wurde eine Applikation entwickelt, die den Anwender unterstützt neue E-OSCE-Prüfungsformulare zu erstellen. So müssen die Benutzer keine komplizierten XML-Dateien von Hand erstellen und haben eine bessere Übersicht über die erstellte Prüfungsstruktur.

[3] http://www.mono-project.com



TabMedEditor zum Editieren und Erstellen von Prüfungsformularen

Erkenntnisse

Ein Schwerpunkt der Arbeit war die Erarbeitung des für die Experten besten User Interfaces. Wir hatten zwar Multitouchgeräte zur Verfügung, fanden allerdings keine guten Anwendungsmöglichkeiten für Multitouch-Gesten. Da das IML keine Zooming- oder Drag&Drop-Effekte sehen wollte, waren sogar die grundlegendsten Techniken ausgeschlossen. Aus diesem Grund entwickelten wir die Applikation mit einfachen Singletouch-Ereignissen. Deren Programmierung erweist sich in der Realität fast genau gleich wie wenn mit einer Maus gearbeitet würde. Es gibt einige spezielle Fälle, bei denen man auf Stylus- anstatt Mouse-Events reagieren muss.

Eine weitere Entscheidung war die Wahl der Webservice-Technologie. Wir haben auf eine REST-basierte HTTPS-Schnittstelle gesetzt, was wir im Nachhinein nie bereuten. Es stellte sich heraus, dass es auch ohne schwergewichtigen Schnittstellen-Technologien möglich ist, Webservices einfach auf mehreren Plattformen zur Verfügung zu stellen und zu konsumieren.

Zu Beginn gingen wir davon aus, dass das Verschlüsseln der Kommunikation und das Signieren der Formular-Dateien ein grosser Aufwand werden würde. Da wir gute Middleware und Frameworks eingesetzt haben, mussten wir überrascht feststellen, dass diese Funktionalität ohne grossen Aufwand implementiert werden konnte.

Ausblick

Mit den aktuell vorhandenen Applikationen kann jetzt schon eine Prüfung durchgeführt werden. Die Prüfungsvorbereitung und Auswertung kann allerdings sehr aufwändig werden, da noch keine Komponenten dafür vorhanden sind und alle Arbeiten manuell durchgeführt werden müssen.

Aufgabenstellung SA Fabian Mettler, Rico Suter

TabMed - Tablet PC Client-Server Applikation für medizinische Evaluationen

1. Auftraggeber und Betreuer

Diese Studienarbeit findet in Zusammenarbeit mit der Institut für Medizinische Lehre (IML) Bern.

Ansprechpartner Auftraggeber:

Philippe Zimmermann (IML) philippe.zimmermann@iml.unibe.ch
Institut für Medizinische Lehre
Direktion Institut für Medizinische Lehre
Konsumstrasse 13
3010 Bern

Betreuer HSR:

Prof. Dr. Markus Stolze, Institut für Software mstolze@hsr.ch

1. Ausgangslage

Bei strukturierten praktischen Prüfungen in der Ausbildung von Ärzten beurteilt ein Fachexperte mit Hilfe eines Fragebogens die Fähigkeiten eines Kandidaten während einer simulierten Konsultation (sog. OSCE-Prüfungen: Objective Structured Clinical Evaluation). Um die Effizienz und Datenqualität der Beurteilungen zu verbessern, soll in Zukunft der Fragebogen nicht mehr auf Papier, sondern direkt elektronisch ausgefüllt werden.

Das Institut für Medizinische Lehre (IML) der Universität Bern und das Institut für Software (IFS) der HSR, mit Unterstützung von Markus Fretz der Firma Crealogix, arbeiten gemeinsam an einer geeigneten Informatik-Lösung. In der Bachelorarbeit MoMEA wurde die Clientsoftware versuchsweise auf einem iPhone realisiert. Erste Resultate aus Usability Tests zeigten, dass die Bildschirmgrösse für die teilweise bereits älteren Fachexperten problematisch ist (d.h. Schrift- und Tastaturgrösse). Ausserdem werden die vorstrukturierten Evaluationsformulare nicht sequentiell ausgefüllt, sondern in der Folge des Auftretens der Ereignisse in der Kommunikation zwischen Student und Patient.

2. Ziele der Arbeit

Basierend auf der Bachelorarbeit MoMEA soll eine Client-Server Applikation für OSCE-Prüfungen entwickelt werden. Für die Client-Applikation ist ein Net- oder Notebook mit Stift- und/oder Touch-Eingabe vorgesehen. Die zu verwendenden Technologien für Client und Server sollen durch eine Studie abgeklärt werden. Die Server-Applikation ist über geeignete Interfaces an die Umgebung des

IML anzubinden. Es ist geplant, die Applikation mit entsprechenden Tests (insbesondere bezüglich Usability) auf ihre Praxistauglichkeit zu prüfen.

3. Zur Durchführung

Mit den HSR-Betreuern finden in der Regel wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen. Besprechungen mit dem Auftraggeber werden nach Bedarf durchgeführt.

Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das den Betreuern, dem technischen Berater und dem Auftraggeber per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

4. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in 5 Exemplaren abzugeben. Auf Wunsch ist für den Auftraggeber eine gedruckte Version zu erstellen. Zudem ist eine kurze Projektresultatdokumentation im Wiki von Prof. Stolze zu erstellen. Weiterhin erwünscht ist die Erstellung eines kurzen Videos oder einer Camtasia Demo sowie die Erstellung einer Kurzzusammenfassung (bis maximal 5 Seiten) welche die wichtigsten Resultate und Erkenntnisse der Arbeit zusammenfasst.

5. Termine

Siehe auch Terminplan auf https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html

Montag, den 14. September 2009	Beginn der SA, Ausgabe der Aufgabenstellung durch die Betreuer
15.9.2009 16:00-17:00	Kick-off Sitzung mit Auftraggeber an HSR 6.112
18.12.2009	Abgabe Kurzbeschreibung an das Abteilungssekretariat mit folgendem Formular gemäss https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html
Freitag, den **18. Dezember 2009, 17:00**	**Abgabe** des Berichtes an die Betreuer

Weitere Termine zur Abgabe von Postern und öffentlichen Präsentation der SA sind von den Studenten am Sekretariat der Abteilung Informatik zu erfragen und sollten entsprechend in einem Sitzungsprotokoll dokumentiert werden.

6. Beurteilung

Eine erfolgreiche SA zählt 8 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Siehe auch http://unterricht.hsr.ch/staticWeb/allModules/10939_M_BAI.html für die Modulbeschreibung der Bachelorarbeiten.

Für die Beurteilung sind die HSR-Betreuer verantwortlich unter Einbezug des Feedbacks des Auftraggebers.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/5
2. Berichte (Abstract, Mgmt Summary, techn. u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/5
3. Inhalt*)	3/5

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit festgelegt.

Im Übrigen gelten die Bestimmungen der Abt. Informatik zur Durchführung von Studienarbeiten.

Rapperswil, den 17. Dezember 2009

Prof. Dr. Markus Stolze
Institut für Software
Hochschule für Technik Rapperswil

18 Dezember | 09

TabMed – Projektplan

Rico Suter & Fabian Mettler

Inhaltsverzeichnis

Einführung

Zweck

Dieses Dokument beschreibt den Projektplan für das Projekt „TabMed“.

Gültigkeitsbereich

Dieses Dokument gilt als Grundlage für das ganze Projekt TabMed und hat deshalb Gültigkeit über die gesamte Projektdauer.

Definitionen und Abkürzungen

Definitionen und Abkürzungen sind im Dokument „Glossar“ zu finden.

Referenzen

- Product Backlog: Dokument „Projektmanagement\ProductBacklog.xls“
- Wiki: http://sinv-56028.edu.hsr.ch/wiki/

Übersicht

In den nachfolgenden Kapitel werden die folgenden Themen beschrieben: Projekt Übersicht, Projektorganisation, Management Abläufe, Product Backlog, Sprint Backlog, Impediment Backlog, Infrastruktur, Qualitätsmassnahmen.

Projekt Übersicht

Ausgangslage

Bei strukturierten praktischen Prüfungen in der Ausbildung von Ärzten beurteilt ein Fachexperte mit Hilfe eines Fragebogens die Fähigkeiten eines Kandidaten während einer simulierten Konsultation (sog. OSCE-Prüfungen: Objective Structured Clinical Evaluation). Um die Effizienz und Datenqualität der Beurteilungen zu verbessern, soll in Zukunft der Fragebogen nicht mehr auf Papier, sondern direkt elektronisch ausgefüllt werden.

Das Institut für Medizinische Lehre (IML) der Universität Bern und das Institut für Software (IFS) der HSR, mit Unterstützung von Markus Fretz der Firma Crealogix, arbeiten gemeinsam an einer geeigneten Informatik-Lösung. In der Bachelorarbeit MoMEA wurde die Clientsoftware versuchsweise auf einem iPhone realisiert. Erste Resultate aus Usability Tests zeigten, dass die Bildschirmgrösse für die teilweise bereits älteren Fachexperten problematisch ist (d.h. Schrift- und Tastaturgrösse). Ausserdem werden die vorstrukturierten Evaluationsformulare nicht sequentiell ausgefüllt, sondern in der Folge des Auftretens der Ereignisse in der Kommunikation zwischen Student und Patient.

Inzwischen wurde von Switch ein Forschungsprojekt bewilligt, um eine E-OSCE Applikation zu entwickeln mit Schwergewicht auf Usability Aspekten. Auf diesem Forschungsprojekt arbeiten von HSR-Seite die beiden IFS-Assistentn Kevin Gaunt und Michael Graf. Ihre Aufgabe besteht unter anderem darin, eine geeignete Umgebung und entsprechende Schnittstellen für diese Studienarbeit bereitzustellen.

Ziele und Aufgabenstellung

In dieser Arbeit soll ein Client-Server System für die Abwicklung von OSCE-Prüfungen (Observed Structured Clinical Examination) entwickelt werden. Ziel soll einerseits sein, am Ende der Arbeit einen funktionstüchtigen und benutzerfreundlichen Prototypen eines Clients zu präsentieren, welcher dem Prüfungsexperten ermöglicht, OSCE-Prüfungen vollständig digital zu beurteilen. Andererseits ist eine Server-Komponente zu entwickeln, welche die Prüfungsdurchführung steuert.

Für die Arbeit stehen mehrere Client-Devices mit Windows Betriebssystem zur Verfügung. Je nach Möglichkeit kann dies z.B. auf Windows 7 vereinheitlicht werden. Für die Entwicklung des Client-Prototypen empfiehlt sich Microsoft .NET.

Dem Server kommen verschiedene Aufgaben zu: Vor Prüfungsbeginn gilt es alle für die Prüfung relevanten Aspekte vorzubereiten. Während der Durchführung synchronisiert der Server die verschiedenen Clients. Nach Prüfungsende müssen die abgelegten Prüfungsergebnisse wieder in den bestehenden IML Webpool integriert werden. Die für die Studienarbeit relevante Funktionalität beschränkt sich auf das Modul der E-OSCE Durchführung.

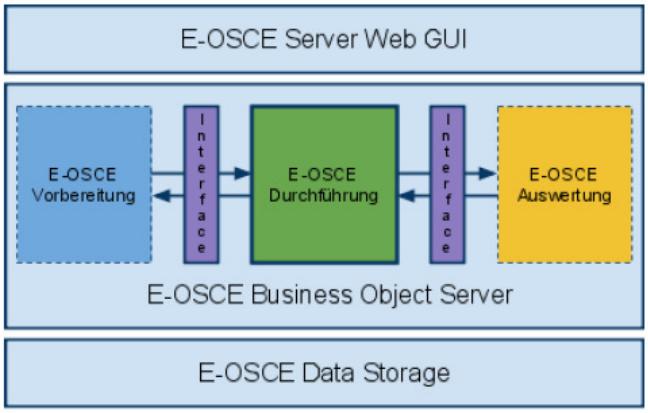


Abbildung: E-OSCE Server

Bezüglich Technologien soll auf Server-Seite auf Java gesetzt werden, die Details sind noch offen.

Annahmen und Einschränkungen

Pro Woche und Projektmitarbeiter wird eine Soll Arbeitszeit von 16 Stunden erwartet. Treten unerwartete Aufwände oder Probleme auf, kann die Arbeitszeit um 2-3 Stunden erhöht werden.

Projektorganisation

Das Projektteam besteht aus zwei gleichgestellten Personen – Fabian Mettler und Rico Suter, wobei jedes Teammitglied für ein spezielles Aufgabengebiet verantwortlich ist. Dies bedeutet allerdings nicht, dass jedes Mitglied nur an seinen eigenen Verantwortlichkeiten arbeitet.

Organisationsstruktur

Projektmitglied	Verantwortlichkeit
Rico Suter	Product Owner, Entwicklung
Fabian Mettler	Entwicklung, Infrastruktur
Kevin Gaunt	Scrum Master

Externe Schnittstellen

Diese Studienarbeit findet in Zusammenarbeit mit dem *Institut für Medizinische Lehre (IML)* der Universität Bern sowie der Firma *Crealogix* statt.

Ansprechpartner Auftraggeber

- Dr. Philippe Zimmermann, Institut für Medizinische Lehre, Universität Bern, philippe.zimmermann@iml.unibe.ch

Technischer Berater

- Markus Fretz, CREALOGIX AG, markus.fretz@crealogix.com

Betreuer HSR

- Prof. Dr. Markus Stolze, Institut für Software mstolze@hsr.ch
- Prof. Hans Rudin, Institut für Software hrudin@hsr.ch
- Kevin Gaunt, Assistent IFS, kgaunt@hsr.ch
- Michael Graf, Assistent IFS, mgraf@hsr.ch

Management Abläufe

Projekt Kostenvoranschlag

Dem Projekt stehen pro Woche und Teammitglied 16 Stunden zur Verfügung. Der Projektstart wurde auf den 14. September 2009 festgelegt und der Abgabetermin auf den Freitag den 18. Dezember 2009. Sollten unerwartete Probleme auftreten, kann die Arbeitszeit pro Woche um 2-3 Stunden erhöht werden.

Projektplan

Software-Entwicklungsmethode

Als Software-Entwicklungsmethode wird Scrum verwendet.

Sprints

Zentrales Element des Entwicklungszyklus von Scrum ist der Sprint. Ein Sprint bezeichnet die Umsetzung einer Iteration.

Sprints Übersicht

Eine Übersicht über die Sprints kann auf folgender Wiki-Seite gefunden werden:
http://sinv-56028.edu.hsr.ch/wiki/index.php/Sprints

Besprechungen

An jedem Montag und Dienstag findet ein kurzes 15-minütiges Daily Scrum mit dem Teammitgliedern statt.

Zu Beginn eines neuen Sprints, findet an jedem Montag um 14:00 das Sprint Review Meeting und die Sprint Retrospective statt. Danach wird mit dem Sprint Planning Meeting der neue Sprint geplant. Bei diesen Sitzungen ist der ScrumMaster, Kevin Gaunt, und eventuell weitere HSR-Betreuer anwesend.

Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das den Betreuern, dem technischen Berater und dem Auftraggeber per E-Mail zugestellt wird.

Meilensteine

Meilenstein	Datum	Produkt
M1.1: Projektplan	28.09.2009	Projektplannung
M1.2: Beta 1	26.10.2009	UI Prototyp
M2.1: Beta 2	23.11.2009	Architektur-Prototyp
M2.2: Beta 3	18.12.2009	Server, Client und Frontend

Artefakte

Meilenstein	Datum	Artefakt
M1.1: Projektplan	28.09.2009	Projektplan
M1.2: Beta 1	26.10.2009	External Design
M2.1: Beta 2	23.11.2009	Software Architektur Dokument
M2.2: Beta 3	18.12.2009	Systemtests-Dokumentation, Usabiltytests-Dokumentation

Product Backlog

Beschreibung

Der Produkt Backlog enhält die Features des zu entwicklenden Produkts. Es umfasst alle Funktionalitäten inklusive technischer Abhängigkeiten.

Product Backlog: Dokument: „Projektmanagement\ProductBacklog.xls“

Sprint Backlog

Beschreibung

Der Sprint Backlog enhält alle Aufgaben, die notwending sind, um das Ziel eines Sprints zu erfüllen.

Der Sprint Backlog ist in der Detail-Ansicht eines Sprints zu finden.
Beispiel: http://sinv-56028.edu.hsr.ch/wiki/index.php/Sprint_1

Übersicht über alle Sprints:
http://sinv-56028.edu.hsr.ch/wiki/index.php/Sprints

Impediment Backlog

Beschreibung

Der Impediment Backlog ist eine Liste von Hindernissen, die das Team daran hindern effizient seine Aufgaben zu erledigen oder das Sprint Ziel zu erreichen.

Der Impediment Backlog ist unter dem folgenden Link zu finden:
http://sinv-56028.edu.hsr.ch/wiki/index.php/Impediment_Backlog

Infrastruktur

Räumlichkeiten

Für gemeinsames Arbeiten und Sitzungen steht uns ein Arbeitsplatz an der HSR zur Verfügung.

Hardware

Server

Die HSR stellt einen virtuellen Server mit folgendener Konfiguration zur Verfügung:

- Windows Server 2003 Standard SP2, 512 MB RAM, 20 GB Disk Drive, 1 Gbps Network

Tablet PC

Für das Projekt wurden drei Client-Devices erworben, deren Tauglichkeit im Verlauf des Projekts evaluiert werden soll.

Geräte-Bezeichnung	Eigenschaften
Asus Eee PC T91	9" Touch Screen, Bedienung per Tastatur/Finger/Stift, ca. 5h Akku, Windows XP Home, 960g, 10/100 Ethernet, Wireless, Bluetooth, USB2, MMC/SD Card-Slot
HP EliteBook 2730p	12.1" TFT Screen, 1280x800, Bedienung per Stift, ca. 6h Akku, Gigabit Ethernet, Wireless, 3G, 1.7kg, USB2, SD Card-Slot
Dell Latitude XT2	12.1" WXGA-LED Screen, Multitouch/Stift/Tastatur Bedienung, Gigabit Ethernet, Wireless, Bluetooth, HSDPA, USB2

Private Notebooks

Primär arbeiten die Projektmitglieder mit ihren privaten Notebooks.

Software

Betriebssystem

- Microsoft Windows 7 Professional
- Microsoft Windows Server 2003 Standard SP2

Programmiersprachen

- Java 6
- C# 3.5 SP1

Entwicklungswerkzeuge

- Netbeans 6.8
- Visual Studio 2008 SP 1

Versionsverwaltung

- Subversion

Projektverwaltung & Dokumentation

- Trac
- MediaWiki
- Microsoft Office 2007

Continuous Integration (Automatisierung)

- Hudson

Backup

Die Daten auf dem virtuellen Server werden nächtlich gesichert. Die Sicherung beinhaltet folgende Daten: SVN-Repository, Wiki, Trac.

Kommunikation

Es werden folgende Kommunikationsmittel eingesetzt um einen erfolgreichen Projektablauf zu unterstützen:

- E-Mail
- Skype (Messenger)
- Wiki
- Besprechungen (wie z.B. Daily Scrum)

Qualitätsmassnahmen

Dokumentation

Wir legen grossen Wert darauf, dass die Dokumentationen stehts aktuell sind. Um diesen Punkt besser zu unterstützen verwenden wir ein Wiki-System.

Sitzungsprotokolle

Sämtliche Sitzungen werden protokolliert. Dadurch wird garantiert, dass Kritik, Vorschläge, Anregungen und Entscheidungen schriftlich festgehalten sind.

Burndown-Diagramm

Das Burndown-Diagramm ist eine grafische, pro Tag zu erfassende Darstellung des noch zu erbringenden Restaufwands pro Sprint. Im Idealfall fällt die Kurse kontinuirlich und der Restaufwand ist am Ende des Sprints null. Das Diagramm lässt anhand der Verlängerung der negativen Steigung bereits während des Sprints erkennen, ob der anfangs geschätze aufwand umgesetzt werden kann.

Styleguides

Für die Dokumente wird eine einheitliche Dokumentvorlage verwendet. Der Programmcode muss nach den definierten Coderichtlinien geschrieben werden.

Zu finden im Wiki: http://sinv-56028.edu.hsr.ch/wiki/index.php/Coderichtlinien

Reviews

Dokumente

Die Dokumente sind vor einer Abgabe von allen Projektteilnehmern durchzulesen. Korrekturen sind im Team zu diskutieren und danach vorzunehmen. Die Dokumente werden zusätzlich von einem Revisor geprüft.

Quellcode

Der Review des Quellcode hat mehrere Ziele. Zum einen dient er dazu, die Quellcoderichtlinien zu überprüfen und durchzusetzen, zum Anderen fördert er die Motivation zum Erstellen von wartbarem Quellcode. Im Vordergrund steht somit die Verbesserung der Produktqualität.

Am Ende jedes Sprints wird ein Quellcode-Review durchgeführt. Die Ergebnisse eines Quellcode-Rewiew werden mit Review-Protokollen festgehalten. Durch dieses Vorgehen wird die Quellcode-Qualität erhöht. Ausserdem erhalten die einzelnen Teammitglieder einen Einblick in die Bereiche, die von ihren Teamkollegen bearbeiteten werden. Das erhöht das Verständnis der Zusammenarbeit der einzelnen Quellcode-Teile.

Folgende Kriterien werden untersucht:

- Einhalten von Standards
- Umsetzung der Anforderungen
- Umsetzung des Design
- Einhalten von Schnitstellenspezifikationen
- Wartbarkeit des Codes

Versionsverwaltung

Die gesamte Dateistruktur wird mit Subversion unter die Versionsverwaltung gestellt. Es werden nur Dokumente mit einem sinnvollen Zwischenstand und lauffähige Quellcodedateien in die Versionsverwaltung übergeben. Das ermöglicht allen Projektteilnehmern den Zugriff auf die aktuellen Dateien.

Durch die Verwendung einer Versionsverwaltung werden daher folgende Punkte erreicht:

- Protokollierungen der Änderungen
  - Es kann jederzeit nachvollzogen werden, wer wann was geändert hat.
- Wiederherstellung von alten Ständen einzelner Dateien

- Somit können versehentliche Änderungen jederzeit wieder rückgängig gemacht werden.

- Archivierung der einzelnen Stände eines Projekts
  - Dadurch ist es möglich, auf alle Versionen zuzugreifen.
- Koordinierung des gemeinsamen Zugriffs auf die Dateien

Tests

Unit-Tests

Während der Entwicklung werden Unit-Tests zur Unterstützung der Qualität und Wartbarkeit eingesetzt. Diese Tests müssen erfolgreich durchlaufen werden, bevor der Quellcode wieder der Versionsverwaltung übergeben wird.

System-Tests

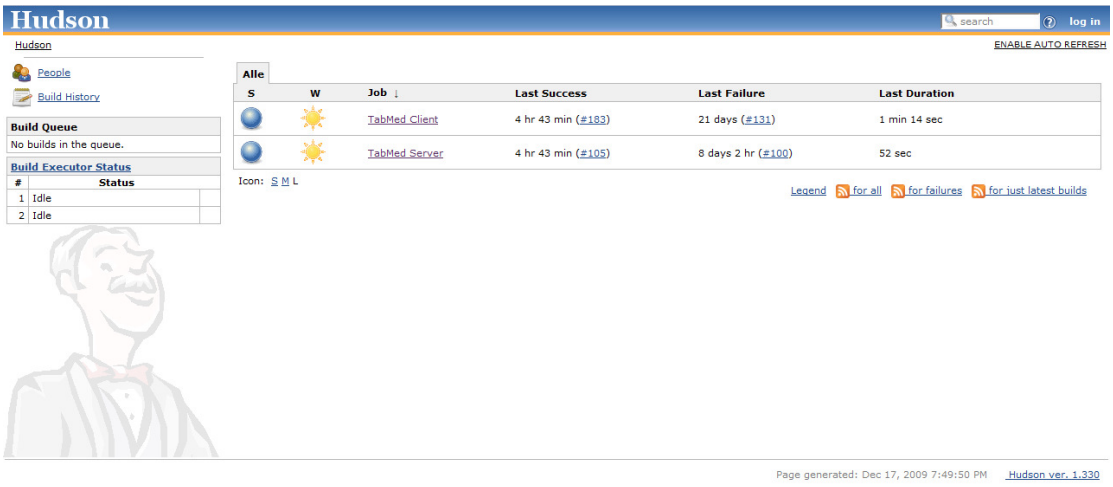
Auf den User Stories basierend werden Systemtests durchgeführt und protokolliert. Diese überprüfen, ob die Software die funktionalen Anforderungen erfüllen.

Usability-Tests

Ein wichtiger Bestandteil dieses Projektes ist die Usability für den Benutzer. Daher werden Usability-Tests durchgeführt und protokolliert. Das Feedback wird ausgewertet und in die weitere Entwicklung miteinbezogen.

Continuous Integration (Automatisierung)

Durch die Verwendung eines Build-Servers wird automatisch überprüft, ob der Quellcode kompilierbar ist und die Unit-Tests erfolgreich durchlaufen.



Zusätzlich ergeben sich folgende Vorteile:

- Integrations-Probleme werden laufend entdeckt und behoben
- Frühe Warnungen bei nicht zusammenpassenden Bestandteilen
- Sofortige Unit-Tests entdecken Fehler schnell
- Kostante Verfügbarkeit eines lauffähigen Standes für Demo-, Test- oder Vertriebszwecke

18 Dezember 09

TabMed – Domain Analyse

Rico Suter & Fabian Mettler

Dokumentinformationen

Änderungsgeschichte

Datum	Version	Änderung	Autor
04.11.2009	0.1	Erste Version	fme & rsu
11.11.2009	0.2	Anpassung Domain Model	rsu
03.12.2009	0.3	Review, Verbesserungen	fme
12.12.2009	0.4	Kleine Anpassungen	rsu
15.12.2009	1.0	Überarbeitung nach Review von Kevin Gaunt	fme

Inhaltsverzeichnis

Entwickelte Applikationen

Dieses Kapitel soll die verschiedenen zu entwickelnden Applikation in diesem Studienarbeit erläutern und welche Requirements sie erfüllen müssen. Für eine genauere Definition der Anforderungen wird auf das Dokument „Anforderungsspezifikation TabMed.pdf“ verwiesen.

TabMedClient

Mit dem Client greift ein Benutzer auf den Server zu, kann eine Prüfung wählen und die damit verbundenen Prüfungsbögen ausfüllen. Die Applikation sollte dabei auf den Experten und dessen Fähigkeiten zugeschnitten sein.

TabMedServer

Der TabMedServer ist eine Java-Applikation, die vorzugsweise im Netzwerk des IML ausgeführt wird. Sie soll eine einfach zu verwendende REST[1]-Schnittstelle dem TabMedClient und MoMEA-Client zur Verfügung stellen. Mit dem Server können Prüfungen erstellt, gestartet und Prüfungsresultate gesammelt werden.

TabMedEditor

Mit dem TabMedEditor können Prüfungsformulare erstellt werden. Diese Applikation von den Prüfungsdesignern verwendet um neue Prüfungen zu erstellen und zu optimieren.

