HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

SEVITEC
SOFTWARE-LÖSUNGEN NACH MASS

Dynamic Document Creation in Windows Phone 7

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2011

Autor :	**Ba Mohamedou Moustapha**
Verantwortlicher :	**Prof. Hansjörg Huser**
Betreuer:	**Prof. Hansjörg Huser**
Industriepartner:	**Sevitec AG, Eschlikon TG**

1. Management Summary

2. Technischer Bericht

3. Projekt und Zeitplanung/Zeiterfassung

4. Anforderungen

5. Architektur und Design

6. Erfahrungsbericht

Management Summary

EMAIL CLIENT DYNAMIC DOCUMENT CREATION

Studienarbeit :	**Integration eines Dynamic Document Creation System in einen Windows Phone 7 Mail Client**
Autor :	**Ba Mohamedou Moustapha**
Verantwortlicher :	**Prof. Hansjörg Huser**
Betreuer:	**Prof. Hansjörg Huser**
	Herr Simon Baer
Industriepartner:	**Sevitec AG, Eschlikon TG**

Ausgangslage

Der Firmenpartner ist Hersteller eines Zusatzsoftware-Produkts zu Microsoft Office (www.idfix.ch), womit Kunden in allen Branchen als Alternative zu selbstprogrammierten Office-Templates auf effiziente Art Geschäftsdokumente und Emails in perfektem Corporate Identity erzeugen können. Dazu gehört auch das Erstellen von formatierten HTML-Emails in Microsoft Outlook, mit Profil-, Sprach- und Zielgruppen-abhängigen Signaturen. Im Rahmen dieser Aufgabe soll diese Funktionalität neu auch auf Smartphone unter Windows Mobile 7.0

verfügbar gemacht werden.

Vorgehen/Technologien

Für die Lösung der Aufgabe wurden folgende Schritte durchgeführt:

- Umfassende Analyse der Anforderungen
- Analyse der Machbarkeit
- Konzeptioneller Design und Architektur

Als wichtiger Teil der Aufgabe wurde die Integration eines HTML-Editors in Windows Phone 7, um Email in HTML-Format versenden zu können.

Folgende Technologien und Tools wurden eingesetzt:

- Visual Studio 2010
- C#, .Net 4.o
- Silverlight for Windows Phone
- Model v View ViewModel
- Windows Communication Foundation(WCF Service)

Resultat

Die Entwickelte Windows Phone 7 App macht folgende Funktionalitäten verfügbar:

- Verfügbare Documenttyps aus der Datenbank laden und im WP7 anzeigen
- Wenn der Nutzer einen Documenttyp auswählt, werden automatisch die dazugehörigen Parameter aus der Datenbank angezeigt
- Parameter in zwei Gruppen unterteilen :
  - CheckboxParameter

- DropDownParameter
- Parameterwerte für die DropDownParameter anzeigen, wenn ein DropDownParameter ausgewählt wird.
- Document anhand ID des gewählten Documenttyps und die gewählte Sprache aus der Datenbank laden und im Editor anzeigen.
- HMTL-Document per Email versenden.

Technischer Bericht

EMAIL CLIENT DYNAMIC DOCUMENT CREATION

Studienarbeit :	**Integration eines Dynamic Document Creation System in einen Windows Phone 7 Mail Client**
Autor :	**Ba Mohamedou Moustapha**
Verantwortlicher :	**Prof. Hansjörg Huser**
Betreuer:	**Prof. Hansjörg Huser**
	Herr Simon Baer
Industriepartner:	**Sevitec AG, Eschlikon TG**

Dokumentinformationen

Änderungsgeschichte

Datum	*Version*	*Änderung*	*Autor*
30.05.2011	0.1	Dokument erstellt	mba
10.06. 2011	1.0	End Version für Abgabe	mba

Inhalt

Einführung

Umfeld

Unternehmen und Produkte

Die Firma Sevitec AG in Eschlikon (Kanton Thurgau) stellt Software-Produkte rep. Individuelle Software-Produkte nach Mass her. Für die anspruchsvollen Kunden wird immer erstklassige Individualsoftware für alle Microsoft Plattformen entwickelt. Sevitec zählt heute rund zwanzig hoch qualifizierte Mitarbeitende.

Kompetenzen der Firma Sevitec

- Microsoft Windows Vista, XP, 2000, NT, CE auf Basis objektorientierter Programmierung mit den Sprachen C#, Visual C++, Visual Basic und XML
- Internet, Extranet oder Intranet mit Microsoft-Technologien wie C#, Java, SOAP, XML, COM+, Visual C++, Visual Basic, Java-Script, VB-Script und SQL Server
- SQL Datenbanken Microsoft und Oracle
- Microsoft Office SharePoint Server / WSS / Webparts
- Individuelle, massgeschneiderte Software
- Garantierte Pauschalpreise
- Günstige Stundensätze
- Unabhängigkeit durch Abgabe des Quellcodes
- Umfassendes und bewährtes Know-how
- Fachmännische Begleitung von der Beratung über die Integration bis zum Betrieb
- Rationelles, zuverlässiges und termintreues Projektmanagement
- Verwendung modernster Microsoft-Technologien
- User Interface Design, Web-Design und Usability

Source: http://www.sevitec.ch/sites/xr/aspx/rx/home.htm#

Produkte

- **Microsoft SharePoint Server 2010 (**Microsoft SharePoint Server 2010)

SharePoint Server 2010 stellt Funktionen bereit, mit denen unverzichtbare Anforderungen erfüllt werden können. Beispielsweise werden Inhalte und Geschäftsprozesse verwaltet, das Suchen und Freigeben von Informationen über Grenzen hinweg vereinfacht und fundierter Entscheidungen ermöglicht. SharePoint Server 2010 unterstützt alle Intranets, Extranets und Webanwendungen im gesamten Unternehmen mit Hilfe einer einzigen integrierten Plattform. Das mühsame Arbeiten mit getrennten Systemen gehört somit der Vergangenheit an.

- **OFFIXX** (OFFIXX: Ein Mausklick - und Word setzt Ihr Corporate Design um)

  Offixx ist ein Microsoft Word-Zusatzprogramm, welches jedes Dokument automatisch Corporate Design-konform formatieren kann.

- **Content Management System SEVIWARE.NET(**Content Management System SEVIWARE.NET)
  SEVIWARE.NET ermöglicht eine bessere Handhabung von tagesaktueller Informationsbereitstellung im Unternehmen.

- **PREMIUM HOSTING für WebSolution(**PREMIUM HOSTING für WebSolutions)
  Die Wartung firmenspezifischer WebSolution sowie Internet Auftritt mit SEVIWARE.NET wird mit PREMIUM HOSTING für WebSolution gewährleistet.

Source : http://www.sevitec.ch/sites/xr/aspx/m.4_1/rx/standard.htm

Windows Phone 7 Email Client

Diese Studienarbeit befasst sich mit dem Versenden von Emails in HTML-Format in einem Windows Phone 7 (WP7). Die HTML-Dokumente sollen

automatisch aus der Datenbank in einen HTML-Editor im WP7 Client geladen, bearbeitet und anschließend per Email verschickt werden können.

Inbetriebnahme

Zum Zweck der Entwicklung und des Testens können nach erfolgreicher Registrierung eine Anzahl von Geräten(3 pro Entwickler, für Studenten 1 pro Account) freigeschaltet werden, auf die dann ein direktes Deployment vom Desktop aus erfolgen kann. Es können über USB

nur maximal 10 (Studenten 3) Applikationen(Apps) auf ein freigeschaltetes Gerät geladen werden. Die fertig entwickelte Apps werden bei Microsoft zur Zertifizierung eingereicht, wobei diese nach technischen, rechtlichen und inhaltlichen Kriterien auf die Einhaltung der veröffentlichen Markplace-Richtlinien geprüft werden (siehe veröffentlichten Marketplace-Richtlinien). Fertige Produkte können dann nach erfolgreichem Test im lokalen Katalog oder international veröffentlicht werden und stehen dann für jeden Windows Phone 7-Nutzer zum Erwerb zur Verfügung.

Windows Phone 7-Anwendungen wie auch jeglicher Client-seitiger Code können mit ausreichendem Aufwand einem Reverse Engineering unterzogen werden. Um dies zu verhindern, trifft Microsoft geeignete Vorkehrungen. Aber es wird Entwicklern geraten im Zweifelsfall wichtige Routinen als Webdienste auszulagern und somit die eigene sogenannte „Intellectual Property“ besser zu schützen. (z.B Tools wie . RunTime Intelligence for Windows Phone).

Problemstellungen bei der Inbetriebnahme

Abgrenzung

Ziel dieser Studienarbeit ist die Entwicklung einer Windows Phone 7 Applikation, welche ein dynamisches Erstellen von Dokumenten unterstützen kann.

Herausforderungen

Die Herausforderungen dieser Studienarbeit finden sich in folgenden Punkten:

- Umfassende Analyse der Anforderungen und der zur Verfügung stehenden Technologien
- Konzeptioneller Design, Architektur und Schnittstellenspezifikation der Lösung.

- Umfassende Machbarkeitsanalyse
- Realisierung eines Funktions-Prototyps bestehend aus verschiedenen Teilsystemen
  (Serverseitig, Client-seitig und ev. Webservice)

Aufgabenstellung

Es soll eine Client/Server –Anwendung für Windows Phone 7(WP7) entwickelt werden, welche mit einem zu realisierendem Webservice in (C#)Kommunizieren kann, der auf eine bestehende Datenbank zugreifen kann und Daten für den WP7 Client dynamisch zu Verfügung stellen kann. Die WP7 Applikation soll mindestens folgende Funktionalitäten verfügbar machen:

- Dokumenttyps laden und anzeigen und auswählbare machen
- Parameter zu dem gewählten Dokumenttyp anzeigen und auswählbar machen.
- Parameter Values zu den gewählten Parametern anzeigen und auswählbar machen.
- Dokument in HTML-Editor laden und darstellen können.
- Dokument per Email verschicken.

Die Wichtigsten Resultate der Studienarbeit stellen folgende Artefakte dar:

- Anforderungen mit Use Cases
- Realisierung eines WebServices für die Bereitstellung der Daten.

- Client/Server –Funktionalität mit Datenzugriff per Entity Framework und DataService.
- Machbarkeitsnachweise für einen WP7 Mail Client mit HTML-Editor.
- (optional) Vergleich von Webtechnologien : REST/SOAP/ODATA/WCF/Thrift etc..

Analyse der Machbarkeit

Bestehender WP7 Email Client erweitern

Diese Variante wäre die Beste Möglichkeit für die Realisierung des Projektes. Für die Entwicklung könnte viel Zeit erspart werden. Aber es gibt leider keine Möglichkeit für uns einen WP7 Email Client zu finden. Da WP7 Email Client in jedem WP7-Gerät integriert ist, wird jeder Email-Client als Bestendteil des Betriebssystems betrachtet. Daher ist die Betriebssystemebene allein den Gerätehersteller vorbehalten. Dies ist allerdings für Softwareentwickler also nicht zugänglich.

Diese Variante ist leider auszuschliessen.

Eigenen WP7 Email Client Entwickeln

Diese Variante wäre viel aufwendiger, aber die Möglichkeit einen eigenen WP7 Email Client zu entwickeln besteht schon. Grundsätzlich werden Anwendungen für WP7 in einem Framework entwickelt, welches auf .NET basiert und Stabilität, Performance und Sicherheit garantiert. In diesem Framework steht eine UI-Technologie zur Verfügung: Silverlight.

„Silverlight for Windows Phone 7“ besonders stellt ein vollwertiges Anwendungsframwork für eigenständige Apps dar. Es basiert auf Silverlight 3 mit folgenden Erweiterungen:

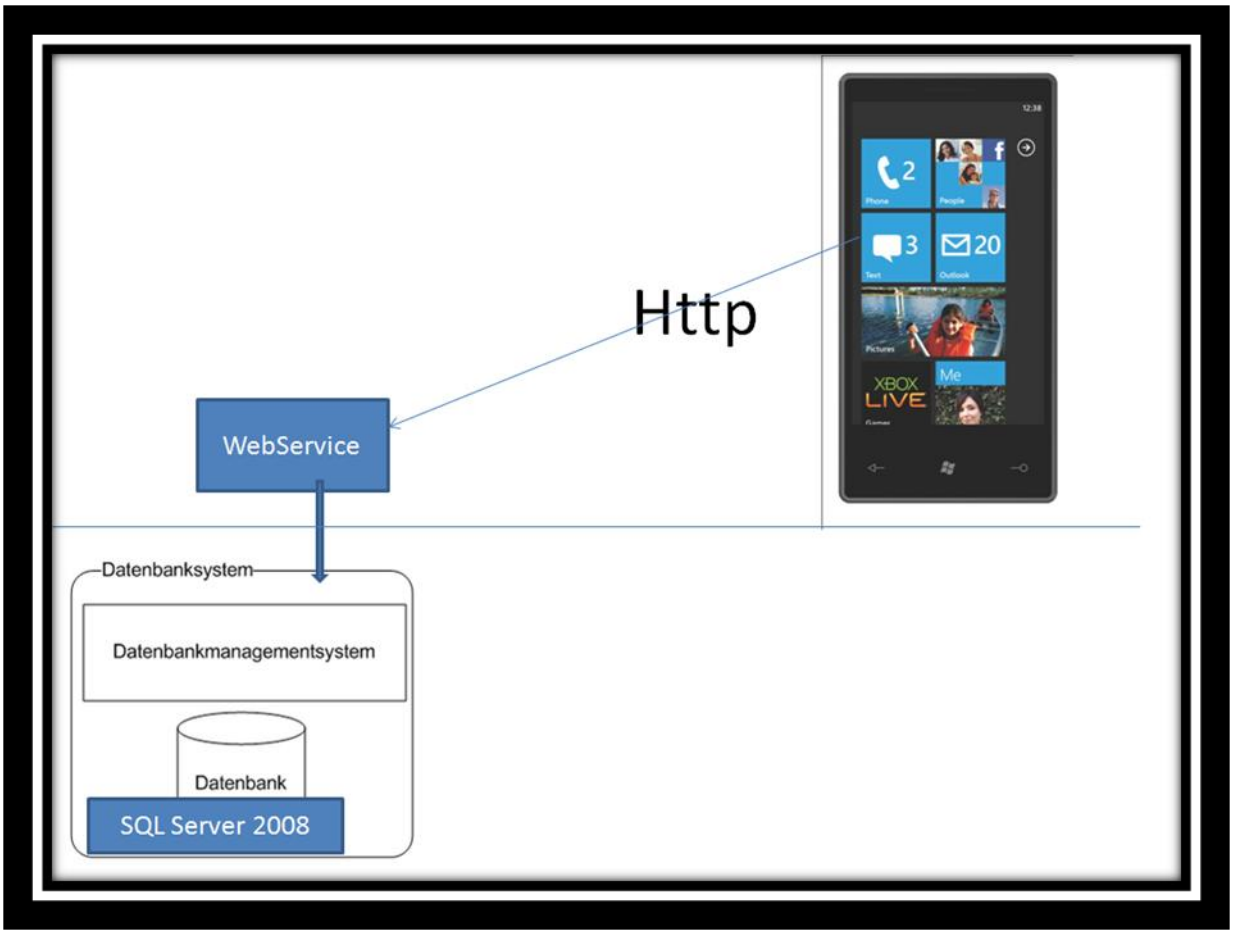
- Höchste Performance
- Input-Integration
- HW-, Medien- & Sensor- Integration
- Adaption für WP7-App-Modell

- Weniger enge Sandbox
- Web ähnliche UI Entwicklung

Um eine eigene Email-App zu entwickeln stellt Microsoft eine API für die Sensorenansteuerung Verfügung. Unter Windows Phone 7 werden alle Sensoren über sehr mächtige Klassen gekapselt, die auf einfache Art und Weise über standardmäßigen C#-Code angesprochen werden können. Der teilweise sehr umfangreiche und komplexe Umgang mit API-Calls, wie bei anderen Plattformen, entfällt daher komplett. Zusätzlich ermöglicht Microsoft unter Windows Phone 7 dem Entwickler aber auch noch den Zugriff auf eine Vielzahl von weiteren Funktionen der Windows Phone 7 Hubs. Das Mittel zur Nutzung der Funktionen aus dem Hub sind die sogenannten Launcher- und Chooser-Tasks. Ein Launcher-Task dient dabei dazu eine bestimmte Funktion auszulösen (z.B. einen Telefonanruf starten), ein Chooser-Task liefert zusätzlich ein Ergebnis zurück, z.B. ein Bild aus der Gallery auswählen. Die Klasse EmailComposeTask ermöglicht das Senden von Email mit dem eigenen Email-Client.

Systemübersicht

Physikalische Systemarchitektur



Anforderungen

Funktionale Anforderungen

Kurze Auflistung der Funktionalen Anforderungen:

- Alle verfügbaren Dokumenttypen aus der Datenbank anzeigen lassen
- Pro. Dokumenttyp dazugehörigen Parameter aus der Datenbank anzeigen lassen
- Parameter in zwei Gruppen teilen : CheckBoxParameter und DropDownParameter

- Pro. DropDownParameter dazugehörigen Werte aus der Datenbank anzeigen lassen.
- HTML-Dokument aus der Datenbank im Editor anzeigen lassen
- HTML-Dokument bearbeiten und per Email verschicken
- Konfigurationsseite für Email und WebService erstellen
- Benutzer-Login sollen einmalig eingegeben werden

Anwendungsfälle (Use Cases)

Die folgenden Anwendungsfälle wurden erfolgreich implementiert und getestet:

- Documenttyp wählen
- Parameter wählen
- Parameter-Werte wählen
- HTML-Document erstellen
- HTML-Document im Editor laden
- HTML-Document per Email senden.

Nichtfunktionale Anforderungen

Allgemeine Überlegungen und daraus resultierende Anforderungen:

- Gute Benutzerführung und eine möglichst einfache Bedienung.
- Jede Benutzeroberfläche ist schlicht und zeigt nur das Wesentliche

Bedienbarkeit

Die Bedienung soll sich soweit möglich an der in Entwicklung stehenden Konfigurationswebseite orientieren.

Trotzdem sollen die Designprinzipien für Windows Phone 7 Applikationen berücksichtig werden.

Datenschutz

Die Windows Phone 7 Applikationen soll durch eine Benutzername/Passwort-Kombination geschützt werden. Die

Verbindung zwischen dem Smartphone und dem Gateway soll vor *Sniffing*-Attacken geschützt werden.

Datensicherheit

Datensicherheit oder Backupstrategien sind für dieses Projekt nicht relevant. Alle Informationen kommen aus der Datenbank und unterliegen strenge Zugriffsrechte.

Wartbarkeit

Die Software soll einfach erweiterbar sein und das Hinzufügen von neuen Funktionen erlauben.

Architektur und Design

Webservice-Schnittstelle

Folgende Anforderungen stellen sich an eine Webservice Architektur für Smartphone(mobile Geräte):

- Plattformunabhängig: Applikationen für weitere mobile Plattformen wie Android oder iPhone sollen
  entwickelt werden können.
- Schlank und einfach: Auf mobilen Geräten ist es wichtig die eingeschränkten Ressourcen nicht zu
  verschwenden.
- Keine Bibliotheken notwendig: Sobald Bibliotheken benötigt werden, müssten diese für jede mobile
  Plattform zur Verfügung stehen.

REST erfüllt die erwähnten Anforderungen. REST ist nicht direkt eine Softwarearchitektur, es ist mehr ein

Architekturstil. Wenn REST im Kontext von HTTP verwendet wird, beschreibt es einige Basisprinzipien wie

Webservices aussehen sollen und wie Client und Server kommunizieren, basierend auf den Konzepten des „World

Wide Web“ (WWW). Heute wird REST von vielen grossen Webapplikationen wie z.B. Facebook, Flickr usw. für ihre

Webservice-Schnittstelle nach aussen benutzt.

Der Unterschied zwischen REST und anderen Webservice Architekturen wie z.B. SOAP liegt darin, dass REST mit dem

bestehenden Vokabular und Konzepten des „World Wide Web“, wie z.B. URIs, Fehler-Codes (404, 500, …),

Methoden (GET, POST, PUT, …) arbeitet, während in SOAP für jede Applikation eigene Konzepte (Methoden)

definiert werden müssen. REST ist ein Architekturstil zum Erstellen von Client / Server-Anwendungen. SOAP ist eine Protokollspezifikation für den Datenaustausch zwischen zwei Endpunkten.Für die Abfrage einer Benutzerliste würde zum Beispiel in SOAP eine neue Methode

‚*getAllUsers()*‘ definiert werden, während in REST für dieselbe Aufgabe das existierende HTTP URI Konzept benutzen wird *‚GETUsers/All‘*.

Damit ein Client fähig ist REST-Webservice zu benutzen, muss er nur in der Lage sein über HTTP zu kommunizieren. Weil eine HTTP-Kommunikation auf allen Mobile- und Desktop-Plattformen möglich ist, wird mit REST auf einfache Weise ein plattformunabhängiger Webservice bereitgestellt. Die Entwicklung eines REST-Webservice auf Server Seite gestaltet sich z.B. mit .NET sehr einfach. Die WCFBibliothek in .NET bietet die Möglichkeit REST-Webservices bereitzustellen. Auf Client Seite ist keine Bibliothek vorhanden, um REST-Webservice zu konsumieren. Dass ist aber nicht weiter ein Problem, weil gar keine Bibliothek benötigt wird. Die Basisfunktionen für eine HTTP-Kommunikation reichen völlig aus und existieren auf jeder Plattform.

Vergleichen von REST mit dem remote Procedure Call (RPC)-Stil des Client / Server-Anwendungen erstellen werden genauer. RPC ist eine Formatvorlage (anstelle eines Protokolls, ist SOAP) Erstellen von Client / Server-Anwendungen in der ein Proxy (im Allgemeinen aus Metadaten generiert) wird im Adressbereich des Clients verwendet, um mit dem Server zu kommunizieren und die Proxyschnittstelle imitiert Schnittstelle des Servers. Obwohl SOAP-RPC-Format nicht erforderlich ist, werden die meisten modernen SOAP-Toolkits in Richtung ausgerichtet (mindestens diese Standardeinstellung um) mithilfe von RPC.

Im Gegensatz zu RPC, REST verfügt nicht über den Proxy Metadaten generiert d. h., die der Client weniger auf den Dienst gekoppelt. Da REST auf die Semantik des HTTP-basiert, können auch Anforderungen für Daten (GET-Anforderungen) zwischengespeichert. RPC-Systeme verfügen im Allgemeinen keine solche Infrastruktur (und sogar bei RPC mit SOAP über HTTP, können

nicht SOAP-Antworten zwischengespeichert, da SOAP das Verb HTTP POST verwendet, das als unsicher betrachtet wird). SOAP-eschews absichtlich HTTP, insbesondere damit SOAP, über andere Protokolle zu arbeiten, so dass Sie tatsächlich etwas disingenuous SOAP-basierte Dienste Webdienste aufrufen.

Meiner Sicht ist, dass REST und SOAP verwendet werden, können um ähnliche Funktionalität zu implementieren, aber im Allgemeinen SOAP verwendet, wenn ein bestimmtes Feature von SOAP erforderlich ist und die Vorteile von REST es andernfalls die beste Option machen.

Datenübertragungsformat (JSON)

Als Datenübertragungsformat für die REST-Webservices wird JSON verwendet. JSON steht für JavaScript Objekt

Notation und ist ein kompaktes, textbasiertes Dateiformat, welches einfach und schnell lesbar und

interpretierbar ist. Es wird häufig für den Datenaustauch zwischen Applikation verwendet.

Eine Alternative wäre XML, doch XML hat strukturbedingt einen viel grösseren Overhead als JSON und das Interpretieren von XML ist um einiges langsamer.

Datenübertragungsformat (SOAP)

Das „Simple Objekt Access Protokoll“ (SOAP) bildet eine Basis für den XML-basierten Informationsaustausch zwischen Applikationen. Derzeit wird die Idee durch das World Wide Web Konsortium weiterentwickelt und standardisiert. Mögliche Anwendungsszenarien von SOAP stellen sowohl einfache unidirektionale Benachrichtigungsdienste als auch komplexe Interaktionsszenarien, wie entfernte Methodenaufrufe, dar.Da SOAP selbst keine eigenen Sicherheitsmechanismen vorsieht, erfordert die Verwendung im sicherheitsrelevanten Umfeld den Einsatz zusätzlicher Mechanismen. Ihren Einsatz und ihr Zusammenspiel beschreibt dieser Beitrag. SOAP ermöglicht durch seine Nutzung für Web Services im Zusammenspiel mit WSDL und anderen Technologien, neue, dynamische Ansätze zur Interaktion zwischen kommerziellen und/oder wissenschaftlichen Anwendungen. Ein Problem von SOAP ist allerdings die Performanz. SOAP-Nachrichten werden in XML kodiert und verbrauchen somit erheblich mehr Speicherplatz, als herkömmliche Binärformate. Die gute Interoperabilität und universelle Einsetzbarkeit stehen im Gegensatz zu Ansprüchen hoher Performanz. Dies liegt vor allem an dem primär textbasierten Ansatz von XML. Eine typische SOAP-Nachricht kann 4-10-mal so gross sein, wie eine entsprechende Binärkodierung der Daten.

Source : http://www2.cs.uni-paderborn.de/cs/ag-engels/ag_dt/Courses/Lehrveranstaltungen/WS0405/PSWebservices/finals/gretencord.pdf

Entwicklung von WMD2C (WP7 App)

Einleitung

Die *Windows Mobile Dynamic Document Creation System Applikation*(WMD2C) wurde realisiert nach dem Client-Server-Prinzip mit Entity Framework Datenmodell als Access Layer, Windows Communication Foundation(WCF) als Service Access Layer. Die Service Access Layer (Servicezugriffsschicht) wird von einem Windows Phone 7 Client benutzt, um auf den SQL –Server 2008 zuzugreifen. Die Aufgabe der Servicezugriffsschicht ist es, von einer konkreten Kommunikationsschnittstelle zu abstrahieren(in diesem Fall der Web Service).

In diesem Kapitel werden die drei wichtigsten Schritte für die Entwicklung von WMD2C beschrieben. Zuerst werden die Installation der Datenbank und das Entity Framework Datenmodell beschrieben, danach folgt die Beschreibung des WCF Services und anschliessend wird der WP7 Client beschrieben.

