HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

ins institute for networked solutions

Business Intelligence mit Silverlight

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2011

Autorin: Anita Hollenstein
Betreuer: Professor Hansjörg Huser
Projektpartner: Jürg Jucker, S3CC GmbH

Erklärung über eigenständige Arbeit

Ich erkläre hiermit,

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

Ort, Datum: _____

Name: Anita Hollenstein

Unterschrift: _____

Verzeichnis Dokumente

Business Intelligence mit Silverlight

Aufgabenstellung

Business Intelligence mit Silverlight

Aufgabenstellung für Anita Hollenstein

Einführung

Silverlight ist ein RIA Framework für die Entwicklung von performanten Webbasierten Applikationen mit einem desktop-ähnlichen Benutzerverhalten. Diese Arbeit soll aufzeigen, wie mit Silverlight moderne und interaktive BI-Clients entwickelt werden können. Eine Demo-Applikation soll das Erarbeitete veranschaulichen.

Aufgabenstellung

- Analyse des Umfelds
  - Welches sind Anforderungen an eine moderne BI-2.0 Lösung.
  - Untersuchen von bestehenden Demo-Applikationen mit modernen UI's.
- Architektur eine BI-Lösung mit Silverlight-Clients
  - Kommunikation Silverlight-Datenquelle (SQL-Server, AS-Server)
    - Analyse von bestehenden Lösungen
    - Service-Schnittstelle: WCF-Dataservices, eigene Service-Schnittstelle
  - Client basierend auf MVVM, Client-Funktionalität
  - Serverseitige Funktionalität
  - Datenquelle: SQL-Server (relationales DW) und/oder Analysis-Services (Cube mit MDX-Abfragen) (Testdatenbanken AdventureWorks oder Contoso)
- User Interface mit Silverlight
  - Evaluation von BI-Controls für Pivot-Dialoge, Charts, Dashboards, KPI's (Silverlight Controls, Telerik, Devexpress, ComponentArt)
    Unterstützung von Navigations-, Drill-Down-, Roll-Up-Operationen.

- Implementation eines Silverlight basierenden Show-Cases: UI mit MySite (Userspezifische Darstellung mit KPI's, Dashboard, Charts, Map-Controls), Export nach Excel etc.

Resultate

- Beschreibung der Architekturvarianten
- Charakterisierung der typischen UI-Komponenten und Interaktionen eines modernen Silverlight-BI-Clients
- Ausführbare Demo-Applikation mit Dokumentation

Projektpartner

Auftraggeber

Jürg Jucker, S3CC GmbH

Projektteam

Anita Hollenstein (ahollens@hsr.ch)

Betreuung HSR

Hansjörg Huser, hhuser@hsr.ch, Tel: 055 222 49 12 (HSR Raum 6.010)

Projektabwicklung

Termine:

- Beginn der Arbeit: **Mo., 21. Feb. 2011**
- Abgabetermin Kurzfassung/Poster zum Review: Di. **31.Mai.2011** an hhuser@hsr.ch
- Abgabetermin (inkl. Poster): **Fr. 3.Juni 2011**, 17.00 Uhr
- Zwischenbesprechung/Review mit Auftraggeber nach Projektplan

Arbeitsaufwand

Für die erfolgreich abgeschlossene Arbeit werden 8 ECTS angerechnet. Dies entspricht einer Arbeitsleistung von 240 Stunden pro Student. Bei einer 14-wöchigen Laufzeit sind dies ca. 2 Arbeitstage pro Woche.

Hinweise für die Gliederung und Abwicklung des Projektes:

Gliedern Sie Ihre Arbeit in 4 bis 5 Teilschritte. Schliessen Sie jeden Teilschritt mit einem Meilenstein ab. Definieren Sie für jeden Meilenstein, welche Resultate dann vorliegen müssen!