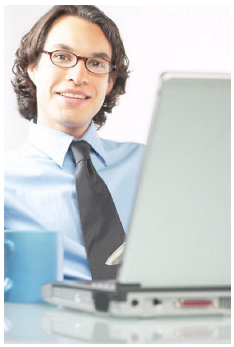
TabMedTerminal

Das TabMedTerminal ist eine einfache Konsolen-Applikation um auf vorhandene Server zuzugreifen und darauf vorhanden Prüfungen und deren Rotationen zu starten. Es ist zwar eine Konsolen-Applikation, sie ist allerdings so einfach gestaltet, dass sie nach einer kurzen Einführung von durchschnittlichen Computerbenutzern verwendet werden kann.

[1] http://en.wikipedia.org/wiki/Representational_State_Transfer

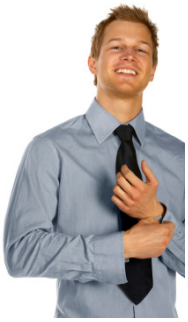
Personas

Erich Tschudi, Administrator



Erich Tschudi ist 35 Jahre alt und arbeitet als IT-Supporter beim IML. Bei OSCE-Prüfungen ist er dafür verantwortlich, dass die Prüfungen ordnungsgemäss durchgeführt werden.

Markus Frei, Experte

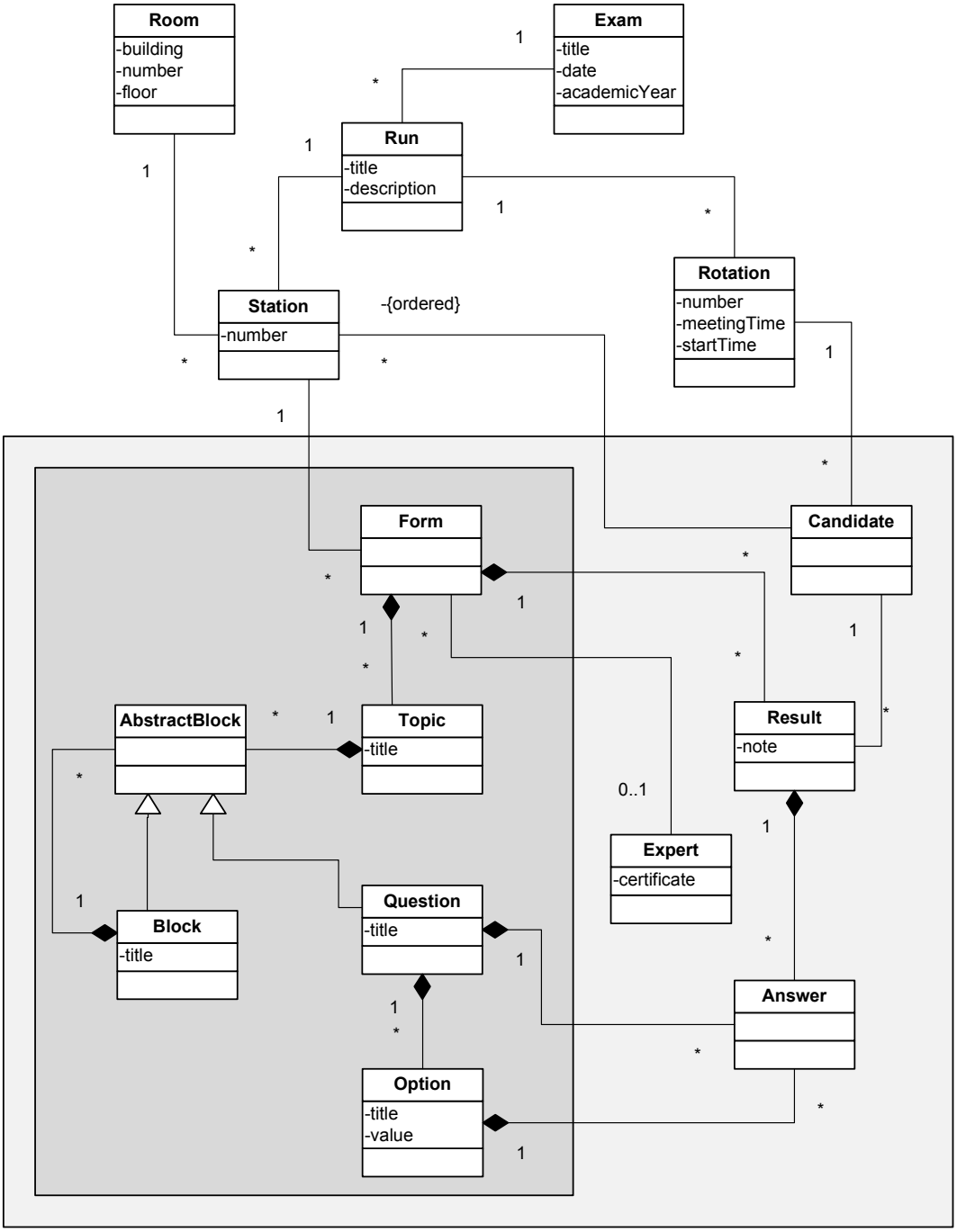


Markus Frei ist 42 Jahre alt und hauptberuflicher Dozent an der medizinischen Fakultät in Bern. Werden OSCE-Prüfungen durchgeführt, wird er oft angefragt, ob er als Experte fungieren möchte – und ist meist dabei. Er ist ein durchschnittlicher Computer-Anwender und verwendet hauptsächlich den Webbrowser und Programme wie Microsoft Word.

Domain Modell

Strukturdiagramm

Nachfolgend wird ein Diagramm der Domäne des TabMed-Projektes dargestellt. In den darauffolgenden Kapiteln werden die wichtigsten Domain-Klassen näher beschrieben und Konzepte zum Diagramm erklärt.



Die beiden Rahmen im Diagramm stellen die zwei möglichen „Zustände“ der Formulare dar. Der dunkle Rahmen zeigt das nicht ausgefüllte Formular, ohne Antworten, Kandidat- und

Experten-Informationen. Im hellen Rahmen ist das komplett ausgefüllte Formular dargestellt wie es auf dem Server wieder abgelegt wird nach der Bearbeitung.

Der Experte und das dazugehörige Zertifikat werden von einem USB-Stick eingelesen und dem ausgefüllten Formular angehängt. Desweiteren wird das Formular bei jedem Speicherungsvorgang automatisch signiert um das Dokument vor Fälschungen zu sichern.

Alle dargestellten Klassen, die nicht zum Formular gehören, werden grundsätzlich vom Server verwaltet und vom Client nur dargestellt. Einzige Ausnahme ist natürlich die „Answer“-Klasse.

User Stories

Beschreibung

Eine User Story ist eine in Alltagssprache formulierte Software-Anforderung. Sie setzt sich in diesem Projekt aus folgenden Attributen zusammen:

Attribut	Beschreibung
ID	Jede User Story besitzt eine eindeutige Identifikationsnummer.
Titel	Der Titel der User Story, der kurz und knapp die Anforderung beschreibt.
Sprint	Eine User Story ist zu einem bestimmten Sprint zugeordnet.
Beschreibung	Beschreibung der User Story, welche nicht mehr als zwei Sätze umfasst.
Arbeitspakete	Jede User Story wird in mehrere Arbeitspakete unterteilt.

TabMedClient

S6 Prüfung auswählen **Sprint 2**

Der Benutzer kann aus einer Liste von Prüfungen auswählen.

Arbeitspakete:

- Paper Prototype (External Design)
- Entwicklung UI Prototyp für "Prüfung auswählen“
- Domain Model so weit als nötig implementieren

S8 Station auswählen **Sprint 2**

Der Benutzer kann aus einer Liste von Stationen auswählen.

Arbeitspakete:

- Paper Prototype (External Design)
- Entwicklung UI Prototyp für "Station auswählen"
- Domain Model so weit als nötig implementieren

S3 Benutzer verbindet sich mit Server **Sprint 2**

Der Benutzer kann eine Server-Adresse eingeben und sich mit diesem Server verbinden.

Arbeitspakete:

- Login View erstellen
- REST-Schnittstelle / -Klasse implementieren

S15 Auf Prüfungsstart warten **Sprint 2**

Die Client-Applikation kann auf das Start-Signal des Servers warten und beim Empfang des Signals die jeweilige Rotation starten.

Arbeitspakete:

- Waiting View erstellen
- Long Polling (siehe SAD) implementieren

S9 Prüfungsbogen ausfüllen	**Sprint 3**

Der Benutzer kann das Prüfungsformular des aktuellen Kandidaten ausfüllen. Dabei soll das Bearbeiten des Formulars so einfach und intuitiv wie möglich sein.

Arbeitspakete:

- Paper Prototype (External Design) erstellen
- Entwicklung UI Prototyp für "Prüfung ausfüllen"
- Prüfungsformulare (Form XML, siehe SAD) parsen

S11 Prüfungsbogen speichern	**Sprint 3**

Beim Verlassen des aktuellen Formulars soll das Formular lokal und auf dem USB Stick gespeichert werden.

Arbeitspakete:

- Speichern des Prüfungsbogen als XML

S10 Kandidat auswählen	**Sprint 3**

Der Experte wählt einen Kandiaten aus der Kandidatenliste der aktuellen Durchführung aus und kann ihn danach bearbeiten (User Story S9).

Arbeitspakete:

- Paper Prototype (External Design) erstellen
- Entwicklung UI Prototyp für "Kandidat auswählen"

S13 Prüfungsbogen übermitteln	**Sprint 4**

Das gespeicherte Formular (User Story S11) wird an den Server übermittelt.

Arbeitspakete:

- Übermitteln mit REST-Klasse

S14 Benutzer kann sich an- und abmelden	**Sprint 4**

Der Benutzer kann sich am System an- und abmelden. Dabei soll ein USB-Stick verwendet werden. Ist der USB-Stick eingesteckt, ist der Benutzer angemeldet, wird er herausgezogen, wird der Bildschirm gesperrt.

Arbeitspakete:

- Event wenn Device rein-/raus abfangen und scannen
- Applikation sperren, wenn kein USB Stick vorhanden ist

S16 Notizen	**Sprint 5**

Von der Ansicht zum Ausfüllen des Formulars (User Story S9) kann auf eine Notizen-Ansicht gewechselt werden. Die geschriebenen Notizen sollen dann in das XML-Formular (-Objekt) eingebettet werden.

Arbeitspakete:

- Schreibbarer Bereich erstellen
- Konvertierung zu GIF und Einbettung in XML

S12 Prüfungsbogen signieren **Sprint 5**

Beim Speichern der Formulare (User Story S9) sollen diese automatisch mit einem Private-Key auf dem USB-Stick signiert werden, sodass nachgewiesen werden kann, dass der richtige Experte die Prüfung gespeichert hat.

Arbeitspakete:

- Laden des Zertifikats vom USB-Stick
- Signierung des Prüfungsformulars mit dem Benutzerzertifikat

S34 Uhr implementieren **Sprint 5**

Eine Uhr soll die noch verbleibende Zeit für das Ausfüllen des aktuellen Formulars anzeigen.

Arbeitspakete:

- GUI Komponente designen und implementieren
- Zeitberechnung anhand der Startzeit

TabMedServer

S18 Server akzeptiert einkommende Verbindung **Sprint 2**

Der Server hört auf einkommende HTTPS-Verbindungen und kann sie an spezielle Listener weiterleiten.

Arbeitspakete:

- Schnittstellen definieren und dokumentieren
- Implementation der Schnittstelle als Gerüst

S24 Liste an den Client senden **Sprint 2**

Der Client kann die Prüfungs-, Durchlaufs- und Stationsliste vom Server abrufen.

Arbeitspakete:

- Domain Model implementieren
- Rückgabe der Stationenliste implementieren

S19 Prüfungsrotation starten **Sprint 2**

Die Prüfung soll per URL (REST) gestartet werden. Dabei soll es eine URL für den Client geben, bei dem mit Long Polling auf den Start gewartet wird, und eine URL um eine Rotation zu starten.

Arbeitspakete:

- Thread-Synchronization implementieren

S23 Kandidatenliste der aktuellen Rotation und Station an Client schicken **Sprint 3**

Nach der Auswahl der Station soll die Kandidatenliste für alle Rotationen und gewählte Durchführung vom Client heruntergeladen werden können.

Arbeitspakete:

- Erstellen der Kandidatenliste (XML)
- Abrufen der Kandiatenliste und parsen des XML
- Bereitstellen der Liste durch REST

S20 Prüfungsbogen auf Server speichern **Sprint 4**

Nach dem Ausfüllen (oder während) kann der Prüfungsbogen auf dem Server gespeichert werden.

Arbeitspakete:

- Daten per POST empfangen und korrekt speichern

S17 Prüfungen laden **Sprint 4**

Prüfungen von Verzeichnis einlesen.

Arbeitspakete:

- Definition und Anpassung des XML Schemas (Exam XML)
- XML-Dateien einlesen und parsen

S25 Status der Clients ermitteln **Sprint 6**

Die aktuell verbundenen Clients sollen mithilfe eines Timeouts gespeichert werden können.

Arbeitspakete:

- Implementation der Liste
- Konfiguration (Timeout-Länge) in Datei auslagern

TabMedEditor

S32 Prototyp erstellen **Sprint 4**

Hier soll ein erster, einfacher Prototyp des Editors erstellt werden.

Arbeitspakete:

- Paper Prototype (External Design) erstellen
- Gemeinsame Klassen (TabMedEditor und TabMedClient) in gemeinsame DLL auslagern
- Prototyp entwickeln

S33 Verbessern **Sprint 4**

Prototyp (User Story 32) erweitern.

Arbeitspakete:

- Refactorings
- Ribbon User Interface implementieren

TabMedTerminal

S27 Prüfungsrotation starten **Sprint 6**

Das Terminal kann sich mit einem Server verbinden, die aktuellen Prüfungen anzeigen und eine Prüfungsrotation starten.

Arbeitspakete:

- Verbindung mit dem Server
- Auflisten der vorhanden Prüfungen
- Prüfungen starten

TabMedFrontend

S28 Prüfung hochladen	**Sprint 6**

Der Benutzer kann mit der Applikation ein Verzeichnis auswählen, in dem eine vorhandene Prüfung enthalten ist. Das TabMedFrontend lädt die Prüfung dann automatisch auf den Server.

Arbeitspakete:

- Prüfungs-Dialog erstellen
- Prüfung auf den Server laden

S26 Status der Clients anzeigen	**Sprint 6**

Die Applikation kann auf dem Server eine Liste der aktuell verbundenen Clients abrufen und darstellen.

Arbeitspakete:

- Status vom Server abfragen
- Status im GUI anzeigen

S31 Benutzer anmelden	**Sprint 6**

Beim Verbinden soll der Benutzer ein Passwort für die Anmeldung auf dem Server angeben können. Der Server kann nur mit dem korrekten Passwort verwaltet werden.

Arbeitspakete:

- SSL und Authentifizierung auf dem Server einrichten
- REST-Klassen für Anmeldung anpassen

18 Dezember 09

TabMed – External Design

Rico Suter & Fabian Mettler

Dokumentinformationen

Änderungsgeschichte

Datum	Version	Änderung	Autor
11.11.2009	0.1	Erste Version	fme & rsu
19.11.2009	0.2	Kleine Anpassungen	rsu
03.12.2009	0.3	Review, Verbesserungen	fme
12.12.2009	0.4	Kleine Anpassungen	rsu
15.12.2009	0.5	Ergänzungen nach Review von Kevin Gaunt	fme
17.12.2009	1.0	Endgültie Korrektur	rsu

Inhaltsverzeichnis

Einleitung

In diesem Dokument wird das GUI des TabMedClients näher erläutert. Es wird gezeigt, wie die Views zu erreichen sind, welche Prototypen entwickelt wurden und anhand welcher Kriterien die definitive Prototyp-Auswahl gemacht wurde.

Die Entscheidungen wurden in erster Linie mit dem IFS getroffen. Unsicherheiten bezüglich Benutzung des Systems wurden auch direkt mit dem IML geklärt. So wurden mehrere Sitzungen direkt mit dem IML abgehalten um aktuelle Prototypen vorzuführen und wichtige Entscheidungen zu treffen.

In diesem Dokument werden oft englische Bezeichnungen aus der Domäne verwendet. Die Bedeutung dieser Wörter kann im Dokument „Domain Analyse“ nachgeschlagen werden.

Varianten

In den folgenden Kapiteln werden verschiedene Varianten für das GUI evaluiert, wobei unter anderem mit Usability-Experten vom IML die Vor- und Nachteile erarbeitet wurden.

Perspective Wall

Eine Möglichkeit für das GUI war das Implementieren einer Perspective Wall[1]. Mit dieser Technik steht nur ein bestimmter Ausschnitt als Arbeitsbereich zur Verfügung: Zum Beispiel in Form eines Balken von 5 cm Höhe, welcher den Arbeitsbereich definiert. Da mit dieser Technik nur ein bestimmter Ausschnitt eines Formularblocks ausgefüllt werden kann, kommt diese Variante nicht in Frage. Die Experten müssen ganze Frageblöcke möglichst einfach ausfüllen können, ohne das vorherige Verschieben einer „Lupe“.

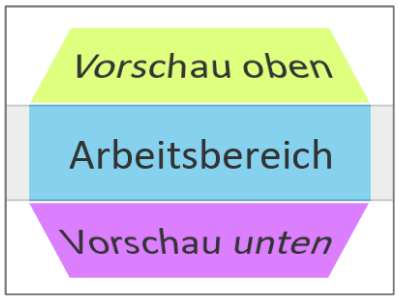


Abbildung: Perspective Wall

Semantic Zoom

Eine weitere Möglichkeit war der Einsatz einer Technik die sich Semantic Zoom[2] nennt. Dabei werden die Daten je nach Detailgrad mit mehr oder weniger Informationen angezeigt und wenn der Benutzer genauere Informationen möchte, kann er einzelne Objekte näher betrachten. Dies haben wir auch so umgesetzt im Prototyp 1 und noch stärker im Prototyp 2. Unsere Testpersonen vom IML waren davon allerdings nicht begeistert. Das Zooming von Formularblöcken erzielte nämlich einen negativen Effekt, da die Blöcke innerhalb des Bildschirms hin und her sprangen, wenn ein Block sehr gross war. Für die Experten des IML war dieses Springen mit Nervosität und Unruhe gleich zu setzen, was dazu führte, dass wir im fertigen Protypen komplett auf visuelle sowie informationsbezogene Zoomingeffekte verzichten.

Abbildung: Semantic Zoom

Querformat oder Hochformat

Zu Beginn des Projekt war es wichtig zu entscheiden, ob man die Applikation im Querformat oder Hochformat benutzen möchte.

Querformat (Portrait Mode)

Vorteile

- Es lassen sich viele Informationen nebeneinander darstellen:
  - Prüfungsbogen und Notizen
  - Informationen in Tabellen / Listen (Spalten)
  - Statusinformationen oben und unten: Statusbar

[1] http://portal.acm.org/citation.cfm?id=108870
[2] http://en.wikipedia.org/wiki/Zooming_user_interface

Nachteile

- Der ganze Prüfungsbogen hat nicht vollständig Platz, da der Bildschirm nicht genug hoch ist. Dies bedingt eine Scroll-Operation des Benutzers.

Hochformat (Landscape Mode)

Vorteile

- Ein Prüfungsbogen kann wie auf einer A4-Seite knapp auf dem ganzen Bildschirm angezeigt werden
- Der Tablet-PC liegt besser in der Hand: Intuitiv wie bei einem Papierblock

Nachteile

- Durch die Verwendung des Hochformats lassen sich keine Informationen nebeneinander darstellen, da zu wenig Platz vorhanden ist (die horizontalen Topic-Anzeigen mit Topic-Titel im Prototype #3 sind nicht möglich)

Fazit

Da die Vorteile, wie auch die Nachteile, der beiden Formate mehr oder weniger ausgeglichen sind, wurden zwei unterschiedliche Prototypen (Quer-/Hochformat) entwickelt. Diese beiden Prototypen wurden den Prüfungsexperten des IML vorgestellt, um die Richtung der weiteren Entwicklung zu bestimmen. Die Prüfungsexperten haben sich vor allem für das Hochformat positiv ausgesprochen, da eine gute Gesamtübersicht über die Prüfung gewährleistet ist. Das bedeutet, dass man so viele Informationen (z.B. Bewertungskriterien) wie möglich von einer Prüfung sehen sollte.

GUI Map

In der folgenden Abbildung sind die einzelnen Ansichten des Clients zu sehen. Die ersten vier Einrichtungsschritte werden grundsätzlich von einem Administrator ausgeführt. Nach dem Auswählen der Station wartet die Applikation auf den globalen Prüfungsstart, der vom Server (TabMedServer) ausgelöst wird. Die eigentliche Prüfung findet in den letzten drei Ansichten statt. Aus diesem Grund müssen sie sehr einfach und schnell bedienbar sein.

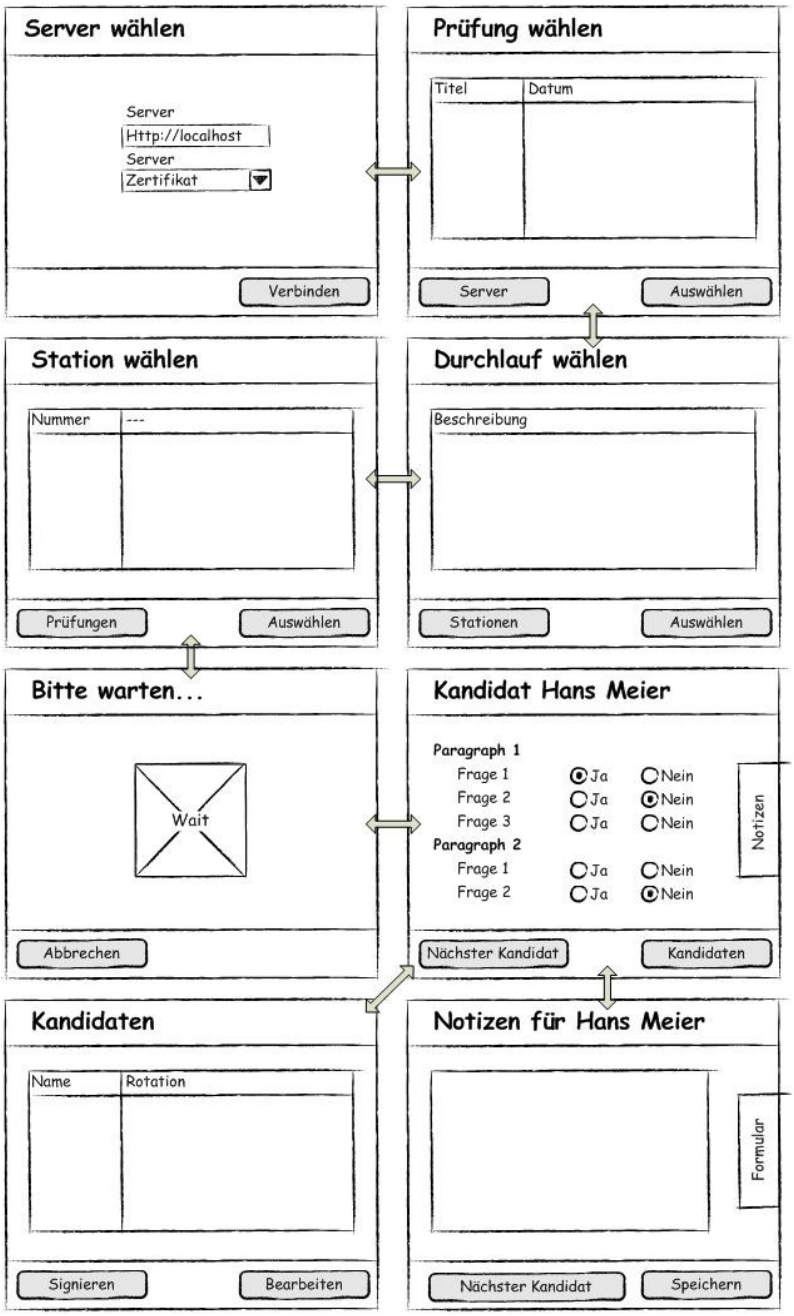


Abbildung: Schematischer Ablauf einer Prüfungssession

Anmerkung: In der Abbildung fehlen noch einige Ansichten, wie zum Beispiel der Lock-Screen beim Fehlen des USB-Sticks oder allfällige Fehlermeldungen. Die ersten vier Bildschirme werden meist von den Administratoren bedient, alle anderen von den Experten.

Prototypen

Vor der Evaluation durch Experten des IML wurden drei verschiedene, einfache Prototypen entwickelt. Am 2. November 2009 wurden diese Prototypen dann in Bern vorgestellt. Dort entschied man sich für die Weiterentwicklung des ersten Prototypen (Prototype #1).

In den folgenden Kapiteln werden die entwickelten Prototypen vorgestellt und deren Vor- und Nachteile aufgezeigt.

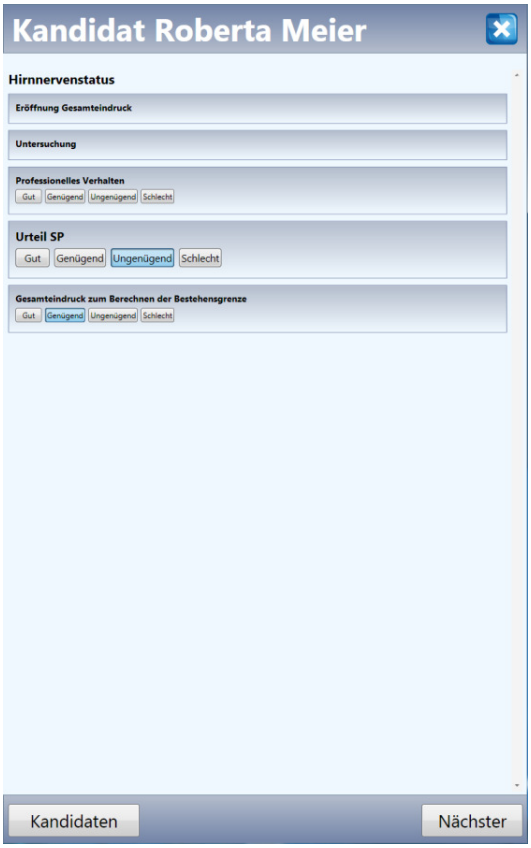
Prototype #1 – Hochformat

Hier sieht man den ersten Prototyp. Es wird pro Topic ein Block erstellt. Es können einzelne Blöcke vergrössert werden, wobei alle anderen in den kleinen Zustand zurückspringen.

Dieser Prototyp wurde schlussendlich als die zu weiterentwickelnde Variante gewählt. Alle Zooming-Effekte und unnötigen Animationen (wie zum Beispiel das Vergrössern eines Blockes) sollen dabei allerdings vermieden werden.