Installation Datenbank & Entity Datenmodell

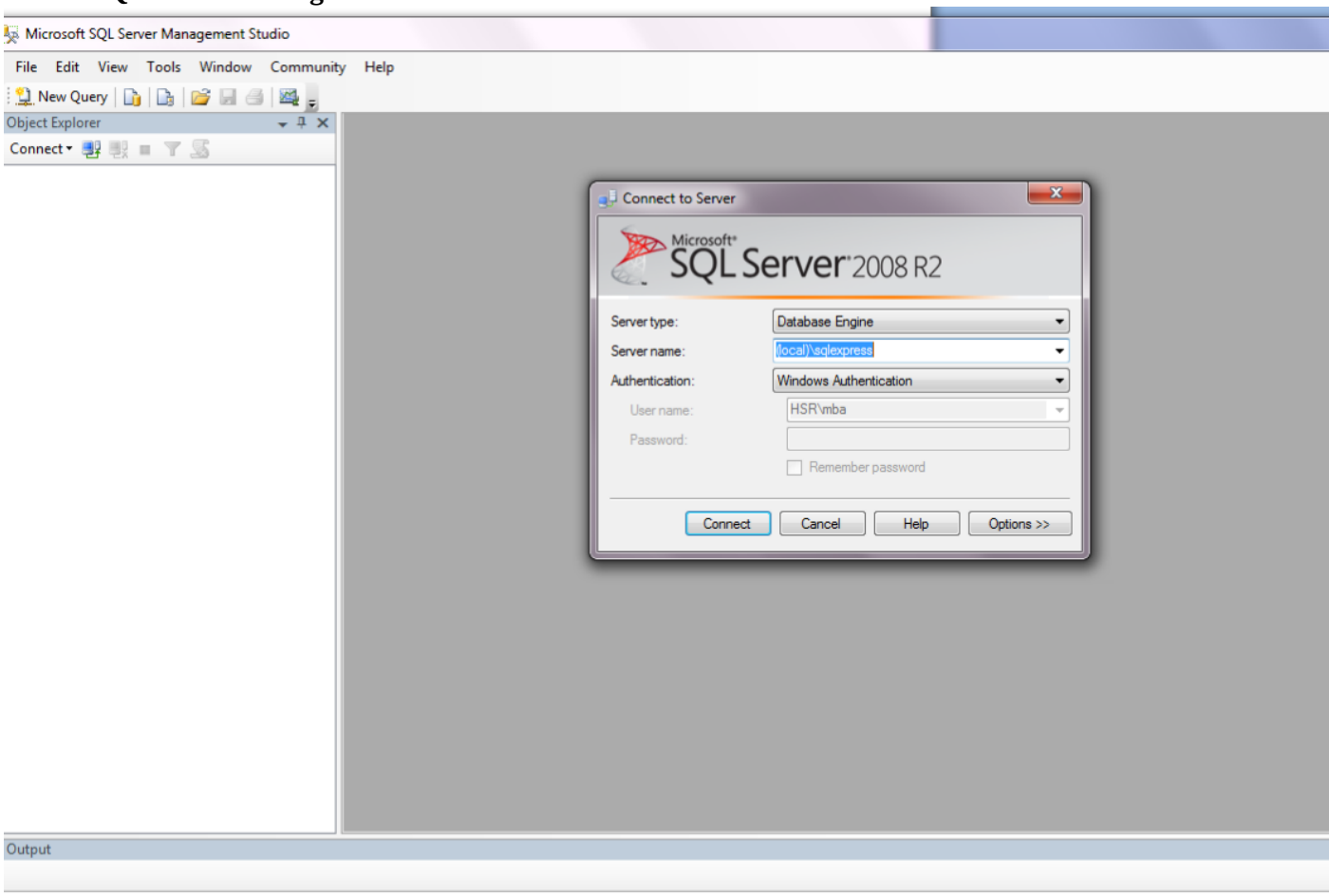
Datenbank

Datenbank-Tabellen

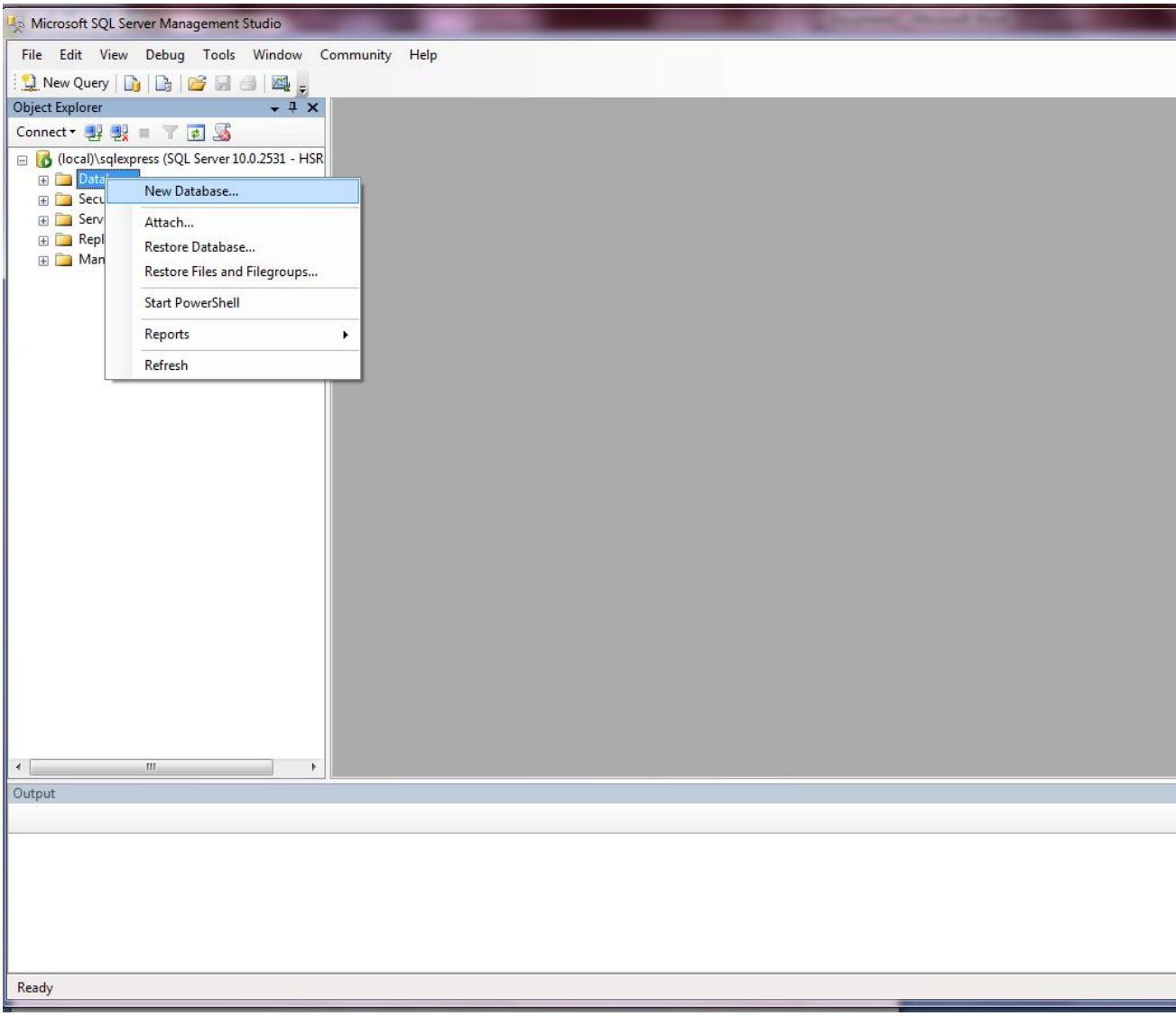
Die Datenbank(WMD2CDatabase) für das WMD2C-Projekt besteht aus 4 Tabellen: Documenttyps, Documents, Parameters und ParameterValues. Mit Hilfe von Microsoft SQL Server Management Studio wurde die notwendige Datenbank (WMD2CDatabase) erzeugt.

Datenbank Erzeugen

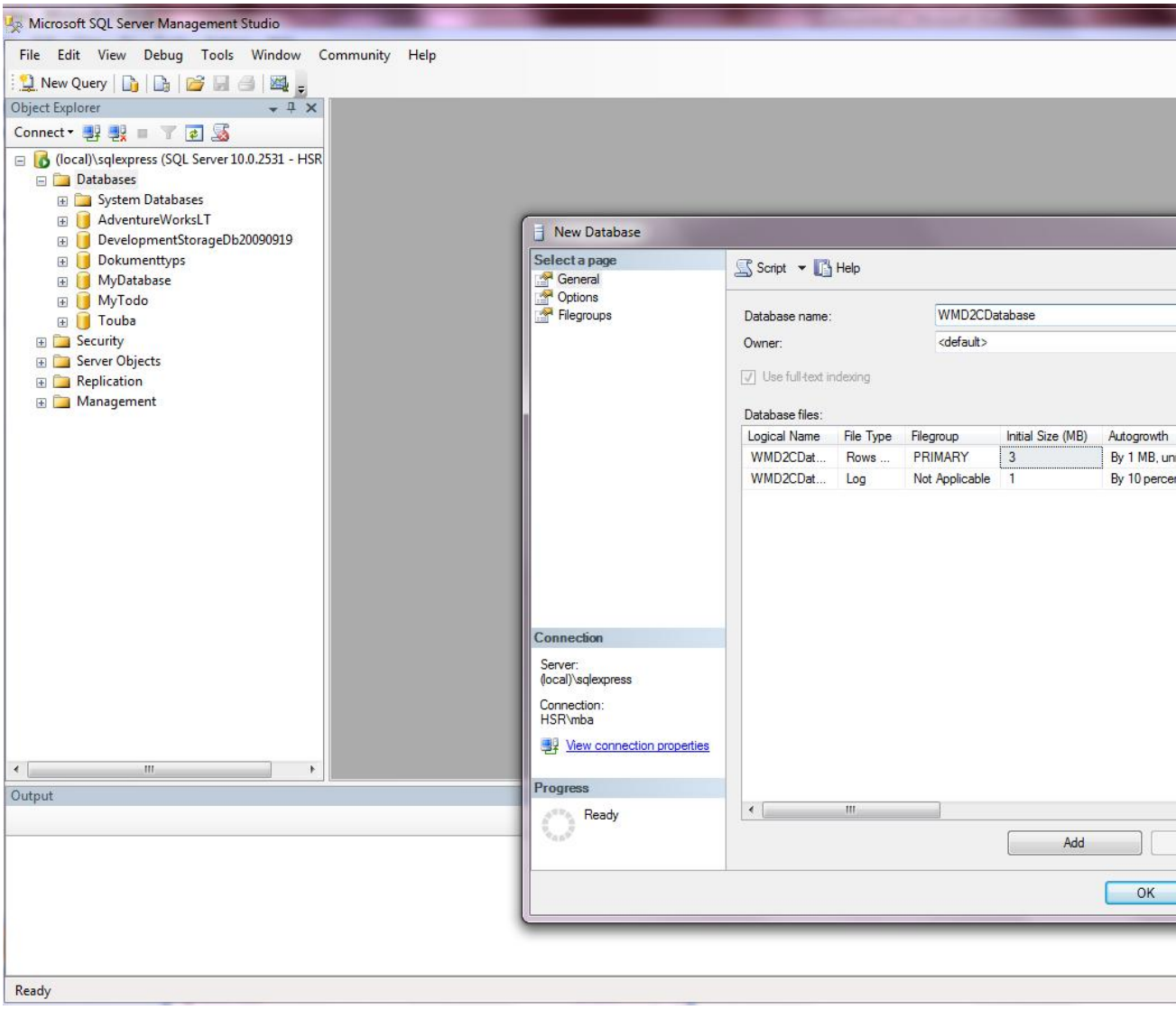
1. **SQL Server Management Studio starten und mit dem Server verbinden**



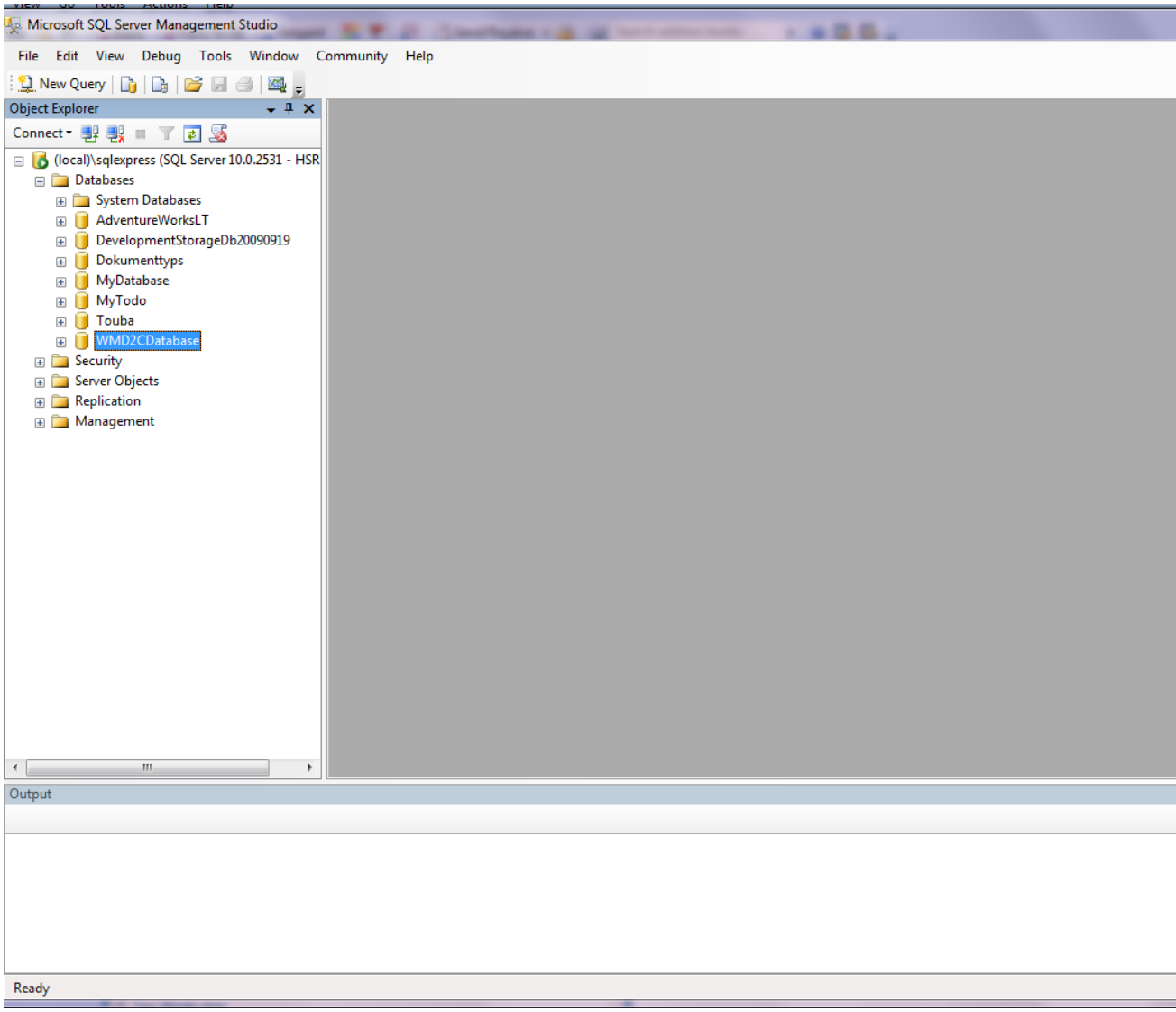
2. **Datenbank erzeugen (Rechtsklick auf Database und New Database auswählen)**



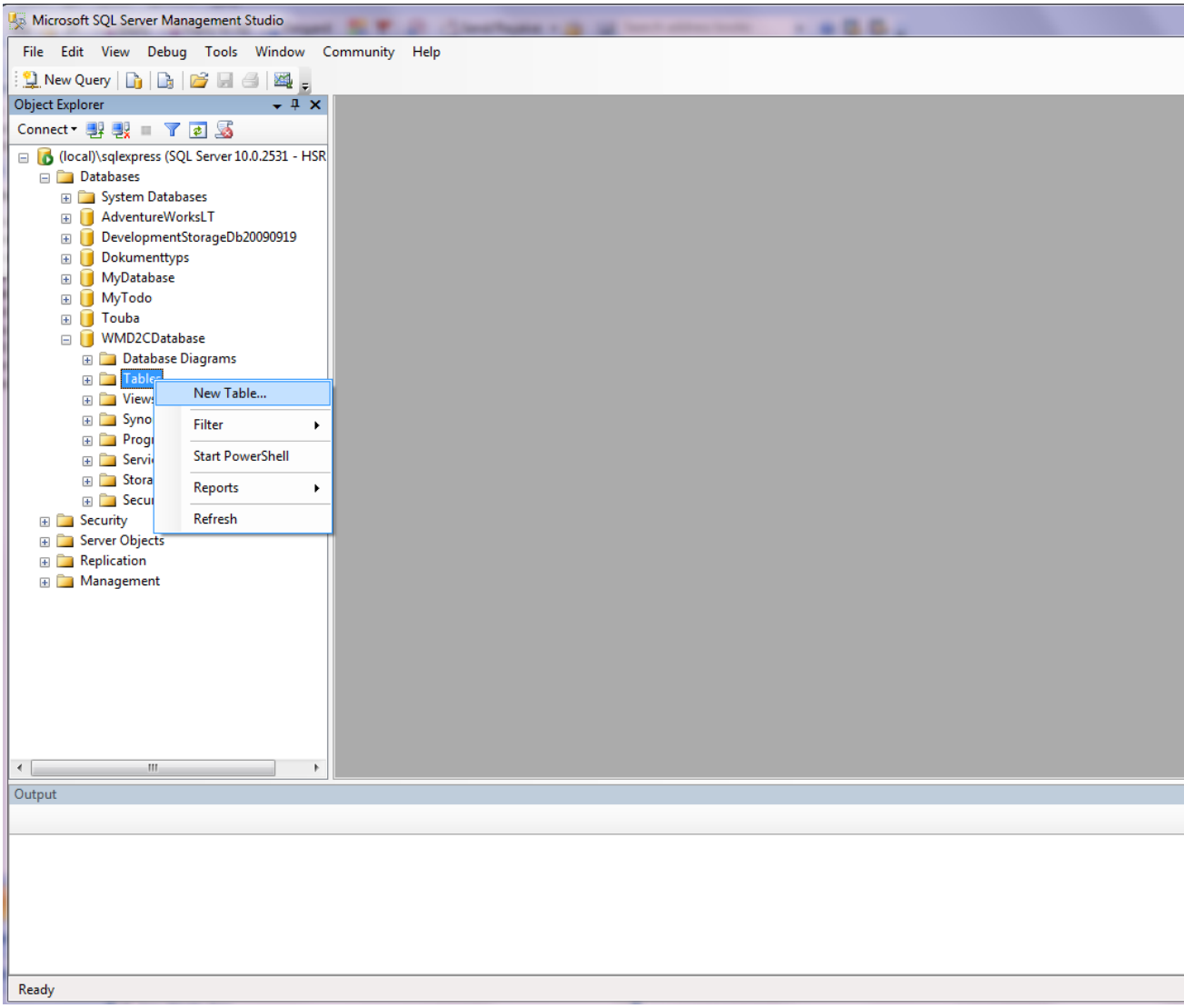
3. **Name für die Datenbank eingeben und auf Ok drücken (Hier WMD2Databae)**



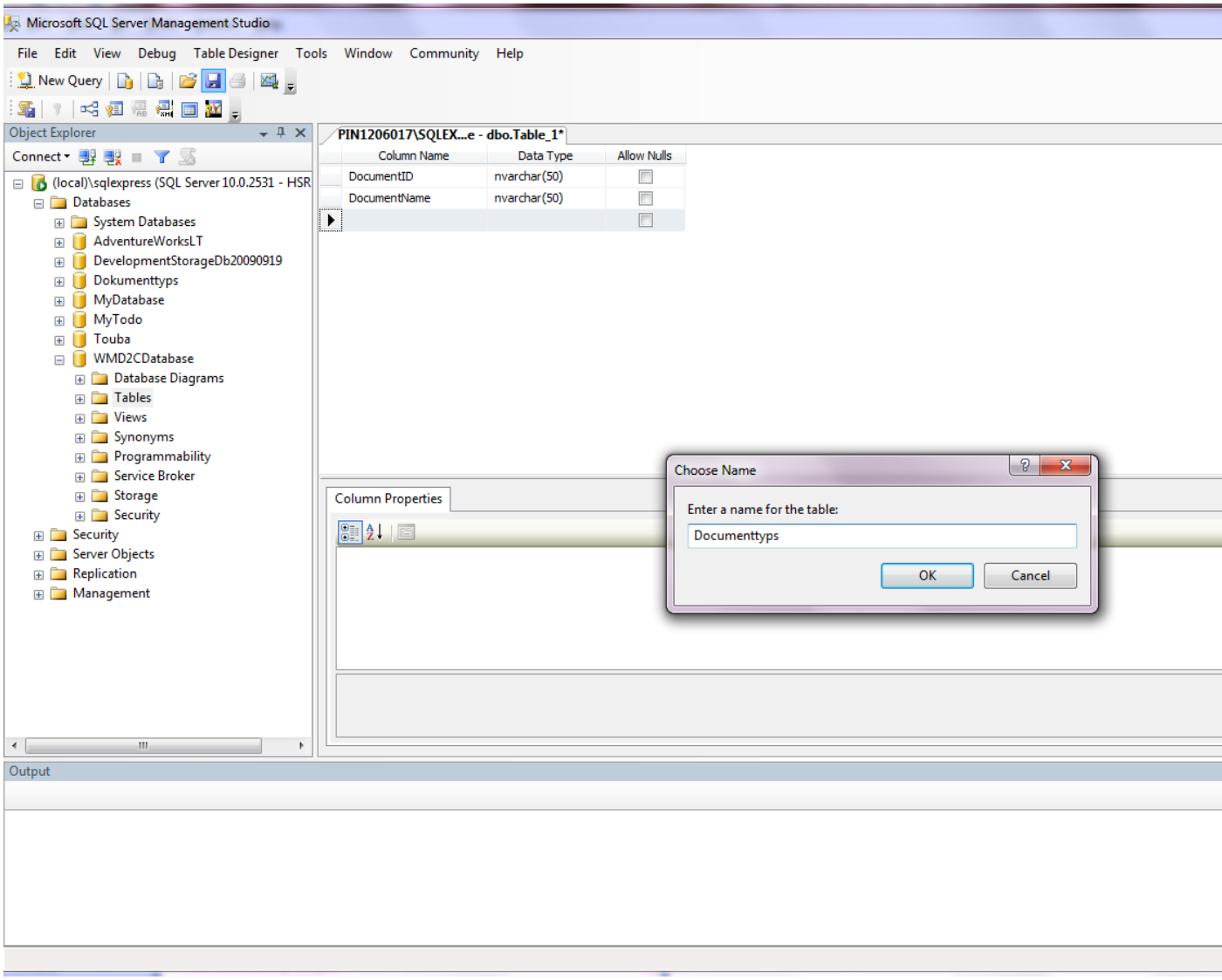
4. **Datenbank ist nun ersichtlich (WMD2CDatabase)**



5. **Nun Tabelle kann erzeugt werden (Rechtsklick auf Table und New Table auswählen)**

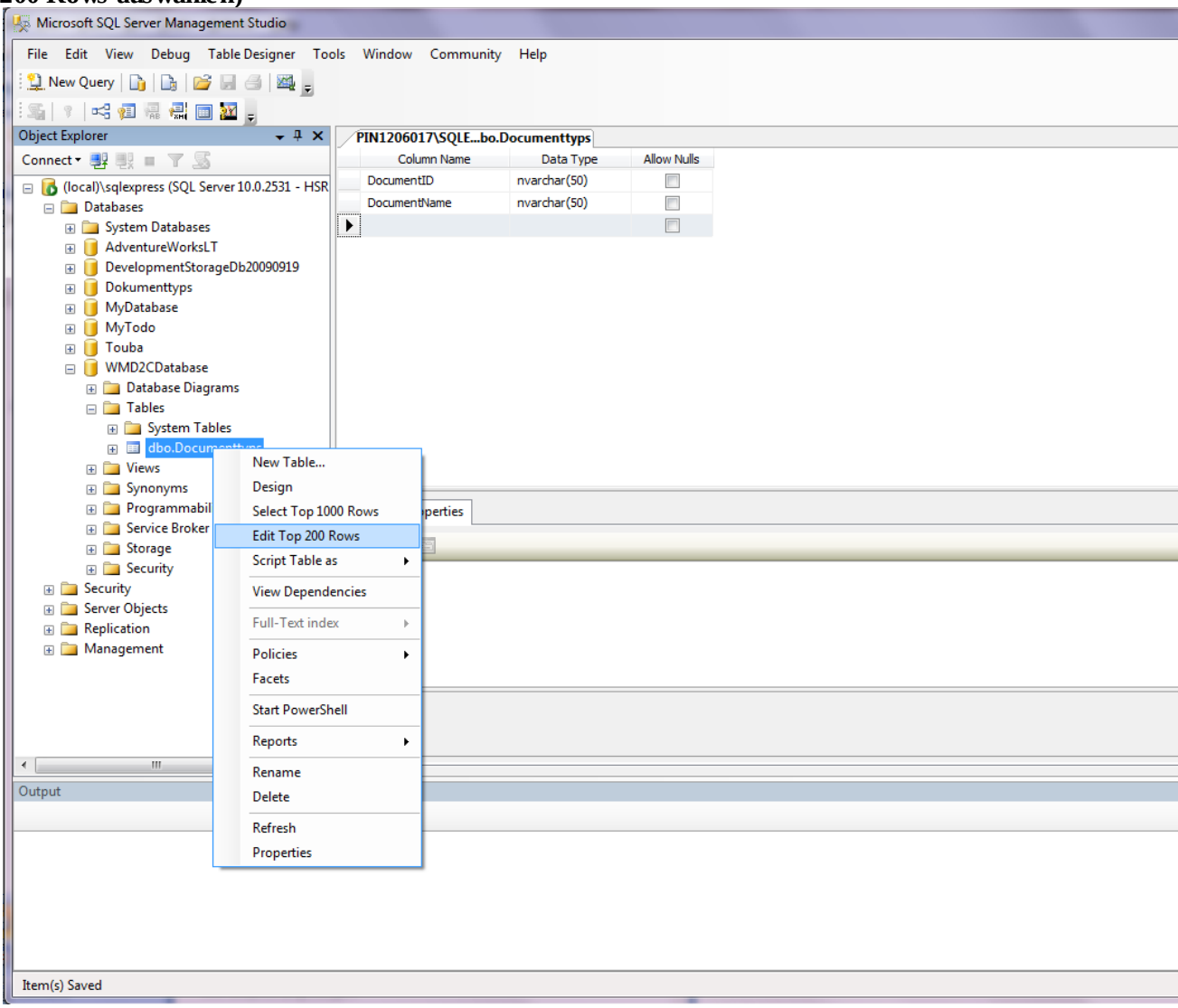


Nun haben wird Documenttyps-Tabelle bestehend aus zwei Spalten erzeugt.

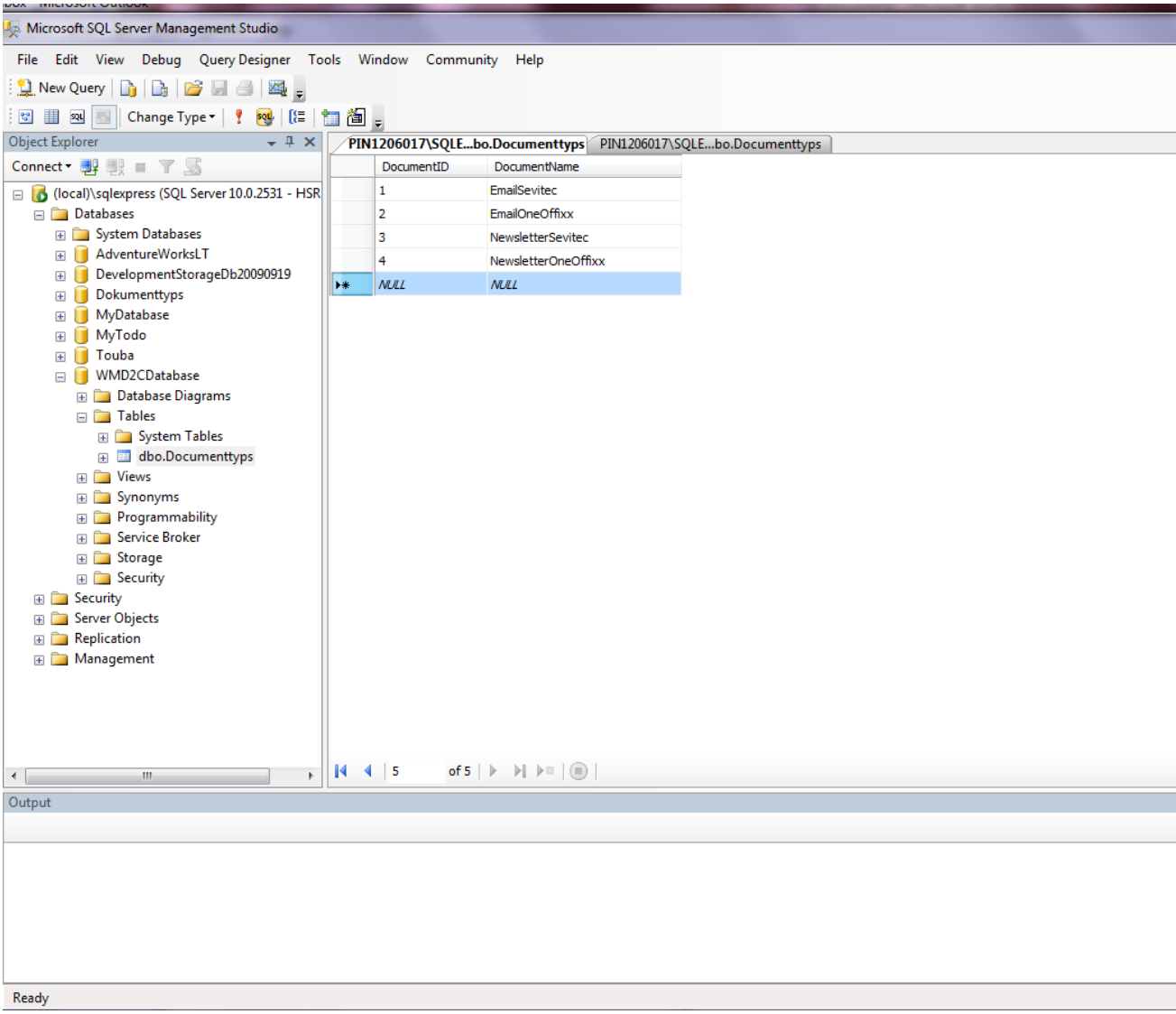


- DocumentID (Primary Key) [nvarchar(50)]
- DocumentName [nvarchar(50)]

6. **Documenttyps-Tabelle mit TestDaten füllen (Rechtsklick auf die Tabelle und Edit Top 200 Rows auswählen)**



7. **Documenttyps-Tabelle mit TestDaten füllen (Rechtsklick auf die Tabelle und Edit Top 200 Rows auswählen)**



Wir haben nun die erste Tabelle mit TestDaten gefüllt:

- **EmailSevitec**
- **EmailOneOffixx**
- **NewsletterSevitec**
- **NewsletterOneOffixx**

Die Tabellen Documents, Parameters und ParameterValues werden mit genauso erzeugt**.**

Tabelle : Documents

- DocID (Primary Key) [nvarchar(50)]
- DocumentID (Foreign key) [nvarchar(50)]
- Sprache [nvarchar(50)] allow Null
- DocName[nvarchar(50)]
- Content [ntext]