Folgende Teilschritte bzw. Meilensteine sollten Sie in Ihrer Planung vorsehen:

Schritt 1: Projektauftrag inkl. Projektplan (mit Meilensteinen),

- Meilenstein 1: Review des Projektauftrages abgeschlossen. Projektauftrag von Auftraggeber und Dozent genehmigt
  Letzter Meilenstein: Systemtest abgeschlossen
  Termin: ca. eine Woche vor Abgabe
- Entwickeln Sie Ihre SW in einem iterativen, inkrementellen Prozess: Planen Sie möglichst früh (spätestens in der Mitte des Projektes) einen ersten lauffähigen Prototypen mit den wichtigsten und kritischsten Kernfunktionen. In die folgenden Phasen können Sie dieses Kernsystem schrittweise ausbauen und testen.
- Falls Sie in Ihrer Arbeit neue oder Ihnen unbekannte Technologien einsetzen, sollten Sie parallel zum Erarbeiten des Projektauftrages mit dem Technologiestudium beginnen.
- Setzen Sie konsequent Unit-Tests ein! Verwalten Sie ihre Software und Dokumente auf einem SVN-Repository (oder auf einer vergleichbaren Umgebung). Stellen Sie sicher, dass der Betreuer jederzeit Zugriff auf das Repository hat und dass das Projekt anhand des Repositories jederzeit wiederhergestellt werden kann.
- Achten Sie auf die Einhaltung guter Programmier- und Designprinzipien
- Halten Sie sich im Übrigen an die Vorgaben aus dem Modul SE-Projekt.

Projektadministration

- Führen Sie ein individuelles Projekttagebuch aus dem ersichtlich wird, welche Arbeiten Sie durchgeführt haben (inkl. Zeitaufwand). Diese Angaben sollten u.a. eine individuelle Beurteilung ermöglichen.
- Dokumentieren Sie Ihre Arbeiten laufend. Legen Sie Ihre Projektdokumentation mit der aktuellen Planung und den Beschreibungen der Arbeitsresultate elektronisch in einem Projektordner ab. Dieser Projektordner sollte jederzeit einsehbar sein (z.B svn-Server oder File-Share).

Inhalt der Dokumentation

Bei der Abgabe muss jede Arbeit folgende Inhalte haben:

- Dokumente gemäss Vorgabe: https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html
- Aufgabenstellung
- Technischer Bericht
- Projektdokumentation
- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind.

- Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen.
- Projekttagebuch, Dokumentation des Projektverlaufes, Planung etc.

Form der Dokumentation:

- Bericht (Struktur gemäss Beschreibung) in Ordner(1 Exemplar für HSR)
- Alle Dokumente und Quellen der erstellten SW auf CD, CD's sauber angeschrieben (2 Ex.).

Fortschrittsbesprechung:

Regelmässig findet zu einem fixen Zeitpunkt eine Fortschrittsbesprechung statt.
Teilnehmer: Dozent und Studenten, bei Bedarf auch Vertreter der Auftraggeber
Termin: jeweils Monat, 13h, Raum 6.010 (Abweichungen werden rechtzeitig kommuniziert)
Traktanden

- Was wurde erreicht, was ist geplant, welche Probleme stehen an
- Review von Code/Dokumentation (Abgabe jeweils einen Tag vor dem Meeting

Falls notwendig, können weitere Besprechungen / Diskussionen einberufen werden.
Sie erstellen zu jeder Besprechung ein Kurzprotokoll, welches Sie spätestens 2 Tage nach der Sitzung per e-mail an den Betreuer senden.

Rapperswil, 21. Feb. 2011
Hansjörg Huser

Abstract

Abteilung	**Informatik**
Name der Studierenden	**Anita Hollenstein**
Studienjahr	**FS 2011**
Titel der Studienarbeit	**Business Intelligence mit Silverlight**
Examinator	**Prof. Hansjörg Huser**
Themengebiet	**Internet-Technologien und -Anwendungen / Software**
Projektpartner	**Jürg Jucker, S3CC GmbH**
Institut	**Institut für vernetzte Systeme**

Übersicht
Business Intelligence bezeichnet die systematische Analyse von Unternehmensdaten. Die Daten werden gesammelt, ausgewertet und leicht verständlich dargestellt.