Abbildung:
Formular mit vergrössertem Topic

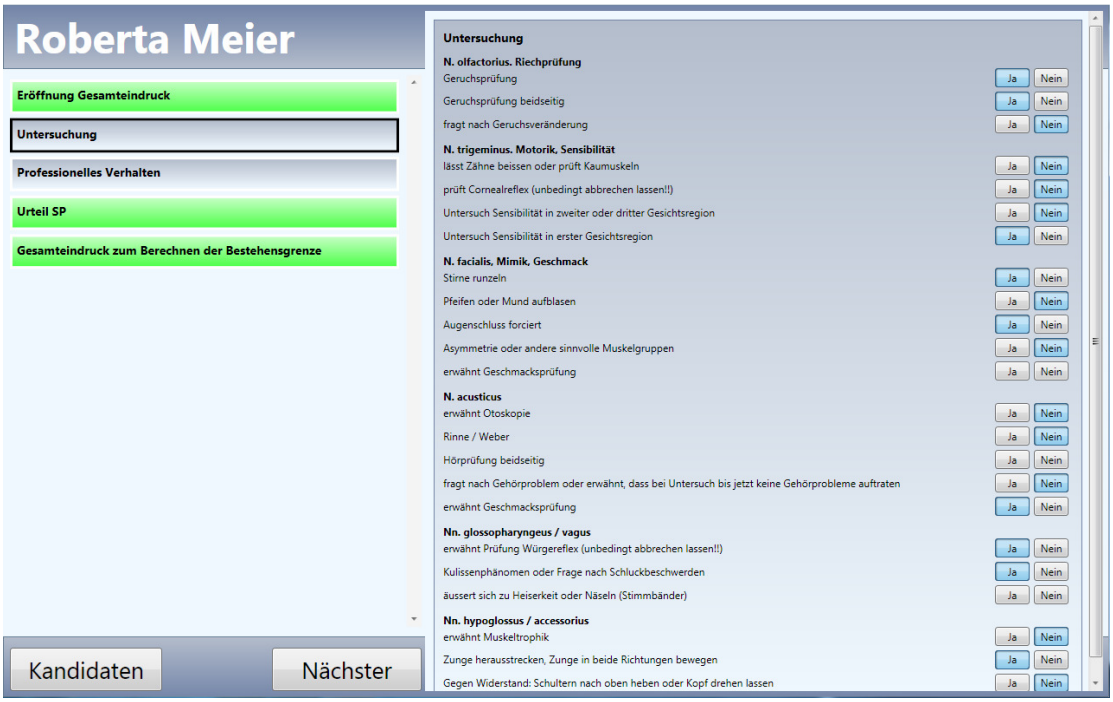
Prototype #2 – Hochformat mit Titelausblendung



Dies ist eine Weiterentwicklung des ersten Prototypen, wobei Blöcke, die weiter als ein Block vom aktuell vergrösserten Block entfernt sind, auf deren Titel reduziert werden.

Diese Variante wurde als zu unruhig empfungen, da ganze Topics augeblendet werden und darunterliegende Topics grosse Positions-änderungen erfahren. Diesem Pro–blem könnte mit einer guten Zooming-Strategie entgegengewirkt werden; das Problem wird allerdings trotzdem nicht gelöst, denn es ist so gar nicht möglich, dass Blöcke am gleichen Ort bleiben, während andere komplett ausgeblendet werden.

Prototype #3 – Querformat



Die vorhandene Flexibilität bei der zwischen einzelnen Topics gewechselt werden kann, ist so nicht nötig. Es ist viel wichtiger, dass grosse Topics komplett dargestellt werden können. Aus diesem Grund hat sich das IML trotzdem demokratisch für den Protype #1 entschieden.

Entscheidungen

Eingabe

Da die Eingabe per Finger schwierig und unpräzise ist – wahrscheinlich aufgrund der aktuellen Hardware – haben wir uns für eine Bedienung ausschliesslich mit dem Stylus entschieden. Ein weiterer Vorteil dieser Variante ist, dass der Benutzer nicht dauernd zwischen Finger und Stylus wechseln muss, wenn er Prüfungen ausfüllt und Notizen schreibt. Ein Problem ist auch, dass die Prüfungen in Zimmern mit hohen Temperaturen abgehalten werden, wobei die Benutzer stark schwitzen. Desweiteren kommt es bei gewissen Stationen vor, dass die Experten ihre Hände mit einem Desinfektionsmittel reinigen. Dadurch könnte die Oberfläche einen Schaden mit sich tragen, wenn die Experten nach dem Reinigen ihrer Hände auf dem Bildschirm mit den Fingern arbeiten.

Wir hatten zwar Multitouchgeräte zur Verfügung, es gab allerdings keine guten Anwendungsmöglichkeiten für Multitouch-Gesten, da das IML keine Zooming oder Drag&Drop-Effekte sehen wollte. Aus diesem Grund entwickelten wir die Applikation mit einfachen Single-Touch-Ereignissen.

Scrollbar

Über das Verwenden von Scrollbars wurde viel diskutiert. Schlussendlich wurde mit dem IML entschieden, dass eine Scrollbar im Prüfungsformular eingesetzt werden soll. Denn die Verwendung einer Scrollbar ist intuitiv und von den Benutzern bereits eine bekannte Technik zur Auswahl von verdeckten Inhalten. Schlussendlich haben wir drei Varianten zum Scrollen implementiert:

- Scrollbar
- Drag down/up
- Anklicken der TopicButtons

Dadurch kann jeder Benutzer seine bevorzugte Variante verwenden.

Buttons und Controls

Da die Eingaben mit einem Stift gemacht werden, sollten die Buttons am rechten Rand zu finden sein, da ein Rechtshänder sonst mit seiner Hand die Fragentexte verdecken würde.

Es wäre möglich, eine spezielle Version oder Option für Linkshänder anzubieten. Auf dies wird im Rahmen dieser Studienarbeit allerdings verzichtet, da der Grossteil der Bevölkerung Rechtshänder sind.

Desweiteren wurde der Einsatz von ToggleButtons/Checkboxen diskutiert. Das Problem bei diesen Controls ist die Validation, da in diesem Falle nicht geprüft werden kann, ob die Frage ausgefüllt wurde. Es wurde entschieden, Controls mit drei Zuständen zu verwenden, wodurch die Validation gewährleistet wird.

Für die Notenauswahl (schlecht – gut) wurden auch verschiedene Controls besprochen. Das Problem – zum Beispiel bei einem Slider – ist dasselbe wie bei den Checkboxen: Man kann nicht validieren, ob die Frage beantwortet wurde.

Validation

Die Anzeige, welche Bereiche bereits korrekt ausgefüllt wurden, soll pro Fragebereich (Topic) angezeigt werden. Dabei sollen nicht nur Farben (rot / grün) sondern auch Symbole verwendet werden, da viele Personen eine Rot-Grün-Sehschwäche[3] haben.

Da nicht immer die komplette Prüfung dargestellt werden kann, soll eine kleine Übersicht über alle Frageblöcke dargestellt werden. Dies wird dann so gelöst, dass pro Block ein kleines Icon dargestellt wird, bei dem man den Validationsstatus ablesen kann.

Notizen

Da die Experten Notizen zu dem Prüfungsformular machen können, musste auch dafür eine geeignete Lösung gefunden werden. Eine Variante war das Überlagern eines Notiz-Layers über den Prüfungsbogen. So wäre es auch möglich einzelne Stellen zu unterstreichen. Man entschied sich allerdings gegen diese Lösung, da so nicht einfach ersichtlich ist, ob man Notizen schreibt oder das Formular ausfüllt. Desweiteren beziehen sich die Notizen selten auf eine bestimmte Fragen, sondern viel mehr auf die Prüfung an sich.

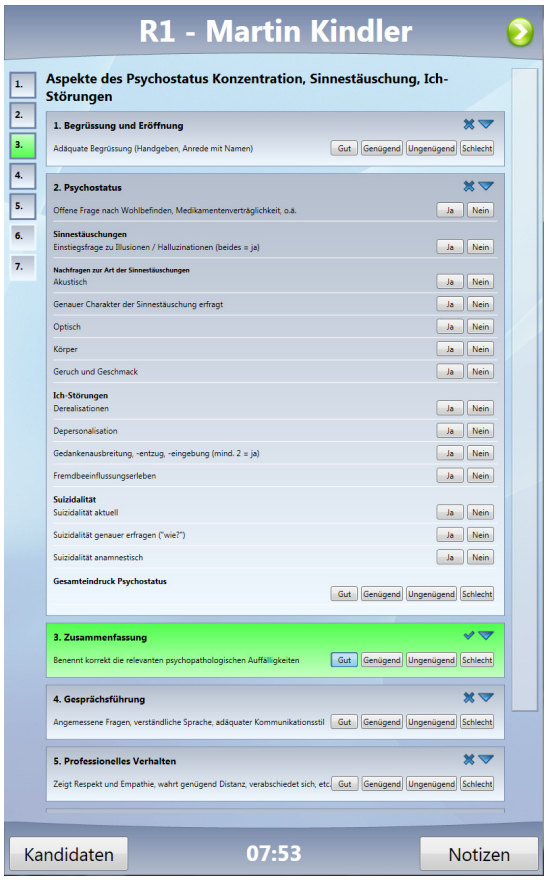
Kandidatenliste

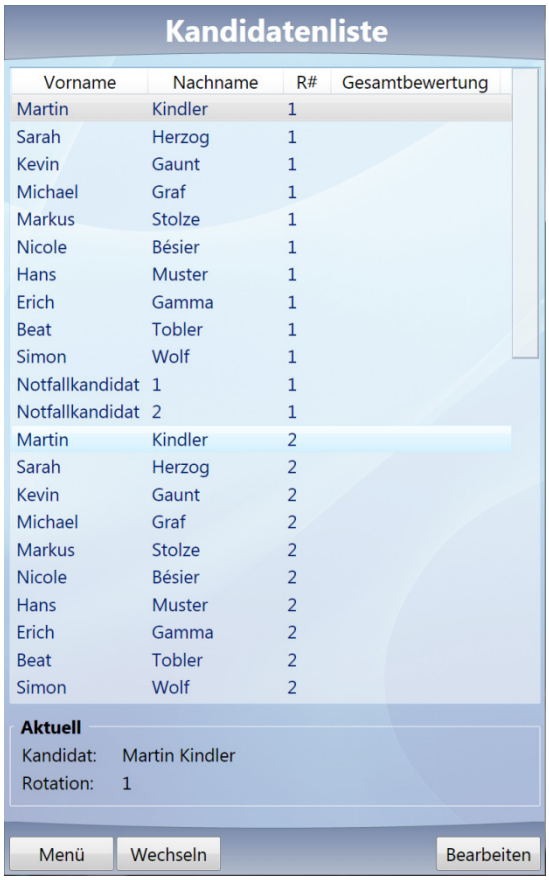
Da die Experten während der Prüfung einzelne Bewertungen noch anpassen und vergleichen wollen, muss in der Gesamtübersicht, beziehungsweise in der Kandidatenliste, die Gesamt-bewertung angezeigt werden können.

[3] http://de.wikipedia.org/wiki/Rot-Gr%C3%BCn-Sehschw%C3%A4che

Finales Design

Wir haben den ersten Prototypen konsequent weiterentwickelt und viele kleine Verbesserungen, die unter anderem im Kapitel „Entscheidungen“ getroffen wurden, umgesetzt. Nachfolgend werden ein paar Screenshots der fertigen Applikation gezeigt.





- Die komplett ausgefüllten Topics werden grün und mit einem Haken angezeigt
- Für eine bessere Augenführung wurden dünne Linien eingeführt
- Die Vierecke in der linken oberen Ecke geben eine schnelle Übersicht über die Kompletionszustand der Bereiche
- In der Kandidatenliste kann über den Menü-Button eine Rotation manuell gestartet
- Notizen können mit dem Stylus erstellt werden

18 Dezember 09

TabMed – SAD

Rico Suter & Fabian Mettler

Dokumentinformationen

Änderungsgeschichte

Datum	Version	Änderung	Autor
11.11.2009	0.1	Erste Version	fme & rsu
13.11.2009	0.2	Erstellung des Kapitel „Prozesse und Threads“	fme
13.11.2009	0.3	Erstellung des Kapitel „Grösse und Leistung“	fme
19.11.2009	0.4	Weitere Anpassungen	rsu
20.11.2009	0.5	Kapitel XML und REST erweitert	rsu
25.11.2009	0.6	Dateistrukturen und vieles mehr	rsu
30.11.2009	0.7	Verbesserungen in allen Kapiteln	rsu
02.12.2009	0.8	Review, Verbesserungen in allen Kapiteln	fme
03.12.2009	0.9	Review	fme & rsu
03.12.2009	0.10	Ergänzungen in allen Kapiteln	fme
07.12.2009	0.11	Kapitel „Timeouts“ und „Konfiguration“, weitere kleine Verbesserungen	rsu
13.12.2009	0.12	Komplettes Review	rsu (lwi)
17.12.2009	1.0	Letzte Änderungen	rsu

Inhaltsverzeichnis

Einleitung

In dieser Dokumentation wird die Architektur aller Applikationen, die in diesem Projekt entwickelt wurden, erläutert. Dazu gehört der TabMedServer, der TabMedClient, der TabMedEditor und das TabMedTerminal. Im Kapitel „Übersicht“ erhält der Leser einen ersten Einblick in den Aufbau des gesamten Systems. In den darauf folgenden Kapiteln wird genauer auf die Architektur der Applikationen eingegangen.

Architektonische Ziele und Einschränkungen

Ziele

- **Exchangeability, Maintainability:** Die einzelnen Komponenten sollen wie aus einem Baukasten zusammengesetzt und ausgetauscht werden können. Das GUI darf somit nur über eine minimale Schnittstelle mit der Business Logik verbunden sein.
- **Scalability:** Der Server soll mehrere Clients versorgen können.
- **Usability:** Die Benutzeroberfläche der TabMed-Applikationen muss einfach, intuitiv und übersichtlich gestaltet sein.
- **Fault Tolerance:** Die ganze Applikation muss stabil sein. Falls zum Beispiel unerwartet ein Netzwerkausfall auftritt, muss die Applikation mit Offlinedaten weiterarbeiten können. Damit steht einer erfolgreichen Durchführung der Prüfung bei solch einem Problemfall nichts im Weg.
- **Security:** Die Software muss sicher sein: Daten müssen vor Änderungen von Unbefugten geschützt werden und es dürfen keine Daten verloren gehen.

Einschränkungen

- Sobald die Prüfung gestartet wurde, soll der Client mit einem Offline-Datenbestand arbeiten. Bei jedem Speichervorgang wird allerdings das Prüfungsformular des Kandidaten an den Server gesendet, sofern eine Verbindung zum Server besteht. Sollte ein Netzwerkunterbruch vorliegen und die Daten sind noch nicht auf den Client vorhanden, so kann der Client nicht verwendet werden.
- Da der Server, welcher das Prüfungsstartsignal sendet, ausfallen kann, muss die Prüfung jederzeit manuell gestartet werden können.

Übersicht

Das Projekt TabMed beruht auf einer Client/Server-Architektur, wodurch die vorher erwähnten Anforderungen erfüllt werden.

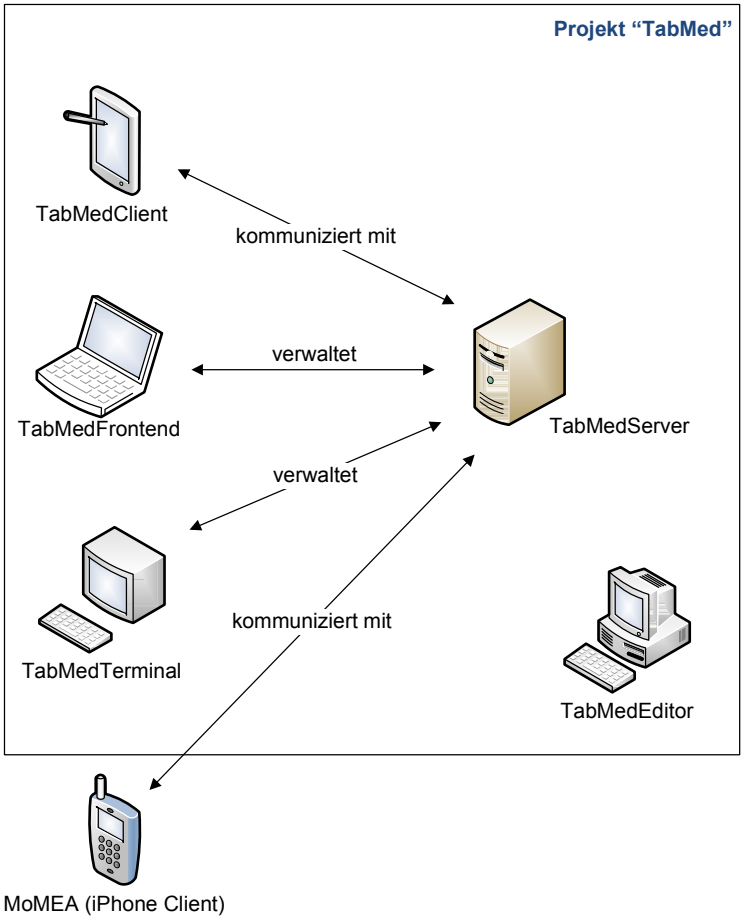


Abbildung: Client/Server Architektur des Projekts

TabMedServer

Der Server ist ein wichtiger und zentraler Baustein im gesamten System. Er ist so aufgebaut, dass mehrere parallele Prüfungen verwaltet werden können. Sobald eine Prüfung erstellt wurde und die benötigten Daten (Rotationen, Stationen, Kandidaten, Experten, usw.) vorhanden sind, kontrolliert und steuert der Server die Prüfung. Er ist somit für die Koordination der Clients zuständig. Das bedeutet, dass der Server die nötigen Daten für die Clients in einer plattformunabhängigen Form (XML) zur Verfügung stellt. Die Clients kommunizieren mit dem Server über eine REST-Schnittstelle, wodurch sie die benötigen Informationen und Signale (z.B. Prüfungsstart) erhalten.

Für die Verarbeitung der HTTPS-Requests wird der Grizzly-Servlet-Container[1] verwendet, da er auch als Embedded-Variante funktioniert und somit kein Servlet-Container auf dem Server installiert werden muss. Der Grizzly-Webserver wird so eingerichtet, dass die Verbindungen verschlüsselt per SSL übertragen werden und dass sich die Benutzer mit einem Passwort authentifizieren müssen.

[1] https://grizzly.dev.java.net

Der Server kann auf einem beliebigen Rechner ausgeführt werden, auf den die Clients Zugriff haben. Da die Server-Applikation in Java geschrieben wurde, kann theoretisch ein beliebiges Betriebssystem verwendet werden. Der Server wurde unter Windows 7, Windows Vista und Ubuntu Linux mit der Java Version 1.6 erfolgreich getestet.

TabMedClient

Der Client stellt das eigentliche User Interface zu den Prüfungen dar und ist die zentrale Applikation dieser Studienarbeit. Um die benötigten Daten zu erhalten kommuniziert er mit dem Server über ein REST-Schnittstelle oder liest sie au einem lokalen Verzeichnis aus. Da der Client dieselben XML-Formulare bearbeiten muss, wie der Server, beinhalten beide Projekte fast dieselbe Domain-Objekte. Da beim Server allerdings Java eingesetzt wird und auf dem Client .NET, mussten die Klassen neu implementiert werden.

Jeder Experte erhält bei einer Prüfung einen Tablet-PC, auf dem dann die TabMedClient-Applikation ausgeführt wird.

TabMedTerminal

Das Terminal ist eine einfache Schnittstelle für die Steuerung des Servers. Es ermöglicht das Starten von Prüfungen.

Das TabMedTerminal wird vom Prüfungsverantwortlichen hauptsächlich verwendet, um die aktuell laufende Prüfung zu starten. Da es eine einfache .NET-Konsolen-Applikation ist, kann sie auch unter Linux oder Mac OS X mithilfe von Mono[2] ausgeführt werden.

TabMedFrontend

Das TabMedFrontend ist eine weitere Applikation um den Server zu steuern. Es wird dabei auf ein grafisches User Interface gesetzt um dem Benutzer eine einfachere Serververwaltung als das TabMedTerminal zu bieten. Diese Applikation kann auch die aktuell verbundenen Clients anzeigen.

TabMedEditor

Der Editor ist nur für das Erstellen und Bearbeiten von Prüfungensformularen verantwortlich. Dies bedeutet, dass er keinerlei Verbindungen mit dem Server oder anderen Komponenten innerhalb des Projekts aufnimmt. Mit dem Programm können neue Formular-XML-Dateien (siehe „Form XML“ im Kapitel „Datenspeicherung“) erstellt oder bearbeitet werden.

Diese Applikation wird grundsätzlich von den Erstellern der Prüfung verwendet. Da sie in .NET mit WPF entwickelt wurde, kann sie nur auf Microsoft Windows Systemen ausgeführt werden.

MoMEA

Der iPhone Client (MoMEA) ist nicht Bestandteil dieser Arbeit, sondern wird vom IFS (Ansprechsperson: Michael Graf) weiterentwickelt.

[2] Getestet mit Version 2.6, http://www.mono-project.com

Logische Architektur

Da das TabMed-Projekt aus diversen Teilprojekten besteht, wird in diesem Kapitel die logische Architektur für jedes Teilprojekt einzeln beschrieben. TabMed verwendet in jedem Teilprojekt generell eine mehrschichtige Architektur. Dies hat den Vorteil, dass die Applikationen einfacher zu verstehen und besser testbar sind. Ganz nach dem „Open Close"-Prinzip: „Software Entitäten (Klassen, Module, Funktionen, usw.) sollten offen für Erweiterungen sein, aber geschlossen für Modifikationen" [3].

Alle Anwendungen, die mit C# programmiert wurden, wird mindestens Version 3.5 des .NET-Frameworks benötigt. Weitere externe Abhängigkeiten haben diese Anwendungen allerdings nicht.

Spezielle Design Pattern

M-V-VM Pattern (.NET WPF)

Das Model-View-ViewModel Pattern wurde speziell auf die neue Technologie "Windows Presentation Foundation" zugeschnitten. Es beschreibt eine saubere Trennung der View vom Model durch ein sogenanntes ViewModel. Dies wird unter anderem durch das DataBinding und das Commanding realisiert.

Dieses Pattern wird in allen Applikationen mit einem WPF-GUI eingesetzt – namentlich TabMedClient, TabMedFrontend und TabMedEditor.

Weitere Informationen zum Pattern kann unter folgender Adresse gefunden werden: http://msdn.microsoft.com/en-us/magazine/dd419663.aspx

Databinding

Beim DataBinding wird eine Verbindung zwischen dem User-Interface und dem Model definiert. Bei einer Änderung der gebundenen Elemente werden somit die Änderungen automatisch auf das User Interface und das Model übertragen. Als Beispiel wäre das Ausfüllen von TextBox Elementen in einem Formular zu nennen. Die dahinterliegenden Daten werden je nach Definition der Bindung während der Eingabe oder erst nach einer korrekten Validierung automatisch geändert.

Commanding

Die Commands in WPF sind im Prinzip nicht anderes als EventHandlers. Der Unterschied zwischen den normalen EventHandlers und Commands liegt darin, dass ein Command keiner spezifischen Schaltfläche zugeordnet ist, sondern dass der Auslöser des Commands von der dazugehörigen Logik getrennt ist. Dadurch ist es möglich in XAML (WPF-UI-Sprache) Commands zu definieren, welche noch gar nicht im .NET-Code existieren, da diese erst zur Laufzeit gebunden werden. Somit kann das User Interface ohne Logik vorbereitet werden und der Entwickler kann die Logik nachträglich implementieren.

[3] Meyer, Bertrand (1988). Object-Oriented Software Construction. Prentice Hall. ISBN 0136290493

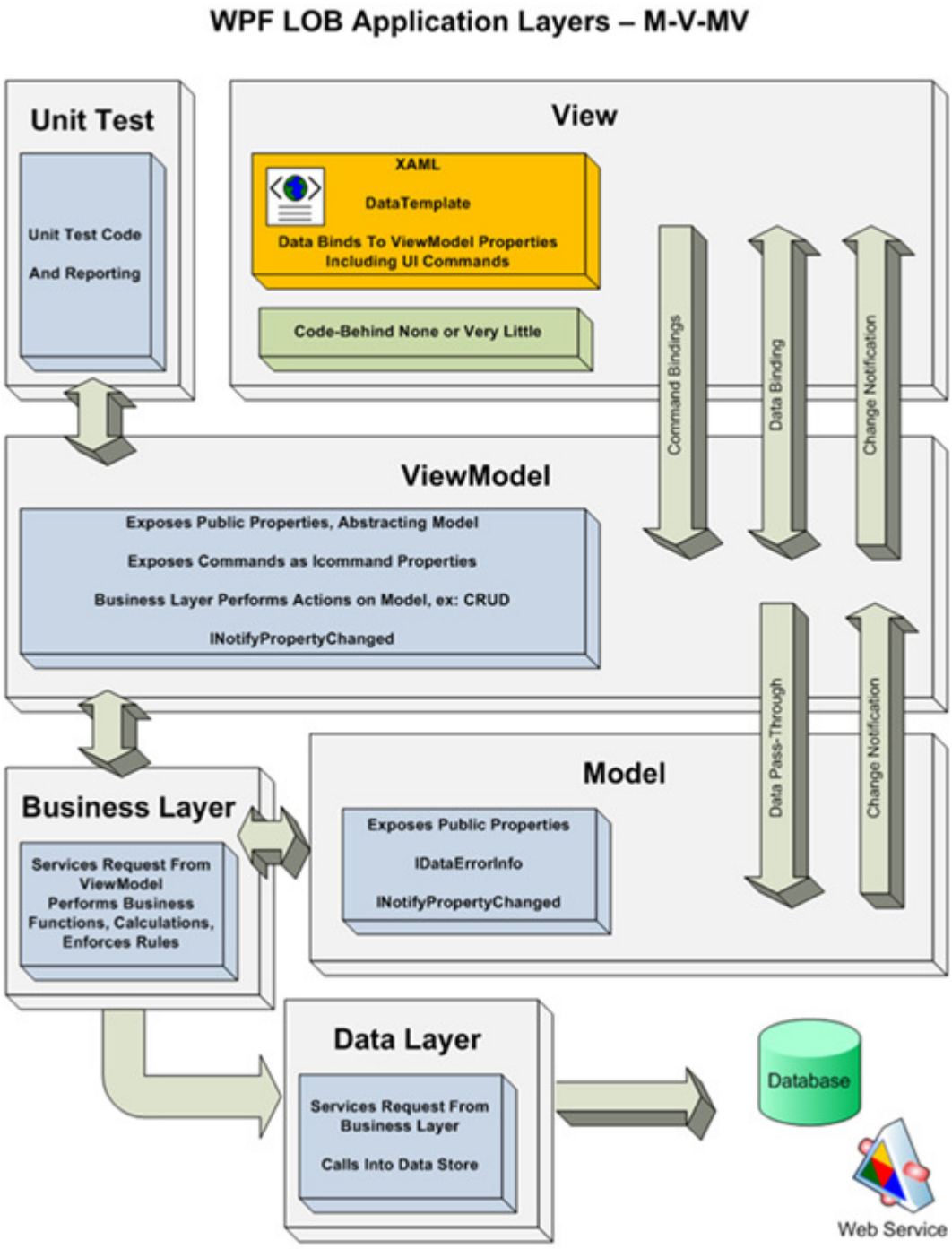


Abbildung: Verwendung des M-V-VM Pattern [4]

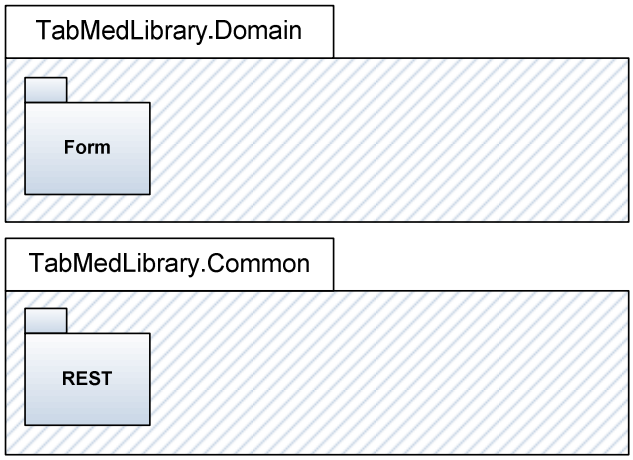
Die meisten im Diagramm oben dargestellten Komponenten haben wir im TabMedClient auch entwickelt. Die Views und ViewModels sind mit dem gleichen Namen zu finden und die Models sind die Klassen in der Domäne. Da wir fast keine Business-Logik haben, ist ein Grossteil davon direkt in den ViewModels oder Domain-Klassen zu finden. Unsere Daten werden über das Session-Objekt ausgelesen und sind dann im ViewModel zu finden. Mehr Informationen zum Aufbau des TabMedClients sind im Kapitel „TabMedClient“ zu finden.