Tabelle: Parameters

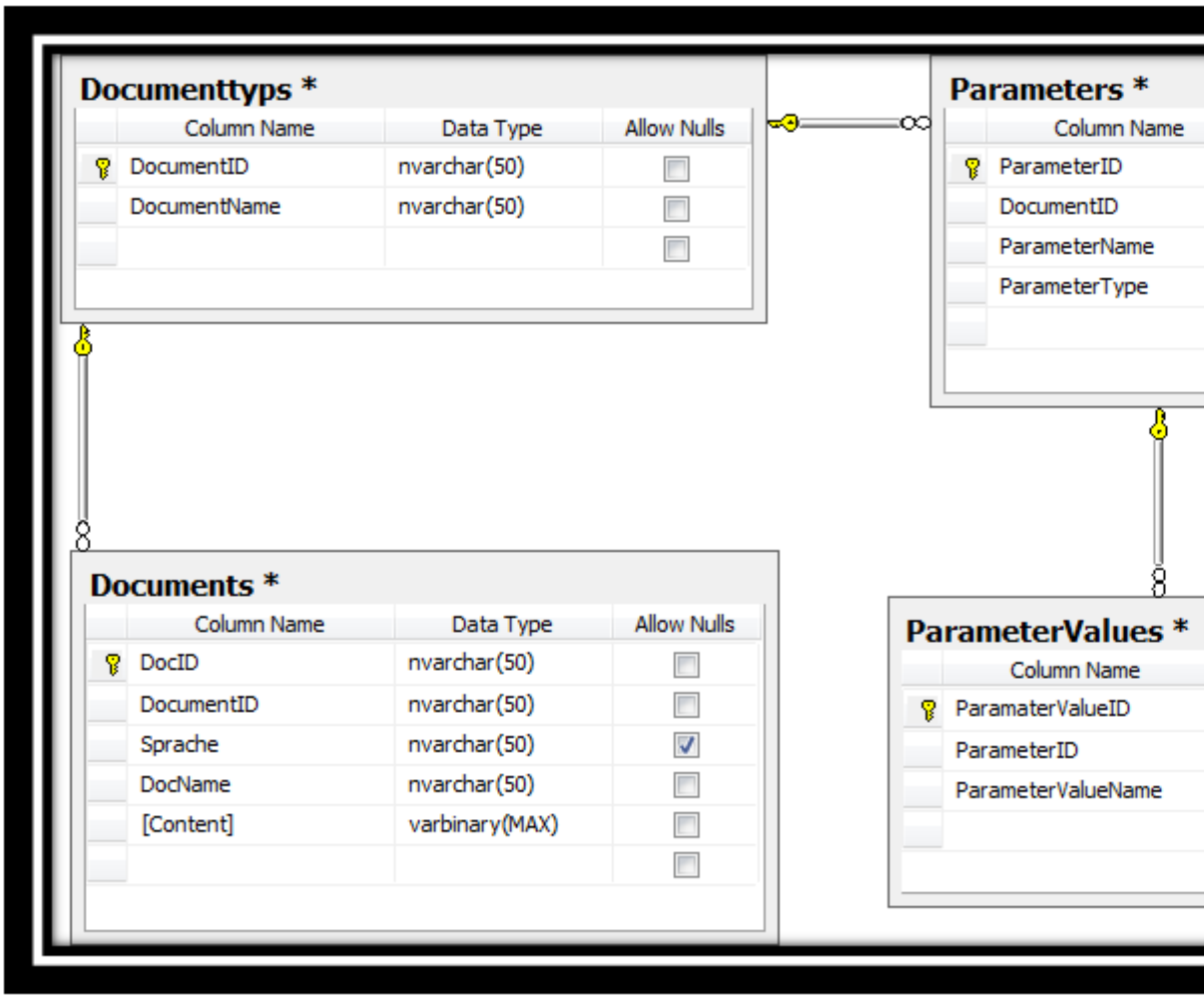
- ParameterID (Primary Key) [nvarchar(50)]
- DocumentID (Foreign key) [nvarchar(50)]
- ParameterName [nvarchar(50)]
- ParameterType[nvarchar(50)]

Tabelle: ParameterValues

- ParameterValueID (Primary Key) [nvarchar(50)]
- ParameterID (Foreign key) [nvarchar(50)]

- ParamValueName [nvarchar(50)]

Die SQL Server Datenbank ist einfach gehalten mit folgender Struktur:



Ein Parameter besitzt die Spalte „ParameterType“, welche 2 Werte haben kann:

- ParameterType = 10 **CheckBoxParameter**
- ParameterType = 20 **DropDownParameter**

Entity Framework Datenmodell

Für das Erzeugen eines Entity Datenmodells sind folgende Schritte erforderlich:

In Visual Studio 2010

- **New Solution erstellen (Projektmappe)**
- **Ein Projekt von Type „Class Library“ in die Solution hinzufügen (WCFEntityData)**

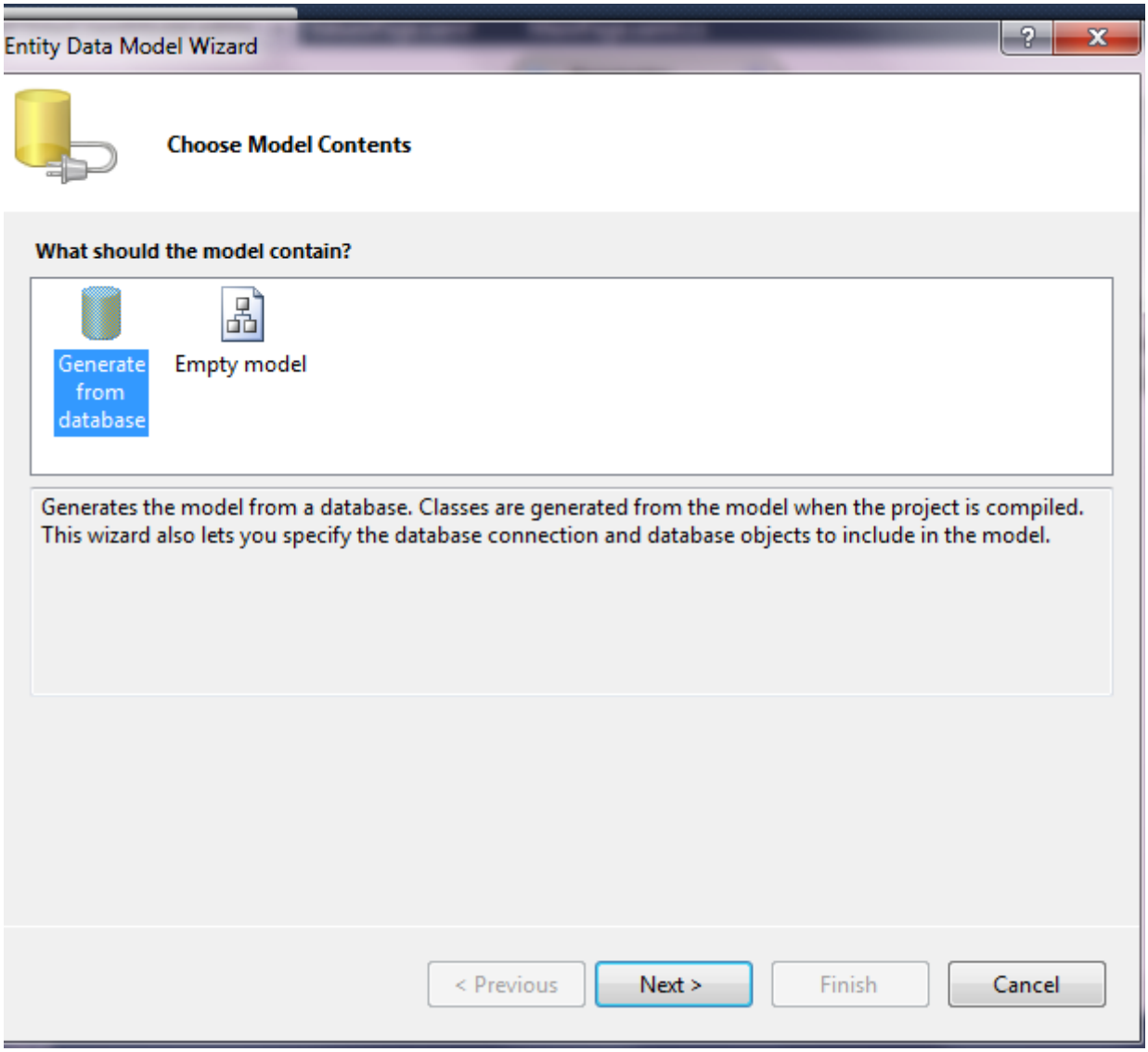
Im WCFEntityData –Projekt

- Referenzieren die Bibliothek (.dll) EF4CTP, welche in C:\Program Files (x86)\Microsoft ADO.NET Entity Framework 4.1 RC\Binaries zu finden sein sollte.
- Die automatisch erzeugte Klasse Class1.cs entfernen.
- Folgende Projekt-Ordner hinzufügen **:**
  - **Model**
  - **Entities**
  - **Interfaces**
  - **Repositories**
  - **SmtpMail**

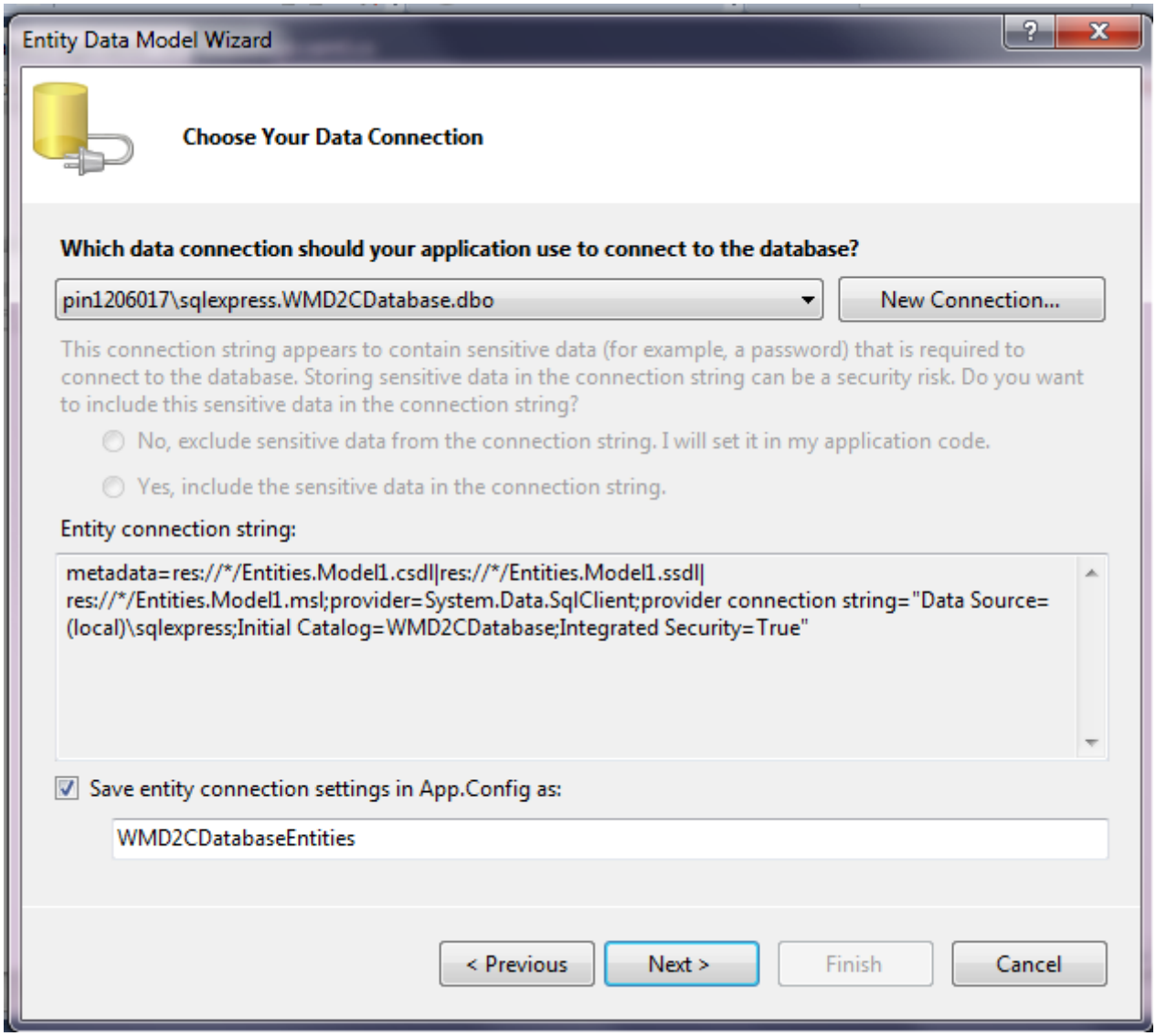
Im Ordner Model

Der Data Access Layer wird mit dem ADO.NET Entity Framework implementiert. Dieser Layer beinhaltet eigentlich das *.edmx Datenmodell (z.B. WP7MailClientModel.edmx), welches den Datenzugriff auf die darunterliegende SQL Server Datenbank ermöglicht.

- Add new „**ADO.Net Entity Data Model**“ mit dem Namen (WP7MailClientModel)

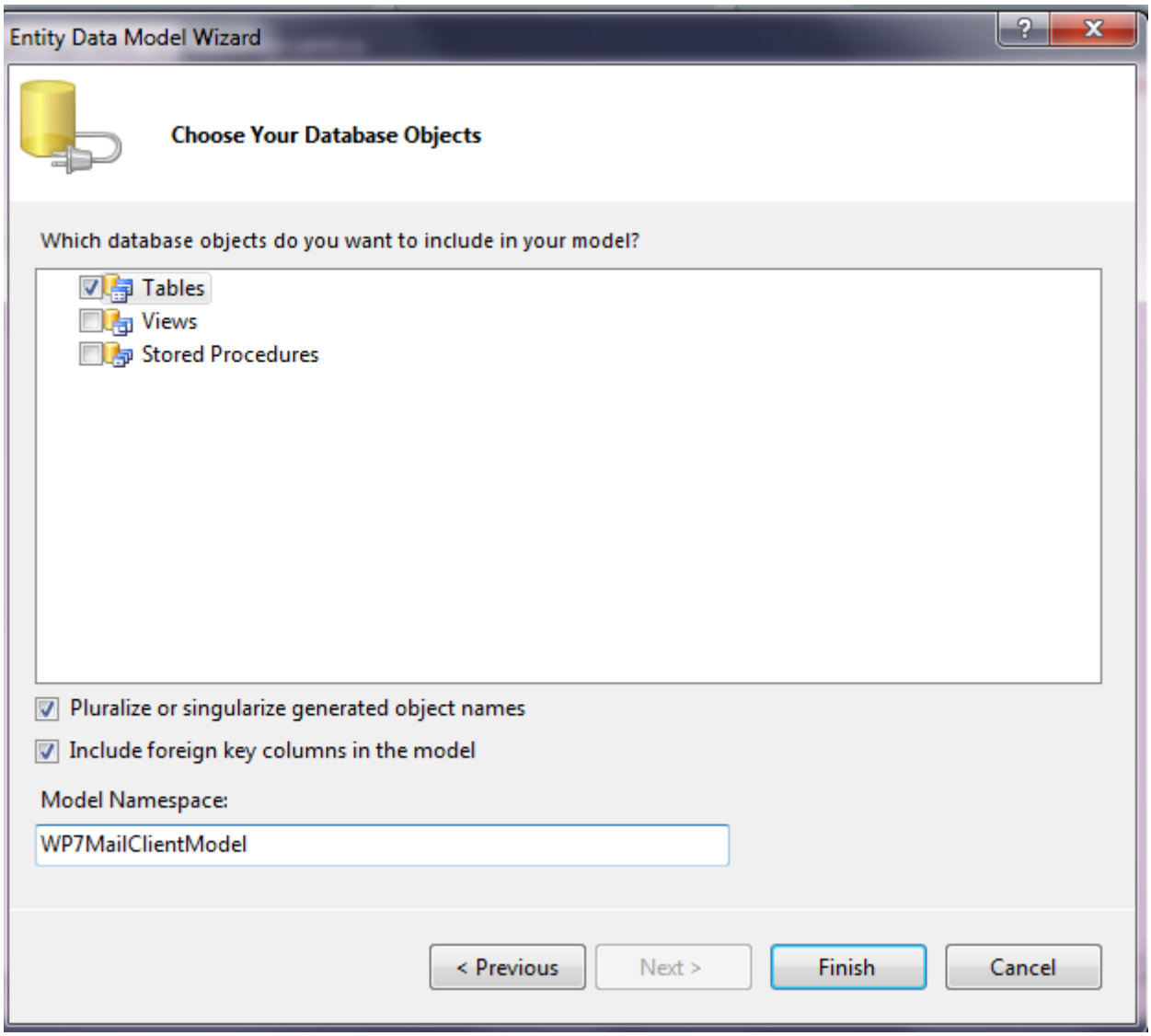


Auf „Next“ drücken und die erzeugte Datenbank für das Projekt auswählen (WMD2CDatabase)

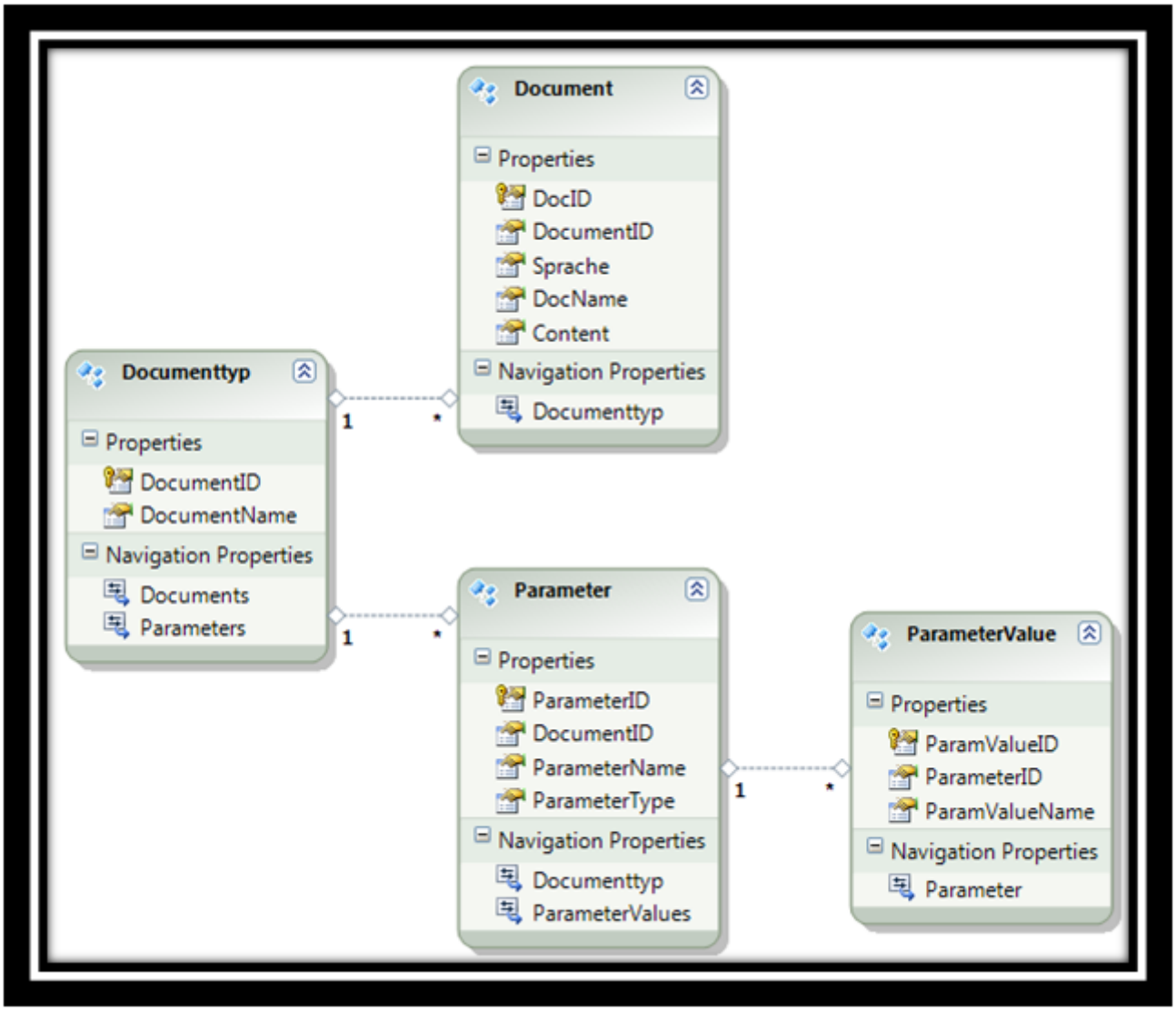


Auf „Next“ drücken und „Tables“ auswählen, um alle Tabellen in der Datenbank(WMD2CDatabase) auszuwählen

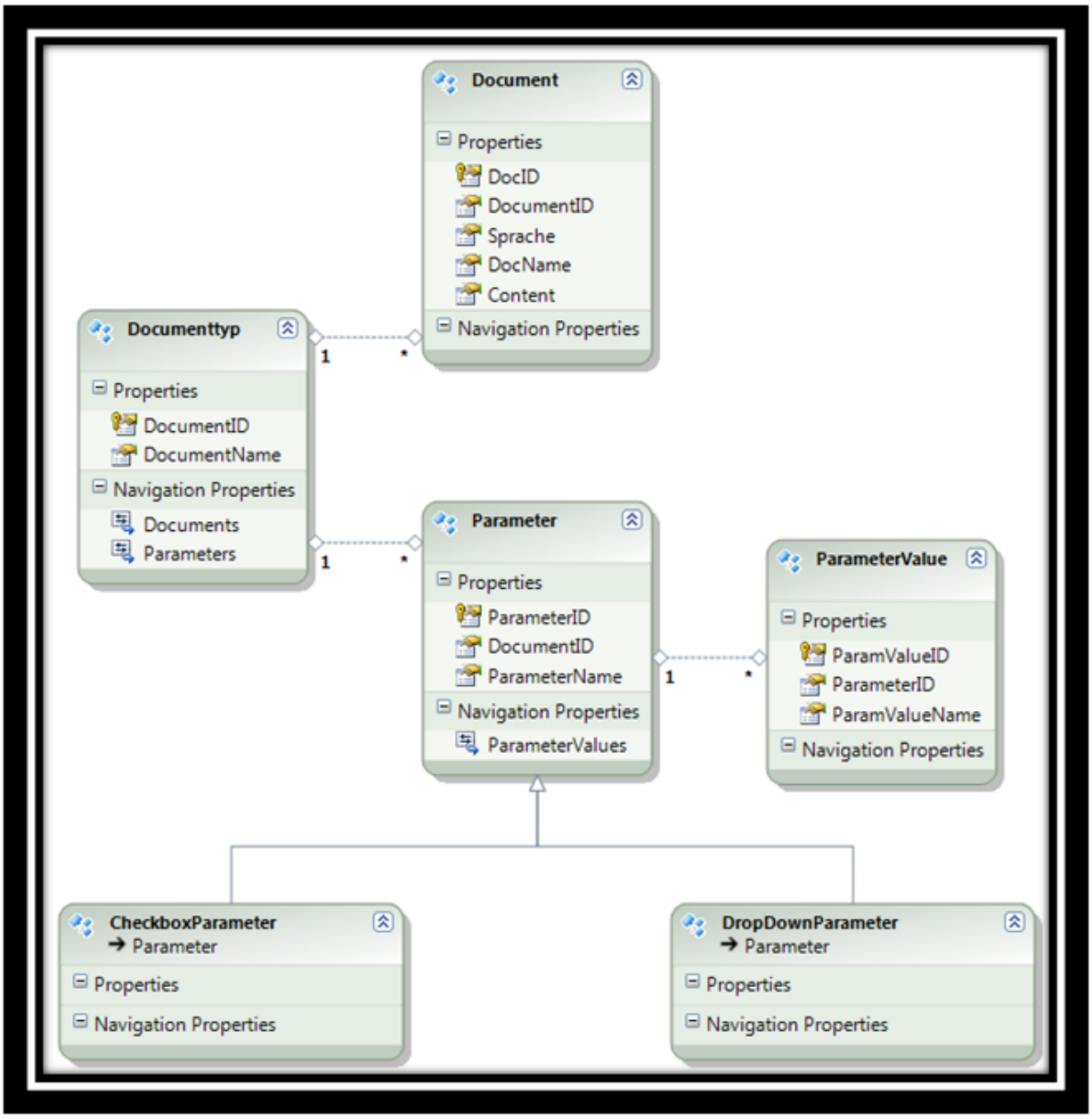
.



Auf „Finish“drücken und Das Datenmodell wird automisch im Projekt erzeugt.



Für die Lösung mit der Verwendung von Enumeratoren bezüglich ParameterType sieht das erweiterte Datenmodell folgend aus:



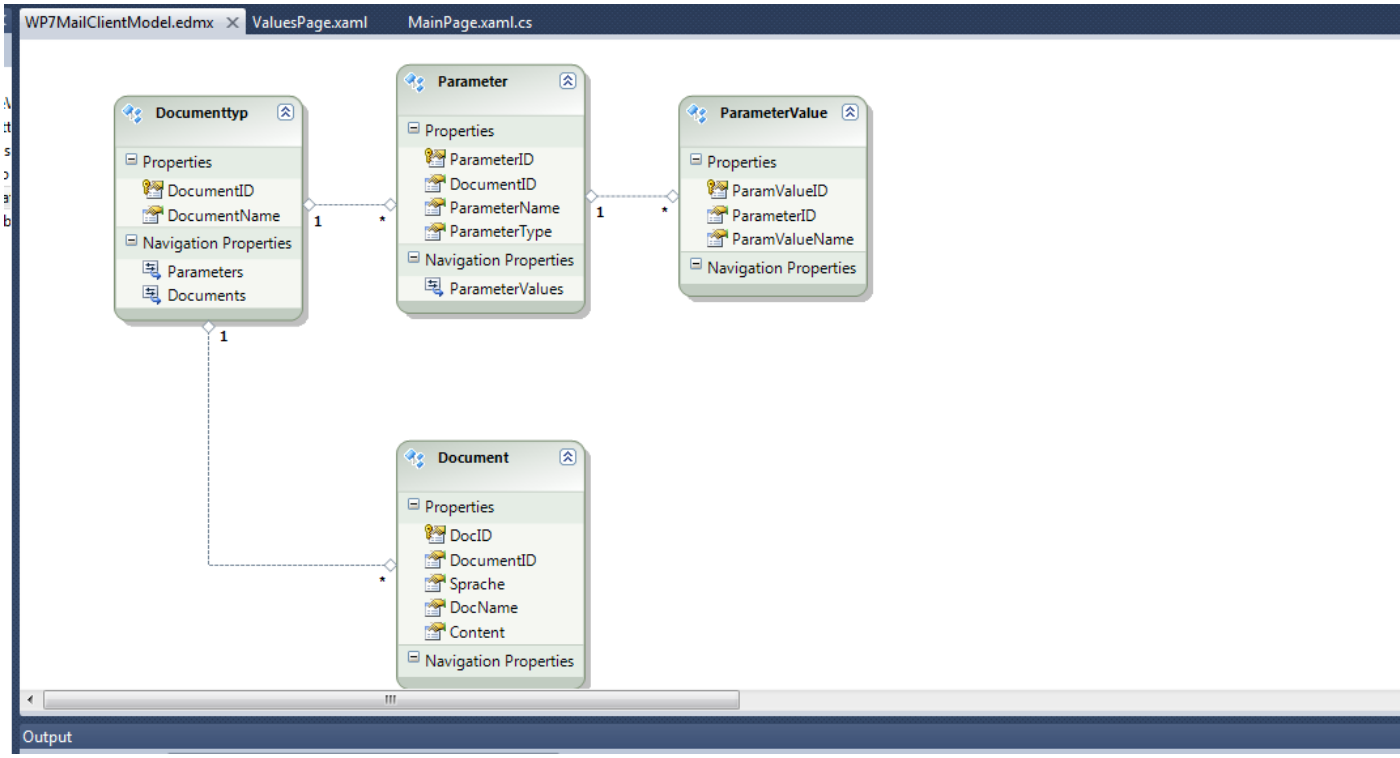
Im Datenmodell müssen überflüssige „Navigation Properties“ entfernt werden, um eine zirkuläre Abhängigkeit zwischen den Entitäten zu vermeiden. Die zirkuläre Abhängigkeit kann Probleme bei der Serializierung mittels WCF verursachen.

Die Entitätstypen entsprechen den im konzeptionellen Modell definierten Entitäten. In Entity Framework werden Entitäten und Beziehungen zugeordnet, die in einem konzeptionellen Modell für eine Datenquelle definiert werden. Entity Framework stellt Funktionen bereit, mit denen die folgenden Schritte durchgeführt werden können: Materialisieren der von der Datenquelle als Objekte zurückgegebenen Daten, Nachverfolgen von Änderungen an den Objekten, Behandeln der Parallelität, Zurückgeben von Objektänderungen an die Datenquelle und Binden von Objekte an Steuerelemente.

Folgende „**Navigation Properties**“ wurden deswegen entfernt:

- **Documents** (in der Entität Document)
- **Documents**(in der Entität Parameter)
- **Parameters**(in der Entität ParameterValue)

In den „**Propreties**“ des Models muss „**Code Generation Strategy**“ auf „**None**“ gesetzt werden, um zu verhindern, dass die Entity Klassen vom Entity Framework automatisch erzeugt werden.



Jedem Property des Models muss den Concurrency Mode auf „Fixed“ gesetzt werden, um OptimisticConcirrency zu aktivieren.