[4] http://karlshifflett.wordpress.com/2008/11/08/learning-wpf-m-v-vm

TabMedLibrary

Die TabMedLibrary ist eine DLL-Bibliothek, die hauptsächlich die eigentliche Domänen-Logik für den TabMedEditor, das TabMedTerminal, das TabMedFrontend und den TabMedClient enthält. Die Domain wurde extra ausgelagert, damit alle Projekte sie verwenden können.

Namespaces



Common

Alle benötigen Hilfsklassen sind in diesem Namensraum zu finden.

- **REST**
  In diesem Unterpaket befindet sich die Klasse für die Kommunikation mit dem Server mittels REST-HTTPS-Protokoll. Diese Klassen wurden selber implementiert und benötigen daher neben dem .NET-Framework keine weiteren Biblitheken. Im Kapitel „Kommunikation“ unter „Programmierung“ werden alle Klassen genauer erläutert.

Domain

In diesem Paket sind die Domainklassen, welche die Business Logik der Applikation enthalten, zu finden.

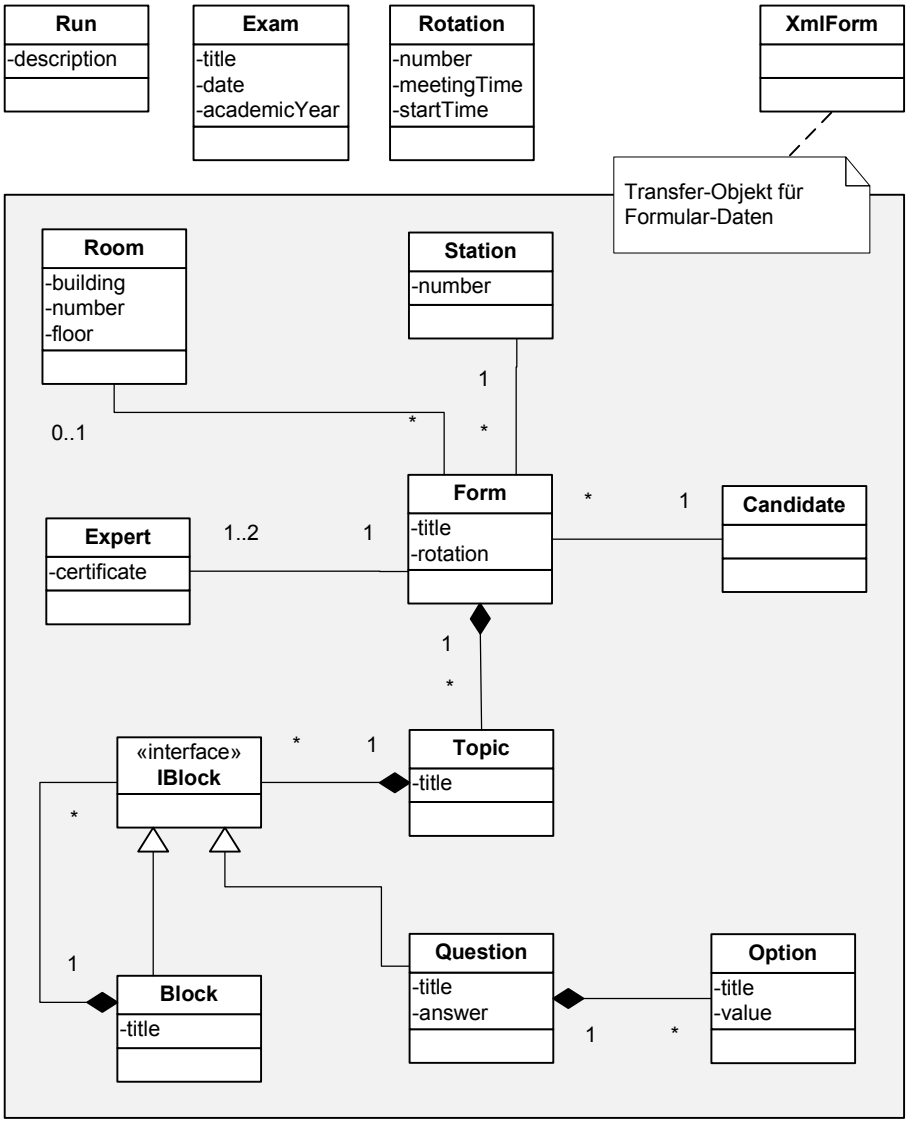


Abbildung: Klassendiagramm des Namespaces "Domain"

Da die Klassen „Run", „Exam" und „Rotation" jeweils per ID mit REST-HTTPS-URLs geladen werden, sind auf Code-Ebene keine Relationen zwischen diesen Klassen nötig.

Die in der Abbildung mit grauem Hintergrund dargestellten Klassen sind im Subnamespace „Domain.Form" zu finden und können aus einer einzigen Formular-XML-Datei gelesen und wieder gespeichert werden. Diese Formulare werden nach der Auswahl der Station automatisch mithilfe eines „XmlForm"-Objektes auf den Client übertragen und werden sogleich lokal gespeichert. Von diesem Ort können sie dann zu einem späteren Zeitpunkt wieder geladen werden und in effektive Form-Objekte transformiert werden.

Da ein Formular nur ausgefüllt werden kann, wenn ein USB-Stick zur Experten-Identifizierung vorhanden ist, wird das „Expert"-Objekt, das mit dem Formular verknüpft ist, automatisch beim Einstöpseln eines Sticks erstellt. So kann jederzeit ein anderer Experte einspringen. Die Signatur des Zertifikats vom USB-Sticks zeigt, wer den Bogen bearbeitet hat.

TabMedClient

Grafische Oberfläche

Die Architektur ist so aufgebaut, dass für alle Views ein ViewModel vorhanden ist. In diesem ViewModel sind dann alle Daten, die die View benötigt, enthalten. Über dieses ViewModel kann dann auch auf die aktuelle Session zugegriffen werden um zum Beispiel auf den aktuellen Server zuzugreifen.

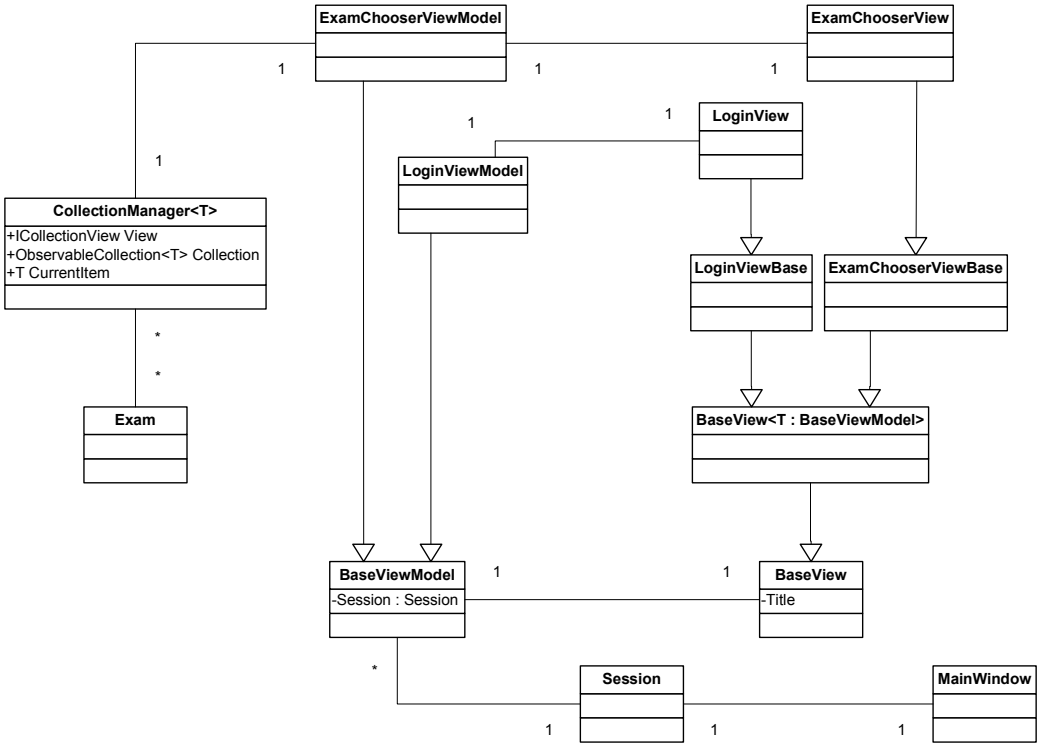


Abbildung: Klassendiagramm eines Teils des Namespaces "UI"

Der CollectionManager vereinigt eine ICollectionView mit einer ObservableCollection, damit die Funktionalitäten beider Klassen für das GUI und die Datenverwaltung vorhanden sind. Eine CollectionView wird benötigt, damit ein Cursor bzw. eine Selektion für die Collection vorhanden ist.

Der BaseView ist generisch, damit automatisch ein dazu passendes BaseViewModel erzeugt werden kann und damit der View einfach und ohne Casting auf das assoziierte ViewModel zugreifen kann.

Da die XAML-View (grafische Beschreibung der View) noch keine generischen Klassen unterstützt, wurde jeweils zwischen dem spezifischen View und dem BaseView eine Hilfsklasse für XAML erstellt (im Diagramm zum Beispiel „LoginViewBase"). Im XAML-Code kann dann mit dieser Hilfsklasse ohne Generics gearbeitet werden.

Für das Projekt wurden einige benutzerdefinierte Controls entwickelt. In der folgenden Abbildung werden die einzelnen entwickelten Controls dargestellt.

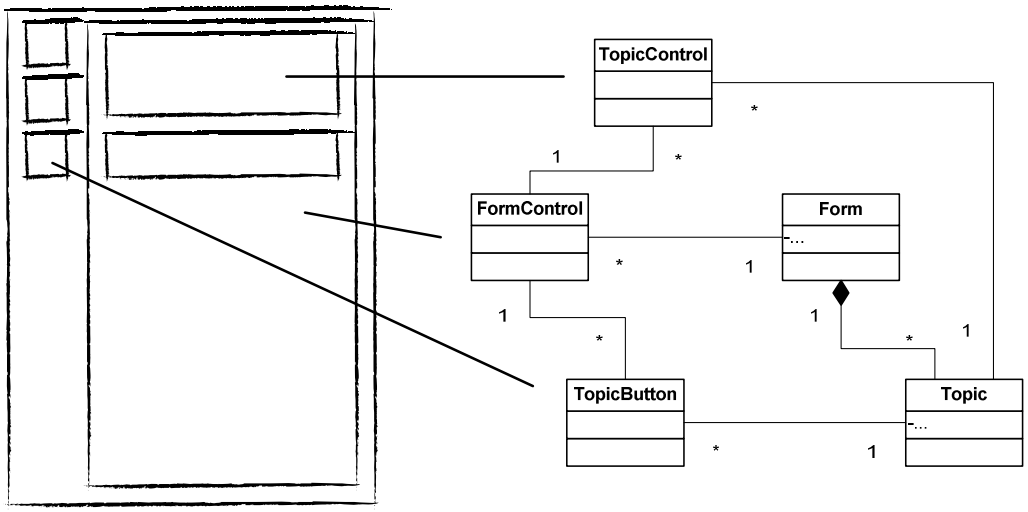
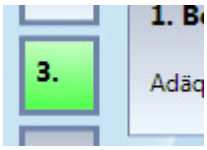


Abbildung: UI-Klassen

Ein FormControl wird benötigt um ein komplettes Formular-Objekt im GUI anzuzeigen. Dabei erstellt das FormControl automatisch weitere TopicControls für jedes Topic. Die TopicButtons am linken Rand (Abbildung rechts) müssen allerdings manuell durch das zugrunde liegende Fenster erstellt und mit dem FormControl verknüpft werden.



Das GUI der gesamten Applikation funktioniert grundsätzlich stapelbasiert. Die einzelnen Views und deren Transitionen können daher sehr einfach mit einem Zustandsdiagramm ohne Übergangsaktionen dargestellt werden.

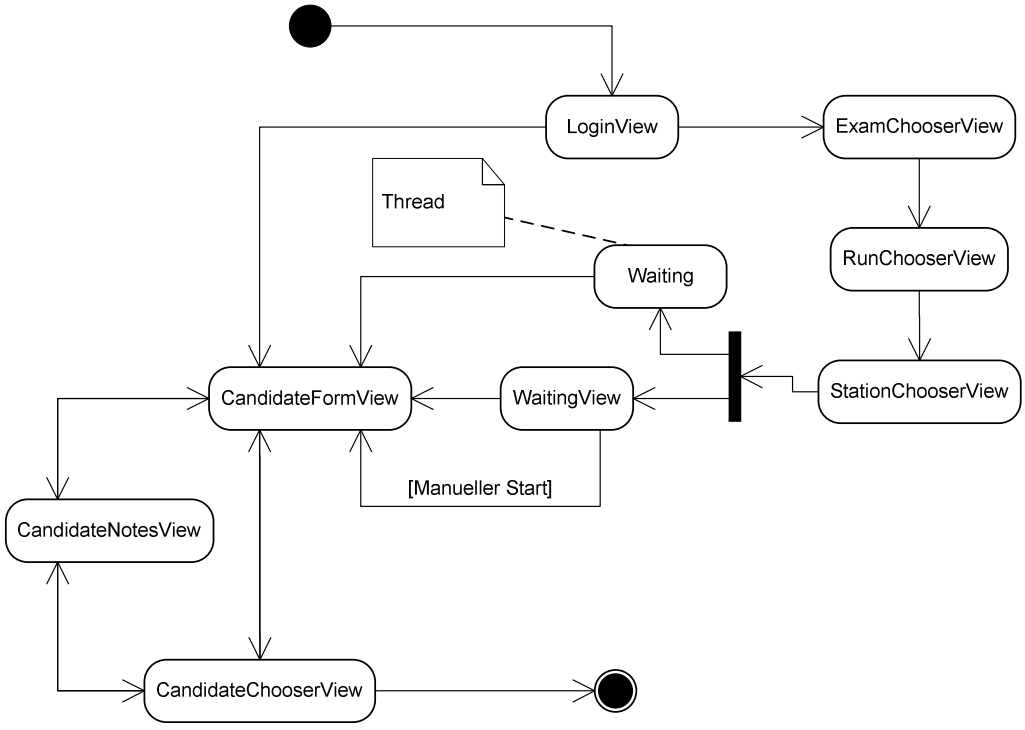


Abbildung: Ablauf der Views

Wichtig ist zu wissen, dass der Thread, der auf den Rotationsstart wartet, nur gestartet wird, wenn die Prüfung über den WaitingView gestartet wurde.

Session

In jedem ViewModel kann direkt auf die aktuelle Session zugegriffen werden. In diesem Objekt ist der aktuelle Zustand gespeichert, wie zum Beispiel die aktuelle Rotation und deren Startzeitpunkt. Desweiteren kann in der Session der Thread, welcher auf die Startsignale wartet, gestartet werden. In der Session kann auch auf die aktuelle Server-Verbindung sowie die HTTP-Request-Queue zugegriffen werden.

XML-Signierung

Die Logik für das Signieren der XML-Dateien ist in der Klasse „XmlSignature“ zu finden. Es wurde eine Funktion zum Auslesen eines Zertifikats aus einer Datei, zum Signieren und Validieren entwickelt.

Beim Signieren eines XML wird in die XML-Daten ein weiteres Tag eingefügt, das die Signatur enthält[5]. Beim Validieren kann dann dieser Teil ausgelesen werden und mit dem Rest validiert werden.

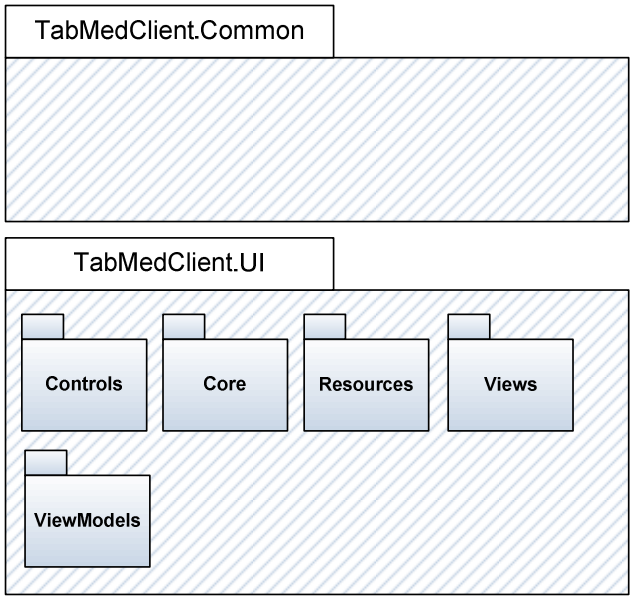
Erkennung von USB-Stick

Um ein Formular zu signieren, wird das Zertifikat auf dem aktuellen USB-Stick ausgelesen. Da der Bildschirm gesperrt sein muss, wenn kein Stick vorhanden ist, musste ein Mechanismus entwickelt werden, der erkennt wenn neue Geräte auftauchen oder verschwinden. Da das .NET-Framework keine solchen Events direkt unterstützt, musste auf die Win32-API zugegriffen werden.

Die Klasse „DeviceListener“ registriert eine Hook-Methode im System. Diese Methode erhält dann Statuscodes, die ausgewertet werden und C#-Events auslösen.

Namespaces

Die Logik wurde auf drei Namensräume aufgeteilt: Das User Interface, die Domäne und weitere Klassen. Der Quellcode der Domäne wurde zudem in eine seperate Bibliothek ausgelagert, da sie auch vom TabMedEditor verwendet wird.



[5] Enveloped Signature: http://en.wikipedia.org/wiki/XML_Signature

Common
Hier sind alle benötigten Hilfsklassen zu finden.

User Interface (UI)
In der UI-Schicht befinden sich alle relevanten Klassen, die das User Interface betreffen. Für eine bessere Gliederung sind die jeweiligen Klassen zusätzlich in weiteren Namensräumen verschachtelt.

- **Controls**
  Dieses Paket beinhaltet zusätzliche WPF User Controls, welche für das User Interface erstellt wurden.

- **Core**
  In diesem Paket sind alle Kern-Klassen zu finden, welche von den UI Elementen verwendet werden.
  Dazu gehören zum Beispiel die vorher erwähnten BaseView- und BaseViewModel-Klassen.

- **Resources**
  WPF-Ressourcen bieten einen einfachen Weg allgemein definierte Objekte und Werte wieder zu verwenden. Diese Resourcen sind in diesem Paket enthalten.

- **Views**
  Die einzelnen Views der Applikation sind in diesem Paket finden.

- **ViewModels**
  Die ViewModels, welche die Schnittstelle zwischen Views und Models spielen, sind im Paket "ViewModels" zu finden.

Dateistruktur
Die empfangenen sowie die ausgefüllten Formular-Dateien werden lokal und auf dem USB-Stick gespeichert. In der folgenden Tabelle ist abzulesen, wie die Daten auf dem Medium abgelegt werden.

TabMedClient.exe
config.xml
Data/
[EXAM_ID]
[RUN_ID]
[STATION_ID]
forms/
1.xml
2.xml
exam.xml
run.xml
station.xml
candidates/
[CANDIDATE_ID]
[STATION_ID]
time1.xml
time2.xml

Dieselbe Dateistruktur, allerdings nur der Pfad für „candidates/“, wird auch auf dem USB-Stick zur Speicherung verwendet.

Konfiguration

Die Konfiguration wird aus der XML-Datei „config.xml“ ausgelesen, die im gleichen Verzeichnis wie die TabMedClient-Applikation zu finden ist.

```
<?xml version="1.0"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <Server>http://localhost:9998</server>
        <Password>123456</Password>
        <RESTTimeout>20</RESTTimeout>
</Config>
```

In der folgenden Tabelle sind alle möglichen Felder aufgezeigt. Werden einige weggelassen, wird automatisch deren Standardwert verwendet.

Name	Beschreibung	Standard
Server	Server, der im Eingabefeld angezeigt werden soll. Dieser Wert wird automatisch in der Datei gesetzt, wenn ein anderer Server eingegeben wird.	
Password	Passwort für den angegebenen Server.	
RESTTimeout	HTTP-Timeout von REST-Anfragen (mehr im Kapitel „Kommunikation / Timeouts“. Angabe in Sekunden.	20

TabMedServer

Nach dem Starten der Server-Applikation, wird der Grizzly-Webserver geladen, konfiguriert und dann gestartet. Danach kann mit HTTP-REST-Zugriffen auf das Interface der Service-Klasse zugegriffen werden. Das Verarbeiten dieser Anfragen, das Weiterleiten und Verwalten mehrerer Worker-Threads übernimmt alles der vorkonfigurierte Webserver.

Abhängigkeiten durch Libraries

Da der TabMedServer im Grizzly-WebServer läuft und auf dem Jersey-REST-Framework aufgebaut ist, werden einige Jar-Libraries benötigt. In der folgenden Tabelle werden diese Libraries zusammen mit der verwendeten Version dargestellt. Desweiteren setzt der Server Java 1.6 voraus.

Dateiname	Version
activation.jar	1.0
annotations-api-6.0.20.jar	6.0.20
asm-3.1.jar	3.1
grizzly-http-webserver-1.9.18-e.jar	1.9.18-e
grizzly-servlet-webserver-1.9.18-e.jar	1.9.18-e
http.jar	1.0
jaxb-api.jar	1.0
jaxb-impl.jar	1.0
jaxb-xjc.jar	1.0
jdom-1.0.jar	1.0
jersey-core-1.0.3.1.jar	1.0.3.1
jersey-server-1.0.3.1.jar	1.0.3.1
jettison-1.0-RC1.jar	1.0

jsr173_api.jar	1.0
jsr311-api.jar	1.0
rome-0.9.jar	0.9
wadl2java.jar	1.0

SSL mit Grizzly

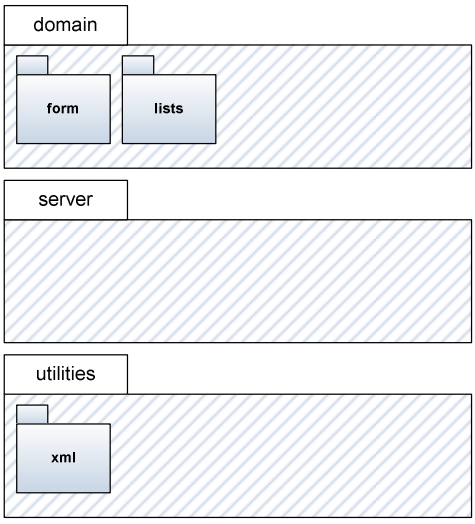
Beim Erstellen des GrizzlyWebServer-Objekts kann angegeben werden, ob HTTPS verwendet werden soll. Ist diese Option aktiviert muss beim Server ein SSLConfig-Objekt gesetzt werden. Dieses Konfigurations-Objekt beinhaltet den Pfad zur Keystore-Datei, welches das Server-Zertifikat beinhaltet. Für die Erstellung dieser Datei wird auf das Dokument „Benutzeranleitung“ verwiesen.

Damit sich die Clients beim Server authentifizieren können muss beim Server ein Security-Filter eingeflochten werden. Die Server-Passwörter können dann in diesem Filter angegeben werden. Mittels Annotationen können dann einzelnen Webservice-Methoden Zugriffsrollen vergeben werden, sodass nur autorisierte Benutzer diese aufrufen können.

```
@GET
@RolesAllowed({"admin"})
@Path("/reload")
public String reload() throws IOException {
       ...
```

Packages

Die Packages sind aufgeteilt in „service“, „domain“ und „utilities“. Das jax.ws.rs-Framework stellt alle Klassen im Package „service“ über ein REST-Interface den Server-Benutzern zur Verfügung. Beim Server-Start werden alle Prüfungen im Prüfungs-Verzeichnis eingelesen und in Objekte der Domain transformiert. Dies geschieht mittels XML-Readern aus dem „utilities.xml“-Package.



Domain

In folgendem Diagramm sind die Domain Klassen für die Business Logik des Servers abgebildet. Ein Prüfungs-XML (siehe Kapitel „Datenspeicherung“, „Exam XML“) kann mithilfe der „ExamReader“-Klasse in Objekte des abgebildeten Klassendiagramms überführt werden.

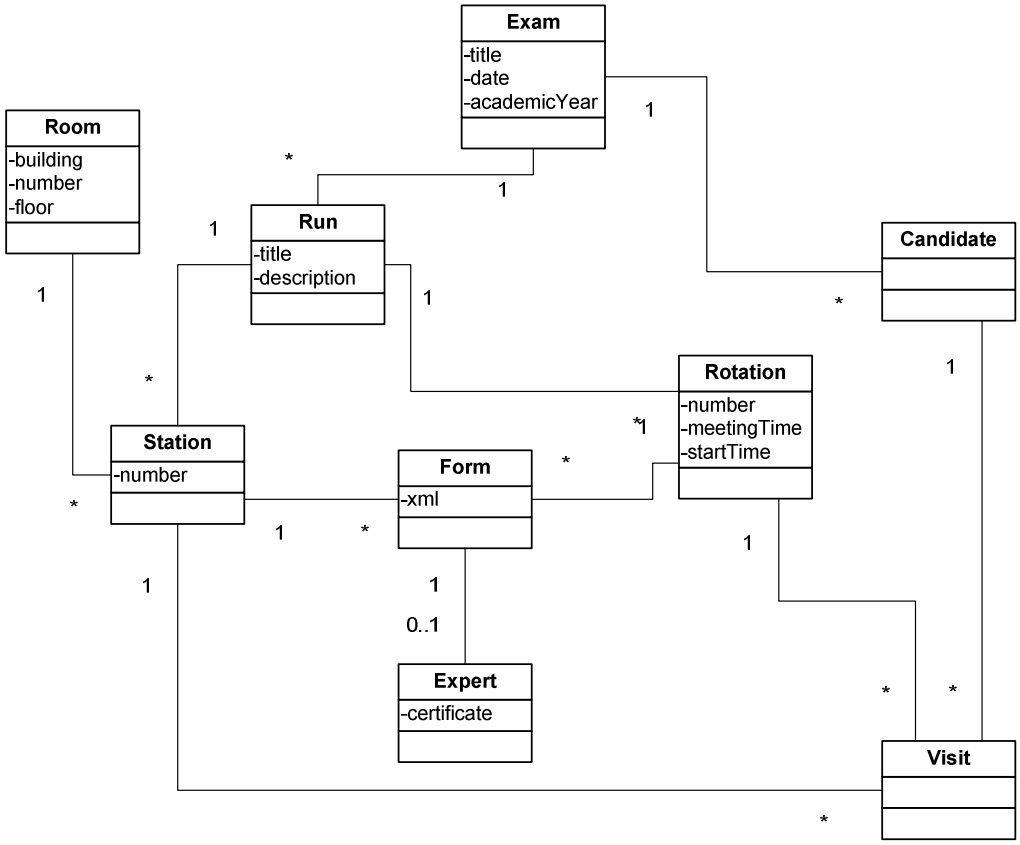


Abbildung: Klassendiagramm des Packages „Domain“

Im Subpackage „lists“ wurde für viele Domain-Objekte eine spezielle List-Klasse implementiert. Diese Klasse wird nur benötigt, damit die Objekte richtig nach XML serialisiert werden können. Es geht also nur um die XML-Annotationen. Einzige Ausnahme ist die DeviceList, denn sie enthält die Logik, welche anhand des Timeouts überprüft, welche Geräte momentan verbunden sind.

Die Annotation auf Klassenebene setzt den Namen des Root-Tags, das immer im Plural geschrieben ist. Als Tag-Namen für die einzelnen Objekte wird dann die Annotation der Methode verwendet.

```
@XmlRootElement(name = "Exams")
public class ExamList {
    Collection<Exam> exams;

    ...

    @XmlElement(name = "Exam")
    public Collection<Exam> getExams() {
        return exams;
    }
}
```

Server

Alle in diesem Package enthaltenen Klassen werden vom Grizzly-Server auf Annotations untersucht und dann, wenn eine REST-Annotation gefunden wurde, in die Liste möglicher REST-Abfragen aufgenommen. Die Klasse „Service“ stellt alle REST-Methoden zur Verfügung.

Utilities

In diesem Paket sind alle benötigten Hilfsklassen für den Server zu finden. Dazu gehören zum Beispiel die XML-Parser für die verschiedenen Dokumente.

Dateistruktur

Die verwendete Dateistruktur ist grundsätzlich ein Subset der Struktur des Clients und sieht folgendermassen aus:

TabMedServer.jar				
lib/				
exams/				
	[EXAM_ID]			
		exam.xml		
		forms/		
		candidates/		
			[CANDIDATE_ID]	
				[STATION_ID]
				timestamp1.xml
				timestamp2.xml
				timestamp3.xml

Java-Applikation

Nach der Kompilation der Sourcen steht eine Jar-Datei zur Verfügung. Dieses Programm ist eine reine Konsolenapplikation, die ohne weitere Applikationen oder Container ausgeführt werden kann.

Konfiguration

Die Konfiguration wird in der XML-Datei „config.xml“ vorgenommen. Der Server sucht diese Datei im aktuellen Arbeitsverzeichnis. Ist keine Datei vorhanden werden die in der Tabelle angebenen Standardwerte verwendet.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<config>
        <port>22</port>

        <minThreads>5</minThreads>
        <maxThreads>50</maxThreads>

        <waitTimeout>10<waitTimeout/>
        <sessionTimeout>30</sessionTimeout>

        <adminPassword>123456<adminPassword/>
        <userPassword>123456</userPassword>

        <keystorePassword>secret</keystorePassword>
</config>
```

In der folgenden Tabelle sind alle möglichen Felder zu finden. Werden einige Felder nicht definiert, wird automatisch der Standardwert aus der Tabelle verwendet.

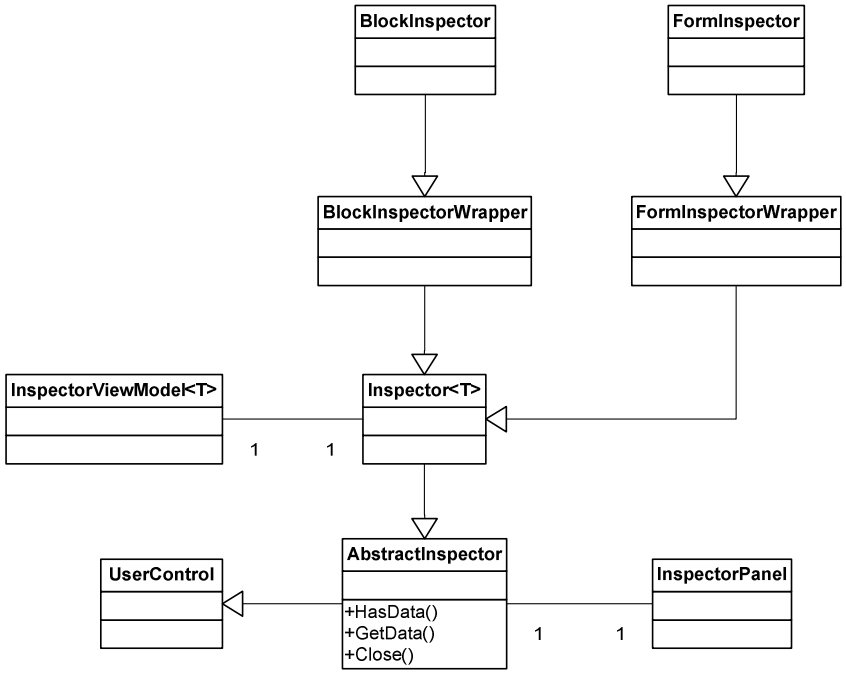
Name	Beschreibung	Standard
port	Server, der im Eingabefeld angezeigt werden soll. Dieser Wert wird automatisch in der Datei gesetzt, wenn ein anderer Server eingegeben wird.	9998
minThreads	Minimale Anzahl an Threads im Thread-Pool des Servers.	5
maxThreads	Maximale Anzahl an Threads im Thread-Pool des Servers. Gute Berechnungsgrundlage: Sollte doppelt so gross sein wie die Anzahl Clients, die den Server gleichzeitig verwenden.	50
waitTimeout	Zeit, nach der eine Antwort an den Client gesendet wird, ob eine Rotation gestartet wurde oder nicht (in Sekunden).	10
sessionTimeout	Zeit, nach der ein Client nicht mehr aktiv ist (in Sekunden).	30
adminPassword	Passwort um neue Prüfungen hochzuladen und und den Server zu aktualisieren. Dieses Passwort muss im TabMedFrontend eingegeben werden.	123456
userPassword	Passwort um Formulare herunterzuladen und ausgefüllte hochzuladen. Dieses Passwort muss im TabMedClient angegeben werden.	123456
keystorePassword	Keystore-Passwort, damit der Server das Zertifikat laden kann.	secret

Für eine genauere Beschreibung zu den Timeouts wird auf das Kapitel „Kommunikation / Timeouts“ verwiesen. Für die Erstellung von Serverzertifikaten wird auf die Benutzer-dokumentation verwiesen.

TabMedEditor

Grafische Oberfläche mit Inspektoren

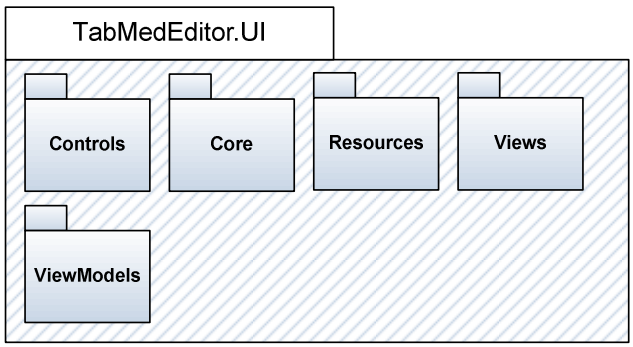
Für das GUI wurden einfache Inspektoren implementiert. Dabei wird im Hauptfenster ein „InspectorPanel“ erstellt, der die Inspektoren verwaltet und auswechselt.



Wie im TabMedClient wird auch hier eine Hilfsklasse (zum Beispiel BlockInspectorWrapper) eingesetzt, damit man die Klasse besser in XAML verwenden kann, da dort keine Generics unterstützt werden. Die abstrakte Klasse „AbstractInspector“ erbt direkt von der WPF-Klasse UserControl und verlangt alle Methoden, die vom InspectorPanel-Objekt benötigt werden. Die generische Klasse Inspector<T> implementiert einen einfacheren Zugriff auf das im „Inspector“ gehaltene Objekt. Die effektiven Inspektor-Klassen werden dann zusammen mit XAML-Code implementiert.

Namespaces

Die Logik des Editors wurde in Namensräume aufgeteilt. Da der Editor für die Business Logik die TabMed Library (DLL) verwendet, existiert nur der UI-Namensraum.



User Interface (UI)

In der UI-Schicht befinden sich alle relevanten Klassen, die das User Interface betreffen. Für eine bessere Gliederung sind die jeweiligen Klassen zusätzlich in weiteren Paketen verschachtelt.

- **Controls**
  Dieses Paket beinhaltet zusätzliche WPF User Controls, welche für das User Interface erstellt wurden.

- **Core**
  In diesem Paket sind alle Kern-Klassen zu finden, welche von den UI Elementen verwendet werden. Ein Beispiel wäre die Klasse „RelayCommand“ zur Verarbeitung von ICommands.

- **Resources**
  WPF Resourcen bieten einen einfachen Weg allgemein definierte Objekte und Werte wieder zu verwenden. Diese Resourcen sind in diesem Paket enthalten.

- **Views**
  Die einzelnen Views der Applikation sind in diesem Paket finden.

- **ViewModels**
  Die ViewModels, welche die Schnittstelle zwischen Views und Models spielen, sind im Paket "ViewModels" zu finden.

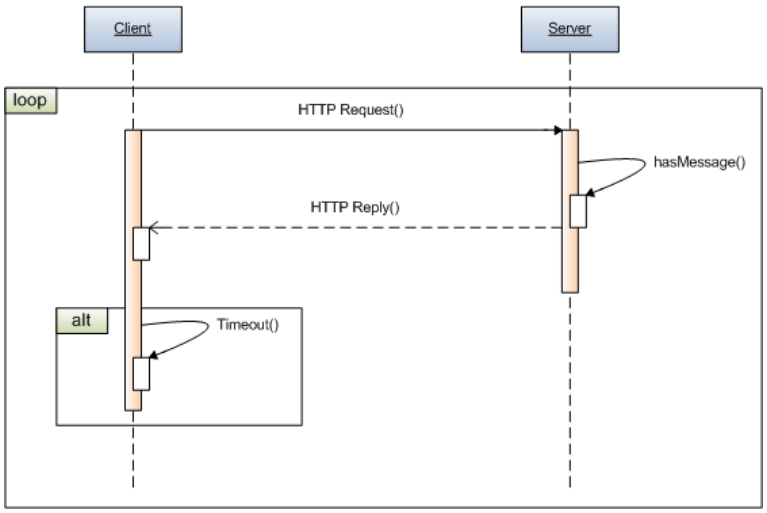
Kommunikation

Im Projekt TabMed werden die Prüfungen zentral vom Server verwaltet. Der Server kennt alle relevanten Daten einer Prüfung: Experten, Kandidaten, Rotationen, Stationen und Prüfungsbogen. Diese Daten werden vom Verzeichnis „exams“ ausgelesen und können mit dem TabMedFrontend hochgeladen oder manuell in dieses Verzeichnis kopiert werden. Er kontrolliert den Start der einzelnen Rotationen und koordiniert somit die einzelnen Clients. Die Daten stellt der Server über einen REST (Representational State Transfer) WebService zur Verfügung, wodurch die Clients die benötigten Daten für die Abbildung ihrer Problem Domain erhalten. Für den Prüfungsstart wird die Technik des Long-Polling verwendet, die im nächsten Kapitel beschrieben ist.

Long Polling

Long Polling ist eine Variation der traditionellen Polling-Technik und erlaubt das emulierte Anstossen (Pushing) von Informationen von einem Server an den Client. Der Client setzt wie gewohnt beim Polling eine Abfrage ab. Wenn der Server aber keine Informationen zur Verfügung hat, wird keine leere Antwort gesendet, sondern der Server hält die Anfrage offen und wartet bis neue Informationen zur Verfügung stehen. Falls neue Informationen bereit stehen, oder nach einem Timeout, wird eine Antwort an den Client gesendet. Ist die Antwort kein Startsignal sondern ein Timeout, dann verbindet sich der Client sofort wieder mit dem Server um erneut auf ein Startsignal zu warten.

Long Polling ist somit keine eigentliche Push-Technologie, kann allerdings dafür verwendetet werden, falls keine richtigen Push-Notifications möglich sind.

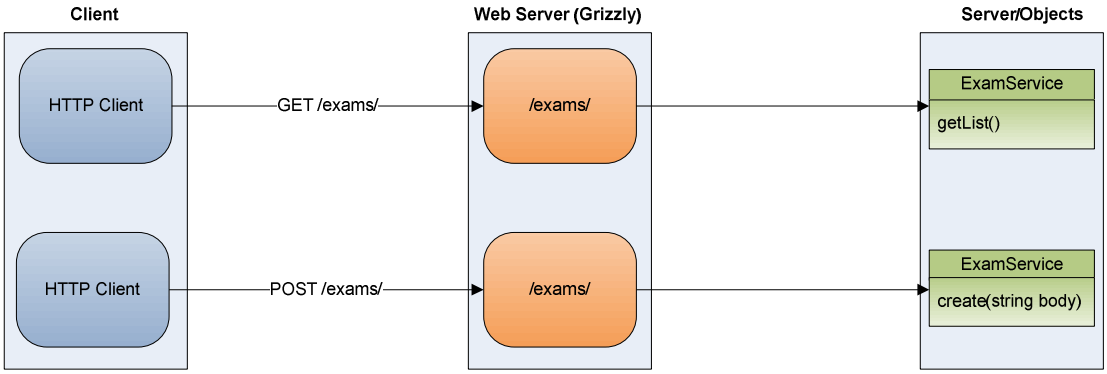


Die Technik des Long Polling wird in TabMed für das Starten der Prüfungsrotationen verwendet. Wenn der Client alle benötigen Daten vom Server heruntergeladen hat und die Prüfung noch nicht gestartet wurde, erscheint ein Warten-Bildschirm. Die Antwort der Anfrage des Client wird somit beim Warten hinausgezögert und die Verbindung offen gehalten, bis dem Server signalisiert wird, dass die Prüfung gestartet werden soll. Ein Nachteil dieser Polling-Technik ist, dass der Server keine Verbindungen zum Client aufbauen kann.

Es ist zu beachten, dass der Web Server so eingerichtet sein muss, dass viele Clients gleichzeitig verbunden sein können. So wird gewährleistet, dass die Clients zeitgleich gestartet werden.

REST Architektur

Representational State Transfer (REST) ist ein Architektur-Stil für netzwerkbasierte Software. Dabei wird die ursprüngliche Idee der Technologien und den Protokollen des Word Wide Web verwendet. REST beschreibt wie man Daten-Objekte oder Ressourcen definiert und adressiert. Zusätzlich strebt es einen einfachen Austausch von Informationen und hohe Skalierbarkeit durch zustandslose Kommunikation an. [6]



Protokoll

Das Zugriffs-Protokoll ist sehr einfach gehalten und kann grundsätzlich mit jedem HTTP-fähigen Client verwendet werden.

Beschreibung	Typ
http://server/exams Alle Prüfungen anzeigen. Gibt ein XML mit mehreren Prüfungen aus.	GET
http://server /exams/[ID] Einzelne Prüfung anzeigen. Gibt ein XML mit einer Prüfung aus.	GET
http://server /exams/[ID]/runs/[ID]/stations/[ID]/wait Wartet auf einen Prüfungsstart warten. Wird eine Zahl grösser als „0“ ausgegeben, wurde die Prüfung gestartet. Der Rückgabewert gibt an, welche Rotation gestartet wurde. Bei „0“ trat ein Fehler auf (wie zum Beispiel ein Timeout): Hier sollte „wait“ nochmals aufgerufen werden (Long Polling). Die Run-ID und Station-ID werden nur benötigt, um den aktuellen Status im TabMedFrontend anzeigen zu können.	GET
http://server /exams/[ID]/rotations/[ID]/start Startet die gewählte Prüfung. Wird „1“ zurückgeliefert ist alles korrekt abgelaufen und das Start-Signal wurde an alle wartenden Clients gesendet. Bei „0“ trat entweder ein Fehler auf, oder keine Clients haben auf den Start gewartet. Es werden alle Durchführungen gleichzeitig gestartet.	GET
http://server /exams/[ID]/devices Zeigt alle verbundenen Geräte der gewünschten Prüfung an.	GET

[6] **Weitere Informationen:**
http://en.wikipedia.org/wiki/REST
Richardson, Leonard; Ruby, Sam (2007), RESTful Web Services, O'Reilly, ISBN 0596529260

http://server /exams/[ID]/runs/[ID]/rotations Zeigt alle Rotationen eines Durchlaufes an.	GET
http://server /exams/[ID]/runs Alle Durchläufe einer Prüfung anzeigen.	GET
http://server /exams/[ID]/runs/[ID]/stations Alle Station eines Durchlaufs anzeigen.	GET
http://server /exams/[ID]/runs/[ID]/stations/[ID]/forms Formulare für jeden Kandidaten zu einer bestimmten Station anzeigen	GET
http://server /exams/[ID]/runs/[ID]/stations/[ID]/forms/[CANDIDATEID] Formular speichern. Dies ist eine POST-Aktion wobei der Body das zu speichernde Formular repräsentiert.	POST

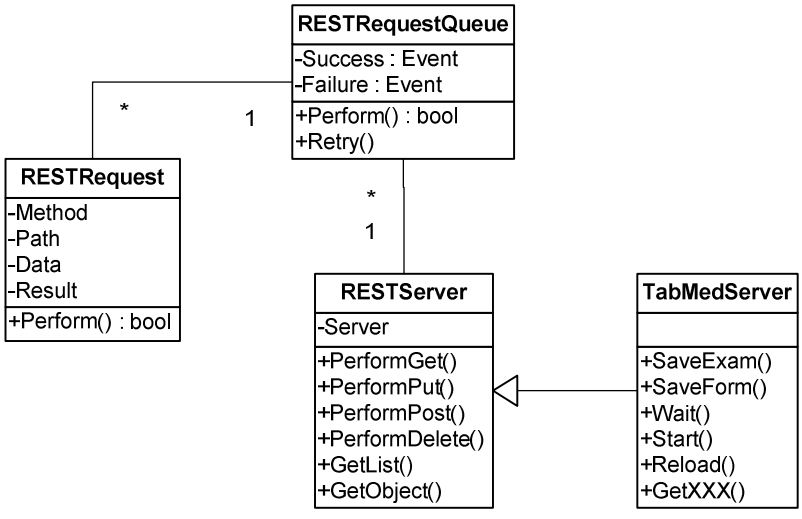
Für die Administration des Servers stehen folgende Befehle zur Verfügung:

http://server /exams/reload Alle Prüfungen aus dem Server-Verzeichnis neu laden.	GET
http://server /exams/[ID] Neue Prüfung mit einer bestimmten ID (= Verzeichnisname auf dem Server) erstellen, wobei der POST-Body das Prüfungs-XML enthält.	POST
http://server /exams/[ID]/forms/[FORM] Neues Formular (mit Dateinamen FORM.xml) für eine bestimmte Prüfung hochladen. Der Request ist mit der POST-Methode aufzurufen und der Body ist die zu erstellende Prüfung.	POST

Die letzten drei Requests werden für die Administration des Servers benötigt und sollten nicht für alle Benutzer verfügbar sein.

Programmierung

Für die REST-Abfragen wurden drei einfache Klassen entwickelt. Diese Klassen sind im Namespace „TabMedLibrary.Common.REST“ zu finden.



Für den Zugriff auf einen HTTP-Server per REST wird in jedem Falle das RESTServer-Objekt benötigt. Mit diesem Objekt können einzelne Requests gemacht werden. Für jede HTTP-Methode gibt es eine eigene Methode. Die Fehlerbehandlung des RESTServer-Objekts wurde komplett mit Exceptions gelöst, was bei der RESTRequest- und RESTRequestQueue-Klassen nicht der Fall ist. Für einfache Lese-Abfragen wie zum Beispiel das Lesen der Prüfungsliste wird direkt die RESTServer-Klasse verwendet.

Da es vorkommen kann, dass keine validen Server-Zertifikate zur Verfügung stehen, ist die Zertifikat-Validierung clientseitig standardmässig ausgeschaltet und die SSL-Verbindung wird nur zur Verschlüsselung verwendet. Mit einem Parameter im RESTServer-Konstruktor kann das Validieren ein- und ausgeschalten werden.

Für Requests, bei denen auch im Fehlerfall weitergearbeitet werden kann, wird die RESTRequestQueue verwendet. Hier wird zuerst ein RESTRequest-Objekt erstellt. Danach wird das RESTRequest-Objekt in die Queue übergeben. Bei dieser Übergabe kann noch angegeben werden, ob frühere, gleiche Anfragen gelöscht werden sollen. Soll zum Beispiel ein Formular gespeichert werden, werden so frühere Anfragen, die fehlgeschlagen sind, aus der Queue gelöscht und nur die aktuelle Anfrage gesendet. Danach versucht die RESTRequestQueue die Anfrage an den Server zu leiten oder speichert sie lokal bei einem Übertragungsfehler. Die RESTRequestQueue-Klasse arbeitet mit Boolean-Rückgabewerten für die Fehlerbehandlung.

Timeouts

Im Server sowie im Client können mithilfe spezieller Konfigurationsdateien die Timeout-Zeiten für die Kommunikation festgelegt werden. Wenn keine Konfiguration vorhanden ist, funktionieren alle Komponenten problemlos miteinander. Will man allerdings diese Werte selbständig konfigurieren, sind einige Gegebenheiten zu beachten, damit das Gesamtsystem noch funktionieren kann.

Komponente	Name	Beschreibung	Standard
Client	RESTTimeout	HTTP-Timeout auf dem Server, **muss** grösser als das WaitTimeout sein, da der Server ansonsten die Verbindung länger geöffnet hat, als der Client und so immer weniger Verbindungen zur Verfügung stehen.	20 s
Server	WaitTimeout	Timeout bis der Server eine Antwort liefert, wenn auf Prüfungsstart gewartet wird (Dauer eines Poll-Zyklus). Muss kleiner sein als das RESTTimeout. Damit nicht immer mehr Verbindungen geöffnet werden.	10 s
Server	SessionTimeout	Zeigt an, wie lange ein Client auf dem Server aktiv ist, bis er wieder aus der Liste gelöscht wird. Dieser Wert sollte mindestens zweimal so gross wie das WaitTimeout sein, damit keine Probleme auftreten.	30 s

Wie diese Konfigurationen in den Applikationen vorgenommen werden können, kann im Kapitel „Logische Architektur“ oder im Dokument „Benutzeranleitung“ nachgeschlagen werden.

Prozesse und Threads

In diesem Kapitel werden die Threads in den jeweiligen architektonischen Elementen beschrieben: Wann sie laufen und welche Aufgaben sie haben. Es werden die folgenden Bereiche unterschieden:

- Server Threads, die in der Server Applikation das Handling für die Kommunikation und das Long Polling übernehmen.
- Client Threads, die zum Beispiel überwachen, ob eine Prüfungsrotation gestartet wurde.

Für die Synchronization der Threads werden folgende Mittel eingesetzt:

- Locks (in .NET-Applikationen)
- Windows Presentation Framework – Dispatcher
- Synchronized-Methoden (in der Java-Klasse „TimeoutLatch“)

TabMedServer

In der Server Applikation wird das Grizzly NIO and Web Framework[7] verwendet, um einen Multi-Threaded Webserver zur Verfügung zu stellen. Grizzly beinhaltet einen Thread Pool für die Verbindungen. Sobald sich ein Client mit dem Server über die REST-Schnittstelle verbindet, wird aus diesem Thread Pool ein unbenutzer Thread dem Client zugeteilt. Dieser Thread übernimmt nun die Kommunikation zwischen dem Client und dem Server.

Wenn der Client die gewünschte Prüfung ausgewählt hat, ladet er sich alle benötigen Daten in seinen lokalen Speicher und wartet auf das Startsignal. Der Client verwendet die Long Polling-Technik um auf das Startsignal zu warten (Siehe Kapitel „Kommunikation“). Um den Start der Clients zu synchronisieren wird ein Latch verwendet.

TabMedClient

In der Client Applikation gibt es zwei wichtige Threads:

1. WaitForStart-Thread
2. TimeCountDown-Thread

Der WaitForStart-Thread wird in der aktuellen Session des MainWindow gestartet. Sobald eine Prüfung ausgewählt wurde, blockiert der Client und zeigt einen Wait Screen an. Im Hintergrund prüft nun der WaitForStart-Thread mittels REST, ob die Prüfung auf dem Server gestartet wurde. Wenn die Prüfungsrotation auf dem Server gestartet wurde, fügt der Thread den nächsten Kandidaten in den WPF Dispatcher ein, wodurch das Prüfungsformular des Kandidaten auf dem Bildschirm angezeigt wird. Zusätzlich wird der WaitForStart-Thead zurückgesetzt und überprüft, ob die nächste Rotation gestartet wurde.

Das UserControl „CountDown“ beinhaltet einen Thread, der sich um das Herunterzählen der Prüfungszeit des Kandidaten kümmert.

[7] https://grizzly.dev.java.net/

Datenspeicherung

Für dieses Projekt wurden einige XML-Schemata definiert, um den Datenaustausch zwischen den verschiedenen Komponenten zu ermöglichen. In diesem Kapitel werden diese Schemata beschrieben.

Exam XML

Das Examl XML beschreibt eine OSCE-Prüfung mit allen Angaben zu den Kandidaten, Durchführungen, Stationen, usw.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<exam>
	<title>Titel der Prüfung</title>
	<description>Beschreibung der Prüfung</description>
	<date>Datum der Prüfung</date>
	<acadamicYear>Studien-Jahrgang der Kandidaten</acadamicYear>
	<examDuration>Stationsdauer (Standard: 8 min)</examDuration>
	<breakDuration>Bearbeitungsdauer (Standard: 2 min)</breakDuration>

	<candidate>
		<id>Eindeutige Studentennummer (z.B. Legi-Nr.)</id>
		<firstName>Vorname</firstName>
		<lastName>Nachname</lastName>
	</candidate>

	<run>
		<title>Titel der Durchführung</title>
		<description>Beschreibung der Durchführung</description>

		<station>
			<number>Stationsnummer</number> TODO: benötigt??
			<form>Dateiname des Formulars[8]</form>
			<title>Titel der Station</title>
			<room>
				<building>Name des Gebäudes</building>
				<floor>Stockwerk</foor>
				<roomNumber>Raumnummer</roomNumber>
			</room>
			<isDouble>gibt an ob Doppelstation („True“/„False“)</isDouble>
			<description>Beschreibung der Station</description>
		</station>

		<rotation>
			<meetingTime>...</meetingTime>
			<startingTime>...</startingTime>

			<visit>
				<stationNumber>Referenz auf Station</stationNumber>
				<candidateID>Referenz auf Kandidat</candidateID>
			</visit>
		</rotation>
	</run>
</exam>
```

[8] Die Datei muss auf dem Server im Verzeichnis „forms/“ zu finden sein. Beispiel: „psychostatus.xml“

Form XML

Es gibt zwei Varianten des Form XML. Zum einen wäre das unausgefüllte Formular, das auf dem Server gespeichert ist und das dann an die einzelnen Clients gesendet wird, wenn die Prüfung gestartet ist. Zum Zweiten wäre das komplett ausgefüllte Formular, das vom Client an den Server gesendet wird und dort archiviert wird.