Im Ordner Model

- Eine neue Klasse einfügen (**WCFEntityContext), welche die ObjektContext-Klasse erweitern muss.**
- using System.Data.Objects;
-

```
using WCFEntityData.Entities;

namespace WCFEntityData.Model
{
  public class WCFEntityContext: ObjectContext
   {
        //Entitäten definieren (POCO)
      public ObjectSet<Documenttyp> Documenttyps {get;set;}
      public ObjectSet<Parameter> Parameters {get;set;}
      public ObjectSet<ParameterValue> ParameterValues {get;set;}
      public ObjectSet<Document> Documents { get; set; }


        //ConnectionString in AppConfig setzen
      public WCFEntityContext():this("name=DB1Entities"){}
      public WCFEntityContext(string
stringConnection):base(stringConnection ,"DB1Entities"){

          Documenttyps = CreateObjectSet<Documenttyp>();
          Parameters =  CreateObjectSet<Parameter>();
          ParameterValues = CreateObjectSet<ParameterValue>();
          Documents = CreateObjectSet<Document>();
          //Turned off to avoid problems with serialization
          ContextOptions.ProxyCreationEnabled = false;
          ContextOptions.LazyLoadingEnabled = false;

          }
```

Die Klasse **WCFEntityContext** soll dem Entity Framework sagen, wie es das Mapping zwischen den Entitäten und den Objekten im Entity Model machen soll.

Mit Entity Framework (EF) können Daten in Form von typisierten CLR-Objekten (Common Language Runtime), bei denen es sich um Instanzen von Entitätstypen handelt, abfragen, einfügen, aktualisieren und löschen.

Bei der ObjectContext-Kalsse(siehe ObjectContext-Klasse) handelt es sich um die primäre Klasse für die Interaktion mit Entitätsobjekten. Eine Instanz der **ObjectContext**-Klasse kapselt folgende Elemente: eine Verbindung zur Datenbank, Metadaten, mit denen das Modell beschrieben wird, und ein ObjectStateManager-Objekt (siehe ObjectStateManager-Objekt)das Objekte

bei Erstellungs-, Aktualisierungs- und Löschvorgängen nachverfolgt. Entity Framework stellt Tools für das automatische Erstellen einer Objektebene entsprechend dem konzeptionellen Modell zur Verfügung. Die Objektebene beinhaltet Entitätstypen und Objektkontextdefinitionen. Die Objektkontextdefinition enthält eine vom **ObjectContext** abgeleitete Klasse, die in der Regel über eine Reihe von Eigenschaften verfügt, die eine Auflistung von Entitäten des angegebenen Typs zurückgeben. Um den Code auf Objektebene zu generieren, wird den Entity Designer(siehe Entity Data Model Designer) oder das Befehlszeilentool EdmGen.exe. verwendet.

Im Ordner Entities

- Entitäten oder POCO (Plain Old CLR Object) definieren.
- Wir müssen dann 4 Klassen definieren: **Documenttyps, Document**, **Parameter** und **ParameterValues.**
  Das Entity Framework kann uns ermöglichen, vorhandene Domänenobjekte zusammen mit dem Datenmodell zu verwenden, ohne Änderungen an den Datenklassen vornehmen zu müssen. Diese POCO-Datenklassen unterstützen meist das gleiche Afrage-, Einfüge-, Aktualisierungs- und Löschverhalten wie die mit den Entity Data Model-Tools erstellten Entitätstypen.

  Bei der Verwendung von POCO-Typen werden am Objektdiagramm vorgenommene Änderungen von Entity Framework nicht automatisch in Echtzeit nachverfolgt. Mithilfe von Momentaufnahmen stellt Entity Framework Änderungen an den Objekten fest. Wenn Lazy Loading verwendet werden soll, muss die Lazy Loading-Proxyerstellung aktiviert werden.

```
//Turned off to avoid problems with serialization

    //Lazy Loading Proxyerstellung deaktiviert
```

```
            ContextOptions.ProxyCreationEnabled = false;
            ContextOptions.LazyLoadingEnabled = false;

            //Lazy Loading-Proxyerstellung aktivieren
            ContextOptions.ProxyCreationEnabled = true;
                ContextOptions.LazyLoadingEnabled = true;
```

Documenttyp

```
[DataContract]
  public class Documenttyp
  {
      [DataMember]
      public string DocumentID { get; set; }
      [DataMember]
      public string DocumentName { get; set; }
      [DataMember]
      public virtual IList<Parameter> Parameters { get; set; }
      [DataMember]
      public virtual IList<Document> Documents { get; set; }
```

Document

```
[DataContract]
    public class Document
    {

        [DataMember]
        public string DocID {get; set;}
        [DataMember]
        public string DocumentID { get; set; }
        [DataMember]
        public string Sprache { get; set; }
        [DataMember]
        public string DocName { get; set;
        [DataMember]
        public string Content {get; set; }









    }
```

Parameter

```
 [DataContract]
public class Parameter
 {
    [DataMember]
    public string ParameterID {get; set; }
    [DataMember]
    public string DocumentID {get; set; }
    [DataMember]
    public string ParameterName {get; set; }
    [DataMember]
    public int ParameterType {get; set; }
    [DataMember]
    public virtual IList<ParameterValue> ParameterValues { get; set; }




 }
```

ParameterValue

```
[DataContract]
public class ParameterValue
 {
    [DataMember]
    public string ParamValueID {get; set ;}
```

```
        [DataMember]
        public string ParamValueName {get; set ;}


        [DataMember]
        public string ParameterID {get; set;}
    }
```

[DataContract] und [DataMember] ermöglichen es dem Entity Framework, die Objekte zu verwenden. Die Documenttyp-Klasse und die Parameter-Klasse haben als DataMember auch eine Liste von Objekten. Diese Liste ist mit dem Schlüsselwort virtual, damit Entity Framework Lazy-Loading machen kann.

Windows Communication Foundation (WCF) kann Proxys nicht direkt serialisieren oder deserialisieren, da der DataContractSerializer (siehe DataContractSerializer)nur bekannte Typen serialisieren und deserialisieren kann, und Proxytypen sind keine bekannten Typen. Wenn POCO-Entitäten serialisiert werden müssen, so muss die Proxyerstellung deaktiviert werden. Oder die ProxyDataContractResolver-Klasse verwenden (siehe ProxyDataContractResolver-), um Proxyobjekte als ursprüngliche POCO-Entitäten zu serialisieren. Um die Proxyerstellung zu deaktivieren, muss die **ProxyCreationEnabled**-Eigenschaft auf false festgelegt werden. POCO-Proxys sollen verwendet werden, wenn eine hocheffiziente und unmittelbare Änderungsnachverfolgung sowie Lazy Loading erforderlich ist. Gleiche Funktionen stehen bei der Verwendung von Proxys zur Verfügung wie bei von *EntittyObjek*t abgeleiteten Typen.Die Domänenklassen werden jedoch weiterhin von Entity Framework getrennt. Um einen Lazy Loading-Proxy und /oder die Proxyerstellung für die sofortige Änderungsnachverfolgung zu aktivieren, müssen jedoch die POCO-Klasse die Anforderungen für die Erstellung von POCO-proxys erfüllen. (siehe Anforderungen für die Erstellung von POCO-Proxys (Entity Framework)).

Im Odrner Interfaces

In diesem Ordner sind zwei wichtigen Klassen für den WCF Service definiert:

- **IDocumentService (Seviceschnittstelle)**

  Das Service-Interface ***IDocumentService*** definiert die Funktionalität für den Service-Layer. Für jede Entität- Documenttyp, Document, Parameter

und ParameterValue – müssen folgende CRUD-Operationen(Create, Read, Update, Delete) unterstützt werden:

- **Alle Entitäten lesen**
- **Eine Entität anhand des Primärschlüssel lesen**
- **Einfügen**
- **Update**
- **Löschen**

Der Rückgabewert der beschriebenen Methoden - wenn vorhanden – ist immer ein DTO(Data Transfer Object) /oder Poxy-Object respektive eine Liste davon, nie eine Entität des Data Access Layers.

```
[ServiceContract]
public interface IDocumentService
{

    [OperationContract]
    [ApplyProxyDataContractResolver]
    IList<Documenttyp> DocumenttypGetAll();

    [OperationContract]
    [ApplyProxyDataContractResolver]
     IList<Parameter> GetDocParamAll(string docType);

    [OperationContract]
    [ApplyProxyDataContractResolver]
    IList<ParameterValue> GetParamValues(string p);

    [OperationContract]
    [ApplyProxyDataContractResolver]
    IList<ParameterValue> GetParamVal();
```

```
//Document Methods.
[OperationContract]
[ApplyProxyDataContractResolver]
IList<Document> GetDocById(Documenttyp doctype);

[OperationContract]
[ApplyProxyDataContractResolver]
IList<Document> GetAllDocument();

[OperationContract]
[ApplyProxyDataContractResolver]

string GetDocBySpr(string id, string sprache);

[OperationContract]

[ApplyProxyDataContractResolver]
    Void SendMailClient(string usr,string pwd,string from, string to, string
CC,string host, string port, string subject,    string body);
```

Damit WCF die Serialisierung/Deserialisierung von Objekten korrekt durchführen kann, wird die ProxyDataContractResolver-Klasse als Attribute für die Service-Methoden markiert.

- **ProxyDataContractResolver (Für die Serializierung/Deserializierung von POCO-Entitäten).**
  Wenn POCO-Entitäten serialisiert werden müssen, wird die ProxyDataContractResolver-Klasse verwendet, um Proxy-Objekte als ursprüngliche POCO-Entitäten zu serialisieren.

Im Odrner SmtpMail

- **WCFMailer-Klasse (Definiert die SendMail-Methode)**

Das Versenden von Emails erfolgt über WCF-Service, da der WP7-Email-Client keine HTML-Email unterstützen kann.

```
using System.Net.Mail;

using System.Net.Mime;


namespace WCFEntityData.Entities

{

   public class SmtpSendMail

   {

      /// <summary>

      /// Sends an mail message

      /// </summary>

      /// <param name="from">Sender address</param>

      /// <param name="to">Recepient address</param>

      /// <param name="bcc">Bcc recepient</param>

      /// <param name="cc">Cc recepient</param>

      /// <param name="subject">Subject of mail message</param>

      /// <param name="body">Body of mail message</param>

      ///

//Zum testen haben wir meinen GMX-Account verwendet!


        //SMTP-SERVER und SMTP-PORT

      private static string smtPServer = "mail.gmx.net";

      private static int serverPort = 25;
```

```
public static void SendMailMessage(string from, string fromName, string
to, string toName, string cc, string ccName, string subject, string body)
{
    //Create objects of MailMessage and SmtpClient –
        //MailMessage-Objekt und SmtpClient-Objekt erzeugen

    MailMessage m = new MailMessage();
    SmtpClient sc = new SmtpClient();

        //Add – From, To, CC, BCC, subject, Body – as shown.

        m.From = new MailAddress(from, fromName);
        m.To.Add(new MailAddress(to, toName));
        m.CC.Add(new MailAddress(cc, ccName));
        //similarly BCC
        m.Subject = subject;

      //IsBodyHtml muss hier auf true gesetzt werden, um HMTL-Mail zu
unterstützen.

        m.IsBodyHtml = true;
        m.Body = body;
```

```
            //SMTP-SERVER und SMTP-Port festlegen


            sc.Host = smtPServer;
            sc.Port = serverPort;
               //Login und Passwort für GMX-Account für die Anmeldung
               // auf GMX-Mail-Server werden überprüft und SMTP-Port
festlegen
            sc.Credentials = new
              System.Net.NetworkCredential("tafaba@gmx.de", "tafa2674");
            sc.EnableSsl = true;
        //Nach erfolgeicher Überprüfung der Credentials, wird die Email
verschickt


            sc.Send(m);


}
```

Business Layer

Projekt „WCFEntityData.BusinessLayer“

Im Business Layer findet der Zugriff auf den Data Access Layer (DAL) statt, sprich hier werden die Daten vom DAL geladen und verändert. Im Business Layer befindet sich eine ***Klasse DocumentRepository***, welche die Business-Operationen implementiert, die für die Implementation des Service- Interface benötigt werden.

Das Handling der OptimisticConcurrencyException – eine Exception welche vom Framework beim Auftreten einer Optimistic Concurrency-Verletzung geworfen wird – soll bei den Update-Methoden gehandhabt werden. Im Falle des Auftretens einer solchen Exception wird eine LocalOptimisticConcurrencyException geworfen, welche die neuen in der Datenbank vorhandenen Werte beinhaltet.3

Falls den Update-Methoden das modifizierte und das Original-Objekt mitgeben wird, können diese nach dem konvertieren wieder dem verwendeten OR-Mapper angehängt werden.

Service Layer

Projekt „DocumentService“

Der Service Layer ist die eigentliche WCF-Serviceschnittstelle und implementiert das Interface **IDocumentService**.

Im Normalfall müsste es hier genügen, eine Instanz der ***DocumentRepository-Klasse*** zu halten und die eingehenden Calls mehr oder weniger direkt an diese weiterzureichen. Der Service-Layer ist in dieser einfachen Applikation also nicht viel mehr als ein „Durchlauferhitzer“.

Windows Phone 7 Client

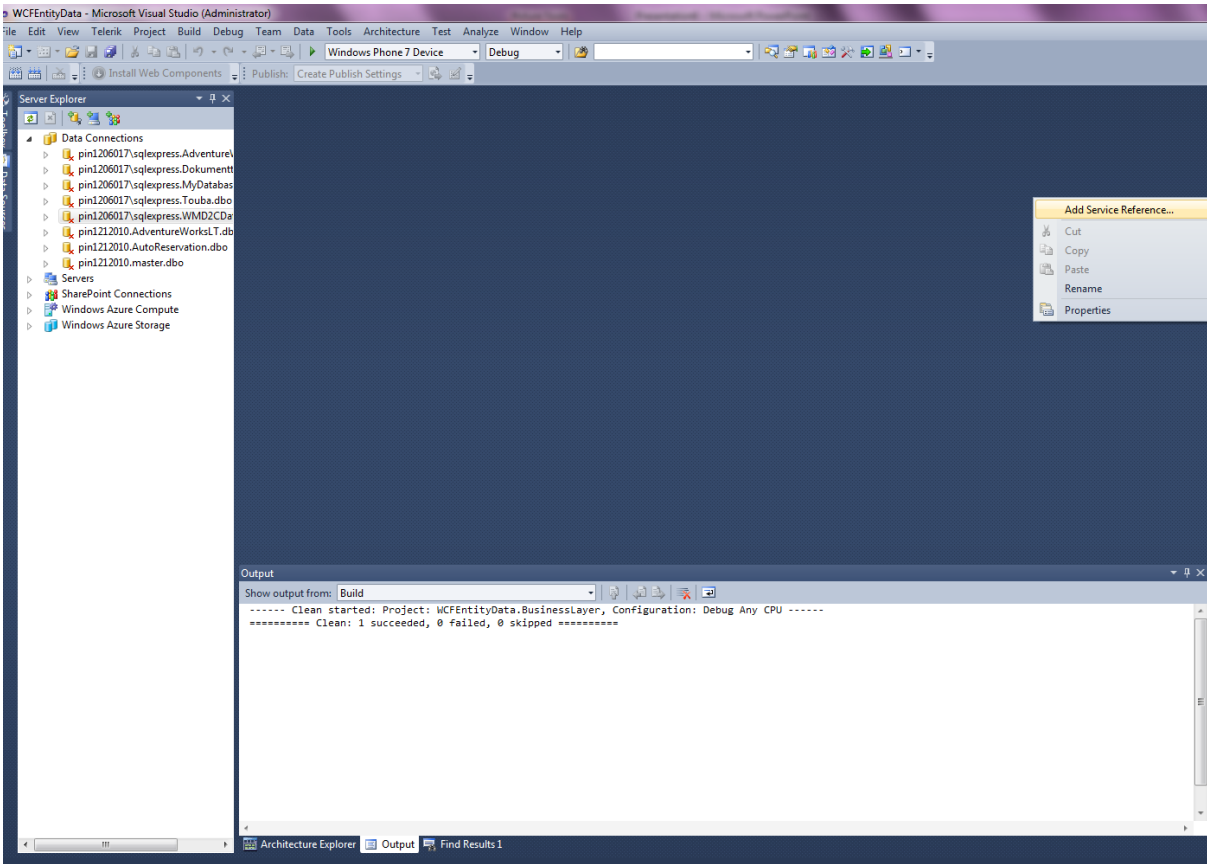
Projekt „WP7TestClient“

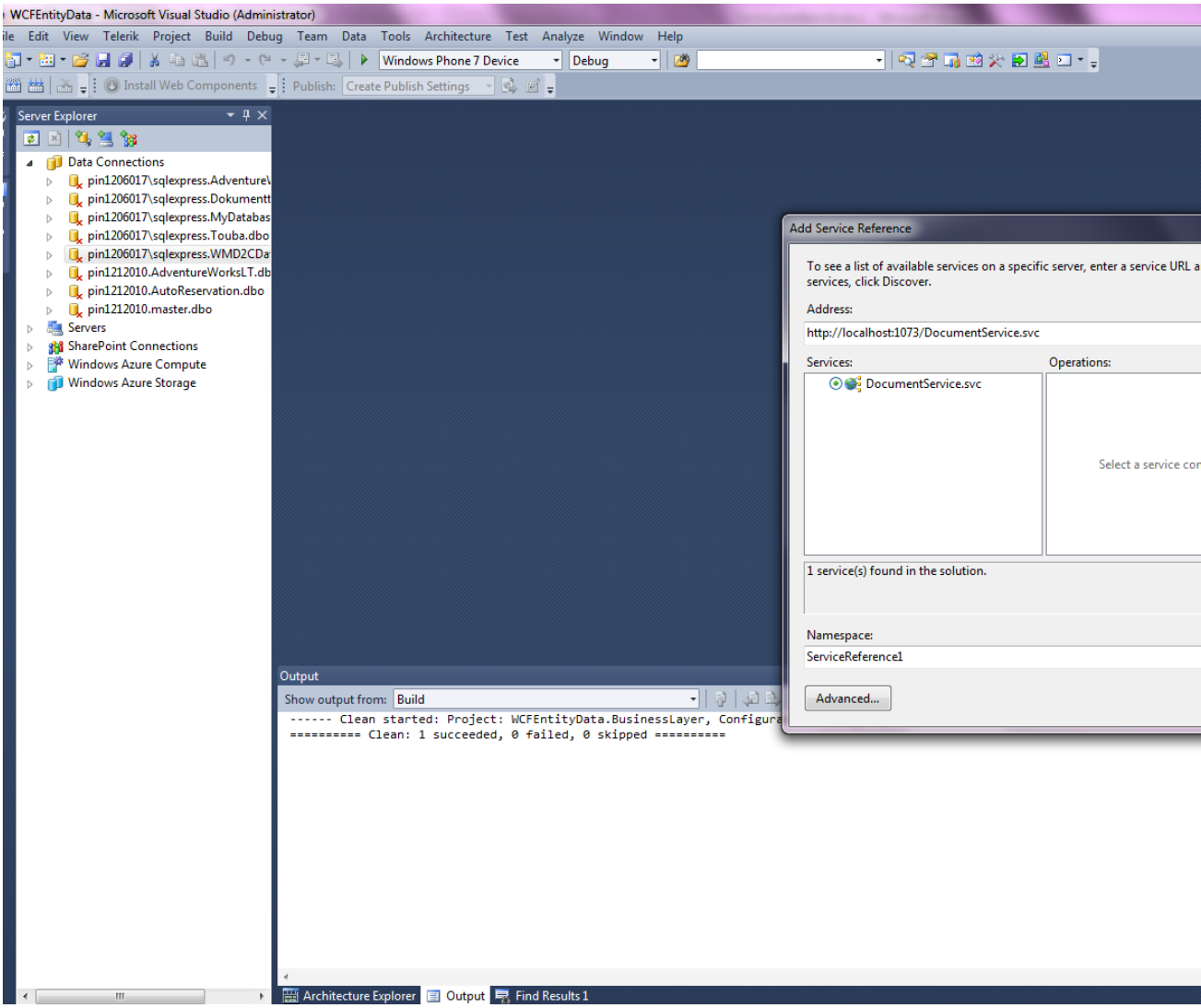
Die Wichtigsten Schritte zur Entwicklung eines Windows Mobile-Anwendung sind wie folgt:

- Erstellen eine Windows Mobile Application
- Erstellen eine Datenbank
- Erstellen eines WCF-Dienst
- Konsumieren des WCF-Dienst von Windows Mobile 7 Application
  Erstellen eine Windows Mobile 7 Application wird im nächsten Kapitel beschrieben. Datenbank und WCF-Dienst sind schon vorhanden. Nun wollen wir den WCF-Service im Windows Phone 7 Client konsumieren.

  Verbrauchen WCFdienst von Windows Mobile 7 Application

  - Dienstverweis hinzufügen zu WP7TestClient (Add Service Reference)


WCFEntityData - Microsoft Visual Studio (Administrator)
File Edit View Telerik Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Windows Phone 7 Device
Debug
Install Web Components
Publish:
Create Publish Settings
Server Explorer
Data Connections
pin1206017\sqlexpress.Adventure\
pin1206017\sqlexpress.Dokumentt
pin1206017\sqlexpress.MyDatabas
pin1206017\sqlexpress.Touba.dbo
pin1206017\sqlexpress.WMD2CDa
pin1212010.AdventureWorksLT.db
pin1212010.AutoReservation.dbo
pin1212010.master.dbo
Servers
SharePoint Connections
Windows Azure Compute
Windows Azure Storage
Add Service Reference...
Cut
Copy
Paste
Rename
Properties
Output
Show output from: Build
------ Clean started: Project: WCFEntityData.BusinessLayer, Configuration: Debug Any CPU ------
========== Clean: 1 succeeded, 0 failed, 0 skipped ==========
Architecture Explorer
Output
Find Results 1



Nach diesem Schritt werden alle Metadaten des WCF-Dienstes im WP7Client kopiert. Im WP7Client kann nun über einen Proxy den WCF – Service konsumiert werden.
Da wir im WP7Client Dokumente dynamisch erstellen wollen, muss die Anwendung mit Listen arbeiten, um die Daten anzeigen zu können. Die WP7 –Anwendung besteht aus vier wesentlichen Views (Seiten):

- MainPage
- ParameterViewPage
- EditorPage
- EmailConfigPage

MainPage.xaml

Die XAML-Datei der MainPage definiert die Oberfläche der Hauptseite. Sei beinhaltet ein ListBox-Control-Element (LongListSelector).

Sie zeigt alle Documenttyps aus der Datenbank (WMD2CDatabase) an.

Die neue Version von Silverlight for Windows Phone Toolkit beinhaltet neuen Controls, welche auf der Projektseite von Silverlight Toolkit auf Codeplex Released worden sind.

Neue Komponenten sind folgende:

- AutoCompleteBox
- ListPicker
- LongListSelector
- Page Transitions

Änderungen gab es bei folgenden Komponenten:

- Gesture Service/Listener
- ContextMenu
- DatePicker
- TimePicker
- ToogleSwitch
- WrapPanel

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
          <controls:Pivot Title="LONGLISTSELECTOR for Documents" >
            <controls:PivotItem Header="Bitte Documenttyp Wählen!">
//Hier wird die Liste definiert.
                <toolkit:LongListSelector
x:Name="documentypListFlatComposite" Background="Transparent"
IsFlatList="True"
              ItemTemplate="{StaticResource documenttypItemTemplate}"
                ListHeaderTemplate="{StaticResource
documenttypListHeader}"

         ListFooterTemplate="{StaticResource documenttypListFooter}"
ItemsSource="StaticResource Documenttyp" >
                </toolkit:LongListSelector>
            </controls:PivotItem>
          </controls:Pivot>
```

```
        </Grid>

    </Grid>
```

MainPage.cs

Hier wird einen WCF –Proxy erzeugt um, die Daten aus der Daten zu laden.

```
//Create Client Proxy

//Statt Localhost im Uri muss die IP –Address von der Maschine, auf welche der Webservice läuft, mitgegeben werden.

        DocumentServiceClient proxy = new DocumentServiceClient();

         proxy.Endpoint.Address = new System.ServiceModel.EndpointAddress(new Uri("http://localhost:81/Service1.svc", UriKind.RelativeOrAbsolute));


        //Load all Documenttyp from Database using WCF Service.


        proxy.DocumenttypGetAllCompleted += new EventHandler<DocumenttypGetAllCompletedEventArgs>(proxy_DocumenttypCompleted);

        proxy.DocumenttypGetAllAsync ();
```

- Die Daten (hier die Documenttyps aus der Datenbank) werden dem View übergegen.

```
void proxy_DocumenttypCompleted (object sender, DocumenttypGetAllCompletedEventArgs e)


        if (e.Result!= null)


            //Populate Documenttyp ListBox with data from Database.

//LongListSelector mit Documenttyps füllen

            documentypListFlatComposite.ItemsSource = e.Result;
```

- Sobald ein Documenttyp ausgewählt wird, müssen die zugehörigen Parameter angezeigt werden. Dafür muss eine Page-Navigation stattfinden, da die Parameter auf der ParameterViewPage angezeigt werden.
- Für die Auswahl von Objekten einer Liste wird ein Event Handler definiert, welcher gefeuert wird, wenn ein Objekt in der Liste ausgewählt wird.
  //Event Handler für die Liste „documentListFlatComposite“
  this.documentypListFlatComposite.SelectionChanged += new SelectionChangedEventHandler (documentypListFlatComposite_SelectionChanged);

Zu ParameterViewPage navigieren

Wählt der Nutzer einen Eintrag in der List aus, soll die App zur DetailsPage (ParameterViewPage) wechseln. Dies wird durch das SelectionChanged-Event der ListBox (documentypListFlatComposite).

Die Page-Navigation in Windows Phone 7 funktioniert über eine definiert *„NavigationService“-Klasse.* Die NavigationService-Klasse hat eine „Navigate“-Methode, welche die Page-Navigation über Uri delegiert.

Da eine Entität anhand des Primärschlüssels gelesen wird, muss die Id des gewählten Documenttyps als QueryString mitgegeben werden.

```
//d.DocumentID

NavigationService.Navigate(new Uri(string.Format("/ParameterViewPage.xaml?DocId={0}", d.DocumentID), UriKind.Relative));
```

ParameterViewPage.Xaml

Die XAML-Datei der ParameterViewPage definiert die Oberfläche für die DetailPage der MainPage. Die ParameterViewPage enthält weitere Informationen zum ausgewählten documentypListFlatComposite-Eintrag. Da wir zwei Typen von Parameter haben, beinhaltet die ParameterViewPage zwei Listbox-Controls. Für CheckboxParameter wird eine Checked-Liste definiert und für die DropDownParameter eine ComboBox-liste.

//CheckBoxParameterin „parameterItemTemplate” anzeigen

```
. <DataTemplate x:Key="parameterItemTemplate">
        <CheckBox>
          <CheckBox.Content>
            <StackPanel Grid.Column="1"  VerticalAlignment="Top">
          <Border Background="{StaticResource PhoneAccentBrush}"
                Margin="{StaticResource PhoneTouchTargetOverhang}"
                Padding="{StaticResource PhoneTouchTargetOverhang}">

              <TextBlock Text="{Binding ParameterName}" />



                </Border>



        </StackPanel>

        </CheckBox.Content>
      </CheckBox>
      </DataTemplate>

       //DropDownParameter in “parameterItemCombo” anzeigen
DataTemplate x:Key="parameterItemCombo">
        <Grid>
          <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto" />
            <ColumnDefinition Width="*" />
```

```
            </Grid.ColumnDefinitions>
          <TextBlock Text="{Binding Name}" Grid.Column="0" />
            <ComboBox Grid.Column="1" Foreground="Black"
ItemsSource="{Binding Values}" DisplayMemberPath="Value">
              <!--<StackPanel VerticalAlignment="Top">
                <Border Background="{StaticResource PhoneAccentBrush}"
                Margin="{StaticResource PhoneTouchTargetOverhang}"
                Padding="{StaticResource PhoneTouchTargetOverhang}">
                </Border>

              </StackPanel>-->
          </ComboBox>
          </Grid>


      </DataTemplate>
```

ParameterViewPage.cs

Diese Datei beinhaltet die ganze Logik für die ParameterViewPage. Folgende Schritte werden hier durchgeführt:

- Parameter eines ausgewählten Documenttyps aus der Datenbank laden
- Parameter nach ParameterType filtern
- ParameterValues für DropDownParameter aus der Datenbank laden.

Parameter eines ausgewählten Documenttyps aus der Datenbank laden

Um die Parameter zu laden, wird die DocumentID des ausgewählten Documenttyps benötigt. Diese ID wurde mit dem QueryString im Uri für die Page-Navigation zu der ParameterViewPage geschickt.
(NavigationService.Navigate(new Uri(string.Format("/ParameterViewPage.xaml?DocId={0}", d.DocumentID), UriKind.Relative));)

Die ParameterViewPage muss die OnNavigateTo-Methode überschreiben, um die DocumentID aus dem QuerySting zu lesen.