Im folgenden XML werden Attribute, die nur in der ausgefüllten Variante vorhanden sein müssen, **fett und blau** dargestellt.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<form>
	<title>Name der Station</title>
	<candidate>
		<id>ID des Kandidaten</id>
		<firstName>Vorname</firstName>
		<lastName>Nachname</lastName>
	</candidate>
	<expert>
		<title>Titel</title>
		<firstName>Vorname</firstName>
		<lastName>Nachname</lastName>
	</expert>
	<topic>
		<title>Titel des Themas</title>
		<question>
			<title>Titel der Frage (optional)</title>
			<type>mark</type>
			<answer>0</answer>
		</question>
	</topic>
	<notes>GIF-Daten kodiert in Base64</notes>
	<Signature>
		Die Signatur wird durch die Applikation eingefügt.
		Dazu muss auf dem USB-Stick eine pfx-Datei mit dem
		Namen „certificate.pfx“ vorhanden sein.
	</Signature>
</form>
```

Ein Formular baut sich aus mehreren Topic-Blöcken zusammen. Ein Block enthält entweder weitere Blöcke oder Fragen.

Typen

Bei den Fragen kann ein Typ angegeben werden. Ist eine Typangabe vorhanden, werden die einzelnen Optionen der Frage automatisch nach erstellt. In der folgenden Tabelle sind die verschiedenen Typen aufgelistet.

Bezeichnung	Beschreibung
yesNo	Entscheidungsfrage (Ja, Nein)
yesNoPart	Erweiterte Entscheidungsfrage (Ja, Nein, Teils)
ensemble	Gesamteindruck („sehr schlecht“ bis „sehr gut“, 6 Stufen)
rating	Bewertung („gut“ bis „schlecht“, 4 Stufen)
grade	Benotung (Zahlenwerte von eins bis fünf)

Die Gross-/Kleinschreibung der Bezeichnung wird ignoriert.

Expert XML

In der Experten-XML-Datei, die auf dem USB-Stick zu finden ist, werden die Informationen zum aktuellen Experten gespeichert. Diese Informationen werden in jedes Formular, das nach einer Änderung gespeichert wird, eingebettet.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<expert>
	<title>Dr.</title>
	<firstName>Hans</firstName>
	<lastName>Muster</lastName>
</expert>
```

Der Aufbau des XML ist so einfach, dass er hier nicht näher erläutert wird.

Grössen und Leistung

Eine Begrenzung beim Server kommt durch die Verwendung von Grizzly zu tragen. Die Anzahl der Threads im ThreadPool sind durch eine untere und obere Grenze definiert. Die untere grenze definiert die aktiven Threads, die von Anfang an zur Verfügung stehen, während die obere Grenze die maximale Anzahl der verfügbaren Threads beschreibt. Sollten nun mehr Clients eine Verbindung aufbauen, als dass der Server Threads zur Verfügung stellen kann, werden einige Clients erst bedient, wenn wieder Threads zur Verfügung stehen.

Eine visuelle Begrenzung ist beim Client vorhanden, was die Anzahl der Kandidaten und den Umfang des Prüfungsbogen betrifft. Sollten mehr als 100 Kandidaten angezeigt werden, ist es schwierig den gewünschten Kandidaten zu finden, des weiteren verliert man dadurch schnell die Übersicht. Wenn das Prüfungsformular zu gross entworfen wird und es viele Verschachtelungen enthält ist es schwierig eine gute Übersicht zu garantieren. Das Ausfüllen würde somit zu einer Qual für den Experten werden.

Durch diese Begrenzungen werden folgende Empfehlungen abgegeben:

1. Anzahl der Kandidaten: 20 bis 30 Personen
2. WAN-/MAN-/LAN-Verbindung von mindestens einem Mbit/s, da ein ausgefülltes Prüfungsformular ca. eine Grösse von 60Kbyte aufweist
3. Das Prüfungsformular sollte maximal drei Seiten gross sein, um eine gute Gesamtübersicht zu gewährleisten

18 Dezember 09

TabMed – Beta 1 Testdokumentation

Rico Suter & Fabian Mettler

Dokumentinformationen

Änderungsgeschichte

Datum	Version	Änderung	Autor
16.11.2009	0.1	Erste Version	fme & rsu
17.11.2009	1.0	Ergänzungen	fme

Inhaltsverzeichnis

Funktionale Systemtests

T01: Prüfung einrichten – TabMedClient, TabMedServer

Definition

Der Administrator verbindet sich mit dem Server. Im nächsten Schritt wählt er eine Prüfung, eine Durchführung und eine Station aus. Anschlissend wartet er auf den Prüfungsstart.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T01.1: Benutzer verbindet sich mit dem Server	OK
T01.2: Prüfung auswählen	OK
T01.3: Durchführung auswählen	Ok
T01.4: Station auswählen	OK
T01.5: Auf Prüfungsstart warten	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T01.1: Benutzer verbindet sich mit dem Server

T01.1: Benutzer verbindet sich mit dem Server	
Beschreibung	Der Benutzer verbindet sich durch eine Server-Adresse mit dem Server.
Erwartetes Resultat	Der Benutzer kann eine Server-Adresse eingeben und sich mit diesem Server verbinden.
Test Daten	Server: http://sinv-56028.edu.hsr.ch:22
Resultat	Verbindung ist hergestellt
Kommentar	-
Test Datum	25.10.2009

T01.2: Prüfung auswählen

T01.2: Prüfung auswählen	
Beschreibung	Der Benutzer wählt aus einer Liste von Prüfungen aus.
Erwartetes Resultat	Die gewünschte Prüfung konnte ausgewählt werden.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Prüfung konnte ausgewählt werden.
Kommentar	-
Test Datum	25.10.2009

T01.3: Durchführung auswählen

T01.3: Durchführung auswählen	
Beschreibung	Der Benutzer wählt aus einer Liste von Durchführungen aus.
Erwartetes Resultat	Die gewünschte Durchführung konnte ausgewählt werden.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Durchführung konnte ausgewählt werden.
Kommentar	-
Test Datum	25.10.2009

T01.4: Station auswählen

T01.4: Station auswählen	
Beschreibung	Der Benutzer wählt aus einer Liste von Stationen aus.
Erwartetes Resultat	Die gewünschte Station konnte ausgewählt werden.
Test Daten	TextExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Rotation konnte ausgewählt werden.
Kommentar	-
Test Datum	25.10.2009

T01.5: Auf Prüfungsstart warten

T01.5: Auf Prüfungsstart warten	
Beschreibung	Es wird auf den Prüfungsstart gewartet.
Erwartetes Resultat	Die Applikation zeigt einen Warte-Bildschirm und zeigt Informationen über die Auswahl an.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Der Warte-Bildschirm wird angezeigt.
Kommentar	-
Test Datum	25.10.2009

T02: Kandidaten bewerten - TabMedClient

Definition

Der Benutzer füllt das vorgegebene Prüfungsformular des Kandidaten aus. Für eine zusätzliche Beurteilung notiert sich der Benutzer ergänzende Bemerkungen. Zum Schluss speichert er das ausgefüllte Formular und wählt den nächsten Kandidaten aus.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T02.1: Prüfungsbogen ausfüllen	OK
T02.2: Notizen erfassen	OFFEN
T02.3: Prüfungsbogen speichern	OK
T02.4: Kandidat auswählen	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T02.1: Prüfungsbogen ausfüllen

T02.1: Prüfungsbogen ausfüllen	
Beschreibung	Der Benutzer füllt das Prüfungsformular des aktuellen Kandidaten aus.
Erwartetes Resultat	Der Benutzer kann das Prüfungsformular des aktuellen Kandidaten ausfüllen.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Prüfungsformular ist ausgefüllt.
Kommentar	-
Test Datum	25.10.2009

T02.2: Notizen erfassen

T02.2: Notizen erfassen	
Beschreibung	Der Benutzer wechselt in die Notiz-Ansicht und schreibt einige Notizen auf.
Erwartetes Resultat	Der Benutzer konnte eigene Notizen aufschreiben.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T02.3: Prüfungsbogen speichern

T02.3: Prüfungsbogen speichern	
Beschreibung	Der Benutzer füllt ein Formular aus und speichert dieses anschliessend.
Erwartetes Resultat	Die Änderungen wurden auf den Server, USB-Stick und auf die Festplatte gespeichert.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Das Formular befindet sich auf dem Server, USB-Stick und der Festplatte.
Kommentar	-
Test Datum	25.10.2009

T02.4: Kandidat auswählen

T02.4: Kandidat auswählen	
Beschreibung	Der Benutzer wechselt zwischen zwei Kandidaten.
Erwartetes Resultat	Der Benutzer konnte einen Kandidaten aus der Kandidatenliste der aktuellen Durchführung auswählen und ihn danach bearbeiten.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Der Benutzer konnte zum gewünschten Kandidaten wechseln.
Kommentar	-
Test Datum	25.10.2009

T03: Dongle – TabMedClient

Definition

Der Benutzer kann sich am System an- und abmelden. Dabei soll ein USB-Stick verwendet werden. Ist der USB-Stick eingestekt, ist der Benutzer angemeldet, wird er herausgezogen, wird der Bildschirm gesperrt. Auf dem USB-Stick ist zudem ein Zertifikat vorhanden, mit dem der Prüfungsbogen beim speichern signiert wird.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T03.1: Benutzernameldung	OFFEN
T03.2: Prüfungsbogen signieren	OFFEN

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T03.1: Benutzernameldung

T03.1: Benutzernameldung	
Beschreibung	Der Benutzer meldet sich mit dem USB-Stick am System an und ab.
Erwartetes Resultat	Beim Einstecken des USB-Sticks wird der Bildschirm entsperrt und beim Entfernen wieder gesperrt.
Test Daten	
Resultat	
Kommentar	Noch nicht implementiert.
Test Datum	

T03.2: Prüfungsbogen signieren

T03.2: Prüfungsbogen signieren	
Beschreibung	Der Benutzer speichert einen Prüfungsbogen, mit eingestecktem USB-Stick, und signiert diesen.
Erwartetes Resultat	Nachdem Speichern des Formulars ist das XML signiert.
Test Daten	
Resultat	
Kommentar	Noch nicht implementiert.
Test Datum	

T04: Prüfungen einlesen – TabMedServer

Definition

Der TabMedServer liest die Prüfungen (XML-Daten) aus dem „exams“-Verzeichnis ein und bietet diese zur Auswahl an.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T04.1: Prüfungen einlesen	OFFEN

Bemerkungen

-

T04.1: Prüfungen einlesen

T04.1: Prüfungen einlesen	
Beschreibung	Es werden Prüfungen (XML-Daten) in das „exams“-Verzeichnis abgelegt und der Server wird gestartet.
Erwartetes Resultat	Der Server liest die Prüfungen ein und bietet diese zur Auswahl an.
Test Daten	
Resultat	
Kommentar	Noch nicht implementiert.
Test Datum	

T05: Prüfungsbogen erstellen / bearbeiten – TabMedEditor

Definition

Der Benutzer startet den Editor und erstellt eine Prüfung. Anschliessend öffnet er eine vorhandene Prüfung und bearbeitet gewisse Blöcke.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T05.1: Fomurlar erstellen	OFFEN
T05.2: Bestehendes Formular bearbeiten	OFFEN

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T05.1: Formular erstellen

T05.1: Formular erstellen	
Beschreibung	Der Benutzer erstellt ein neues Prüfungsformular.
Erwartetes Resultat	Es wurde ein neues Prüfungsformular mit Themen, Blöcken und Antwortstypen erstellt.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

T05.2: Bestehendes Formular bearbeiten

T05.2: Bestehendes Formular bearbeiten	
Beschreibung	Der Benutzer bearbeitet ein bestehendes Formular.
Erwartetes Resultat	Im bestehenden Formular konnten Bereiche geändert werden.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

T06: Prüfungsrotation starten – Terminal

Definition

Der Benutzer startet die Prüfungsrotation durch das Terminal.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T06.1: Prüfungsrotation starten	OFFEN

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T06.1: Prüfungsrotation starten

T06.1: Prüfungsrotation starten	
Beschreibung	Die Prüfungsrotatin wird durch das Terminal gestartet.
Erwartetes Resultat	Die Prüfungsrotation ist gestartet.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

T07: Prüfung koordieren – TabMedFrontend

Definition

Der Benutzer meldet sich am Frontend an und ladet eine neue Prüfung auf den Server. Anschliessend lässt er sich den Status der Clients anzeigen und startet die Prüfungsrotation durch das Frontend.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T07.1: Benutzer anmelden	OFFEN
T07.2: Prüfung hochladen	OFFEN
T07.3: Status der Clients anzeigen	OFFEN
T07.4: Prüfungsrotation starten	OFFEN

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T07.1: Benutzer anmelden

T07.1: Benutzer anmelden	
Beschreibung	Der Benutzer meldet sich beim Frontend an.
Erwartetes Resultat	Der Benutzer konnte sich anmelden.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

T07.2: Prüfung hochladen

T07.2: Prüfung hochladen	
Beschreibung	Der Benutzer lädt eine neue Prüfung auf den Server.
Erwartetes Resultat	Die Prüfung ist auf dem Server vorhanden und wird bereit gestellt.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

T07.3: Status der Clients anzeigen

T07.3: Status der Clients anzeigen	
Beschreibung	Der Benutzer lässt sich den Status der Clients anzeigen.
Erwartetes Resultat	Es werden die aktiven Geräte in der Übersicht angezeigt.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

T07.4: Prüfungsrotation starten

T07.4: Prüfungsrotation starten	
Beschreibung	Der Benutzer startet die Prüfungsrotation.
Erwartetes Resultat	Die Prüfungsrotation ist gestartet.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

Nicht funktionale Systemtest

Bemerkungen

Im aktuellen Release werden keine nicht funktionalen Tests durchgeführt. Es wird aber dargstellt, welche Tests später durchgeführt werden müssen.

T08: Leistungsanforderung

Definition

Der Experte steht während der Prüfung unter Zeitdruck. Die Applikation muss schnell auf die Eingaben des Experten reagieren und es dürfen keine Wartezeiten entstehen. Die Applikation soll immer flüssig laufen und es dürfen keine grösseren Wartezeiten als 0.4 Sekunden entstehen. Eine Ausnahme bildet eine eventuell etwas längere Wartezeit beim Senden oder Empfangen des Prüfungsformulars. Für die Wartezeiten ist die Performanz auf den zur Verfügung gestellten Client-Devices massgebend.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T08.1: Wartezeiten bei Eingaben durch den Experten	OFFEN

T08.1: Wartezeiten bei Eingaben durch den Experten

T08.1: Wartezeiten bei Eingaben durch den Experten	
Beschreibung	Es wird gemessen, wie lange die Wartezeiten bei UI-Eingaben ist.
Erwartetes Resultat	Die Wartezeiten sind kleiner als 0.4 Sekunden.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T09: Verfügbarkeit und Datensicherung

Definition

Während des gesamten Zeitraums einer OSCE-Prüfung muss jede Client/Server Verbindung einwandfrei funktionieren. Es kann aber vorkommen, dass ein Gerät ausfällt z.B. wenn es vom Tisch hinunterfällt. In diesem Fall soll an dieser Station das Gerät ersetzt werden können um die Prüfung fortzusetzen. Natürlich hat auch die Server- Applikation stabil zu laufen, da ansonsten die Prüfungsdurchführung unmöglich wird. Es wird angenommen, dass die Infrastruktur durch das IML bereitgestellt wird und einwandfrei läuft. Jedes Gerät hat also immer Zugriff auf das Netz, entweder durch das LAN oder WLAN.

Die Prüfungsergebnisse müssen jeder Zeit sicher gestellt sein. Eine Kandidatenbewertung soll immer auf dem Server gesichert werden. Bei einem Ausfall des Servers z.B. Stromausfall, dürfen die erfassten Resultate nicht verloren gehen.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T09.1: Verbindungsunterbruch	OFFEN
T09.2: Ersetzen eines defekten Clients	OFFEN
T09.3: Datensicherung	OFFEN

T09.1: Verbindungsunterbruch

T09.1: Verbindungsunterbruch	
Beschreibung	Die Verbindung vom Client zum Server wird unterbrochen.
Erwartetes Resultat	Der Client arbeitet ohne Verzögerungen weiter und speichert die UI-Eingaben in einer Warteschlange, um diese später an den Server zu schicken.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T09.2: Ersetzen eines defekten Clients

T09.2: Ersetzen eines defekten Clients	
Beschreibung	Der Client wird durch ein neues Gerät ersetzt.
Erwartetes Resultat	Der Benutzer arbeitet am neuen Client weiter.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T09.3: Datensicherung

T09.3: Datensicherung	
Beschreibung	Verbindung zum Server wird unterbrochen.
Erwartetes Resultat	Die Verbindung zum Server wird unterbrochen. Die Daten werden aber weiterhin auf den USB-Stick und auf die Festplatte gespeichert.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T10: Sicherheit

Definition

Während des gesamten Zeitraums einer OSCE-Prüfung muss die Client/Server Verbindung vor unberechtigten Personen und Manipulationen geschützt sein. Sowohl auf die Prüfungsbogen, wie auch auf die ausgefüllten Prüfungen soll auf keinen Fall, vom Netz aus, unbefugter Zugriff stattfinden können. Nach abgeschlossener Prüfung muss garantiert sein, dass die Prüfungsresultate nicht mehr manipuliert werden können und nicht verlogen gehen. Ein Datenverlust darf niemals auftreten.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T10.1: Verbindungen sind verschlüsselt	OFFEN

T10.1: Verbindungen sind verschlüsselt

T10.1: Verbindungen sind verschlüsselt	
Beschreibung	Die Verbindungen eines Clients zum Server werden mittels Wireshark (Sniffer) überwacht und analysiert.
Erwartetes Resultat	Die Daten im Wireshark sind nur in verschlüsselter Form vorhanden.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T11: Qualitätsanforderungen

Definition

Das Design des Clients soll Personen zwischen 30 und 70 Jahren mit grossem medizinischem, aber tendenziell geringem Informatik-Fachwissen ansprechen. Dabei ist vor allem wichtig, dass eine optimale Benutzbarkeit der Anwendung unter Zeitdruck entwickelt wird.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T11.1: Verständlichkeit	OFFEN
T11.2: Erlernbarkeit	OFFEN

T11.1: Verständlichkeit

T11.1: Verständlichkeit	
Beschreibung	Es wird geprüft, ob die Bedienung der Applikation intuitiv ist.
Erwartetes Resultat	Die Benutzeroberfläche ist grafisch ansprechend und selbsterklärend.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T11.2: Erlernbarkeit

T11.2: Erlernbarkeit	
Beschreibung	Es wird geprüft, ob die Bedienung der Applikation schnell zu erlernen ist.
Erwartetes Resultat	Der Umgang mit der Applikation ist nach einer kurzen Einführung einfach zu erlenen.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

Usability Tests

Bemerkung

Im aktuellen Release werden keine Usability Tests durchgeführt.

18 Dezember 09

TabMed – Beta 2 Testdokumentation

Rico Suter & Fabian Mettler

Dokumentinformationen

Änderungsgeschichte

Datum	Version	Änderung	Autor
16.11.2009	0.1	Erste Version	fme & rsu
17.11.2009	1.0	Ergänzungen	fme

Inhaltsverzeichnis

Funktionale Systemtests

T01: Prüfung einrichten – TabMedClient, TabMedServer

Definition

Der Administrator verbindet sich mit dem Server. Im nächsten Schritt wählt er eine Prüfung, eine Durchführung und eine Station aus. Anschlissend wartet er auf den Prüfungsstart.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T01.1: Benutzer verbindet sich mit dem Server	OK
T01.2: Prüfung auswählen	OK
T01.3: Durchführung auswählen	Ok
T01.4: Station auswählen	OK
T01.5: Auf Prüfungsstart warten	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T01.1: Benutzer verbindet sich mit dem Server

T01.1: Benutzer verbindet sich mit dem Server	
Beschreibung	Der Benutzer verbindet sich durch eine Server-Adresse mit dem Server.
Erwartetes Resultat	Der Benutzer kann eine Server-Adresse eingeben und sich mit diesem Server verbinden.
Test Daten	Server: http://sinv-56028.edu.hsr.ch:22
Resultat	Verbindung ist hergestellt
Kommentar	-
Test Datum	22.11.2009

T01.2: Prüfung auswählen

T01.2: Prüfung auswählen	
Beschreibung	Der Benutzer wählt aus einer Liste von Prüfungen aus.
Erwartetes Resultat	Die gewünschte Prüfung konnte ausgewählt werden.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Prüfung konnte ausgewählt werden.
Kommentar	-
Test Datum	22.11.2009

T01.3: Durchführung auswählen

T01.3: Durchführung auswählen	
Beschreibung	Der Benutzer wählt aus einer Liste von Durchführungen aus.
Erwartetes Resultat	Die gewünschte Durchführung konnte ausgewählt werden.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Durchführung konnte ausgewählt werden.
Kommentar	-
Test Datum	22.11.2009

T01.4: Station auswählen

T01.4: Station auswählen	
Beschreibung	Der Benutzer wählt aus einer Liste von Stationen aus.
Erwartetes Resultat	Die gewünschte Station konnte ausgewählt werden.
Test Daten	TextExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Rotation konnte ausgewählt werden.
Kommentar	-
Test Datum	22.11.2009

T01.5: Auf Prüfungsstart warten

T01.5: Auf Prüfungsstart warten	
Beschreibung	Es wird auf den Prüfungsstart gewartet.
Erwartetes Resultat	Die Applikation zeigt einen Warte-Bildschirm und zeigt Informationen über die Auswahl an.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Der Warte-Bildschirm wird angezeigt.
Kommentar	-
Test Datum	22.11.2009

T02: Kandidaten bewerten - TabMedClient

Definition

Der Benutzer füllt das vorgegebene Prüfungsformular des Kandidaten aus. Für eine zusätzliche Beurteilung notiert sich der Benutzer ergänzende Bemerkungen. Zum Schluss speichert er das ausgefüllte Formular und wählt den nächsten Kandidaten aus.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T02.1: Prüfungsbogen ausfüllen	OK
T02.2: Notizen erfassen	OK
T02.3: Prüfungsbogen speichern	OK
T02.4: Kandidat auswählen	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T02.1: Prüfungsbogen ausfüllen

T02.1: Prüfungsbogen ausfüllen	
Beschreibung	Der Benutzer füllt das Prüfungsformular des aktuellen Kandidaten aus.
Erwartetes Resultat	Der Benutzer kann das Prüfungsformular des aktuellen Kandidaten ausfüllen.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Prüfungsformular ist ausgefüllt.
Kommentar	-
Test Datum	22.11.2009

T02.2: Notizen erfassen

T02.2: Notizen erfassen	
Beschreibung	Der Benutzer wechselt in die Notiz-Ansicht und schreibt einige Notizen auf.
Erwartetes Resultat	Der Benutzer konnte eigene Notizen aufschreiben.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Die Notizen wurden ins Formular eingebettet.
Kommentar	-
Test Datum	22.11.2009

T02.3: Prüfungsbogen speichern

T02.3: Prüfungsbogen speichern	
Beschreibung	Der Benutzer füllt ein Formular aus und speichert dieses anschliessend.
Erwartetes Resultat	Die Änderungen wurden auf den Server, USB-Stick und auf die Festplatte gespeichert.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Das Formular befindet sich auf dem Server, USB-Stick und der Festplatte.
Kommentar	-
Test Datum	22.11.2009

T02.4: Kandidat auswählen

T02.4: Kandidat auswählen	
Beschreibung	Der Benutzer wechselt zwischen zwei Kandidaten.
Erwartetes Resultat	Der Benutzer konnte einen Kandidaten aus der Kandidatenliste der aktuellen Durchführung auswählen und ihn danach bearbeiten.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Der Benutzer konnte zum gewünschten Kandidaten wechseln.
Kommentar	-
Test Datum	22.11.2009

T03: Dongle – TabMedClient

Definition

Der Benutzer kann sich am System an- und abmelden. Dabei soll ein USB-Stick verwendet werden. Ist der USB-Stick eingestekt, ist der Benutzer angemeldet, wird er herausgezogen, wird der Bildschirm gesperrt. Auf dem USB-Stick ist zudem ein Zertifikat vorhanden, mit dem der Prüfungsbogen beim speichern signiert wird.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T03.1: Benutzernameldung	OK
T03.2: Prüfungsbogen signieren	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T03.1: Benutzernameldung

T03.1: Benutzernameldung	
Beschreibung	Der Benutzer meldet sich mit dem USB-Stick am System an und ab.
Erwartetes Resultat	Beim Einstecken des USB-Sticks wird der Bildschirm entsperrt und beim Entfernen wieder gesperrt.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Der Benutzer wurde angemeldet und der Bildschirm freigegeben.
Kommentar	-
Test Datum	22.11.2009