```
  protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {

            base.OnNavigatedTo(e);



            DocumentServiceClient proxy1 = new DocumentServiceClient();
            proxy1.Endpoint.Address = new System.ServiceModel.EndpointAddress(new Uri("http://localhost:81/Service1.svc", UriKind.RelativeOrAbsolute));
            proxy1.GetDocParamAllCompleted += new EventHandler<GetDocParamAllCompletedEventArgs>(proxy1_GetDocParamAllCompleted);


            string doctyp = "";
//Hier wird die DocumentId gelesen


            if (NavigationContext.QueryString.TryGetValue("DocId", out doctyp))
```

```
        {

            //this.DataContext = doctyp;
            ShardeId = doctyp;

            proxy1.GetDocParamAllAsync(doctyp);
        }
    }
```

Parameter nach ParameterType filtern

In der Method: void proxy1_GetDocParamAllCompleted(object sender, GetDocParamAllCompletedEventArgs e)

```
private void proxy1_GetDocParamAllCompleted(object sender, GetDocParamAllCompletedEventArgs e)
    {

        ObservableCollection<Parameter> values = e.Result;
```

- **ParameterType = 10 => CheckBoxParmeter**

```
        foreach (Parameter para in values)
```

```
{


    if (para.ParameterType == 10)
    {
      //CheckBoxParameterTyp  have no values
      //ParamerTyp = 10.

      _checkbox.Add(para);

      //Populate the Checked Listbox with CheckBoxParameterTyp
      parameterListFlatComposite.ItemsSource = _checkbox;




    Else
```

- **ParameterType = 20 => DropDownParameter**

```
    {
      if (para.ParameterType == 20)
      {

        _Combox.Add (para);
```

// Asynchrone-Aufrufe der Methode werden hier berücksichtigte, damit die Daten korrekt geladen werden können

```
          lock (paramLock)
```

```
                {
                    _names1.Add(new Hoi() { ParamId = para.ParameterID, Name
= para.ParameterName, Values = new List<Strings>() });
                }


                        //ParameterValue für DropDownParameter laden.
                client.GetParamValuesAsync(para.ParameterID);




                //Here bind the data to the combo box as DataContext
                parameterListFlatCombo.ItemsSource = _names1; //_Combox;
                parameterListFlatCombo.Tag = result;
```

Die ParameterViewPage beinhaltet noch ein Button-Control-Element (CreateDocument). Beim Anklicken des Buttons wird die Applikation zur EditorPage wechseln. Auf der EditorPage wird das geladene Document aus der Datenbank angezeigt. Die Entität Document wird anhand zwei Parameter aus der Datenbank gelesen: DocumentID (ID des ausgewählten Documenttyps), Sprache (die ausgewählte Sprache). Die ParameterViewPage muss dann beide Parameter der EditorPage übergeben.

Diese Logik wird im Event-Handler des CreateDocument-Buttons realisiert.

```
private void button1_Click(object sender, RoutedEventArgs e)
    {
```

```
            foreach (Hoi ho in parameterListFlatCombo.ItemsSource)
            {
//Dropdown Parameter : Sprache
// Aus dem ComboBox wird die gewünschte Sprache ausgewählt.

                if (((Hoi)parameterListFlatCombo.SelectedItem) == ho)
                {



                    List<Strings> res = ho.Values;
                    foreach(Strings str in  res)
//DocumentId wurde schon aus dem QueryString gelesen, und wird hier weitergeleitet.

                    NavigationService.Navigate(new Uri("EditorPage.xaml?DocId=" + ShardeId + "&Sprache=" + str.Value, UriKind.Relative));
                }


            }
```

EditorPage.xaml

Die XAML-Datei der EditorPage beinhaltet nur ein RichtextBox-Control. Hier wird ein C1RichtextBox von ComponentOne Studio® for Windows Phone (http://www.componentone.com/SuperProducts/StudioWindowsPhone/) verwendet. Es gibt Controls z.B. für WindowsForms, WPF, Silverlight und viele andere. Nun hat ComponentOne auch die CTP für Phone 7 Tools veröffentlicht. Diese beinhalten eine ganze Menge nützlicher Komponenten für die Phone 7 Entwickler.

Unter anderem gibt es:

- CHART
- COVER FLOW
- DOCK PANEL
- HYPER PANEL
- LAYOUT TRANSFORMER
- MAPS
- MASKED TEXTBOX
- NUMERIC TEXTBOX
- PDF VIEWER
- REFLECTOR
- RICHTEXTBOX
- UNIFORMGRID
- WRAPPANEL

Wie man anhand der Komponenten erkennen kann, handelt es sich wirklich um tolle Dinge, die jedem Entwickler das Leben erleichtern!

Um an diese CTP zu gelangen, muss man sich bei ComponenteOne kostenlos registrieren. Anschließend wird einem der kostenlose Download angeboten.

Editor

Im Editor sind drei Button-Controls für die Text-Formatierung definiert:

- **B** (Bold) :
- **I**(Italic)
- **U** (Underline)

EditorPage.cs

Die Logik für das Laden und Anzeigen von HTML-Dokumenten aus der Datenbank wird hier implementiert. Folgende Schritte werden durchgeführt:

- Document anhand DocumentId und Sprache aus der Datenbank laden (Inhalt des Dokuments)
- Inhalt des Document im Editor anzeigen lassen

Document anhand DocumentID(des ausgewählten Documenttyps) und Sprache aus der Datenbank laden (Inhalt des Dokuments)

- OnNavigatesTo-Mehode wieder überschreiben, um die Parameter aus dem QueryString zu lesen.

```
//protected override void OnNavigatedTo(NavigationEventArgs e)

//WCF proxy erzeugen

DocumentServiceClient proxy1 = new DocumentServiceClient();

        proxy1.Endpoint.Address = new
System.ServiceModel.EndpointAddress(new
Uri("http://localhost:81/Service1.svc", UriKind.RelativeOrAbsolute));

        proxy1.GetDocBySprCompleted += new
EventHandler<GetDocBySprCompletedEventArgs>(proxy1_GetDocBySprCom
pleted);


//DocumentId und Sprache aus dem QueryString holen


string doctyp = this.NavigationContext.QueryString["DocId"];

        string sprach = this.NavigationContext.QueryString["Sprache"];


//Document aus der Datenbank laden.


        proxy1.GetDocBySprAsync (doctyp, sprach);
```

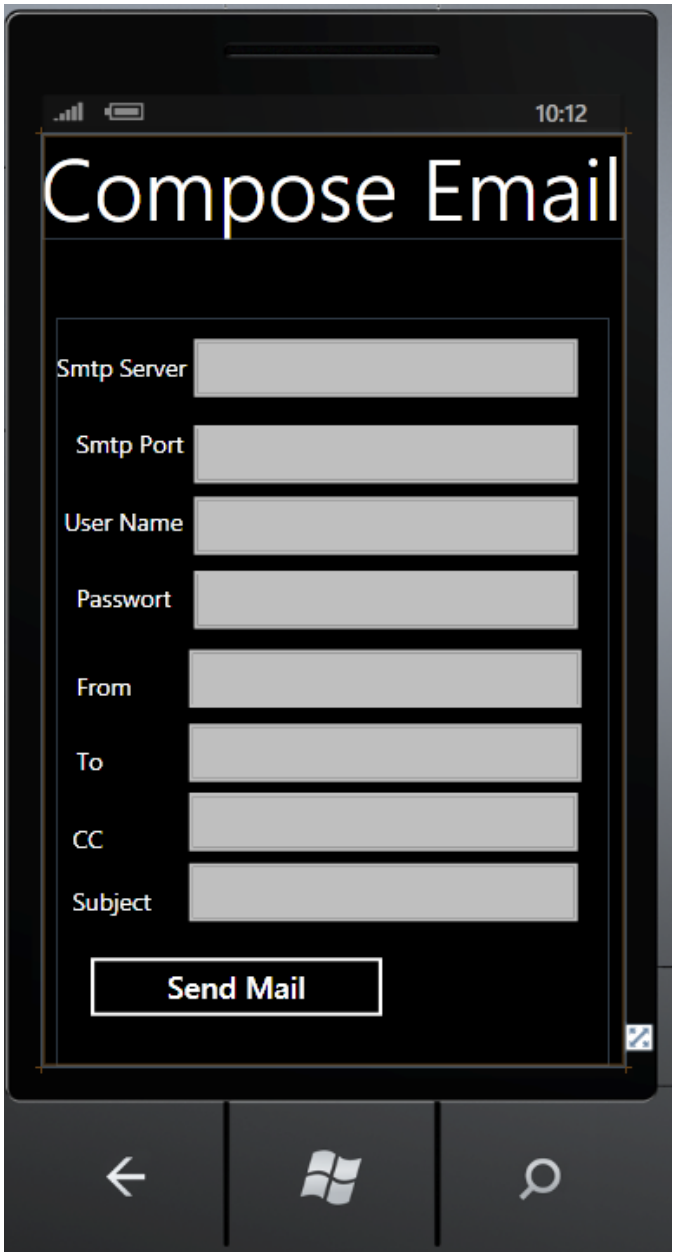
```
//private void proxy1_GetDocBySprCompleted(object sender,
GetDocBySprCompletedEventArgs e)

if (e.Result != null)


        Doc = e.Result;

//Inhalt des Dokuments im Editor anzeigen lassen


        this._rtb.Html = Doc;
```

Inhalts des Documents wird per Email verschickt. Dafür müssen die Daten an die EmailConfigPage weitergeleitet werden. Wenn der Nutzer im Editor das Button-Control „ComposeMail“ anklickt, so wird zu der EmailConfigPage navigiert.

EmailConfigPage.xaml (Konfigurationsseite für Email)

Die XAML-Datei der EmailConfigPage definiert die Oberfläche für die EmailConfigPage.

EmailConfigPage



EmailConfig.xaml

Diese Datei beinhaltet die Logik für das versenden von Emails via SmtpMailClient.Wenn der Nutzer das Button-Control „Send Mail“ anklickt, wird die Service-Methode(SendMailClient) aufgerufen und das Document per Email verschickt.

```
//private void button1_Click(object sender, RoutedEventArgs e)


//Email Input Information
        string _userName ="tafaba@gmx.de";////User
        string _passWort = "tafa2674";// passwordBox1.Password;//Passwort
```

```
            string _from = "tafaba@gmx.de";// fromtxt.Text; //from
            string _to = "mba@hsr.ch";// totxt.Text; //to
            string _CC = "kurtysde@yahoo.fr";// CCtxt.Text; // CC.
            string _smtpServer = "mail.gmx.net";// servertxt.Text; //host
            string _smtpPort = "25";// porttxt.Text; //port
            string _subject = "Test WP7 Mail";// subjtxt.Text;//subject
            string body = this.MessageBody; // Body


//Use SmtpClient from WCF Service to send Mail in WP7 Client.


            DocumentServiceClient client = new DocumentServiceClient();
            client.Endpoint.Address = new
System.ServiceModel.EndpointAddress(new
Uri("http://localhost:81/Service1.svc", UriKind.RelativeOrAbsolute));
            client.SendMailClientCompleted += new
EventHandler<System.ComponentModel.AsyncCompletedEventArgs>(client_S
endMailClientCompleted);
//WCF-Service Methode
            client.SendMailClientAsync(_userName, _passWort, _from, _to,_CC,
_smtpServer, _smtpPort,_subject, body);
```

Einführung in die Entwicklung für Windows Phone 7

Kurze Faktenübersicht

Entwicklungsumgebung ist Visual Studio 2010

- Entwicklungsframework ist ein Silverlight-Derivat genannt *„Silverlight for Windows Phone“*
- Phone-Emulator ist vorhanden
- Echte Phone-Geräte müssen für die Entwicklung registriert werden
- Probleme bei der Registrierung von Entwickler-Phones bei Studenten-Accounts
- App-Verteilung über Windows Phone Marketplace ist kostenpflichtig
- Keine private oder firmeninterne Verteilung von Apps möglich

Entwicklungsumgebung

Um mit der Entwicklung für Windows Phone 7 zu starten, genügt bereits die Installation der Windows Phone

Developer Tools, welche von Microsoft als kostenloser Download angeboten werden (create.msdn.com). Im

Installationspaket befinden sich unter anderem die folgenden Programme und Tools:

- Visual Studio 2010 Express
- Windows Phone Emulator
- XNA Game Studio 4.0
- Microsoft Expression Blend for Windows Phone

Falls bereits eine oder mehrere der enthaltenen Komponenten installiert sind, werden nur die zusätzlich

benötigtes Programm installiert. Die Windows Phone Developer Tools integrieren sich auch automatisch in eine

bereits vorhandene Installation von Visual Studio 2010.

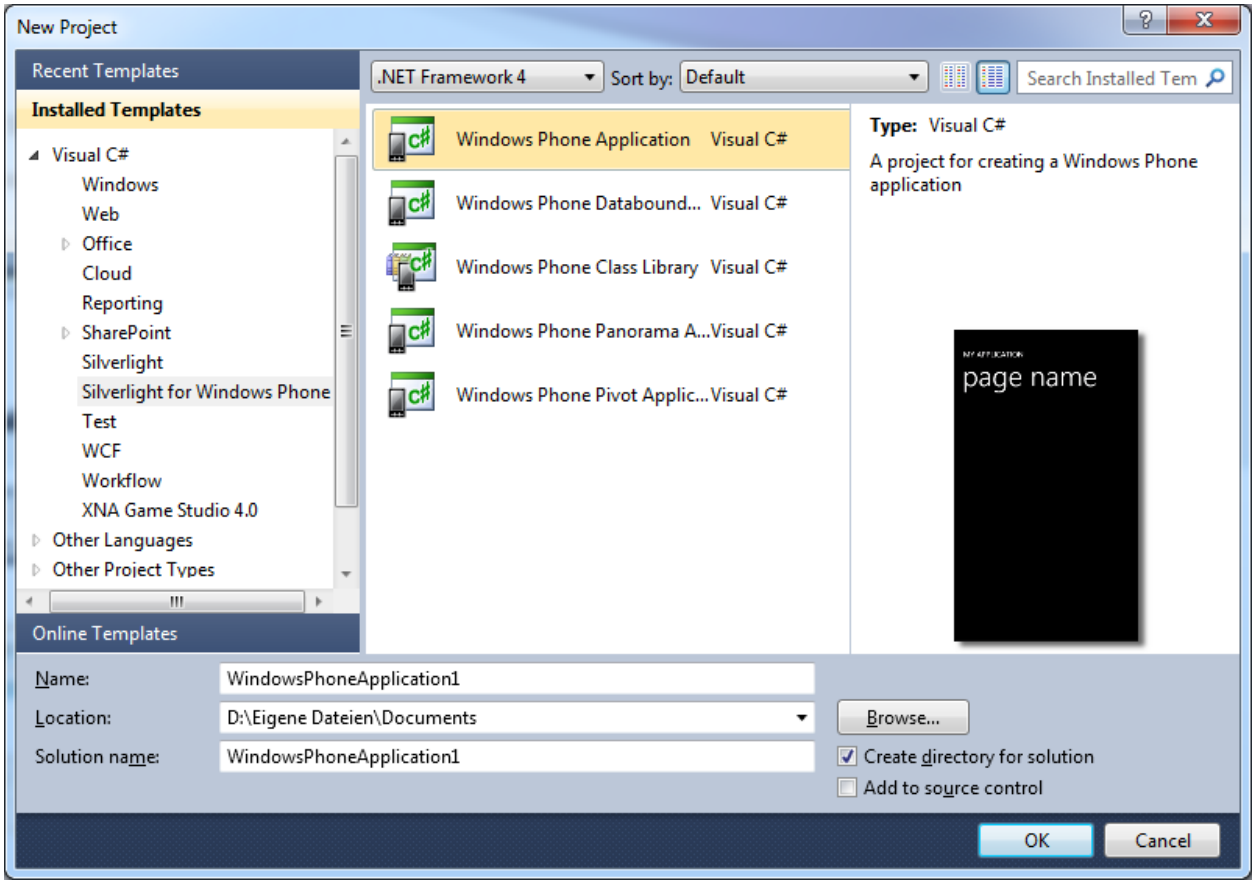


Abbildung 27: Windows Phone Template-Auswahl

Nach der Installation der Entwicklertools stehen die

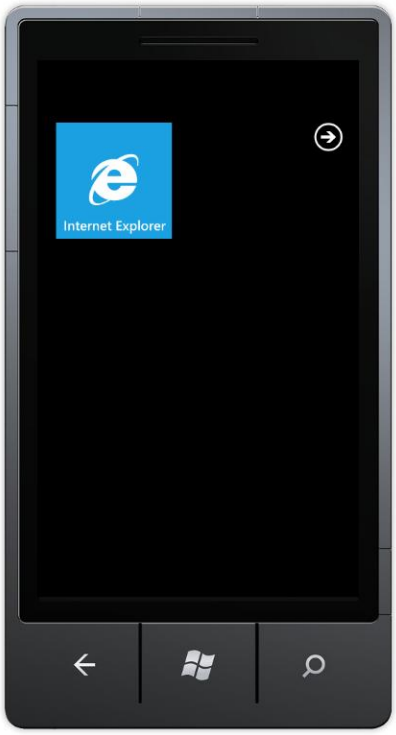
Windows Phone Projektvorlagen in Visual Studio

2010 zur Auswahl (sieh Abbildung links).

Das Framework für die Entwicklung für Windows

Phone 7 nennt sich *Silverlight for Windows Phone*.

Dieses basiert auf einer abgespeckten Version von Silverlight 3.0 mit spezifischen Erweiterungen für die Windows Phone Hardware. Die Laufzeitumgebung (CLR) ist aber nicht eine Silverlight-Portierung, sondern spezifisch für die Smartphone-Hardware optimiert, da diese ja ohne leistungsfähige PCProzessoren und Grafikkarten auskommen muss.



Windows Phone

Emulator

Im Visual Studio gibt es zwei auswählbare Ziele („Targets“) für die Ausführung bzw. das Debuggen von Applikationen. Davon ist eines der Windows Phone Emulator und das andere ein per USB angeschlossenes Windows Phone Gerät. Wenn der Emulator als Ziel ausgewählt ist, wird er automatisch gestartet, sobald man ein Windows Phone Projekt aus dem Visual Studio heraus ausführt (gilt für Release- und Debug-Version). Der Emulator ist Microsoft wirklich sehr gut gelungen. Es ist eigentlich alles so umgesetzt, wie es sich auf einem richtigen Gerät verhält. Die Bedienung der Apps auf dem Emulator erfolgt mit Maus und Tastatur. Der Emulator unterstützt aber auch echtes Multi-Touch, wozu aber natürlich ein entsprechender Touch-Monitor vorhanden sein sollte. Es gibt zwar spezielle Tools, mit denen sogar Multi-Touch mit

zwei angeschlossenen PC-Mäusen simuliert werden kann, aber die Möglichkeiten sind sehr beschränkt und auch etwas realitätsfern. Der Emulator erlaubt auch den Zugriffauf das Netzwerk und somit auch auf das Internet. Somit können Client-Server-Applikationen uneingeschränkt getestet werden. Hardwarespezifische Funktionen wie den Beschleunigungssensor oder die Kamera sowie Kommunikationsmöglichkeiten über Telefon, Email und SMS sind dagegen nur eingeschränkt oder gar nicht verfügbar. Um eine App auf

einem echten Smartphone zu testen, braucht es noch einige zusätzliche Schritte. Diese sind zusammen mit den Voraussetzungen im nächsten

App auf echtem Gerät testen (App Hub Account)

Um eine App auf einem echten Smartphone zu testen, muss das Phone zuerst als Entwicklergerät bei Microsoft registriert werden. Dies ist u.a.

eine Massnahme, um das Abspielen von „Raubkopien“ zu verhindern. Der Registrierungsprozess ist grundsätzlich sehr einfach gehalten. Dazu

muss ein kleines Registrierungstool gestartet werden, das bei der Installation der Windows Phone Developer Tools mitinstalliert wurde. In

diesem Registrierungstool muss nur das Login für den App Hub Account eingegeben werden. Danach wird das über USB angeschlossene Phone auf den angegebenen Account registriert und für die Entwicklung freigeschaltet. App Hub ist Microsofts neue Web-Plattform für die Entwicklung mit Windows Phone 7 und XBOX 360, und ist unter create.msdn.com erreichbar. Um den für die Registrierung des Phones benötigten App Hub Entwickler-Account zu erstellen, ist leider etwas Geduld gefragt. Für diesen Account muss nämlich eine jährliche Gebühr von $99 USD bezahlt werden. Zusätzlich wird von Microsoft eine Identitätsprüfung verlangt, die von der von Microsoft beauftragten Firma GeoTrust durchgeführt wird. Für die Validierung muss normalerweise ein Formular zusammen mit einem Identitätsausweis oder

Führerschein an GeoTrust geschickt werden. Der Validierungsprozess kann einige Tage in Anspruch nehmen, währenddessen keine Phones für die Entwicklung freigeschaltet werden können. Der App Hub Account wird erst freigeschaltet, wenn die Identitätsprüfung erfolgreich und die Gebühr von $99 USD bezahlt worden ist.

Entwickler-Phone Registrierung

App Hub Account für Studenten

Studenten profitieren vom Microsoft DreamSpark Programm (www.dreamspark.com), wodurch keine Kosten für

den App Hub Account anfallen. Dafür gibt es bei den Studenten-Accounts ein Problem bei der Registrierung von

Entwickler-Phones. Die Ursache liegt darin, dass die Validierung von Studenten-Accounts anders abläuft, als bei

den bezahlten Accounts. So wird der Identitätsprüfungsvorgang normalerweise gleich bei Erstellung des Accounts

initiiert. Bei Studenten erfolgt dies aber erst mit dem Einsenden der ersten App für den Marketplace. Da bei der Erstellung des Accounts seitens Microsoft aber nicht genügend auf diesen Ablauf hingewiesen wird, verbleibt man in der Annahme, der Account würde automatisch freigeschaltet. Sobald man aber versucht, ein Phone als Entwicklergerät zu registrieren, scheitert dieser Vorgang mit einer kryptischen Fehlermeldung. Erst durch eine Anfrage beim Microsoft

Support, konnte der Missstand aufgeklärt werden. Der Microsoft Support kann den

Identitätsprüfungsvorgang aber nicht anstossen. Er rät dazu, eine Dummy-App zur Veröffentlichung auf dem Marketplace einzusenden, welche dann automatisch den Identitätsprüfungsvorgang initiiert. Dieses Vorgehen funktioniert tatsächlich. Die Dummy-App wird bei der Validierung auch umgehend zurückgewiesen und somit nicht im Marketplace veröffentlicht. Dass zuerst eine App eingesendet werden muss, bevor auf einem Phone getestet werden kann, ist als Fehler im Prozess anzusehen. Hier sollte Microsoft unbedingt nachbessern und zumindest die Benutzer besser informieren.

Verteilung der Apps über Windows Phone Marketplace

Apps können nur auf für die Entwicklung registrierten Geräten direkt aufgespielt werden. Ansonsten wird die Verteilung der Apps ausschliesslich über den Windows Phone Marketplace abgewickelt. Das während dieser Studienarbeit entwickelte *Windows Mobile Dynamic Document Creation System* wurde nicht im Marketplace veröffentlicht. Die Gründe dafür sind, dass es einerseits nur ein Prototyp ist und andererseits eine firmeninterne Anwendung, welche der Öffentlichkeit nicht zugänglich sein sollte. In diesem Punkt gibt es noch einen gewissen Aufholbedarf seitens Microsoft, da zurzeit keine Möglichkeit besteht, Apps nur firmenintern zu verteilen. Bei der Konkurrenz von Apple wird die firmeninterne Verteilung, als Teil eines grösseren Leistungspakets, gegen eine Gebühr von $299 USD im Jahr angeboten. Apps, welche für die Veröffentlichung auf dem Marketplace eingesendet werden, durchlaufen einen Validierung und Zertifizierungsprozess. Microsoft hat dazu auf dem App Hub eine Anleitung veröffentlicht, welche den Prozess erklärt und alle Anforderungen an die einzusendenden Apps beschreiben. Die Apps werden erst nach erfolgreichem Bestehen aller unterzogenen Prüfungen im Marketplace veröffentlicht.

Zusätzliche Tools und Libraries

Da Silverlight for Windows Phone auf Silverlight 3.0 basiert, sind viele hilfreiche und dringend benötigte

Funktionen, welche in Silverlight 4.0 zur Verfügung stehen, noch nicht vorhanden. Das bedeutet, dass der Einsatz von zusätzlichen Libraries und Frameworks fast unumgänglich ist. Im Folgenden werden zwei wichtige Zusatz-Libraries kurz vorgestellt, die auch in dieser Bachelorarbeit verwendet wurden.

Silverlight for Windows Phone Toolkit

Das *Silverlight for Windows Phone Toolkit* ist von Microsoft selbst entwickelt worden. Es beinhaltet vor allem

zusätzliche Controls sowie Komponenten für eine vereinfachte Handhabung von *Gestures*. Es wird regelmässig

aktualisierte Versionen auf silverlight.codeplex.com veröffentlicht.

Prism - Windows Phone 7 Developer Guide

Prism für Windows Phone 7 ist als Teil des *Windows Phone 7 Developer Guide* verfügbar, welcher unter dem

Microsoft patterns & practices Label veröffentlicht wurde. Prism besteht aus einer Library sowie Beispielprojekten

und Hilfestellungen für die Entwicklung unter Windows Phone 7. Der *Windows Phone 7 Developer Guide* kann von

wp7guide.codeplex.com heruntergeladen werden.

Mit der folgenden Zeile wird eine neue Instanz der *ViewModelLocator*-Klasse erstellt und als

applikationsweite Ressource verfügbar gemacht:

```
<Application.Resources>

<presentation:ViewModelLocator x:Key="ViewModelLocator"/>

</Application.Resources>
```

Asynchrone Service-Architektur

Bei den meisten Client-Server-Architekturen sind Client und Server einfach zu kontrollieren, aber bei dem was dazwischen liegt, der Datenverbindung, gibt es nur wenig Einflussmöglichkeiten. So kann die Verbindung jederzeit gestört, kurzzeitig unterbrochen oder auch für immer unterbrochen werden. Um mit diesen Situationen möglichst vernünftig umzugehen, ohne dem Benutzer das Leben schwer zu machen, wird eine asynchrone Kommunikation verwendet. Damit das aus der Perspektive des Softwaredesigns auch handhabbar ist, wird

eine asynchrone, serviceorientierte Architektur verwendet. Mit dieser werden die REST- bzw. HTTP-spezifischen Kommunikationseigenheiten vor den Service-Konsumenten vollständig versteckt. Die Benutzer eines Services entsprechen in der WP7TestClient-App den *ViewModel*-Klassen. In diesen wird die Benutzerinteraktion in Service-Anfragen umgewandelt, welche fast immer in einer Kommunikation mit dem Server enden. Damit die Benutzung der Services für die *ViewModel*-Klassen so einfach wie möglich ist, wird das bekannte Konzept eines Service-Locators eingesetzt. Um eine Instanz eines bestimmten Services zu erhalten, wird nur eine einzige Zeile Code benötigt:

```
IGatewayAuthentication authenticationService =
Context.ServiceLocator.FindService<IGatewayAuthentication>();
```

Der Aufruf einer Servicemethode sieht dann z.B. folgendermassen aus:

```
authenticationService.Authenticate(UserName, Password,
AuthenticateCallback);
```

Das Methodenargument "*AuthenticateCallback*" ist ein Delegate auf eine Methode, welche bei der Rückgabe des

Resultates aufgerufen wird:

```
private void AuthenticateCallback(AuthenticateEventArgs e)
{
if (e.ResultState.State == OperationState.Successful)
{
// Successfully logged in, do work...
}
else
{
// Error occurred:
MessageBox.Show(e.ResultState.Error.ToString());
}
}
```

Diese drei Schritte bleiben bei der Benutzung aller Services die gleichen:

1. Instanz des entsprechenden Services über den *ServiceLocator* holen
2. Servicemethode aufrufen und eine Callback-Methode angeben
3. Rückgabe des Resultates in der Callback-Methode verarbeiten

Dabei ist nie klar, wie viel Zeit zwischen Serviceaufruf und Rückgabe des Resultates vergeht. Dies muss bei der

Gestaltung sowie dem Verhalten der Benutzeroberfläche berücksichtigt werden.

Der Vorteil an diesem Vorgehen ist, dass die Benutzeroberfläche nie „einfrieren“ kann, wenn eine Serviceanfrage

etwas länger dauert oder im Fehlerfall ein extrem langes Timeout auftritt. Auch können Fehler so viel schöner

behandelt werden und müssen nicht überall mit *Try-Catch*-Blöcken abgefangen werden.

Internationalisierung (I18N)

Statische Texte in der Benutzeroberfläche sind direkt im XAML-Code geschrieben. Da die WP7TestClient-App als Prototyp

entwickelt wurde, war der Fokus nicht auf einer sauberen Unterstützung einer Mehrsprachigen

Benutzeroberfläche. Diese Texte in eine Ressourcen-Datei (*.resx) auszulagern, wäre aber gut umsetzbar.

Im Gegensatz dazu sind Texte, welche als dynamisch generierter Inhalt auf der Benutzeroberfläche dargestellt

werden, bereits in einer Ressourcen-Datei abgelegt. Bei diesen Texten handelt es sich mehrheitlich um

Übersetzungen für Enum-Werte der Datenpunkte. Bei diesen Enumerationen handelt es sich zum Beispiel um

Einheiten wie C°, km/h, m², etc. oder auch um Signal- oder Fehlerzustände usw. Die Texte für diese

Enumerationen werden ebenfalls in der Benutzeroberfläche des Kontrollersimulators benötigt. Daher ist die

Ressourcen-Datei Teil der Common-Bibliothek.

Leistungsfähigkeit (Performance)

Microsoft stellt mit Windows Phone 7 relativ hohe Leistungsanforderungen an die Smartphone-Hersteller, um

eine möglichst schnelle und verzögerungsfreie Bedienung zu garantieren. Man merkt tatsächlich schon bei der

erstmaligen Verwendung eines Windows Phone 7-Gerätes, dass das Betriebssystem speziell auf Performance

optimiert wurde. Alle Systemfunktionen sowie die von Microsoft erstellten Apps reagieren extrem schnell und es

gibt weder Ruckeln noch Ladebildschirme oder Sanduhren. Bei Apps von Drittherstellern sieht dies hingegen ganz

anders aus. Oft kommen Ladebildschirme mit beträchtlichen Wartezeiten vor oder beim Scrollen von längeren

Listen stockt der Bildaufbau. Schon hier zeigt sich also, dass es nicht ganz einfach ist, eine App zu entwickeln, die

über eine ansprechende Optik verfügt und zugleich eine störungsfreie Benutzung erlaubt.

Windows Phone Ausführungsmodell (Execution Model)

Das Ausführungsmodell von Windows Phone 7 erlaubt generell kein Multitasking. Das bedeutet, dass nur eine

Applikation gleichzeitig im Arbeitsspeicher geladen sein darf. Nur gewisse Systemprogramme und Microsoft Apps,

wie z.B. die Musikwiedergabe, dürfen beim Starten einer anderen Applikation im Hintergrund weiterlaufen. Das

Ausführungsmodell zusammen mit einem ausgeklügelten Bedienkonzept lassen den Benutzer aber im Glauben,

dass trotzdem mehrere Applikationen gleichzeitig laufen.

So wird jeweils beim Navigieren durch die einzelnen Seiten einer App, jede Seite auf einen Stapel abgelegt. Durch

Drücken der Starttaste kann eine weitere Applikation gestartet werden, wobei die erste App deaktiviert und aus

dem Speicher entfernt wird. Wenn der Benutzer nun in der aktiven App weiter navigiert, werden wiederum alle

besuchten Seiten auf den Stapel abgelegt. Das geht so weiter, bis der Stapel gefüllt ist und die zuerst geöffneten

Seiten zuunterst wieder aus dem Stapel herausfallen. Durch Drücken der Zurücktaste wird nun die letzte Seite

vom Stapel genommen und wiederhergestellt. Sobald der Benutzer durch wiederholtes Drücken der Zurücktaste

zur letzten Seite der ersten App zurückkommt, wird die zweite App endgültig beendet und die erste App wieder in

den Speicher geladen und aktiviert.

Der Vorgang, bei dem eine App deaktiviert und aus dem Speicher entfernt wird, wird als Tombstoning bezeichnet.

Das „Back-Button“-Konzept, bei dem der Navigationspfad über mehrere Apps hinweg auf einem Stapel abgelegt

wird und beim Drücken der Zurücktaste wiederhergestellt wird, lässt den Benutzer im Glauben, dass die

vorhergehend aufgerufenen Apps die ganze Zeit im Hintergrund weitergelaufen sind. Dabei überlässt es Microsoft

dem Anwendungsentwickler, bei der Deaktivierung der App, den Applikationszustand zu speichern und bei der

erneute Aktivierung, den vorhergehenden Zustand wiederherzustellen. Das gesamte Konzept für die Speicherung und Wiederherstellung des Applikationszustandes ist relativ umfangreich und es sind einige Spezialfälle zu beachten. Microsoft stellt ein paar rudimentäre Hilfsmittel zur

Verfügung, welche die Handhabung etwas erleichtern. Grundsätzlich muss zwischen der Speicherung des Applikations- und des Seitenzustandes unterschieden werden. Mit ersterem ist der globale Applikationskontext gemeint und mit letzterem der Zustand der einzelnen Controls auf der Benutzeroberfläche. Die Zustände der Controls sind z.B. der Inhalt einer TextBox, die Position einer Scrollbar oder das aktuell fokussierte Control. Bei

der SCT-App wird zurzeit nur der Applikationszustand gespeichert. Da der gesamte applikationsweite Zustand

in der ApplicationContext-Klasse enthalten ist, gestaltet sich dessen Speicherung und Wiederherstellung sehr einfach. Das Speichern und Laden der Seitenzustände wurde aus Zeitgründen nicht implementiert. Dies ist auch nicht weiter kritisch, sondern eher als ein „Schönheits-Fehler“ anzusehen. Applikationszustand sowie Seitenzustand sind als flüchtige Daten zu bezeichnen, die nur wieder geladen werden,

falls die Applikation reaktiviert wird. Diese Daten überleben eine endgültige Beendigung der Applikation nicht und werden gelöscht. Daten, welche die Beendigung einer Applikation überleben und bei einer frisch gestarteten Applikation wieder vorhanden sein sollten, müssen im *IsolatedStorage* gespeichert werden. Dies wird beim WP7TestClient z.B. mit den Login-Daten so gemacht.

Benutzeroberfläche (UI)

Kurze Erklärung der benutzten Gesten bei der Touch-Bedienung:

- **Antippen (Tap)**

Die Geste „Antippen“ ist eine kurze Berührung des Bildschirms mit einem Finger in einem definierten

Bereich.

- **Flick-Bewegung (Flick)**

Die Flick-Bewegung ist eine Berührung mit einem Finger und eine schnelle Bewegung nach links oder

rechts.

- **Halten (Hold)**

Die Geste „Halten“ ist eine lange Berührung ohne Bewegung in einem definierten Bereich.

Application Bar

Die *Windows Phone Application Bar* ist sozusagen das Pendant zur Menüleiste oder Toolbar einer

Desktopanwendung. Die *Application Bar* wird zuunterst auf dem Bildschirm angezeigt und wird vollständig von

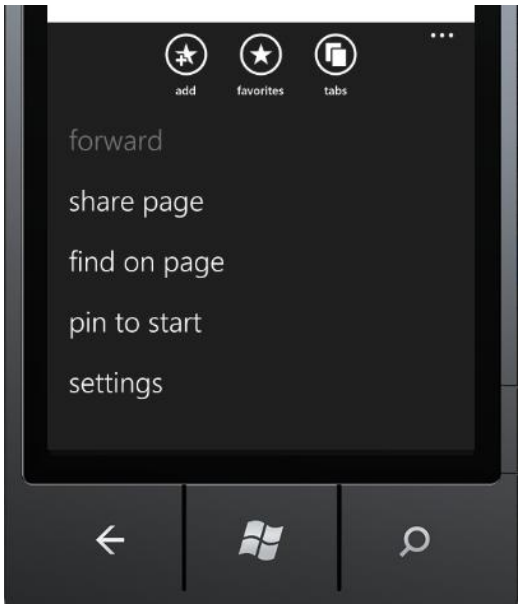
der aktiven App kontrolliert. Es finden bis zu vier Icons auf der *Application Bar* Platz. Zusätzlich existiert noch ein

erweitertes, rein textuelles Menü, das von der *Application Bar* aus aufgerufen werden kann und bis zu 50 weitere

Menüpunkte aufnehmen. Es besteht jederzeit die Möglichkeit, die *Application Bar* sowie einzelne Icons bzw.

Menüpunkte ein- oder auszublenden. Die folgenden Screenshots stammen von der Windows Phone Internet

Explorer App.



Es empfiehlt sich, die *Application Bar* als möglichst statisches Element zu verwenden, da häufiges Ändern der

Icons den Benutzer leicht verwirren könnte.Da die *Application Bar* ein Systemobjekt und nicht direkt Teil des XAML-*Visual-Trees* ist, gilt es ein paar Besonderheiten zu beachten. Obwohl die *Application Bar* im XAML-Code definiert werden kann, funktioniert weder *Databinding* noch Command-Binding. Anstatt dessen ist ein Workaround vorhanden, bei dem die *ApplicationBarButtonCommand*-Klasse der Prism-Library verwendet wird. So können die *Application Bar* Buttons an *Commands* gebunden werden.

Kontextmenü

Obwohl in den Microsoft-eigenen Apps ein Kontextmenü verwendet wird, ist es in den Standardbibliotheken von

Silverlight for Windows Phone nicht enthalten. Dafür wird dieses wichtige Bedienelement im *Silverlight for*

Windows Phone Toolkit nachgeliefert, welches Aussehen und Verhalten des Kontextmenüs aus den Microsofteigenen

Apps imitiert. Bei der Benutzung des Kontextmenüs tritt ein Problem auf: Sobald ein Menüpunkt per Tippen angewählt wird,

wird der Tipp-Befehl auch an das dahinterliegende Bedienelement gesendet.Die Ansichten reagieren also ganz

normal auf Tipp-Befehle, wie wenn gar kein Kontextmenü geöffnet wäre. Dies scheint ein Fehler in der

Implementation des Kontextmenüs oder des verwendeten *Toolkit-Gesture-Listeners* zu sein, welche auf dieselben

Bewegungsmuster reagieren. Aber möglicherweise ist dies auch Auslegungssache, weil diese Bedienelemente

vielleicht gar nicht für eine gemeinsame Benutzung ausgelegt sind

Sobald das Kontextmenü geschlossen wird, was z.B. bei der Auswahl eines Menüpunktes geschieht, wird das Flag nach einem Timeout von 150ms wieder gelöscht:

```
private void ControllerContextMenu_Opened(object sender, RoutedEventArgs
e)

{

IsContextMenuOpened = true;

}

private void ControllerContextMenu_Closed(object sender, RoutedEventArgs e)

{

// Use a timeout of 150ms before resetting the flag to prevent other

// action listeners from being executed after clicking a context menu item.