T03.2: Prüfungsbogen signieren

T03.2: Prüfungsbogen signieren	
Beschreibung	Der Benutzer speichert einen Prüfungsbogen, mit eingestecktem USB-Stick, und signiert diesen.
Erwartetes Resultat	Nachdem Speichern des Formulars ist das XML signiert.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Das Formular (XML-Daten) wurde durch das Zertifikat auf dem USB-Stick signiert.
Kommentar	-
Test Datum	22.11.2009

T04: Prüfungen einlesen – TabMedServer

Definition

Der TabMedServer liest die Prüfungen (XML-Daten) aus dem „exams“-Verzeichnis ein und bietet diese zur Auswahl an.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T04.1: Prüfungen einlesen	OK

Bemerkungen

-

T04.1: Prüfungen einlesen

T04.1: Prüfungen einlesen	
Beschreibung	Es werden Prüfungen (XML-Daten) in das „exams“-Verzeichnis abgelegt und der Server wird gestartet.
Erwartetes Resultat	Der Server liest die Prüfungen ein und bietet diese zur Auswahl an.
Test Daten	Server: http://sinv-56028.edu.hsr.ch:22 Prüfungen: XML Daten
Resultat	Daten sind eingelesen und zur Verfügung gestellt.
Kommentar	-
Test Datum	22.11.2009

T05: Prüfungsbogen erstellen / bearbeiten – TabMedEditor

Definition

Der Benutzer startet den Editor und erstellt eine Prüfung. Anschliessend öffnet er eine vorhandene Prüfung und bearbeitet gewisse Blöcke.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T05.1: Fomurlar erstellen	OK
T05.2: Bestehendes Formular bearbeiten	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T05.1: Formular erstellen

T05.1: Formular erstellen	
Beschreibung	Der Benutzer erstellt ein neues Prüfungsformular.
Erwartetes Resultat	Es wurde ein neues Prüfungsformular mit Themen, Blöcken und Antwortstypen erstellt.
Test Daten	Psychotest als Vorlage.
Resultat	Formular ist erstellt.
Kommentar	-
Test Datum	22.11.2009

T05.2: Bestehendes Formular bearbeiten

T05.2: Bestehendes Formular bearbeiten	
Beschreibung	Der Benutzer bearbeitet ein bestehendes Formular.
Erwartetes Resultat	Im bestehenden Formular konnten Bereiche geändert werden.
Test Daten	Psychotest (XML-Daten)
Resultat	Formular ist bearbeitet.
Kommentar	-
Test Datum	22.11.2009

T06: Prüfungsrotation starten – Terminal

Definition

Der Benutzer startet die Prüfungsrotation durch das Terminal.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T06.1: Prüfungsrotation starten	OFFEN

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T06.1: Prüfungsrotation starten

T06.1: Prüfungsrotation starten	
Beschreibung	Die Prüfungsrotatin wird durch das Terminal gestartet.
Erwartetes Resultat	Die Prüfungsrotation ist gestartet.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

T07: Prüfung koordieren – TabMedFrontend

Definition

Der Benutzer meldet sich am Frontend an und ladet eine neue Prüfung auf den Server. Anschliessend lässt er sich den Status der Clients anzeigen und startet die Prüfungsrotation durch das Frontend.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T07.1: Benutzer anmelden	OFFEN
T07.2: Prüfung hochladen	OFFEN
T07.3: Status der Clients anzeigen	OFFEN
T07.4: Prüfungsrotation starten	OFFEN

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T07.1: Benutzer anmelden

T07.1: Benutzer anmelden	
Beschreibung	Der Benutzer meldet sich beim Frontend an.
Erwartetes Resultat	Der Benutzer konnte sich anmelden.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

T07.2: Prüfung hochladen

T07.2: Prüfung hochladen	
Beschreibung	Der Benutzer lädt eine neue Prüfung auf den Server.
Erwartetes Resultat	Die Prüfung ist auf dem Server vorhanden und wird bereit gestellt.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

T07.3: Status der Clients anzeigen

T07.3: Status der Clients anzeigen	
Beschreibung	Der Benutzer lässt sich den Status der Clients anzeigen.
Erwartetes Resultat	Es werden die aktiven Geräte in der Übersicht angezeigt.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

T07.4: Prüfungsrotation starten

T07.4: Prüfungsrotation starten	
Beschreibung	Der Benutzer startet die Prüfungsrotation.
Erwartetes Resultat	Die Prüfungsrotation ist gestartet.
Test Daten	
Resultat	Nicht nicht implementiert
Kommentar	
Test Datum	

Nicht funktionale Systemtest

Bemerkungen

Im aktuellen Release werden keine nicht funktionalen Tests durchgeführt. Es wird aber dargstellt, welche Tests später durchgeführt werden müssen.

T08: Leistungsanforderung

Definition

Der Experte steht während der Prüfung unter Zeitdruck. Die Applikation muss schnell auf die Eingaben des Experten reagieren und es dürfen keine Wartezeiten entstehen. Die Applikation soll immer flüssig laufen und es dürfen keine grösseren Wartezeiten als 0.4 Sekunden entstehen. Eine Ausnahme bildet eine eventuell etwas längere Wartezeit beim Senden oder Empfangen des Prüfungsformulars. Für die Wartezeiten ist die Performanz auf den zur Verfügung gestellten Client-Devices massgebend.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T08.1: Wartezeiten bei Eingaben durch den Experten	OFFEN

T08.1: Wartezeiten bei Eingaben durch den Experten

T08.1: Wartezeiten bei Eingaben durch den Experten	
Beschreibung	Es wird gemessen, wie lange die Wartezeiten bei UI-Eingaben ist.
Erwartetes Resultat	Die Wartezeiten sind kleiner als 0.4 Sekunden.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T09: Verfügbarkeit und Datensicherung

Definition

Während des gesamten Zeitraums einer OSCE-Prüfung muss jede Client/Server Verbindung einwandfrei funktionieren. Es kann aber vorkommen, dass ein Gerät ausfällt z.B. wenn es vom Tisch hinunterfällt. In diesem Fall soll an dieser Station das Gerät ersetzt werden können um die Prüfung fortzusetzen. Natürlich hat auch die Server- Applikation stabil zu laufen, da ansonsten die Prüfungsdurchführung unmöglich wird. Es wird angenommen, dass die Infrastruktur durch das IML bereitgestellt wird und einwandfrei läuft. Jedes Gerät hat also immer Zugriff auf das Netz, entweder durch das LAN oder WLAN.

Die Prüfungsergebnisse müssen jeder Zeit sicher gestellt sein. Eine Kandidatenbewertung soll immer auf dem Server gesichert werden. Bei einem Ausfall des Servers z.B. Stromausfall, dürfen die erfassten Resultate nicht verloren gehen.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T09.1: Verbindungsunterbruch	OFFEN
T09.2: Ersetzen eines defekten Clients	OFFEN
T09.3: Datensicherung	OFFEN

T09.1: Verbindungsunterbruch

T09.1: Verbindungsunterbruch	
Beschreibung	Die Verbindung vom Client zum Server wird unterbrochen.
Erwartetes Resultat	Der Client arbeitet ohne Verzögerungen weiter und speichert die UI-Eingaben in einer Warteschlange, um diese später an den Server zu schicken.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T09.2: Ersetzen eines defekten Clients

T09.2: Ersetzen eines defekten Clients	
Beschreibung	Der Client wird durch ein neues Gerät ersetzt.
Erwartetes Resultat	Der Benutzer arbeitet am neuen Client weiter.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T09.3: Datensicherung

T09.3: Datensicherung	
Beschreibung	Verbindung zum Server wird unterbrochen.
Erwartetes Resultat	Die Verbindung zum Server wird unterbrochen. Die Daten werden aber weiterhin auf den USB-Stick und auf die Festplatte gespeichert.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T10: Sicherheit

Definition

Während des gesamten Zeitraums einer OSCE-Prüfung muss die Client/Server Verbindung vor unberechtigten Personen und Manipulationen geschützt sein. Sowohl auf die Prüfungsbogen, wie auch auf die ausgefüllten Prüfungen soll auf keinen Fall, vom Netz aus, unbefugter Zugriff stattfinden können. Nach abgeschlossener Prüfung muss garantiert sein, dass die Prüfungsresultate nicht mehr manipuliert werden können und nicht verlogen gehen. Ein Datenverlust darf niemals auftreten.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T10.1: Verbindungen sind verschlüsselt	OFFEN

T10.1: Verbindungen sind verschlüsselt

T10.1: Verbindungen sind verschlüsselt	
Beschreibung	Die Verbindungen eines Clients zum Server werden mittels Wireshark (Sniffer) überwacht und analysiert.
Erwartetes Resultat	Die Daten im Wireshark sind nur in verschlüsselter Form vorhanden.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T11: Qualitätsanforderungen

Definition

Das Design des Clients soll Personen zwischen 30 und 70 Jahren mit grossem medizinischem, aber tendenziell geringem Informatik-Fachwissen ansprechen. Dabei ist vor allem wichtig, dass eine optimale Benutzbarkeit der Anwendung unter Zeitdruck entwickelt wird.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T11.1: Verständlichkeit	OFFEN
T11.2: Erlernbarkeit	OFFEN

T11.1: Verständlichkeit

T11.1: Verständlichkeit	
Beschreibung	Es wird geprüft, ob die Bedienung der Applikation intuitiv ist.
Erwartetes Resultat	Die Benutzeroberfläche ist grafisch ansprechend und selbsterklärend.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

T11.2: Erlernbarkeit

T11.2: Erlernbarkeit	
Beschreibung	Es wird geprüft, ob die Bedienung der Applikation schnell zu erlernen ist.
Erwartetes Resultat	Der Umgang mit der Applikation ist nach einer kurzen Einführung einfach zu erlenen.
Test Daten	
Resultat	Noch nicht implementiert
Kommentar	
Test Datum	

Usability Tests

Bemerkung

Im aktuellen Release werden keine Usability Tests durchgeführt.

18 Dezember 09

TabMed – Beta 3 Testdokumentation

Rico Suter & Fabian Mettler

Dokumentinformationen

Änderungsgeschichte

Datum	Version	Änderung	Autor
16.11.2009	0.1	Erste Version	fme & rsu
17.11.2009	1.0	Ergänzungen	fme

Inhaltsverzeichnis

Funktionale Systemtests

T01: Prüfung einrichten – TabMedClient, TabMedServer

Definition

Der Administrator verbindet sich mit dem Server. Im nächsten Schritt wählt er eine Prüfung, eine Durchführung und eine Station aus. Anschlissend wartet er auf den Prüfungsstart.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T01.1: Benutzer verbindet sich mit dem Server	OK
T01.2: Prüfung auswählen	OK
T01.3: Durchführung auswählen	Ok
T01.4: Station auswählen	OK
T01.5: Auf Prüfungsstart warten	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T01.1: Benutzer verbindet sich mit dem Server

T01.1: Benutzer verbindet sich mit dem Server	
Beschreibung	Der Benutzer verbindet sich durch eine Server-Adresse mit dem Server.
Erwartetes Resultat	Der Benutzer kann eine Server-Adresse eingeben und sich mit diesem Server verbinden.
Test Daten	Server: http://sinv-56028.edu.hsr.ch:22
Resultat	Verbindung ist hergestellt
Kommentar	-
Test Datum	22.11.2009

T01.2: Prüfung auswählen

T01.2: Prüfung auswählen	
Beschreibung	Der Benutzer wählt aus einer Liste von Prüfungen aus.
Erwartetes Resultat	Die gewünschte Prüfung konnte ausgewählt werden.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Prüfung konnte ausgewählt werden.
Kommentar	-
Test Datum	22.11.2009

T01.3: Durchführung auswählen

T01.3: Durchführung auswählen	
Beschreibung	Der Benutzer wählt aus einer Liste von Durchführungen aus.
Erwartetes Resultat	Die gewünschte Durchführung konnte ausgewählt werden.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Durchführung konnte ausgewählt werden.
Kommentar	-
Test Datum	22.11.2009

T01.4: Station auswählen

T01.4: Station auswählen	
Beschreibung	Der Benutzer wählt aus einer Liste von Stationen aus.
Erwartetes Resultat	Die gewünschte Station konnte ausgewählt werden.
Test Daten	TextExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Rotation konnte ausgewählt werden.
Kommentar	-
Test Datum	22.11.2009

T01.5: Auf Prüfungsstart warten

T01.5: Auf Prüfungsstart warten	
Beschreibung	Es wird auf den Prüfungsstart gewartet.
Erwartetes Resultat	Die Applikation zeigt einen Warte-Bildschirm und zeigt Informationen über die Auswahl an.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Der Warte-Bildschirm wird angezeigt.
Kommentar	-
Test Datum	22.11.2009

T02: Kandidaten bewerten - TabMedClient

Definition

Der Benutzer füllt das vorgegebene Prüfungsformular des Kandidaten aus. Für eine zusätzliche Beurteilung notiert sich der Benutzer ergänzende Bemerkungen. Zum Schluss speichert er das ausgefüllte Formular und wählt den nächsten Kandidaten aus.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T02.1: Prüfungsbogen ausfüllen	OK
T02.2: Notizen erfassen	OK
T02.3: Prüfungsbogen speichern	OK
T02.4: Kandidat auswählen	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T02.1: Prüfungsbogen ausfüllen

T02.1: Prüfungsbogen ausfüllen	
Beschreibung	Der Benutzer füllt das Prüfungsformular des aktuellen Kandidaten aus.
Erwartetes Resultat	Der Benutzer kann das Prüfungsformular des aktuellen Kandidaten ausfüllen.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Prüfungsformular ist ausgefüllt.
Kommentar	-
Test Datum	22.11.2009

T02.2: Notizen erfassen

T02.2: Notizen erfassen	
Beschreibung	Der Benutzer wechselt in die Notiz-Ansicht und schreibt einige Notizen auf.
Erwartetes Resultat	Der Benutzer konnte eigene Notizen aufschreiben.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Die Notizen wurden ins Formular eingebettet.
Kommentar	-
Test Datum	22.11.2009

T02.3: Prüfungsbogen speichern

T02.3: Prüfungsbogen speichern	
Beschreibung	Der Benutzer füllt ein Formular aus und speichert dieses anschliessend.
Erwartetes Resultat	Die Änderungen wurden auf den Server, USB-Stick und auf die Festplatte gespeichert.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Das Formular befindet sich auf dem Server, USB-Stick und der Festplatte.
Kommentar	-
Test Datum	22.11.2009

T02.4: Kandidat auswählen

T02.4: Kandidat auswählen	
Beschreibung	Der Benutzer wechselt zwischen zwei Kandidaten.
Erwartetes Resultat	Der Benutzer konnte einen Kandidaten aus der Kandidatenliste der aktuellen Durchführung auswählen und ihn danach bearbeiten.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Der Benutzer konnte zum gewünschten Kandidaten wechseln.
Kommentar	-
Test Datum	22.11.2009

T03: Dongle – TabMedClient

Definition

Der Benutzer kann sich am System an- und abmelden. Dabei soll ein USB-Stick verwendet werden. Ist der USB-Stick eingestekt, ist der Benutzer angemeldet, wird er herausgezogen, wird der Bildschirm gesperrt. Auf dem USB-Stick ist zudem ein Zertifikat vorhanden, mit dem der Prüfungsbogen beim speichern signiert wird.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T03.1: Benutzernameldung	OK
T03.2: Prüfungsbogen signieren	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T03.1: Benutzernameldung

T03.1: Benutzernameldung	
Beschreibung	Der Benutzer meldet sich mit dem USB-Stick am System an und ab.
Erwartetes Resultat	Beim Einstecken des USB-Sticks wird der Bildschirm entsperrt und beim Entfernen wieder gesperrt.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Der Benutzer wurde angemeldet und der Bildschirm freigegeben.
Kommentar	-
Test Datum	22.11.2009

T03.2: Prüfungsbogen signieren

T03.2: Prüfungsbogen signieren	
Beschreibung	Der Benutzer speichert einen Prüfungsbogen, mit eingestecktem USB-Stick, und signiert diesen.
Erwartetes Resultat	Nachdem Speichern des Formulars ist das XML signiert.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Das Formular (XML-Daten) wurde durch das Zertifikat auf dem USB-Stick signiert.
Kommentar	-
Test Datum	22.11.2009

T04: Prüfungen einlesen – TabMedServer

Definition

Der TabMedServer liest die Prüfungen (XML-Daten) aus dem „exams“-Verzeichnis ein und bietet diese zur Auswahl an.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T04.1: Prüfungen einlesen	OK

Bemerkungen

-

T04.1: Prüfungen einlesen

T04.1: Prüfungen einlesen	
Beschreibung	Es werden Prüfungen (XML-Daten) in das „exams“-Verzeichnis abgelegt und der Server wird gestartet.
Erwartetes Resultat	Der Server liest die Prüfungen ein und bietet diese zur Auswahl an.
Test Daten	Server: http://sinv-56028.edu.hsr.ch:22 Prüfungen: XML Daten
Resultat	Daten sind eingelesen und zur Verfügung gestellt.
Kommentar	-
Test Datum	22.11.2009

T05: Prüfungsbogen erstellen / bearbeiten – TabMedEditor

Definition

Der Benutzer startet den Editor und erstellt eine Prüfung. Anschliessend öffnet er eine vorhandene Prüfung und bearbeitet gewisse Blöcke.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T05.1: Fomurlar erstellen	OK
T05.2: Bestehendes Formular bearbeiten	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T05.1: Formular erstellen

T05.1: Formular erstellen	
Beschreibung	Der Benutzer erstellt ein neues Prüfungsformular.
Erwartetes Resultat	Es wurde ein neues Prüfungsformular mit Themen, Blöcken und Antwortstypen erstellt.
Test Daten	Psychotest als Vorlage.
Resultat	Formular ist erstellt.
Kommentar	-
Test Datum	22.11.2009

T05.2: Bestehendes Formular bearbeiten

T05.2: Bestehendes Formular bearbeiten	
Beschreibung	Der Benutzer bearbeitet ein bestehendes Formular.
Erwartetes Resultat	Im bestehenden Formular konnten Bereiche geändert werden.
Test Daten	Psychotest (XML-Daten)
Resultat	Formular ist bearbeitet.
Kommentar	-
Test Datum	22.11.2009

T06: Prüfungsrotation starten – Terminal

Definition

Der Benutzer startet die Prüfungsrotation durch das Terminal.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	

Übersicht

Test	Status
T06.1: Prüfungsrotation starten	OFFEN

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T06.1: Prüfungsrotation starten

T06.1: Prüfungsrotation starten	
Beschreibung	Die Prüfungsrotatin wird durch das Terminal gestartet.
Erwartetes Resultat	Die Prüfungsrotation ist gestartet.
Test Daten	Server: https://sinv56028.edu.hsr.ch:22
Resultat	OFFEN
Kommentar	Nicht nicht implementiert.
Test Datum	-

T07: Prüfung koordieren – TabMedFrontend

Definition

Der Benutzer meldet sich am Frontend an und ladet eine neue Prüfung auf den Server. Anschliessend lässt er sich den Status der Clients anzeigen und startet die Prüfungsrotation durch das Frontend.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Fabian Mettler

Übersicht

Test	Status
T07.1: Benutzer anmelden	OK
T07.2: Prüfung hochladen	OK
T07.3: Status der Clients anzeigen	OK
T07.4: Prüfungsrotation starten	OK

Bemerkungen

Die einzelnen Tests müssen in der vorgegeben Reihenfolge durchgeführt werden, da sie sich auf die vorherigen Tests beziehen.

T07.1: Benutzer anmelden

T07.1: Benutzer anmelden	
Beschreibung	Der Benutzer meldet sich beim Frontend an.
Erwartetes Resultat	Der Benutzer konnte sich anmelden.
Test Daten	Server: https://sinv56028.edu.hsr.ch:22 Passwort: 123456
Resultat	Benutzer ist angemeldet.
Kommentar	-
Test Datum	17.12.2009

T07.2: Prüfung hochladen

T07.2: Prüfung hochladen	
Beschreibung	Der Benutzer lädt eine neue Prüfung auf den Server.
Erwartetes Resultat	Die Prüfung ist auf dem Server vorhanden und wird bereit gestellt.
Test Daten	Server: https://sinv56028.edu.hsr.ch:22 Passwort: 123456 TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Prüfung ist auf dem Server vorhanden.
Kommentar	-
Test Datum	17.12.2009

T07.3: Status der Clients anzeigen

T07.3: Status der Clients anzeigen	
Beschreibung	Der Benutzer lässt sich den Status der Clients anzeigen.
Erwartetes Resultat	Es werden die aktiven Geräte in der Übersicht angezeigt.
Test Daten	Server: https://sinv56028.edu.hsr.ch:22 Passwort: 123456 Clients: Dell Latitude XT2, Lenovo X200
Resultat	Status der Clients wird angezeigt.
Kommentar	-
Test Datum	17.12.2009

T07.4: Prüfungsrotation starten

T07.4: Prüfungsrotation starten	
Beschreibung	Der Benutzer startet die Prüfungsrotation.
Erwartetes Resultat	Die Prüfungsrotation ist gestartet.
Test Daten	Server: https://sinv56028.edu.hsr.ch:22 Passwort: 123456 TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Prüfungsration ist gestartet.
Kommentar	-
Test Datum	17.12.2009

Nicht funktionale Systemtest

T08: Leistungsanforderung

Definition

Der Experte steht während der Prüfung unter Zeitdruck. Die Applikation muss schnell auf die Eingaben des Experten reagieren und es dürfen keine Wartezeiten entstehen. Die Applikation soll immer flüssig laufen und es dürfen keine grösseren Wartezeiten als 0.4 Sekunden entstehen. Eine Ausnahme bildet eine eventuell etwas längere Wartezeit beim Senden oder Empfangen des Prüfungsformulars. Für die Wartezeiten ist die Performanz auf den zur Verfügung gestellten Client-Devices massgebend.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Rico Suter

Übersicht

Test	Status
T08.1: Wartezeiten bei Eingaben durch den Experten	OK

T08.1: Wartezeiten bei Eingaben durch den Experten

T08.1: Wartezeiten bei Eingaben durch den Experten	
Beschreibung	Es wird gemessen, wie lange die Wartezeiten bei UI-Eingaben ist.
Erwartetes Resultat	Die Wartezeiten sind kleiner als 0.4 Sekunden.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Wartezeit ist unter 0.4 Sekunden.
Kommentar	-
Test Datum	17.12.2009

T09: Verfügbarkeit und Datensicherung

Definition

Während des gesamten Zeitraums einer OSCE-Prüfung muss jede Client/Server Verbindung einwandfrei funktionieren. Es kann aber vorkommen, dass ein Gerät ausfällt z.B. wenn es vom Tisch hinunterfällt. In diesem Fall soll an dieser Station das Gerät ersetzt werden können um die Prüfung fortzusetzen. Natürlich hat auch die Server- Applikation stabil zu laufen, da ansonsten die Prüfungsdurchführung unmöglich wird. Es wird angenommen, dass die Infrastruktur durch das IML bereitgestellt wird und einwandfrei läuft. Jedes Gerät hat also immer Zugriff auf das Netz, entweder durch das LAN oder WLAN.

Die Prüfungsergebnisse müssen jeder Zeit sicher gestellt sein. Eine Kandidatenbewertung soll immer auf dem Server gesichert werden. Bei einem Ausfall des Servers z.B. Stromausfall, dürfen die erfassten Resultate nicht verloren gehen.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Rico Suter

Übersicht

Test	Status
T09.1: Verbindungsunterbruch	OK
T09.2: Ersetzen eines defekten Clients	OK
T09.3: Datensicherung	OK

T09.1: Verbindungsunterbruch

T09.1: Verbindungsunterbruch	
Beschreibung	Die Verbindung vom Client zum Server wird unterbrochen.
Erwartetes Resultat	Der Client arbeitet ohne Verzögerungen weiter und speichert die UI-Eingaben in einer Warteschlange, um diese später an den Server zu schicken.
Test Daten	Server: https://sinv-56028.edu.hsr.ch:22
Resultat	Der Client arbeitet im Offline Modus weiter.
Kommentar	-
Test Datum	17.12.2009

T09.2: Ersetzen eines defekten Clients

T09.2: Ersetzen eines defekten Clients	
Beschreibung	Der Client wird durch ein neues Gerät ersetzt.
Erwartetes Resultat	Der Benutzer arbeitet am neuen Client weiter.
Test Daten	Client 1: Lenovo X200 Client 2: Lenovo T400
Resultat	Benutzer arbeitet ohne grossen Unterbruch mit dem neuen Client weiter.
Kommentar	-
Test Datum	17.12.2009

T09.3: Datensicherung

T09.3: Datensicherung	
Beschreibung	Verbindung zum Server wird unterbrochen.
Erwartetes Resultat	Die Verbindung zum Server wird unterbrochen. Die Daten werden aber weiterhin auf den USB-Stick und auf die Festplatte gespeichert.
Test Daten	Server: https://sinv-56028.edu.hsr.ch:22
Resultat	Daten sind auf dem USB-Stick und auf der Festplatte vorhanden.
Kommentar	-
Test Datum	17.12.2009

T10: Sicherheit

Definition

Während des gesamten Zeitraums einer OSCE-Prüfung muss die Client/Server Verbindung vor unberechtigten Personen und Manipulationen geschützt sein. Sowohl auf die Prüfungsbogen, wie auch auf die ausgefüllten Prüfungen soll auf keinen Fall, vom Netz aus, unbefugter Zugriff stattfinden können. Nach abgeschlossener Prüfung muss garantiert sein, dass die Prüfungsresultate nicht mehr manipuliert werden können und nicht verlogen gehen. Ein Datenverlust darf niemals auftreten.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Rico Suter

Übersicht

Test	Status
T10.1: Verbindungen sind verschlüsselt	OK

T10.1: Verbindungen sind verschlüsselt

T10.1: Verbindungen sind verschlüsselt	
Beschreibung	Die Verbindungen eines Clients zum Server werden mittels Wireshark (Sniffer) überwacht und analysiert.

Erwartetes Resultat	Die Daten im Wireshark sind nur in verschlüsselter Form vorhanden.
Test Daten	Server: https://sinv-56028.edu.hsr.ch:22
Resultat	Datenpakate sind verschlüsselt.
Kommentar	-
Test Datum	17.12.2009

T11: Qualitätsanforderungen

Definition

Das Design des Clients soll Personen zwischen 30 und 70 Jahren mit grossem medizinischem, aber tendenziell geringem Informatik-Fachwissen ansprechen. Dabei ist vor allem wichtig, dass eine optimale Benutzbarkeit der Anwendung unter Zeitdruck entwickelt wird.