new Thread(() =>

{

Thread.Sleep(150);
```

```
IsContextMenuOpened = false;
}).Start();
}
```

Windows Phone Themes

Auf dem Windows Phone 7 gibt es Einstellungen, um die Farben für die Darstellung den eigenen Vorlieben anzupassen. Grundsätzlich sind zwei *Themes* vorhanden. Ein dunkles, bei dem die Grundfarbe Schwarz ist und ein helles, welches Weiss als Grundfarbe hat. Zusätzlich zur Grundfarbe kann eine Akzentfarbe definiert werden. Die Akzentfarbe wird für Bereiche/Texte benutzt, welche selektiert sind oder grafisch hervorgehoben werden sollen.

List Picker

Das Combobox-Bedienelement wird auf dem Windows Phone nicht angeboten, da es aus der Welt der Desktopanwendungen stammt und nicht für die Touch-Bedienung ausgelegt ist. Trotzdem bleibt natürlich die eigentliche Problemstellung auf dem Smartphone dieselbe: Wie kann der Benutzer aus einer gegebenen Liste von Elementen eines auswählen, ohne dass das Bedienelement viel Platz benötigt. Als Lösung bietet das *Silverlight for Windows Phone Toolkit* ein *List Picker*-Bedienelement an. Der *List Picker* zeigt normalerweise nur das ausgewählte Element an. Mit Antippen wir der *List Picker* zu einer vollwertigen Liste vergrössert, woraus ein anderes Element durch ein weiteres Antippen „herausgepickt“ werden kann. Danach wird die Liste wieder verkleinert und es wird nur noch das neu ausgewählte Element angezeigt. Der *List Picker* zeigt dieses Verhalten jedoch nur bei weniger als sechs Elementen. Bei mehr Elementen wird das Bedienelement nicht einfach vergrössert, sondern es erscheint ein bildschirmfüllendes Popup, das eine scrollbare Liste der Elemente zeigt. Sobald ein Element ausgewählt wurde, schliesst sich das Popup wieder. Der *List Picker* wurde erst in der November-Aktualisierung des *Silverlight for Windows Phone Toolkits* veröffentlicht und scheint noch ein paar Bugs zu haben. So führt z.B. das *Databinding* in gewissen Situationen zu Fehlern, aber nur wenn der Visual Studio Debugger nicht angeheftet ist.

Schlussfolgerung

Zusammenfassung

Diese Studienarbeit hat zum Ziel die Entwicklung eine Windows Phone 7 App, welche Emails in HTML-Format versenden kann. Nach der Analyse der Anforderungen und der bestehenden Technologien ist mir klar geworden, dass es nur um einen Prototype handeln soll. Ein realistisches Konzept der Lösung wurde erstellt. Windows Phone 7 Built-In Email-Client kann keine HTML-Emails versenden, aber es ist doch möglich irgendwie HTML-Email im WP7 zu versenden.

Beurteilung der Resultate

Alle funktionalen Anforderungen wurden erfüllt. Die WMD2C-Anwendung macht folgende Funktionalitäten verfügbar:

- Verfügbare Dokumentyps aus der Datenbank laden und anzeigen
- Dazugehörige Parameter eines ausgewählten Documenttyps aus Datenbank laden und anzeigen.
- Parameter nach ParameterType filtern : CheckboxParameter, DropdownParameter
- Dokument anhand Sprache und Documenttyp-ID aus der Datenbank laden und anzeigen.
- HTML-Dokument per Email versenden wie z.B :

Hallo **BOLD** roter text „ e

Ausblick

Die WMD2C-App eröffnet neue Wege für Entwicklung von Windows Phone 7 App. Dynamic Document Creation System in Windows Phone 7 kann mit WMD2C weitere Innovationen in die WP7-Entwicklung bringen. WMD2C kann noch weiterentwickelt werden ,um ein besseres Produkt auf dem Markt bringen zu können.

Projektplan

EMAIL CLIENT DYNAMIC DOCUMENT CREATION

Studienarbeit :	**Integration eines Dynamic Document Creation System in einen Windows Phone 7 Mail Client**
Autor :	**Ba Mohamedou Moustapha**
Verantwortlicher :	**Prof. Hansjörg Huser**
Betreuer:	**Prof. Hansjörg Huser**
	Herr Simon Baer
Industriepartner:	**Sevitec AG, Eschlikon TG**

Dokumentinformationen

Änderungsgeschichte

Datum	*Version*	*Änderung*	*Autor*
28.02.2011	0.1	Meilensteine, Iterationen (Tabellen)	mba
01.03.2011	0.2	Riskomangement,Management Abläufe,Infrastruktur,Qualitätsmanagement, Arbeitspakete Dokument-Layout	mba
23.03.2011	0.3	Meilensteine ergänzen, Dokument bearbeiten	
09.06.2011	1.0	Änderungen in der Planung anpassen	mba

Inhalt

Einführung

Zweck

Dieses Dokument beschreibt die Projektplanung für das Projekt **WMD2C**

Gültigkeitsbereich

Dieses Dokument gilt als Grundlage für das ganze Projekt **WMD2C** und hat deshalb Gültigkeit über die gesamte Projektdauer.

Definitionen und Abkürzungen

Beschrieben in *XX_Glossar/Glossar.doc*

Referenzen

01_Projektantrag/Projektantrag.doc

02_Projaktplan/Projektplan.xls

02_Projektplan/Risikomanagement.xls

SE2 Projekt Dokumentvorlagen

Rudin, H. (02. Februar 2011). MS1ReviewProjektplanungCheckliste1-0. Rapperswil, St.Gallen: HSR.

Rudin, H. (02. Februar 2011). SE2Projekt-FS11-Anleitung. Rapperswil, St.Gallen: HSR.

Übersicht

Das nachfolgende Kapitel gibt eine kurze Beschreibung des Projekt WMD2C, beschreibt auch Sinn und Zweck sowie die Zielsetzung des Projekt WMD2C. Anschliessend werden in weiteren Kapiteln die Projektorganisation, Management Abläufe, Risikomanagement, Arbeitspakete, Infrastruktur sowie die Qualitätsmassnahmen beschrieben.

Projekt Übersicht

Das Projekt WMD2C beschäftigt sich mit der Integration eines dynamischen Dokument Erstellungssystem in einen Windows Mobile 7 Mail Client. Es soll einen HTML-Mail Client als WP7 App entwickelt werden.

Definitionen und Abkürzungen

Abkürzung	Beschreibung
WP7	**W**indows **P**hone **7** : Neuansatz einer mobilen Plattform von Microsoft für den Consumer- und Business-Bereich
SWO	**S**emester**w**oche (Beginn mit Frühjahrsemester am 22.02.2011)
WMD2C	**W**indows **M**obile **D**ynamic **D**ocument **C**reation: : Name der WP7 Email app
HSR	Hochschule für Technik Rapperswil
App	Kurzform für Application : ist im Allgemeinen jede Form von Anwendungsprogrammen

Zweck und Ziel

Der Industriepartner Sevitec AG stellt Zusatzsoftware-Produkte zu Microsoft Office(www.idfix.ch) her, womit Kunden in allen Branchen als Alternative zu selbstprogrammierten Office-Templates auf effiziente Art Geschäftsdokumente und Emails in perfektem Corporate Identity erzeugen können. Dazu gehört auch das Erstellen von formatierten HTML-Emails in Microsoft Outlook, mit Profil-, Sprach- und Zielgruppen-abhängigen Signatur.

Ziel diese vorliegende Arbeit ist die Integration dieser Funktionalität in Smartphones unter Windows Mobile 7.

Annahmen und Einschränkungen

Das Projekt WMD2C wird im Rahmen einer Studienarbeit im Frühjahrsemester 2011 durchgeführt und darf eine Dauer von 14 Wochen nicht überschreiten. Der Projektplan muss über die Dauer des Frühjahrsemester (14 Wochen) entsprechend erstellt werden.

Pro Woche wird eine Soll Arbeitszeit von Sechzehn Stunden erwartet. Treten unerwartete Aufwände oder Probleme auf, kann die Arbeitszeit auf maximal 10 Stunden erhöht werden.

Projektorganisation

Diese vorliegende Arbeit wird als Einzelarbeit durchgeführt. Das Projektteam besteht aus nur einer Person, die für alle Aufgabe im Projekt WMD2C zuständig ist. Herr Huser ist verantwortlich für das Projekt WMD2C und ist für die Betreuung der Arbeit zuständig.

Organisationsstruktur

M.M. Ba.

Person	Kürzel	Zuständigkeit
Ba Mustafa	mba	Realisierung des Projektes

Externe Schnittstellen

Person	Kürzel	Zuständigkeit
Ba Mustafa	mba	Realisierung des Projektes

Hansjörg Huser	HH	Dozent an der HSR und Leiter vom Institut INS, Beratung, Betreuung, Kontrolle(Reviews),Bewertung
Patrik Dietschweiler	PD	Software-Entwickler im Institut INS an der HSR, Technische Probleme im Bereich Windows Phone 7 Entwicklung
Simon Baer	SB	Software-Entwickler bei der Firma Sevitec AG, Betreuung des Projektes in Eschlikon

Management Abläufe

Projekt Kostenvoranschlag

Für die Realisierung des Projektes stehen pro Woche 16 Stunden zur Verfügung. Der Projektstart ist am 21 Februar 2011 d.h Beginn des Frühjahrsemesters an der Hochschule für Technik Rapperswil (HSR) und das Projektende am 3 Juni 2011. Falls unerwartete Probleme auftreten sollen, kann der Arbeitsaufwand pro Woche um 10 Stunden erhöht werden.

Projektplan

Zeitplan

Siehe separates Dokument im Ordner: 02_Projektplan/Zeitplan.xls

Meilensteine

Meilenstein	Geplant	Erreicht	Artefakte

MS1:Projektplan	07.03.2011	07.03.2011	• Projektplan • Zeitplan
MS2:Anforderungen und Analyse	18.03.2011	30.03.2011	• Anforderungspezifikation • (nichtfunktionale Anforderungen, UseCases) • Domain Modell
MS3: Prototyp	04.04.2011	06.04.2011	• Architekturprototyp (Smatphone-seitig,Serverseitig) “proof of concept” • Client/Server Architektur Entwurf
MS4: Konzept Analyse	18.04.2011	19.04.2011	• Alpha Release • Konzept Analyse • Webservice Interfaces • Design Entwurf
MS5: Design	02.05.2011	02.05.2011	• Beta Release • Software Architektur und Design Spezifikation • UI Design Studie
MS6: Final	16.05.2011	20.05.2011	• Finale Release • Technischer Bericht Entwurf
MS7: Demo	26.05.2011	23.05.2011	• Email –Demo mit dem Produkt WMD2C •
MS8: Abgabe	03.06.2011	10.06.2011	• Technischer Bericht • Abgabe von ProjektDokumente (CD und Papierform)

Besprechungen (Meetings)

Pro Woche finden zwei Meetings statt. Vorgesehen sind ein kurzes Meeting am Dienstag sowie ein umfangreiches jeweils am Freitag. Die Meetings dienen

dazu, die anstehenden Arbeiten sowie anfallende Probleme zu besprechen. Während den Besprechungen wird Protokoll geführt.

Besprechung	Teilnehmer	Datum/Zeit	Ort
Mit Betreuer	mba,PD,HH	Mittwochs, 9.30 -10.30	HSR
Mit Industriepartner	SB,mba	Donnerstags,9.30 – 10.30	Sivitec AG Eschlikon TG

Iterationen

Die nachfolgende Tabelle zeigt eine Übersicht über die verschiedenen Iterationsphasen und deren Inhalt. Genauere Informationen bezüglich der zeitlichen Abläufe können dem Zeitplan entnommen werden.

Iteration	Beschreibung	Ende	Dauer in Woche
Inception	• Projektantrag, • Einarbeitung in der Windows Phone 7 Technologie	SWO1	1

	• Analyse vom Kontext		
Elaboration 1	• Dokumentvorlagen, • Projektplan(Zeitplan, Risiko ,Konfigurationsverwaltung • Domain Model, Entwurf des externen Designs • Konzeptioneller Design, Analyse der Anforderungen und der zur Verfügung stehenden Technologien, • Schnittstellenspezifikation der Lösung	SWO3	2
Elaboration 2	• Design Model, • Logische Architektur der Packages, • 100% Use Cases im Fully dressed Format(zumindest der nichtoptionalen Features, • Korrektur des Externen Design, • Client/Server Analyse incl. Funktions-Prototyps aus verschiedenen Teilsystemen, • internes Design des UI incl. Prototyp einer graphischen Oberfläche.	SWO5	2
Construction 1	• Prototyp (Serverseitig, Smartphone-seitig und evtl. Webservices), • Ausbau Server /Windows Mobile 7 Mail Client –Kommunikation, • Projekt Automation, • Prototyp des UI weiter ausbauen.	SWO7	2
Construction 2	• Einfaches Dokument dynamisch im Windows Mobile 7 Mail Client erzeugen. • Funktionale Anforderungen des Systems implementieren (wie Erstellen von formatierten HTML-Mails in Microsoft Outlook, mit Profil-, Sprach- und Zielgruppen-abhängigen Signaturen.	SWO10	3
Construction 3	• Erstellen eines dynamischen Dokument mit WMD2C, • Implementierung zusätzlicher Features, • Bugfixing	SWO12	2

Transition 1	• Integration von WMD2C in einem Windows Phone 7, • Vorbereitung Schlusspräsentation und Abgabe	SWO14	2

Software Releases

Datum	Beschreibung	Release
04.04.2011	Architekturprototyp über alle Layer	• Prototyp
18.04.2011	Grundfunktionalität:HTML-Email mit Logo	• Alpha
16.05.2011	Voller Funktionsumfang(alle Pflicht Anforderungen), Systemtest erfolgreich durchgeführt	• Beta
26.05.2011	Email-Demo	• Final

Risiko Management

ID	Risiko	Auswirkung	Massnahmen	S	W	G
R01	Ausfall eines Teammitglieds	Die Arbeit wird bei einer Einzelarbeit beendet.	Eine Ersatzperson für die Arbeit einplanen.	360	1%	3.6
R02	Datenverlust	Teil der Arbeit geht verloren	SVN bzw. Team Foundation Server einsetzen mit häufigem Commit	10	10%	1

R03	WP7 hat unbekannte Restriktionen(gegenüber normalem Silverlight)	Funktionen nicht umsetzbar, keine Schöne Lösung	Technologiestudium bzw. Prototyp möglichst früh	20	30%	6
R04	Fehler in der Zeitplanung	Meilensteine können nicht eingehalten werden.	Zeitreserve einplanen, Funktionen streichen	80	40%	32
R05	WP7 Mail Client – Architektur nicht klar definiert – Emulator erlaubt nicht das Senden von Email	Lösungskonzept nicht klar für die Spezifikation	Frühzeitig mit WP7 Mail Client experimentieren. WP7 Device frühzeitig Registrieren.	80	40%	32
R06	Ausfall Entwicklungsrechner	Arbeitsstation auf dem entwickelt wird, fällt aus	- Ausweichen auf Firmen, oder Notebook - Ersatzrechner beantragen.	15	5%	0.8
R07	Windows Phone 7 Gerät nicht verfügbar	Keine Test und Demos mit echter Hardware möglich	Tests und Demo mit Emulator	0	100 %	0
R08	Keine Erfahrung mit Entwicklung für Smartphone	Schwierigkeiten bei Gestaltung Komplexer GUI's für Smartphone	Zeitreserven für Design	30	40%	12
R09	Aufgabenstellung unklar	Verzögerungen, Fehlleistungskosten	Möglich frühe Abklärungen mit entsprechenden Personen	80	20%	16
R10	Auftreten unerwartet Probleme bei der Implementation des WP7 Mail Client	Termine für die Meilensteine werden nicht eingehalten.	Möglich früh die Machbarkeit-Studie anfangen	10	30%	3
Gesamtrisiko der Studienarbeit in Stunden				361		

S: Max. Schaden in Stunden[h]

W: Wahrscheinlichkeit[%]
G: Gewichteter Schaden in Stunden(G=W*S)

Arbeitspakete

Projektmanagement					
ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeit/ Risiken
A101	Projektplanung (Initial)	Projektplanungs-Dokument verfassen und übergeben	mba	35	
A102	Infrastruktur Aufbau	SVN-Zugang organisieren und Ablagestruktur definieren	mba	17	
A103	Projektplan(laufend)	Arbeitspakete und Zeitplanung weiterführen / verfeinern	mba	16	A101

Requirements					
ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeit/ Risiken
A201	Anforderungen, nichtfunktional	Definition aller nichtfunktionalen Anforderungen	mba	4	
A202	Dokument-Erstellung und Use Case, Brief	Dokumenterstellung und Definitionen aller funktionalen Anforderungen im „Brief Format“	mba	10	A201
A203	Use Case Diagramm	UseCase Diagramm erstellen	mba	3	A202
A204	Use Cases, „fully dressed“	Für die wichtigsten funktionalen	mba	20	A202,A203

		Anforderungen, die UseCase im ‚fully dressed Format erstellen.			
A205	Abklärungen	Abklärungen zu Anforderungen inklusive User Interface	mba	18	

Analyse					
ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeit/ Risiken
A301	Domain Analyse(OOA)	Domain Modell erarbeiten	mba	16	A202
A302	Sicherheitskonzept	Varianten Analyse für ein Sicherheitskonzept	mba	8 (optional)	
A303	SOA Web-Technologie	Varianten Analyse für SOA Web Technologie in Bezug auf Mobile Devices(REST,SOAP,WCF)	mba	20 (optional)	

Design					
ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeit/ Risiken
A401	Software Architektur	Beschreibung der Software-Architektur	mba	40	A2*,A3*
A402	Design(OOD)	Beschreibung des Software-Designs(Package – Klassen – und Sequenzdiagramme)	mba	30	A2*,A3*
A403	Web-Service Interface	Definition eines Web-Service Interface für die Inbetriebnahme über ein Mobile-Device	mba	16	A2*
A404	UI Design Studie für Mobile Device	Studie für ein auf die Anforderungen von Mobile-Devices optimiertes User	mba	25	

		Interface			

Implementation					
ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeit/ Risiken
A501	Architekturprototyp	Implementation des Architekturprototypen für ein „proof of concept“	mba	60	A2*,A301, A401,A402
A502	Implementation	Umfasst die Komplette Implementation. Wird am Ende der Elaborationsphase 2 genauer aufgeteilt.	mba	163	A2*,A3*, A4*
	Refactoring Architekturprototyp	Refactoring des Architekturprototypen	mba	5	A501
	Zugriff auf Webservices von WP7 Client	Client /Server Kommunikation starten	mba	4	

Test					
ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeit/ Risiken
A601	Systemtest	Beinhaltet den manuellen, auf die Auslieferung ausgerichteten Test der gesamten Applikation vor jedem Release.	mba	6	A2*,A5*, A602 Falls Fehler vorhanden Mehraufwand bei A503
A602	Testdokumentation	Erstellen der Testdokumentation.	mba	14	A2*,A5*

Dokumentation					
ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeit/ Risiken
A701	Kurzbericht	Verfassen und online Abgabe des Kurzberichts.	mba	10	
A702	Technischer Bericht	Verfassen des Technischen Berichts	mba	40	
A703	Management Summary	Management Summary erstellen	mba	12	
A704	Dokument Abschluss	Abschluss und Bereinigung aller Dokumente	mba	16	
A706	Schlussbericht	Schreiben des persönlichen Schlussberichtes	mba	6	

Technologie Studien					
ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeit/ Risiken
A011	Einarbeiten in WP7	Wissen zur Entwicklung für das Windows Phone 7 beherrschen	mba	20	
A012	Einarbeiten in Mobile UI Design	Wissen über UI-Design für Mobile-Devices beherrschen	mba	10	

Sitzungen					
ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeit/ Risiken
A02	Reviews	Reviews mit dem	mba/	37	

1		Betreuer zur Überprüfung der Meilensteine.	Huser		
A022	Sitzungen	Sitzung mit dem Betreuer über den aktuellen Projektstand	mba/ Huser	26	

Qualitätssicherung					
ID	Name	Beschreibung	Verantwortlich	Aufwand [h]	Abhängigkeit/ Risiken
A031	Code Review	Code-Review mit dem Betreuer, Debugging	mba/ Huser	8	
	Reserve	Reserve 100h	mba	100	

Infrastruktur

Hardware

- HSR-Rechner am Arbeitsplatz an der HSR
- SVN-Server der HSR(svn.hsr.ch)
-

Software

- Microsoft Visual Studio 2010(Entwicklungsumgebung)
- Microsoft Expression Blend 4 (GUI Design)
- Notepad2(XML-Dateien)
- FxCop (Überprüfen auf Microsoft Coding-Guidelines)
- GhostDoc(Unterstützung für Kommentare in .Net)
- MsBuild (Build –Automatiom)
- Adobe Photoshop CS4 (Icon/Grafiken)

Dokumentation

- Microsoft Word 2010 (Dokumentation)
- Microsoft Excel 2010 (Planung, Dokumentation)
- Adobe Acrobat 9 Professional(PDF-Dokument Generieren)
- Enterprise Architekt 7.0 (UML-Modeling)
- Doc-O-Matic (Quellcode-Dokumentation)

Versionsverwaltung

- Subversion (SVN-Server der HSR)
- TortoiseSVN als Client

Qualitätsmassnahmen

8.1.Team und Kommunikation

Das Projektteam besteht nur aus einer Person, deswegen muss die Koordination mit dem Betreuer gut abgewickelt sein. Für eine erfolgreiche Lösungsorientierte Kommunikation dienen die verschiedene Sitzungen und Reviews über die gesamte Projektdauer. (14 Wochen).

8.2. Dokumentation und Planung

Eine Gute Dokumentation ist in erste Linie massgeblich für den Erfolg eines Projektes. Die Projektdokumentation wird daher laufend aktualisiert und angepasst. Eine gute Zeitplanung ermöglicht ein besseres Zeitmanagement im Projekt. Dadurch können Verzögerungen frühzeitig erkannt werden. Eine Zeiterfassung wird laufend vom Teammitglied geführt und alle aufgewendeten Stunden werden in der Deteilplanung eingetragen.

8.3. Reviews

Ein wichtiger Faktor für die Qualitätssicherung stellen die Reviews im Projektverlauf dar. Deswegen werden regelmässig Dokumentsreviews und auch Code-Reviews mit dem Betreuer durchgeführt. Für diese Sitzungen werden Sitzungsprotokolle als Grundlage für weitere Vorgehensweise festgehalten.

8.4 Backup/Datensicherung

Daten bzw. Dokumente sowie Sourcecodes werden regelmässig im SVN-Verwaltunsgtool eingecheckt. Dies garantiert jederzeit eine Datensicherung auf dem SVN-Server.

8.5 Tests

Alle Funktionalen Anforderungen, welche auch anhand Use Cases spezifiziert wurden, werden getestet. Die Testresultate werden in einem Testplan ausgeführt.

8.6 Kodierungsrichtlinien

Wir werden Microsofts Coding Standard und Best Practices folgen um eine zuverlässige und wartungsfreundliche Email-Anwendung zu realisieren.

Anforderungsspezifikation

EMAIL CLIENT DYNAMIC DOCUMENT CREATION

Studienarbeit :	**Integration eines Dynamic Document Creation System in einen Windows Phone 7 Mail Client**
Autor :	**Ba Mohamedou Moustapha**
Verantwortlicher :	**Prof. Hansjörg Huser**
Betreuer:	**Prof. Hansjörg Huser** **Herr Simon Baer**
Industriepartner:	**Sevitec AG, Eschlikon TG**

Dokumentenverwaltung

Dokumenthistorie

Version	Status	Datum	Verantwortlicher	Änderungsgrund
0.1	In Bearbeitung	05.03.2011	mba	Dokumentvorlage erstellt
0.1	In Bearbeitung	06.03.2011	mba	Kapitel 1, Kapiel2, bis kapitel4
0.1	In Bearbeitung	07.03.2011	mba	Use Case Diagram, use Cases Brief
0.1	In Bearbeitung	14.03.2011	mba	Dokument überarbeiten
0.1	In Bearbeitung	21.03.2011	mba	Anforderungen ergänzen, Use Cases anpassen.
0.1	In Bearbeitung	22.03.2011	mba	UI Storyboard

Review

Folgende Personen haben das Dokument gelesen und geprüft:

Herr Simon Baer : Donnerstag,10.März 2011

Herr Simon Baer: Mittwoch,23.März 2011

Inhaltsverzeichnis

Einführung

Zweck

Das Anforderungsdokument beschreibt die Anforderungen an das zu beschaffendes Software-System. Die Nicht-Funktionale Anforderungen sowie die Funktionale Anforderungen in Form von Use Cases werden in diesem Dokument spezifiziert.

Gültigkeitsbereich

Die Gültigkeit dieses Anforderungsdokuments erstreckt sich über die gesamte Projektdauer (14 Wochen).Die Änderungen werden

Definitionen und Abkürzungen

Abkürzung	Beschreibung
WP7	**W**indows **P**hone **7** : Neuansatz einer mobilen Plattform von Microsoft für den Consumer- und Business-Bereich
WCF	**W**indows **C**ommunication **F**oundation : Service-orientierte Kommunikationsplattform von Microsoft für verteilte Anwendungen .
WMD2C	**W**indows **M**obile **D**ynamic **D**ocument **C**reation : Name der WP7 Email app
REST	**R**epresentational **S**tate **T**ransfer : Software-Architekturstil für verteilte Hypermedia-Systeme wie z.B World Wide Web
Silverlight	Silverlight ist Microsofts Technologie für sog. Rich Internet Applications (RIA)
XNA	XNA ist Microsofts .NET-basiertes Spieleframework für hochperformante 2D- und 3D-Spiele

Referenzen

Projektantrag/Aufgabenstellung.pdf

. Allgemeine Beschreibung

In folgenden Kapiteln dieses Dokuments wird das Produkt WMD2C zuerst kurz beschrieben. Danach folgt eine Beschreibung von funktionalen und nicht-funktionalen Anforderungen. Das letzte Kapitel beschreibt ausführlich die Use Cases.

Produkt Perspektive

Die Coorporate Identity (CD) spielt eine sehr wichtige Rolle für die Abwicklung der Geschäfte bei Unternehmen aller Branche, da sie die Persönlichkeit eines Unternehmens darstellt. . Da Dokumente und Email ein unverzichtbares Kommunikationsmittel für die Geschäftswelt darstellen, müssen Geschäftsdokumente sowie Email mit perfetem Coorporate Identity erstellt werden.Die Firma Sevitec AG hat mit dem Software –Produkt ONEOFFIX eine Lösung für Microsoft-Word Dokumente geliefert. Die Integration dieser Software in Microsoft-Word ermöglicht eine dynamische Dokument-Erstellung mit Coorporate Identity, Sprach und Profil sowie Benutzer-Abhängige Signaturen. Für Geschäftleute , die unterwegs Geschäfte abwickeln , wäre eine portierung solche Software-Produkte auf Mobile-Telefone von sehr grossen Bedeutung. Dazu soll einen WP7 Mail Client als WP7 App entwickelt werden.

Produkt Funktion

WMD2C soll folgende Funktionalitäten zur Verfügung stellen:

- Email in HTML-Format in Microsoft Outlook erstellen.
- Geschäftsdokumente in perfektem Corporate Identity erzeugen.
- Dokumenttyp laden
- Dokument-Parameter auswählen
- Benutzer- Signatur im Dokument einfügen
- Dokument erstellen
- Dokument steuern :
    - Element (Parameter) Einblenden
    - Element(Parameter) Ausblenden
    - Elemente (Parameter) nachträglich ändern

Dokumenttypen und Parameter sind in der Datenbank abgelegt und sollen daher nicht statisch programmiert. Der ganze Prozess soll dynamisch ablaufen. Es sollte jederzeit möglich sein in der Datenbank einen neun Dokumenttyp oder Parameter eingefügt werden können. Beim nächsten Erstellen eines Dokuments soll der neue Dokumenttyp bzw. Parameter angezeigt werden.

Benutzer Charakteristik

Die Zielgruppe für WMD2C beschränkt sich auf Geschäftsleute aller Branchen, welche Geschäftsdokumente mit Corporate Identity (CI) per Email mit einem Windows Phone 7 Gerät verschicken wollen.

Einschränkungen

WP7 basiert auf Microsoft Embedded-Betriebssystem CE. Die Betriebssystemebene (APIs) ist für Softwareentwickler allerdings nicht zugänglich, dies ist allein den Geräteherstellern vorbehalten. Für die Entwicklung von Anwendungen (Apps) steht eine .Net-basierte Anwendungsplattform (**Silverlight**) zur Verfügung.

Als WP7 Email App konzipiert kann WMD2C nur auf WP7-Geräte lauffähig sein.

Annahmen

In der Einarbeitungsphase konnte ein Konzept für das WMD2C System definiert werden. Dieses Grundkonzept besteht aus einem WebService, einer Datenbank (SQL-Server 2008) und WP7- Gerät. Der WebService stellt Daten wie (Dokumentvorlagen, Parameter) aus der Datenbank für das WP7-Gerät zu Verfügung. Im Rahmen dieser Studienarbeit wird angenommen, dass der Zugriff auf Webservices von WP7 –Geräte möglich wäre.

Abhängigkeiten

. Spezifische Anforderungen

Die Spezifischen Anforderungen werden im Kapitel 4 Anhand Use Case beschrieben. Die anderen Anforderungen, welche nicht aus Benutzer-Perspektive beschrieben werden können werden in folgendem Kapitel beschrieben.

Zugriff auf Datenbank

Die Dokumentvorlagen sollen auf einem SQL-Server 2008 liegen. WMD2C soll auf diese Daten via Internet zugreifen können. Für die Bereitstellung der Daten aus der Datenbank soll einen WCF-Service realisiert werden.

Login

Der Zugriff auf dem Webservice soll über Username /Passwort aus via Smartphone möglich sein.

Sicherheit , Performance, Stabilität

Anwendungen für WP7 werden grundsätzlich in einem Framework entwickelt, welches auf .NET basiert und Stabilität, Performance und Sicherheit garantiert. In diesem Framework stehen zwei UI-Technologien zur Verfügung: Silverlight und XNA.

Unter Windows Phone 7 stellt Silverlight ein vollwertiges Anwendungsframework für eigenständige Programme dar, läuft also nicht im Browser. Es basiert auf Silverlight 3 mit Erweiterungen und Anpassungen, bspw.

- Durchgehende Hardwarebeschleunigung für höchste Performance
- Integration des Software-Keyboards für Eingabeelemente
- Multitouch-Unterstützung
- Zugriff auf die Gerätehardware (Kamera, GPS, Sensoren) über spezielle APIs
- Anpassung an das Anwendungsmodell von WP7 (voneinander isolierte Anwendungen
- Weniger enge Sandbox(für Cross-Domain-Zugriff)

Benutzbarkeit

Geräte mit WP7 verfügen grundsätzlich über kapazitive Touchscreens, welche für Fingerbedienung optimiert sind und Multitouch-Funktionalität (4 oder mehr Touchpoints) unterstützen. Die einheitliche Bildschirmauflösung ist WVGA(800x480), so dass aufwändiges Testen mit verschiedenen Geräten entfällt. Unterstützung von Portrait- und Landscape-Modus gewährleistet. Zwecks Navigation sind drei Buttons auf der Vorderseite (Back, Home, Suche). Hardware-Tastatur oder andere Buttons und Sensoren sind optional.

Verständlichkeit

Das GUI soll intuitiv zu bedienen sein. Es wird erwartet, dass die eingesetzten Icons eine interne sowie externe Konsistenz aufweisen. Die Grundelemente werden daher stark an das Original-WP7Mail Client angel

Wiederherstellbarkeit

Bei einem Systemabsturz sollen alle relevanten Daten und letzte Benutzeränderungen, sicher abgespeichert sein.

Verbrauchsverhalten

Speicher-Verbrauch vom Original-WP7 Mail Client gilt auch für das System WMD2C.

Modifizierbarkeit

Allfällige Änderungen am WP7 Mail Client -Funktionalitäten müssen innerhalb von 2 Manntagen implementiert werden können.

Installierbarkeit

WMD2C kann als WP7 App einfach auf jedes WP7-Gerät installiert werden.

Schnittstellen

Ein Webservice definiert die Schnittstelle für die Kommunikation mit dem WP7 Email Client.

Datenbankschnittstelle

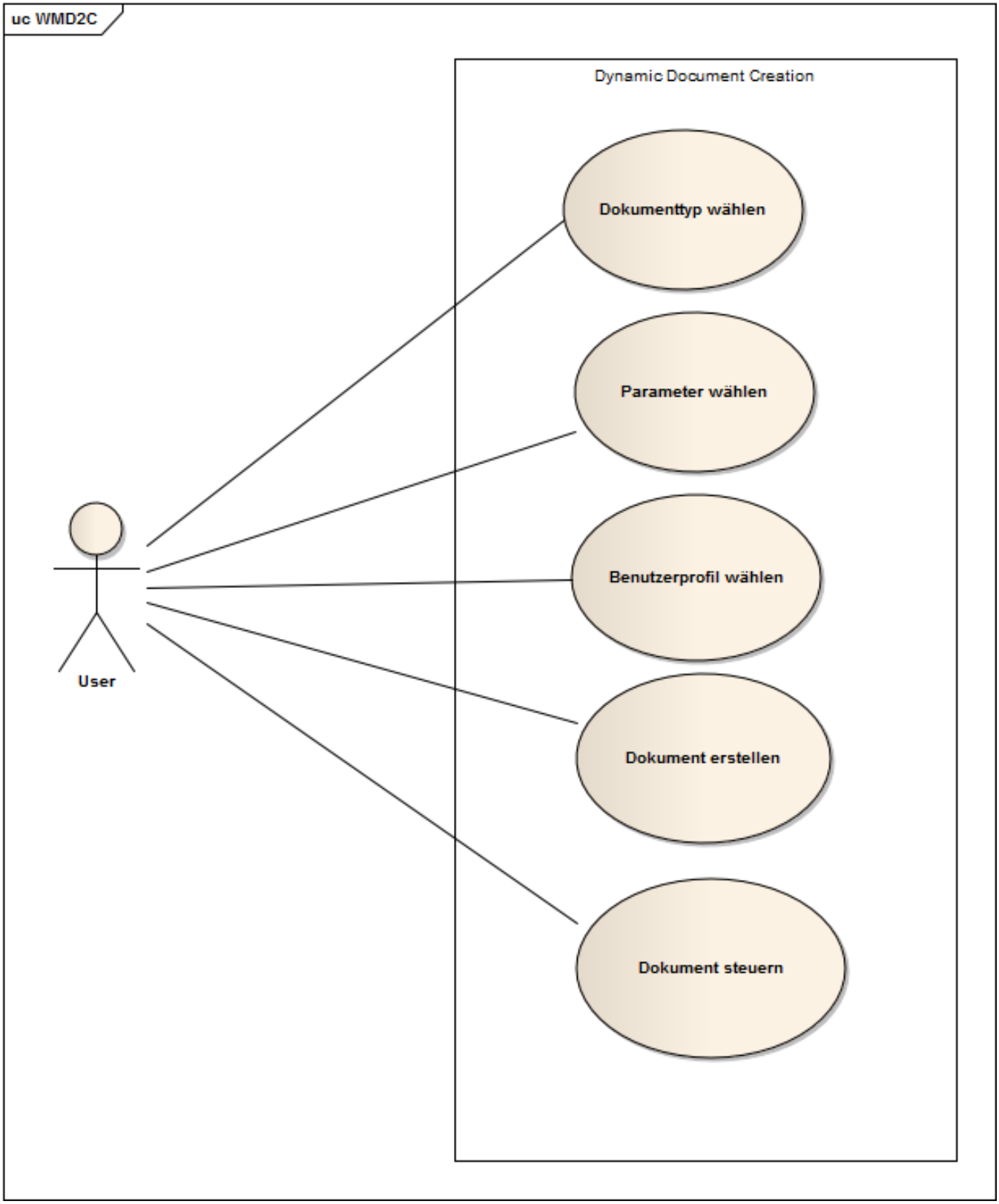
Es wird eine Datenbankschnittstelle benötigt, da Dokument-Vorlage auf eine Datenbank gespeichert sein muss. SQL-Server 2008 wird als Datenbank im Einsatz kommen.

Lizenzanforderungen

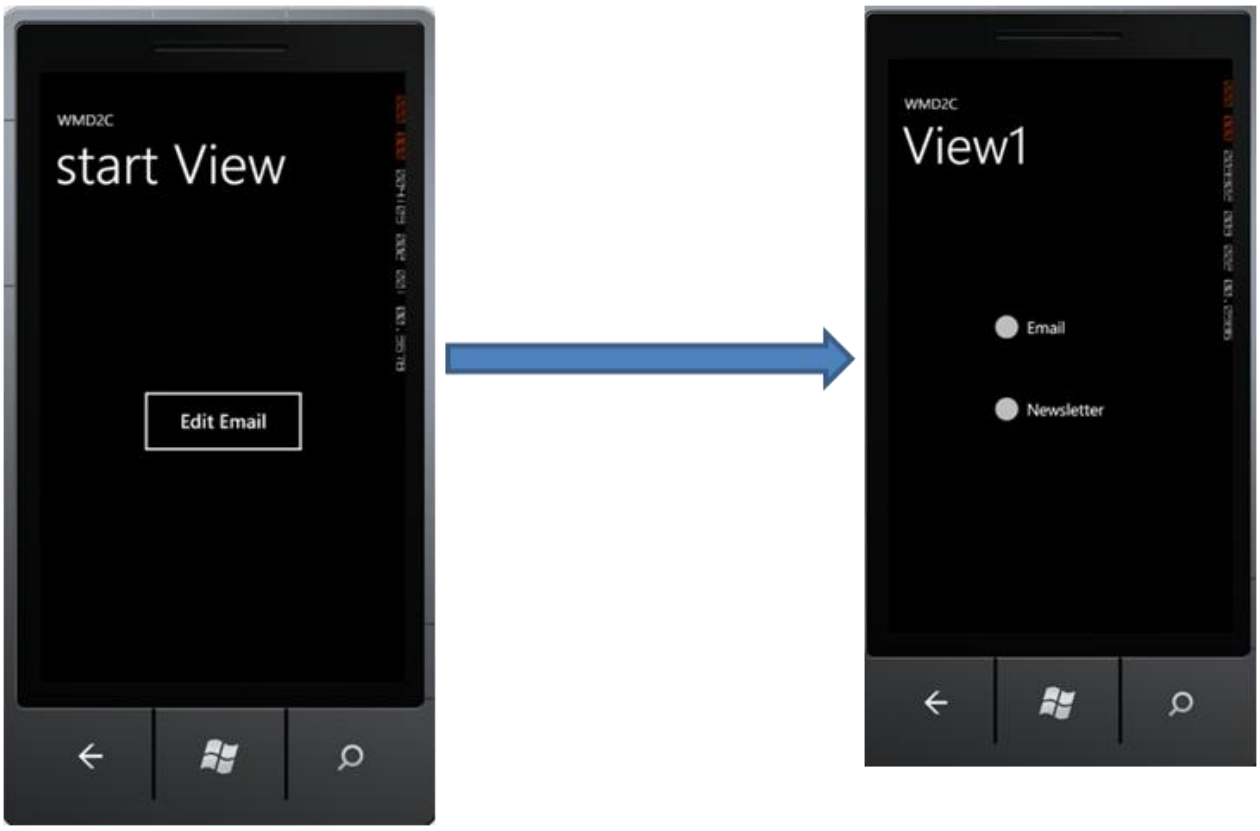
Es werden keine Lizenzen benötigt.

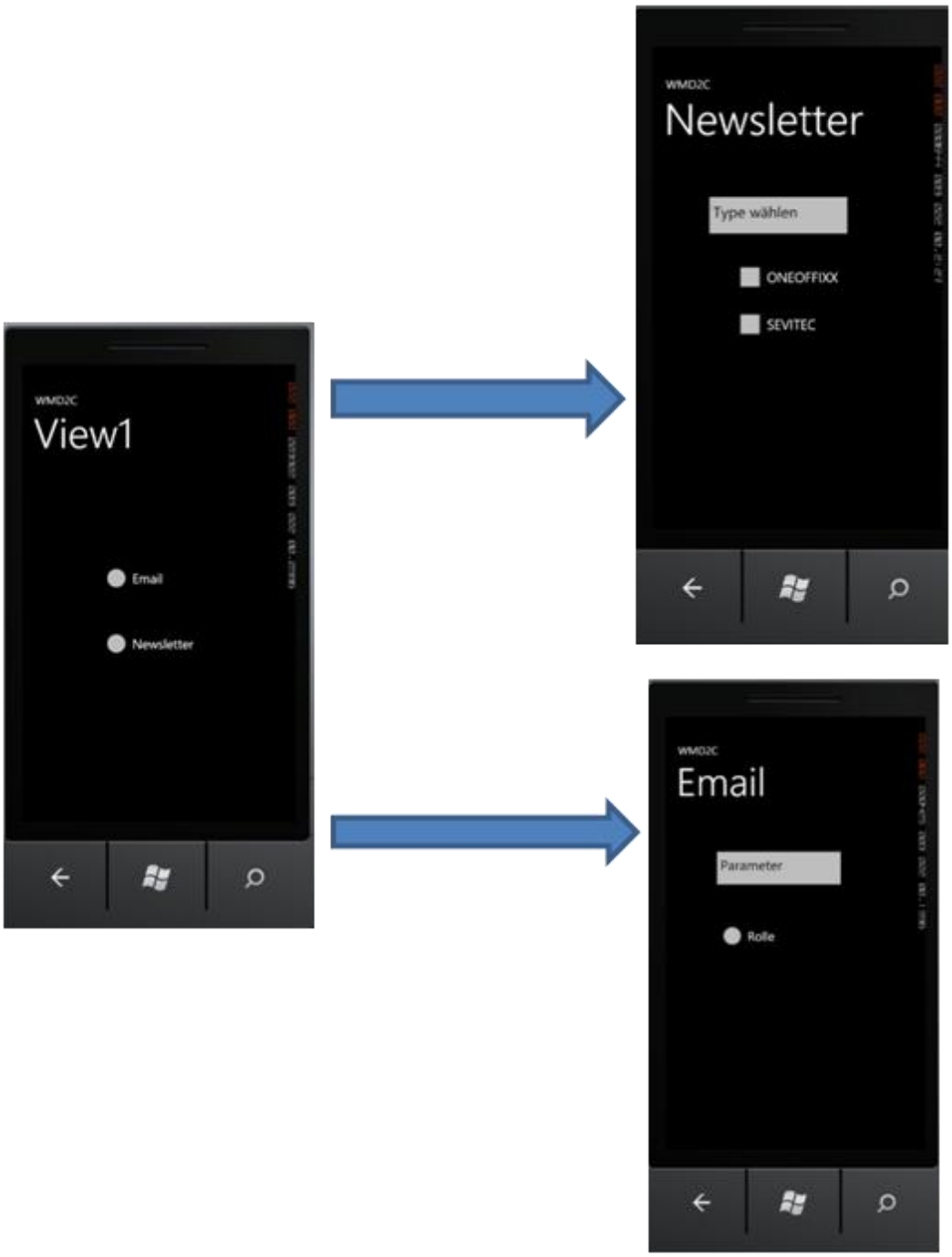
. Use Cases

Use Case Diagramm

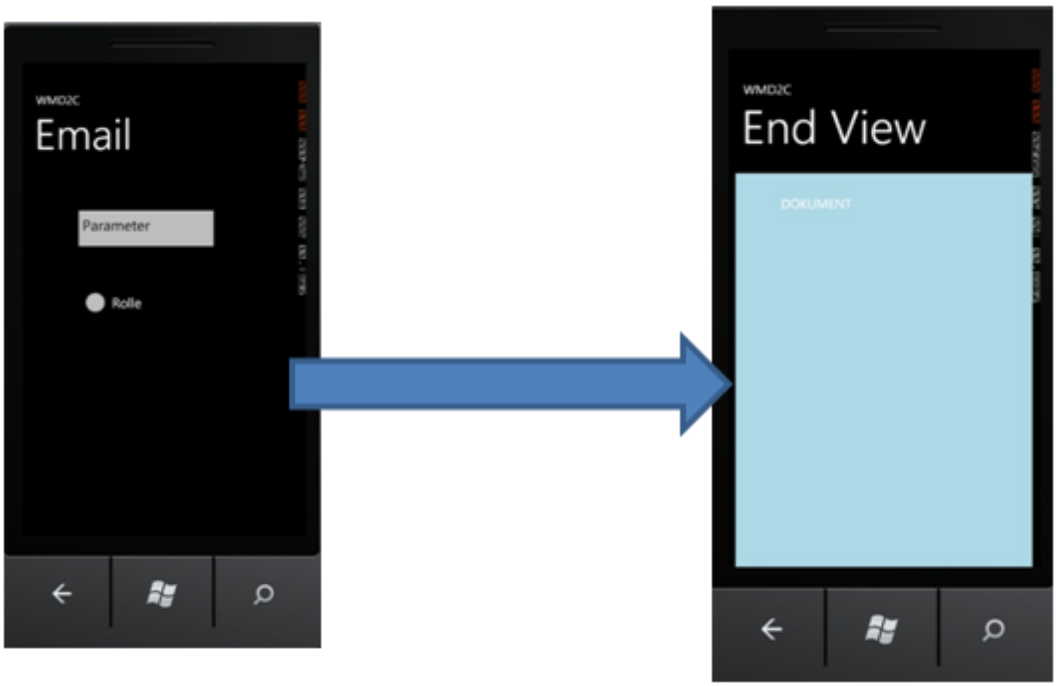


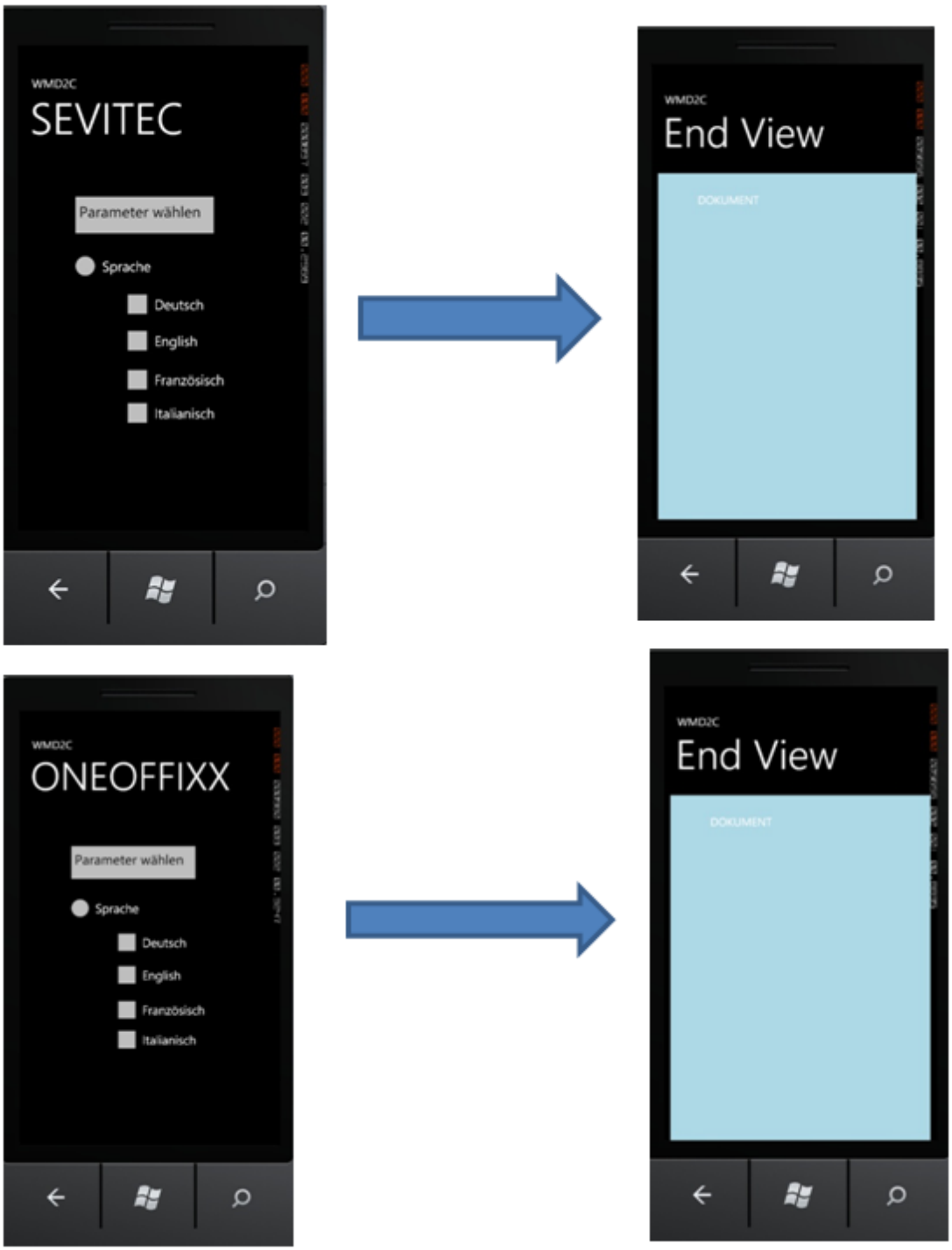
UI Prototype & Storyboard


WMD2C
start View
Edit Email
WMD2C
View1
Email
Newsletter


WMD2C
View1
Email
Newsletter
WMD2C
Newsletter
Type wählen
ONEOFFIXX
SEVITEC
WMD2C
Email
Parameter
Rolle


WMD2C
Newsletter
Type wählen
ONEOFFIXX
SEVITEC
WMD2C
SEVITEC
Parameter wählen
Sprache
Deutsch
English
Französisch
Italianisch
WMD2C
ONEOFFIXX
Parameter wählen
Sprache
Deutsch
English
Französisch
Italianisch


WMD2C
Email
Parameter
Rolle
WMD2C
End View
DOKUMENT



Aktoren & Stakeholders

Der **primäre Aktor** ist Geschäftsmann Nutzer eines WP7-Geräts.

. Use Cases Brief

UC1: Dokumenttyp auswählen

Der Benutzer kann aus einer Drop-Down Liste den gewünschten Dokumenttyp auswählen. Dokumenttypen sollen im allgemein in Gruppen zusammengefasst werden. Es werden zwei Gruppen definiert: Eine Gruppe „Email“ und eine Gruppe „Newsletter“. Die Gruppe „Newsletter“ besteht aus zwei Dokumenttypen: „ONEOFFIXX“ und „SEVITEC“.

UC2: Parameter auswählen

Im allgemein sollen die möglichen Parameter, die der Benutzer auswählen kann, vom Dokumenttyp abhängig sein.

Dokumenttyp „Email „ hat die Parameter: Logo, Rolle, Dokumenttyp „Newsletter“ hat als Parameter : Sprache.

UC3: Benutzerprofil auswählen

Der Benutzer kann das gewünschte Benutzerprofil festlegen .Der Benutzerprofil wird als Signatur für das Dokument verwendet. Das Benutzerprofil soll einmal definiert und gespeichert werden.

UC4: Dokument erstellen

Der Benutzer kann aus den gewählten Dokumenttyp und Parametern ein dynamisches Dokument erstellen bzw. eine formatierte HTML-Email erstellen. Bevor das Mail gesendet wird, muss der Benutzer es bearbeiten können.

Dokument steuern

Der Benutze kann nachträgliche Änderungen im Dokument vornehmen. Ein Dokument-Bereich lässt sich dadurch

Ein -/ und Ausblenden.

. Use Cases Brief

UC1: Dokumenttyp auswählen

Überblick	**Der Benutzer will ein dynamisches Dokument mit dem WP7-Gerät erstellen. Für die Erstellung eines Dokuments muss den Dokumenttyp definiert werden.**
Vorbedingungen	• Email-Anwendung (WMD2C) ist auf WP7-Gerät installiert und lauffähig. • Email-Account von Benutzer auf dem WP7-Gerät vorhanden.
Nachbedingungen	• Dokumenttyp wurde erfolgreich ausgewählt.

Standardablauf	1. Benutzer startet den WP7 Email Client 2. System zeigt Email-Operationen an. 3. Benutzer wählt die Option Dokument –Erstellen aus. 4. System zeigt Dokumenttypen an : Email oder Newsletter 5. Benutzer wählt den gewünschten Dokumenttypen aus.
Erweiterungen	4a. Falls Benutzer den Dokumenttyp Newsletter auswählt 4b. System zeigt zwei Newsletter-Typen an : ONEOFFIX oder SEVITEC 4.c Benutzer wählt Newsletter-Typ aus.
Spezielle Anforderungen	
Häufigkeit des Auftretens	
Offene Fragen:	

UC2: Parameter auswählen

Überblick	**Der Benutzer muss zum gewählten Dokumenttyp die Parameter definieren.**
Vorbedingungen	• UC1
Nachbedingungen	• Parameter wurden erfolgreich festgelegt.
Standardablauf	1. System zeigt die definierten Parameter für die Dokumente an : Sprache, Layout, Rolle. 2. Benutzer wählt die gewünschten Parameter aus der Liste aus.
Erweiterungen	2a. Bei der Wahl von Dokumenttyp Email 2a. Benutzer wählt nur Parameter Rolle aus.
Spezielle Anforderungen	
Häufigkeit des Auftretens	
Offene Fragen:	

UC3: Benutzerprofil auswählen

Überblick	**Der Benutzer muss das passende Benutzerprofil im Dokument**

	einfügen.
Vorbedingungen	• UC1 • UC2
Nachbedingungen	• Die Signatur wurde erfolgreich im Dokument eingefügt.
Standardablauf	1. Benutzer wählt über die Funktion „Einstellungen“ im Dokument das Benutzerprofil. 2. System zeigt Eingabefelder für die Identifikation des Benutzers 3. Benutzer gibt folgende Informationen ein : (Name, Vorname, Firma, Rolle) 4. System speichert das Benutzerprofil und fügt es automatisch in jedes erstellte Dokument ein.
Erweiterungen	
Spezielle Anforderungen	
Häufigkeit des Auftretens	
Offene Fragen:	

UC4: Dokument erstellen

Überblick	**Der Benutzer erstellt nun ein dynamisches Dokument mit dem WP7-Gerät.**
Vorbedingungen	• UC1 • UC2 • UC3
Nachbedingungen	• Dokument wurde erfolgreich erstellt
Standardablauf	1. Das gewählte Dokument wird automatisch vom System aus der Datenbank geladen. 2. Das Dokument wird automatisch in den Dokument-Editor geladen. 3. Benutzer kann nun Dokument schicken oder speichern.
Erweiterungen	

Spezielle Anforderungen	
Häufigkeit des Auftretens	
Offene Fragen:	

UC5: Dokument steuern

Überblick	**Der Benutzer will ein erstelltes Dokument ändern.**
Vorbedingungen	• UC4
Nachbedingungen	• Änderungen im Dokument wurden erfolgreich durchgeführt.
Standardablauf	1. Benutzer öffnet das Dokument im WP7 Email Client. 2. Benutzer klickt auf dem zu ändernden Bereich im Dokument an 3. System zeigt die Optionen für die Steuerung eines Dokuments : Dokument-Sektion ein/oder – ausblenden 4. Benutzer klickt auf die gewünschte Option an um die Änderung am Dokument durchzuführen 5. System übernimmt die Änderung im Dokument.
Erweiterungen	
Spezielle Anforderungen	
Häufigkeit des Auftretens	
Offene Fragen:	

EMAIL CLIENT DYNAMIC DOCUMENT CREATION

Studienarbeit :	**Integration eines Dynamic Document Creation System in einen Windows Phone 7 Mail Client**
Autor :	**Ba Mohamedou Moustapha**
Verantwortlicher :	**Prof. Hansjörg Huser**
Betreuer:	**Prof. Hansjörg Huser**
	Herr Simon Baer
Industriepartner:	**Sevitec AG, Eschlikon TG**

Dokumentinformationen

Änderungsgeschichte

Datum	*Version*	*Änderung*	*Autor*
30.03.2011	0.1	Dokument erstellt	mba
10.06.2011	1.0	End Version des Dokuments	mba

Inhalt

Einführung

Zweck

Dieses Dokument beschreibt die Software Architektur des Grundsystems. Design Alternativen und Entscheide sind in diesem Dokument beschrieben

Gültigkeitsbereich

Das Dokument ist für die komplette Dauer des Projektes gültig(14 Wochen).

Definitionen und Abkürzungen

Referenzen

Aufgabenstellung.pdf

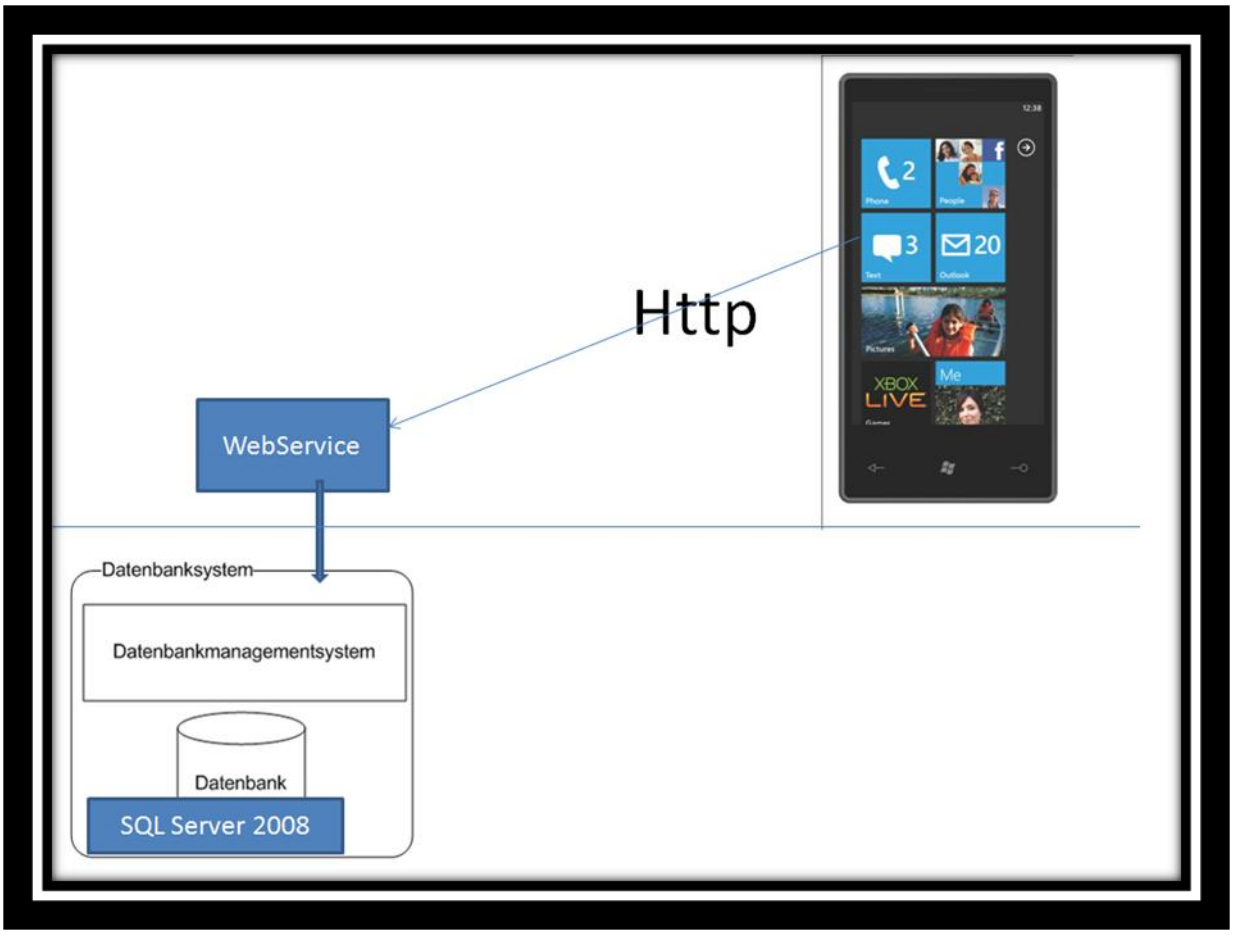
Domainanalyse.pdf

Übersicht

Die folgenden Kapitel beschreiben die gesamte Architektur der Applikation WMD2C.

Physikalische Architektur

Das WMD2C System läuft als WP7 app auf einem Smartphone und kann übers Internet mit einem WebService interagieren. Der WebService kann direkt die nötigen Daten für die Erstellung eines Dokuments aus der Datenbank laden und diese dem WP7 Email Client zur Verfügung stellen. Das folgende Bild zeigt die drei wichtigen Komponenten der physikalischen Architektur des Systems: **WP7 Client, WebService und Datenbank.**



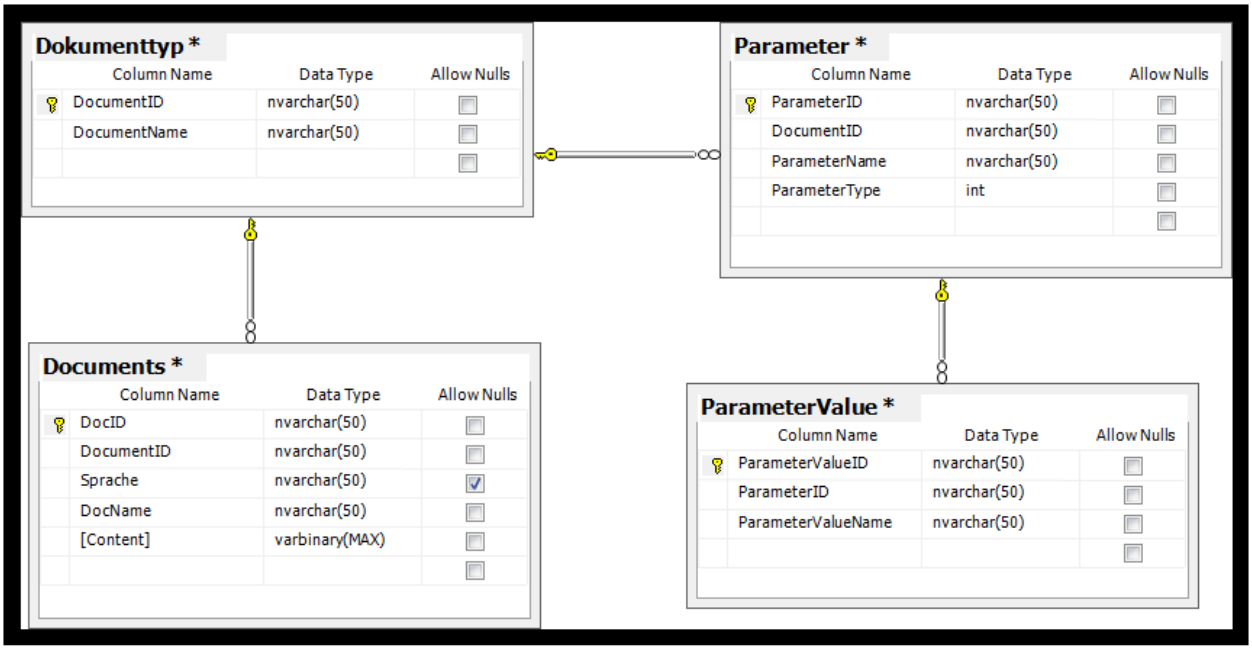
logische Architektur

Datenbank

Die Daten werden auf einer Datenbank abgelegt. Die Parameter, die ausgewählt werden können, sind dann von Dokumenttyp abhängig. In der Datenbank kann jederzeit einen neuen Dokumenttyp oder Parameter eingefügt werden. Beim nächsten Erstellen eines Dokuments werden die neuen Einträge in der Datenbank angezeigt.

Die SQL-Server Datenbank ist einfach gehalten mit folgender Struktur:

Die SQL Server Datenbank ist einfach gehalten mit folgender Struktur.



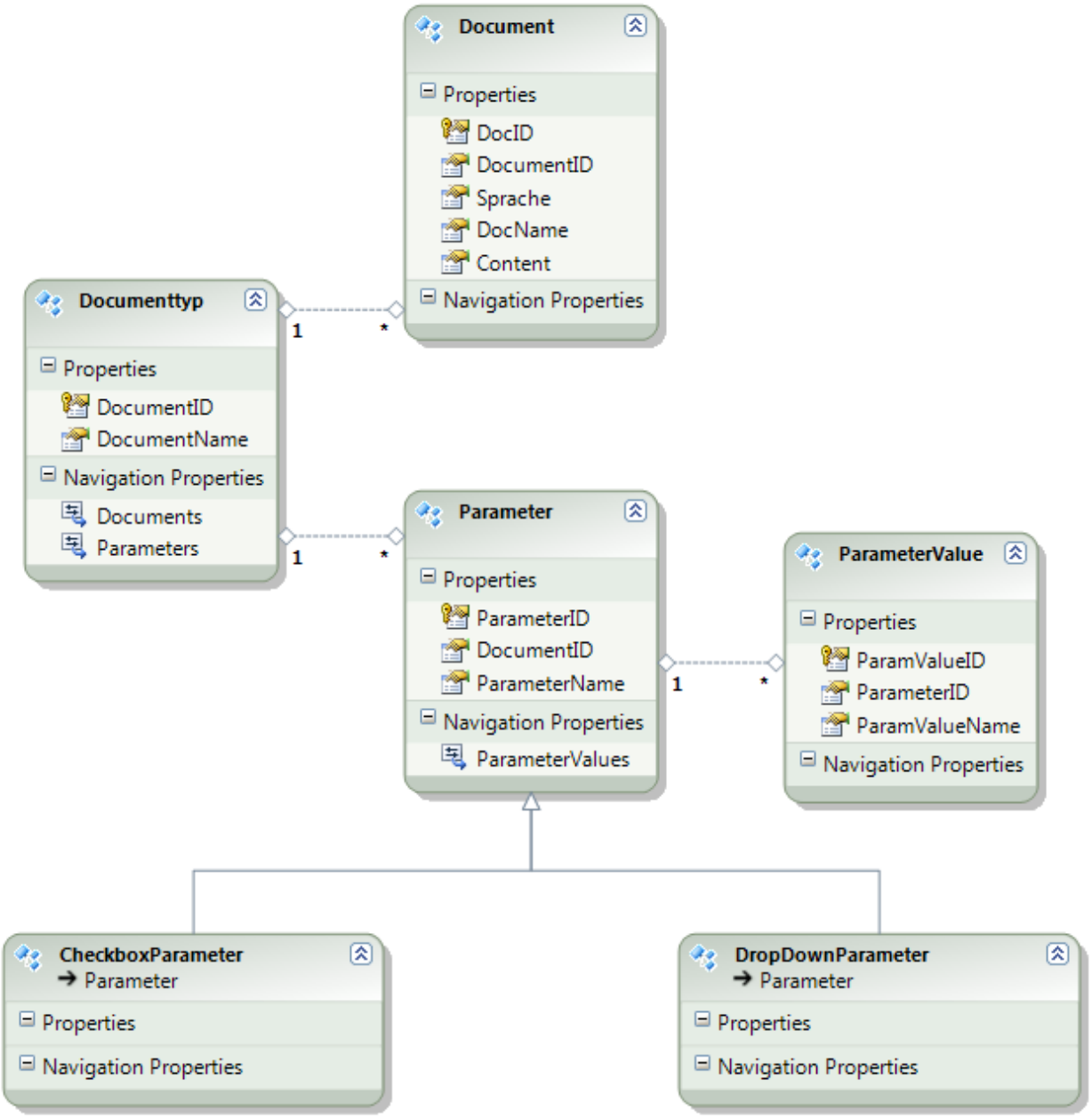
Prozesse und Threads

Jede Windows Phone 7 App wird aus Sicherheitsgründen in einer isolierten Sandbox ausgeführt. Dadurch können Applikationen nicht direkt auf Ressourcen des Geräts zugreifen. Für den Zugriff auf die Ressourcen, wie etwa die Bildbibliotheke, werden daher sogenannte Launcher und Choosers bereitgestellt. Launcher und Choosers sind eigenständige Applikationen. Ein Launcher startet eine integrierte Applikation, welche keine Daten an die aufrufende Applikation zurückliefert. Als Beispiel sei der EmailComposeTask genannt. Dieser öffnet die eingebaute E-Mail-App.Ein Chooser hingegen liefert Daten an die aufrufende Applikation zurück. Beispielsweise öffnet der CameraCaptureTask die Kameraapplikation und liefert als Ergebnis ein neu aufgenommenes Foto an die aufrufende App. Die aktuell vorhandene Windows-Phone-Version verhindert, dass mehrere Apps parallel ausgeführt werden. Verwendet eine App einen Launcher oder Chooser, wird somit die rufende App beendet, in den sogenannten Tombstone-Modus versetzt und faktisch beendet. Dies ist ein essenzielles Konzept der Windows-Phone-Entwicklung.

Data Access Layer

Der Data Access Layer wird mit dem ADO.NET Entity Framework CodeFirst Modell implementiert. Dieser Layer beinhaltet eigentlich nur das *.edmx Datenmodell (z.B. WP7MailClientModel.edmx), welches den Datenzugriff auf die darunterliegende SQL Server Datenbank ermöglicht.

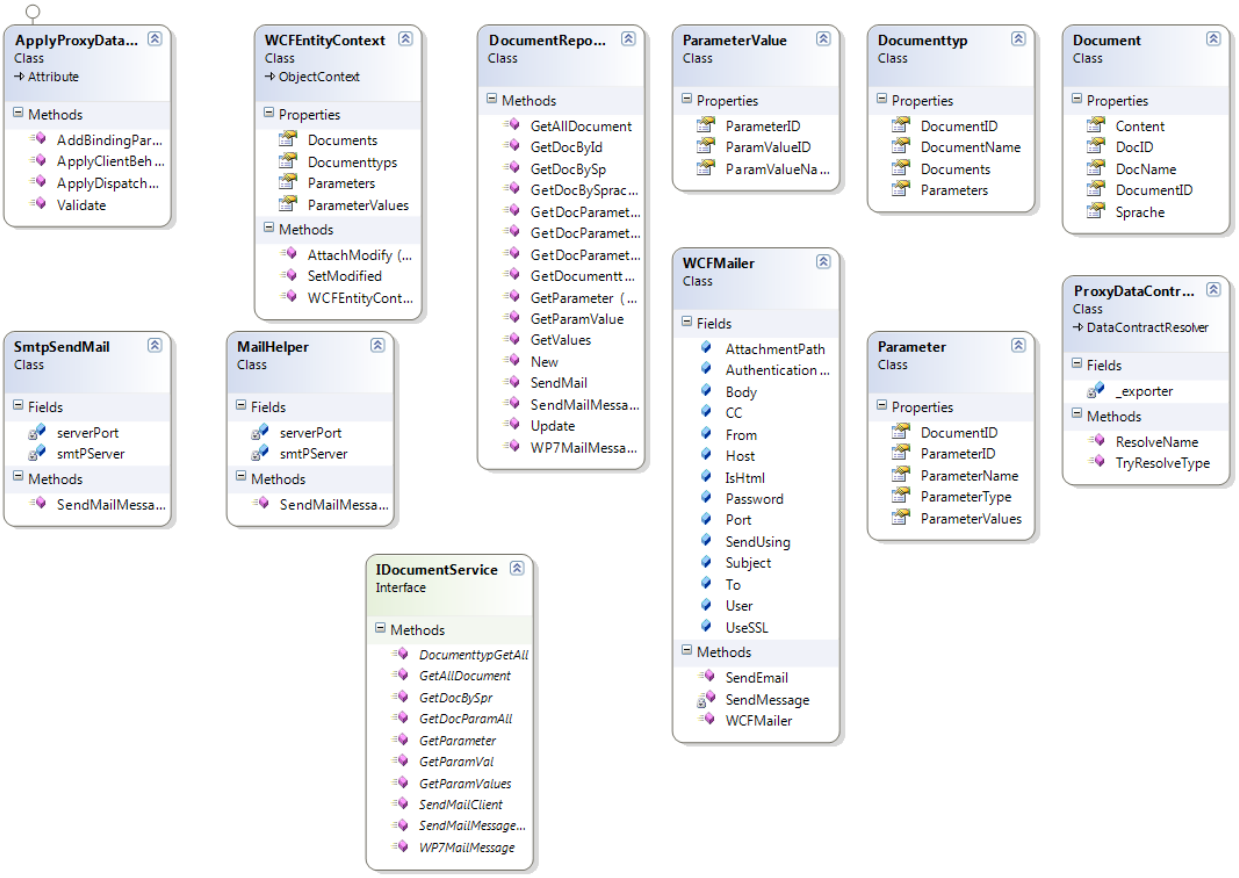
Das Modell sieht folgendermaßen aus:



WCFEntityData

Das WCFEntityData Projekt implementiert das Data Access Layer der Applikation. Dieses Projekt besteht aus drei Komponenten: Entities, Model, Repositories, Interfaces, SmtpMail.

Klassendiagramm



Entities

Für jede Entität im Entity Datenmodell wird eine Klasse definiert. Diese Klassen sind im Ordner „Entities“ abgelegt.

Documents

Diese Klasse definiert ein Daten Transfer Object (DTO), welches ein Mapping mit der entsprechenden Datenbank Tabelle (Documents) realisiert.

Documenttyps

Diese Klasse definiert ein Daten Transfer Object (DTO), welches ein Mapping mit der entsprechenden Datenbank Tabe13.06.201113.06.2011lle (Documenttyps) realisiert.

Parameter

Diese Klasse definiert ein Daten Transfer Object (DTO), welches ein Mapping mit der entsprechenden Datenbank Tabelle (Parameters) realisiert.

ParameterValue

Diese Klasse definiert ein Daten Transfer Object (DTO), welches ein Mapping mit der entsprechenden Datenbank Tabelle (ParameterValues) realisiert.

WCFEntityContext (DTO Converter)

Diese Klasse implementiert die ObjectContext- Klasse im Namespace System.Data.Objects. Die WCFEntityContext-Klasse teilt dem Entity Framework mit, wie es das Mapping zwischen den Entitäten im Datenmodell und den DTO's machen soll.

DocumentRepository

Diese Klasse implementiert die Funktionalität für den Service-Layer. Für jede Entität – Documenttyp, Document, Parameter, ParameterValue- sollen folgende CRUD-Operationen unterstützt werden. Definiert der Business-Layer.

- Alle Entitäten lesen
- Eine Entität anhand des Primärschlüssels lesen
- Einfügen
- Update
- Löschen

Service Layer

DocumentService

WebService-Projekt mit Windows Communication Foundation (WCF) amtiert als Service Layer für die ganze Applikation.

IDocumentService

Diese Klasse ist die eigentliche WCF-Serviceschnittstelle. Definiert alle Service-Operationen für den Windows Phone 7 Client

ProxyDataContractResolver

Diese Klasse implementiert die **DataContractResolver-Klasse.** Die Klasse DataContractResolver erweitert beim WCF-Service die Serialisierungs-und Deserialisierungsprozsse.

Client Layer

WP7TestClient

Diese Windows Phone 7 Client Applikation verwendet den WCF-Service um einen dynamischen Dokument zu erstellen.

MainPage

Diese Klasse definiert die HauptSeite der Applikation. Alle Documenttypen in der Datenbank werden automatisch beim Starten der Applikation angezeigt.

ParameterViewPage

Diese Klasse definiert die Parameter-Seite. Wenn ein Documenttyp aus der HauptSeite ausgewählt wird, navigiert die Applikation von der HauptSeite zur Parameter-Seite. Alle Parameter eines Documenttyps aus der Datenbank werden automatisch angezeigt.

EmailConfigPage

Diese Klasse definiert einen Html-Editor um erzeugte Dokumente darzustellen.

Erfahrungsbericht

EMAIL CLIENT DYNAMIC DOCUMENT CREATION

Autor :	Ba Mohamedou Moustapha
Verantwortlicher :	Prof. Hansjörg Huser
Betreuer:	Prof. Hansjörg Huser Herr Simon Baer
Industriepartner:	Sevitec AG, Eschlikon TG

Ich wollte schon immer etwas mit der .Net Technologie anfangen und vor allem etwas im Mobile Bereich entwickeln. Alles hat bei mir angefangen als ich das Modul „Microsoft Technologie“ bei Herr Huser an der Hochschule für Technik Rapperswil(HSR)besuchte. Ich war immer sehr begeistert und wusste gleich, Aha das ist wirklich etwas für mich. Ich habe also mir gedacht, die beste Möglichkeit sich intensiv mit .Net Technologien zu befassen wäre, eine Arbeit in der Welt von Microsoft zu schreiben. Ich habe vergeblich einen Team-Partner gesucht und ich musste unbedingt meine letzte Studienarbeit absolvieren. Zum Glück habe ich ein sehr kompetenter Professor und vor allem mit Herz gefunden, der mir die Chance gab endlich meine Studienarbeit zu schreiben. Das Thema der Studienarbeit war sehr interessant und ich war sehr froh darüber, dass ich die Arbeit bei meinem Favorit an der HSR schreiben dürfte. Ich habe sehr viel gelernt und immer Grosse Freude von Anfang zum Ende des Projektes empfunden. Die Zusammenarbeit mit dem Industriepartner (Sevitec AG)in Eschlikon war sehr angenehm. Ich war froh, dass die Firma Sevitec für mich Fahrspesen bezahlt. Ich finde das sehr großzügig. Herr Simon Baer ist ein sehr kompetenter Ingenieur und die Zusammenarbeit mit ihm war sehr positiv für mich. Ich habe leider diese Studienarbeit in schwierige Zeiten für mich durchgeführt. Ich hatte die ganze Zeit keinen Kopf frei gehabt. Ich habe viel Zeit investiert um zumindest das Grundkonzept der Lösung zu implementieren. Hauptsache alles ist gut gelaufen und ich bin zufrieden mit mir selbst. Ich bin sehr froh, dass ich die Arbeit selbstständig durchgeführt habe. Natürlich Herr Huser hat mir sehr viel geholfen. Jetzt habe ich die Übung gemacht. Ich möchte

nun Kraft tanken um mein Studium erfolgreich abzuschließen. Ich bedanke mich bei allen für die wertvolle Unterstützung.