Personen

Rolle	Person
Ersteller des Tests	Fabian Mettler
Tester	Rico Suter

Übersicht

Test	Status
T11.1: Verständlichkeit	OK
T11.2: Erlernbarkeit	OK

T11.1: Verständlichkeit

T11.1: Verständlichkeit	
Beschreibung	Es wird geprüft, ob die Bedienung der Applikation intuitiv ist.
Erwartetes Resultat	Die Benutzeroberfläche ist grafisch ansprechend und selbsterklärend.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Die Applikation wirkt grafisch ansprechend und ist intuitiv zu bedienen.
Kommentar	-
Test Datum	17.12.2009

T11.2: Erlernbarkeit

T11.2: Erlernbarkeit	
Beschreibung	Es wird geprüft, ob die Bedienung der Applikation schnell zu erlernen ist.
Erwartetes Resultat	Der Umgang mit der Applikation ist nach einer kurzen Einführung einfach zu erlenen.
Test Daten	TestExam: 4 Kandidaten, 1 Durchführung, 1 Rotation, 1 Station
Resultat	Es wird nach einer längeren Verwendungszeit eine Lernkruve festgestellt.
Kommentar	-
Test Datum	17.12.2009

Usability Tests

Bemerkung

Der Usability-Test wurde vom Institut für medizinische Lehre von Bern durchgeführt. Die Resulate dieses Usability-Tests waren bis zum Abschluss dieses Projekts noch nicht vorhanden.

18 Dezember 09

TabMed – Benutzeranleitung

Rico Suter & Fabian Mettler

Dokumentinformationen

Änderungsgeschichte

Datum	Version	Änderung	Autor
10.12.2009	0.1	Erste Version	fme
17.12.2009	1.0	Ergänzungen	fme & rsu

Inhaltsverzeichnis

TabMedClient

Einleitung

Dieses Dokument beschreibt den Umgang mit der Applikation „TabMedClient“. Die Verwendung der Applikation wird Anhand von drei Szenarien erklärt.

Dongle

Um die Applikation benutzen zu können, bedarf es einen USB-Stick der als Dongle fungiert. Das bedeutet, dass nur mit einem Dongle eine Prüfung ausgefüllt werden kann. Auf dem Dongle muss sich somit eine Konfigurationsdatei und ein Zertifkat des Experten befinden. Dieser Dongle wird von der Prüfungsaufsicht vorbereitet und bereitgestellt.

Benutzerzertifikat

Um ein Benutzerzertifikat mit OpenSSL[1], welches installiert sein muss, zu erstellen gibt man folgende Befehle in eine Kommandozeile ein:

```
# create a file containing key and self-signed certificate
openssl req \
  -x509 -nodes -days 365 \
  -newkey rsa:1024 -keyout mycert.pem -out mycert.pem

# export mycert.pem as PKCS#12 file, certificate.pfx
openssl pkcs12 -export \
  -out certificate.pfx -in mycert.pem \
  -name "My Certificate"
```

Die erstelle Datei “certificate.pfx” muss nun auf den USB-Stick kopiert werden

Expert XML

Zusätzlich zum Zertifikat muss auf dem USB-Stick eine Datei mit dem Namen „expert.xml“ mit folgendem Inhalt vorhanden sein:

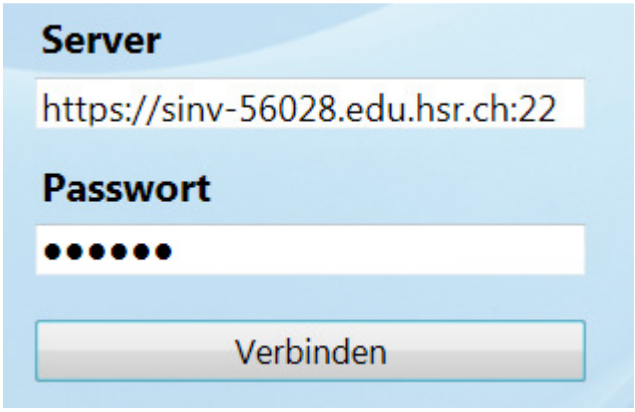
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<expert>
      <title>Dr.</title>
      <firstName>Hans</firstName>
      <lastName>Muster</lastName>
</expert>
```

[1] http://www.openssl.org/

Szenario 1 – Online Modus

Bildschirm 1 – Willkommen

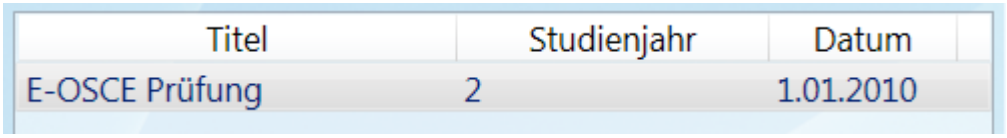
Wenn man die Applikation startet, wird man vom Willkommenbildschirm begrüsst. In dieser Ansicht muss man den gewünschten Server und das Benutzerpasswort eingeben:



Anschliessend drückt man die Schaltfläche „Verbinden“ um die Verbindung mit dem Server aufzubauen.

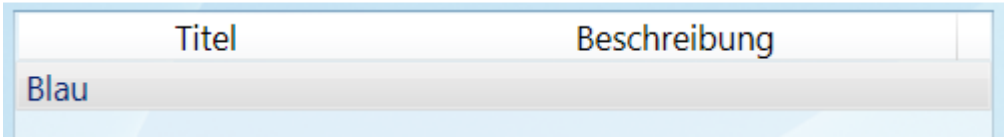
Bildschirm 2 – Prüfung auswählen

Nachdem man sich mit dem Server verbunden hat, werden alle Prüfungen angezeigt, die auf dem Server vorhanden sind:



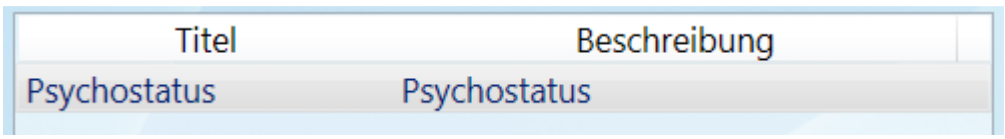
Bildschirm 3 – Durchführung auswählen

Da es von einer Prüfung mehrere Durchführungen geben kann, wählt man auf diesem Bildschirm die gewünschte Durchführung aus.



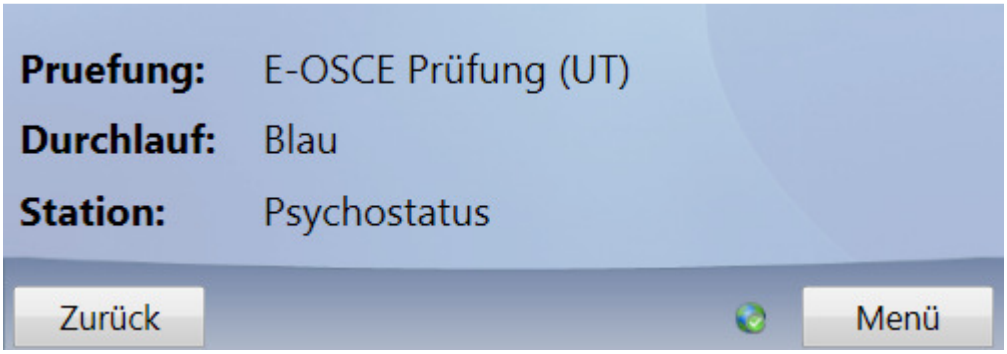
Bildschirm 4 – Station auswählen

Eine OSCE-Prüfung setzt sich aus mehreren Stationen zusammen. In diesem Bildschirm erhält man somit eine Übersicht über alle Stationen und kann die gewünschte Station auswählen.



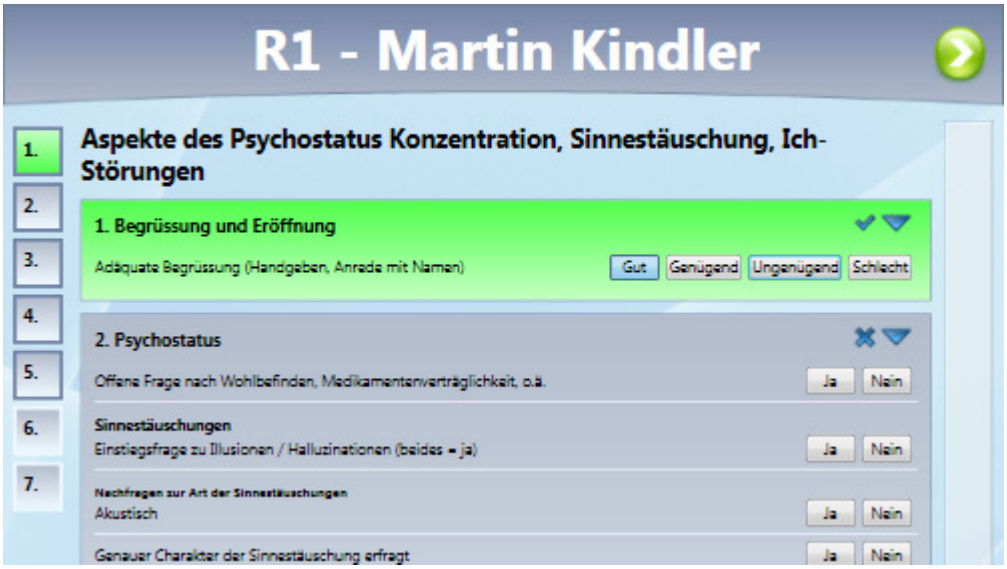
Bildschirm 5 – Warte auf Rotationsstart…

Wenn man die gewünschte Station ausgewählt hat erscheint der Warte-Bildschirm. Dieser Bildschirm wird solange angezeigt, bis die Prüfung auf dem Server durch den Administrator oder Prüfungsleiter gestartet wurde. Zusätzlich werden noch diverse Angaben angezeigt: Prüfungstitel, Durchführung, Station und den Netzwerkstatus. Um eine Prüfung manuell zu starten oder das Programm zu beenden steht ein Menü mit diesen Befehlen zur Verfügung.

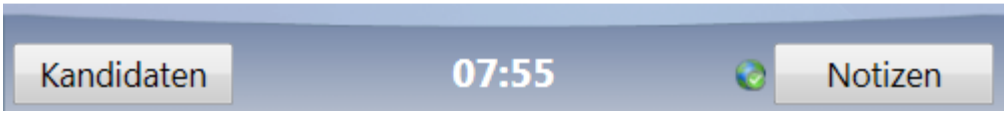


Bildschirm 6 – Prüfungsformular des Kandidaten

Nachdem die Prüfung gestartet wurde, wird das Prüfungsformular des ersten Kandidaten der gestarteten Rotation angezeigt. In dieser Ansicht kann man den Kandidaten nach den vorgegeben Kritieren bewerten.



Unterhalb des Prüfungsbogen wird in der Mitte eine Uhr angezeigt, welche die Zeitdauer für den aktuellen Kandidaten anzeigt.



Es gibt drei „Zeitzustände“:

1. Die Uhr zeigt die verfügbare Prüfungszeit des Kandidaten in weiss an, danach zeigt sie
2. die Reservezeit für die Bewertung des Kandidaten in rot an.

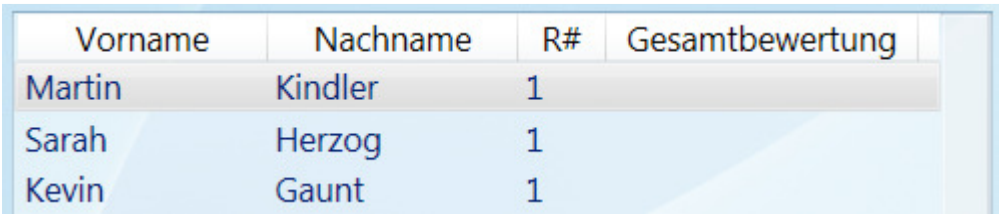
3. Nach Ablauf der Reservezeit wird der Hintergrund rot, dann sollte so schnell wie möglich zum Nächsten Kandidaten gewächselt werden.

Bildschirm 7 – Notizen

Auf diesem Bildschirm kann man diverse Notizen für eine detailiertere Bewertung machen. Es stehen verschiedene Modis zur Auswahl: Schreiben, Radieren, Löschen und Auswählen.

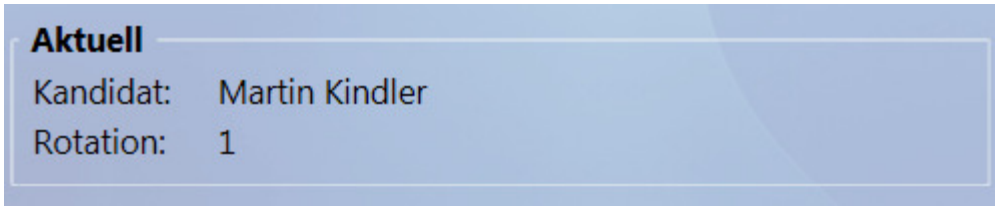
Bildschirm 8 – Kandidatenliste

Wenn man im Prüfungsformular auf den Knopf „Kandidaten“ klickt, gelangt man zur Kandidatenliste. Diese Liste zeigt alle Kandidaten einer Prüfung an, geordnet nach Durchführungsreihenfolge und Rotationsnummer.



Vorname	Nachname	R#	Gesamtbewertung
Martin	Kindler	1	
Sarah	Herzog	1	
Kevin	Gaunt	1	

Um zu wissen welcher Kandidat an der Reihe und welche Rotation aktuell ist, werden diese Informationen unterhalb der Kandidatenliste angezeigt.



TODO: swappen

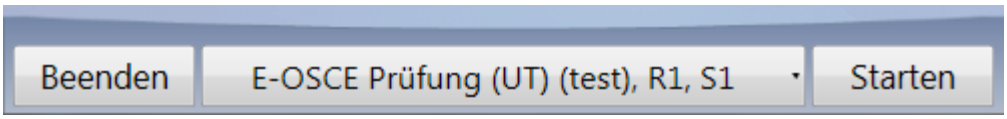
Szenario 2 – Offline Modus

Voraussetzung

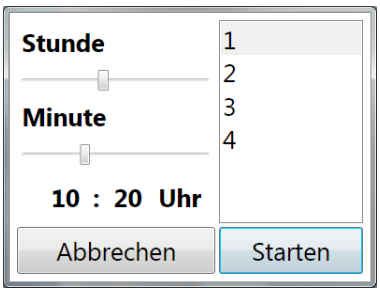
Um den TabMedClient im Offline Modus benutzen zu können, muss die gewünschte Prüfung bereits heruntergeladen sein. Das Herunterladen einen Prüfung erfolgt, wenn man nach dem Verbinden mit dem Server die gewünschte Prüfung ausgewählt hat und den Warte-Bildschirm sieht.

Bildschirm 1 – Willkommen

Bereits heruntergeladene Prüfungen können im Willkommen-Bildschirm durch ein Auswahlmenü, welches sich am unteren Bildschirmrand befindet, gestartet werden. Um die gewünschte Prüfung zu starten wählt man die Prüfung aus und klickt auf „Starten“:



Anschliessend wird ein Dialogfenster geöffnet bei dem man zusätzliche Einstellungen wie Startzeit und Rotationsnummer definieren muss:



Nachdem man die Startzeit definiert und die gewünschte Rotation ausgewählt hat, klickt man auf „Starten“ und der Prüfungsbogen des aktuellen Kandidaten wird angezeigt. Das weitere Vorgehen entspricht nun dem Szenario 1 ab Bildschirm 6.

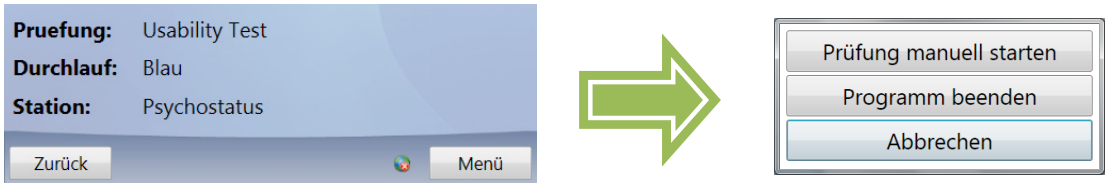
Szenario 3 – Online / Offline Modus

Voraussetzung

Der TabmedClient hat sich mit dem Server verbunden, die gewünschte Prüfung wurde ausgewählt und es wird auf den Rotationsstart gewartet.

Bildschirm 1 – Warte auf Rotationsstart…

Es kann durchaus vorkommen, dass der Server unerwartet ausfällt. Um die Prüfung dennoch durchzuführen, kann man die Prüfung manuell starten. Um die Prüfung manuell zu starten klickt man auf „Menü“ und anschliessend auf „Prüfung manuell starten“:



Wie bereits aus Szenario 2 (Bildschirm 1) bekannt, wählt man nun die gewünschte Zeitdauer und Rotation aus und startet die Prüfung.

TabMedServer

Einleitung

Dieses Dokument beschreibt die Konfiguration und das Betreiben des TabMedServer.

Voraussetzungen

Um den TabMedServer zu starten benötigt man mindestens Java ab Version 1.6.

Konfiguration

Keystore

Die Keystore-Datei enthält das Server-Zertifikat, welches für die SSL Verbindung benötigt wird. Um für den Server ein Zertifikat und eine Keystore-Datei zu erstellen, gibt man folgenden Befehl in eine Konsole ein:

```
keytool -genkey -keystore keystore -alias serverKey –dname
“CN=tabmedserver, OU=IML, O=UNIBE, L=Bern, ST=Bern, C=CH”
```

Keyword	Beschreibung
CN (Common name)	Der allgemeine Servername des Servers. Für einen SSL Web Server, wie der TabMedServer, muss der Common Name der volle qualifizierte Domainnamen sein, z.B.: „www.example.com“
OU (Organization unit name)	Name der Organisationseinheit.
O (Organization)	Name der Organisation.
L (Locality name)	Name der Lokalität, z.B. Stadt
ST (State or province name)	Voller Name des Staats oder der Provinz
C (Country name)	Ländercode nach ISO-Standard, z.B. CH für Schweiz

Config.xml

Die Konfiguration wird in der XML-Datei „config.xml“ vorgenommen. Ist keine Datei vorhanden werden die in der Tabelle angebenen Standardwerte verwendet.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<config>
	<port>22</port>

	<minThreads>5</minThreads>
	<maxThreads>50</maxThreads>

	<waitTimeout>10<waitTimeout/>
	<sessionTimeout>30</sessionTimeout>

	<adminPassword>123456<adminPassword/>
	<userPassword>123456</userPassword>
	<keystorePassword>secret</keystorePassword>
</config>
```

In der folgenden Tabelle sind alle möglichen Felder zu finden. Werden einige Felder nicht definiert, wird automatisch der Standardwert aus der Tabelle verwendet.

Name	Beschreibung	Standard
port	Server, der im Eingabefeld angezeigt werden soll. Dieser Wert wird automatisch in der Datei gesetzt, wenn ein anderer Server eingegeben wird.	9998
minThreads	Minimale Anzahl an Threads im Thread-Pool des Servers.	5
maxThreads	Maximale Anzahl an Threads im Thread-Pool des Servers. Gute Berechnungsgrundlage: Sollte doppelt so gross sein wie die Anzahl Clients, die den Server gleichzeitig verwenden.	50
waitTimeout	Zeit, nach der eine Antwort an den Client gesendet wird, ob eine Rotation gestartet wurde oder nicht (in Sekunden).	10
sessionTimeout	Zeit, nach der ein Client nicht mehr aktiv ist (in Sekunden).	30
adminPassword	Passwort um neue Prüfungen hochzuladen und und den Server zu aktualisieren. Dieses Passwort muss im TabMedFrontend eingegeben werden.	123456
userPassword	Passwort um Formulare herunterzuladen und ausgefüllte hochzuladen. Dieses Passwort muss im TabMedClient angegeben werden.	123456
keystorePassword	Keystore-Passwort, damit der Server das Zertifikat laden kann.	secret

Prüfungsverzeichnis

Die vorbereiteten Prüfungen werden in das Verzeichnis „exams“ abgelegt. Die Prüfungen können entweder manuell in den Ordner kopiert oder mit der Applikation „TabMedFrontend“ neue Prüfungen zum gewünschten Server hochgeladen werden. Das Erstellen der Prüfungen wird vom IML übernommen.

Starten des Server

Um den Server auf einem Windows System zu starten, führt man die Datei „start.bat“ aus. Auf einem Unix System (Linux, Mac OSX, BSD, usw.) führt man die Datei „start.sh“ aus.

18 Dezember | 09

TabMed – Erfahrungsbericht

Rico Suter & Fabian Mettler

Dokumentinformationen

Änderungsgeschichte

Datum	Version	Änderung	Autor
14.12.2009	0.1	Erste Version	rsu
17.12.2009	1.0	Finale Version	rsu & fme

Inhaltsverzeichnis

Persönliche Erfahrungsberichte

Fabian Mettler

Ich konnte bei der Studienarbeit einiges neues lernen, besonders im Bereich Requirement Engineering. Die Arbeit mit dem Industriepartner und dem IFS hat mir sehr gut gezeigt, wie wichtig es ist, von Anfang an bis zum Ende die Anforderungen zu spezifizieren. Denn zum Start hatten beide Parteien teilweise unterschiedliche Ansichten über die Anforderungen. Im Verlauf der Arbeit konnten wir, durch eine gute Zusammenarbeit mit dem IML und dem IFS, die Anforderungen nach den Bedürfnissen der Experten erfüllen.

Interessant war auch die Verwendung von Scrum als Entwicklungsprozess. Wir haben Scrum ein wenig an unsere Bedürfnissen angepasst. Daraus kristallierte sich schnell, dass es ein wirklich starker agiler Prozess ist und ich Scrum für nächste Projekte verwenden will, sofern möglich.

Die verwendeten Technologien, wie z.B. Windows Presentation Foundation, erlaubten uns eine rapide Entwicklung der Software. Wir konnten Änderungen effizient implementieren und nach unseren Wünschen in kürzester Zeit anpassen.

Mit der Betreuung durch Kevin Gaunt und Markus Stolze war ich persönlich zufrieden. Allerdings hätte ich teilweise mehr Feedback zum gesamten Projekt, als nur zu Teilbereichen, erwartet.

Rico Suter

Die Studienarbeit hatte für mich ein paar Aspekte, bei denen ich noch einige Dinge dazugelernt habe. Ein grosser Punkt ist dabei WPF. Wir haben uns dabei mit ein paar Problemen herumgeschlagen und so habe ich recht viel in diesem Bereich gelernt.

Ich fand es interessant, mit einem Industriepartner zusammenzuarbeiten, da dies das gesamte Projekt viel realistischer gemacht hat und wir uns daher viel mehr mit der wirklichen Benutzergruppe auseinandersetzen mussten.

Unser angepasstes Scrum hat gut funktioniert; wir lagen fast immer gut im Zeitplan und haben meist die richtige Anzahl von Stunden pro Woche gearbeitet. Da wir allerdings nur zwei Personen waren, ist dies auch nicht weiter erstaunlich, da Projektmanagement auf ein minimum reduziert werden kann.

Unsere Betreuung durch Kevin Gaunt und Markus Stolze war gut, ich hätte allerdings manchmal ein wenig mehr Feedback erwartet.

Technische Lessons Learned

GUI mit dem Windows Presentation Framework

In dieser Studienarbeit konnten wir wichtige neue Erkenntnisse in der WPF-Programmierung erlernen. Besonders Rico, der bisher nur mit Windows.Forms gearbeitet hat, konnte so einige Erfahrungen sammeln: Hauptsächlich im Umgang mit XAML-Code, Expression Blend und dem MV-V-M-Pattern.

Zertifikate mit .NET

Zu Beginn des Projekts, hatten wir ein wenig Respekt vor der Tatsache, dass wir XML-Daten signieren mussten. Im Laufe des Projekts stellte sich heraus, dass diese Problematik zwar nicht ganz einfach ist, aber doch lösbar. Das .NET-Framework stellt nämlich sehr viele Sicherheits-Funktionen zur Verfügung.

Grizzly-WebServer und SSL

Eine Anforderung des Projekts war, dass der Server einfach zu installieren ist. Daher entschlossen wir uns für den Grizzly-WebServer, da dieser Server auch als Standalone-Applikation verwendet werden kann, nicht wie normale Application-Server.

Zu Beginn entwickelten wir den Server ohne Verschlüsselung, da wir dachten, SSL würde von Grizzly gar nicht unterstützt. Als wir dann gegen Ende der Studienarbeit diese Anforderung auch noch erfüllen wollten, recherchierten wir noch tiefer im Internet. Es stellte sich heraus, dass es mit den richtigen, aber relativ wenigen, Codezeilen doch möglich ist.

REST-basierte HTTP-Webservices

Am Anfang wollten wir eine SOAP-Schnittstelle implementieren, da wir davon ausgingen, dass es so einfacher würde, von anderen Geräten darauf zuzugreifen. Da es allerdings nicht einfach ist, mit dem iPhone SOAP-Services zu konsumieren, entschieden wir uns für die einfacheren REST-basierten HTTP-Webservices.

In diesem Bereich haben wir gesehen, wie man REST-Webservices mit dem JAX-WS-Framework in Java implementiert. Da dies mit ein paar einfachen Annotationen funktioniert, hat man die gegebene API schnell begriffen.

18 Dezember 09

TabMed – Glossar

Rico Suter & Fabian Mettler

Dokumentinformationen

Änderungsgeschichte

Datum	Version	Änderung	Autor
16.12.2009	1.0	Vom Wiki ins Dokument	rsu

Inhaltsverzeichnis

Glossar

Begriff	Beschreibung
CREALOGIX	Partnerfirma, http://www.crealogix.com/
IFS	Institut für Software, http://ifs.hsr.ch
IML	Institut für Medizinische Lehre, http://www.iml.unibe.ch/
MoMEA	Mobile Medizinische Evaluation und Assessment
OSCE	Observed Structured Clinical Examinations
PBL	Product Backlog
PO	Product Owner
REST	Representational State Transfer
SBL	Sprint Backlog
SM	Scrum Master
SOAP	Simple Object Access Protocol
SPSS	Statistikprogramm zur Auswertung der Prüfungsergebnisse
WPF	Windows Presentation Framework

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Studienarbeit: TabMed

Tablet PC Client-Server Applikation für medizinische Evaluationen

Erklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Ort, Datum: Rapperswil, 17. 12. 2009

Name, Unterschrift:

Fabian Mehler

F Mehler

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Studienarbeit: TabMed

Tablet PC Client-Server Applikation für medizinische Evaluationen

Erklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Ort, Datum: Rapperswil, 17.12.2009

Name, Unterschrift:

Rico Suter

Rico Suter

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Vereinbarung

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit „Tablet PC Client-Server Applikation für medizinische Evaluationen" von Rico Suter und Fabian Mettler unter der Betreuung von Prof. Dr. Markus Stolze geregelt.

2. Urheberrecht

Die Urheberrechte stehen der Studentin / dem Student zu.

3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von der Studentin / dem Student, von der HSR sowie vom Institut für Medizinsche Lehre der Universität Bern (IML) nach Abschluss der Arbeit verwendet und weiter entwickelt werden. Sowohl die HSR wie auch das IML dürfen die Rechte für Nutzung und Weiterentwicklung auch an Dritte weitergeben.

Rapperswil, den 17.12.09 .....
Die Studentin/der Student

Rapperswil, den 17. Dez 2009 .....
Der Betreuer / die Betreuerin der Studienarbeit

Rapperswil, den..... .....
Der Studiengangleiter / die Studiengangleiterin