Aufgabe
Inhalt dieser Studienarbeit ist die Realisierung einer Microsoft Silverlight Demo-Applikation. Diese wird veranschaulichen, wie mit Silverlight ein moderner und interaktiver Webclient für Business Intelligence Lösungen entwickelt werden kann.
Als Datenquelle soll dabei ein SQL-Server mit einem relationalen Data Warehouse oder ein Analysis-Service mit Cube (OLAP Würfel) dienen. Für die Kommunikation mit der Datenquelle gibt es verschiedene Möglichkeiten von Service-Schnittstellen. Diese werden analysiert und die beste Variante implementiert.
Die Benutzeroberfläche wird aus bereits bestehenden Business Intelligence Controls wie z.B. Diagrammen, Graphen, Tabellen und Karten zusammengesetzt. Diese Komponenten werden aufgrund ihrer Datenschnittstelle und den verschiedenen Navigations-, Drill-Down- und Roll-Up-Operationen untersucht und ausgewählt.
Die Business Intelligence Applikation muss mit grossen Datenmengen umgehen können. Es dürfen sowohl beim Programmstart als auch bei Operationen keine langen Verzögerungen aufgrund des Ladens von Daten auftreten. Zudem sollen für den Benutzer keine zusätzlichen Kenntnisse erforderlich und die Benutzeroberfläche selbsterklärend sein.

Ergebnis
Als Demo-Applikation wurde ein Sales Dashboard entwickelt. Dieses stellt mittels diversen Controls die Verkaufszahlen eines Unternehmens mit verschiedenen Details auf einen Blick dar. Der Benutzer kann den Zeitbereich der anzuzeigenden Daten selber festlegen und verändern. Zudem kann er mittels Drill-Down- und Roll-Up-Operationen die Daten detailliert oder weniger detailliert darstellen lassen.
Die Kommunikation zwischen Client und Server erfolgt über WCF RIA Services. WCF RIA Services ermöglicht eine einfache und strukturierte Entwicklung von Rich Internet Applikationen wie Silverlight.
Die Datenabfrage auf den SQL-Server erfolgt über den Aufruf von Stored Procedures, welche in der relationalen Datenbank abgelegt sind. Dies ermöglicht, dass die abgefragten Daten bereits auf der Datenbank gefiltert werden und somit eine effizientere Datenübertragung erfolgt.

Business Intelligence mit Silverlight

Projektplan und Zeiterfassung

Inhaltsverzeichnis

1 Einführung

1.1 Zweck

Dieses Dokument dient der Aufzeichnung des Projektplans für die Studienarbeit. Es dient als Grundlage für das gesamte Projekt. Weil nicht von Anfang an der genaue Ablauf des Projekts geplant werden kann, wird das Dokument laufend angepasst

1.2 Gültigkeitsbereich

Der Projektplan ist während der gesamten Projektdauer gültig.

1.3 Definitionen und Abkürzungen

- BI: Business Intelligence
- MS: Meilenstein
- MVVM: Model-View-ViewModel
- UI: User Interface

2 Projektübersicht

Business Intelligence bezeichnet die systematische Analyse von Unternehmensdaten. Die Daten werden gesammelt, ausgewertet und leicht verständlich dargestellt.

2.1 Zweck und Ziel

Ziel dieses Projektes ist es, eine Applikation zu entwickeln, welche aufzeigt wie sich Microsoft Silverlight für Business Intelligence Lösungen einsetzen lässt.

3 Projektorganisation

3.1 Projektmitglieder

Anita Hollenstein

3.2 Betreuer

Prof. Hansjörg Huser

4 Management Abläufe

4.1 Projekt Kostenvoranschlag

Das Projekt startet am 21. Februar 2011 und endet am 3. Juni 2011. Insgesamt dauert das Projekt 15 Wochen. Der Projektumfang beträgt ca. 240 Arbeitsstunden, dies entspricht ca. 16 Stunden pro Woche.

4.2 Projektplan

4.2.1 Zeitplan

Der in Tabelle 4 (am Ende des Kapitels 4) ersichtliche Zeitplan bietet eine Übersicht über den gesamten Projektablauf. Er zeigt, wann an welchen Artefakten gearbeitet wird und bietet einen Überblick über die Iterationen und die gesetzten Meilensteine. Der Plan wird fortlaufend angepasst.

4.2.2 Iterationsplanung

Die Zeitplanung legt sieben Iterationen von einer bis drei Wochen fest. Die Artefakte Projektdokumentation, Projekttagebuch und Literaturverzeichnis/Glossar ziehen sich über alle Iterationen hinweg und werden deshalb in den in der Tabelle 1 aufgeführten Zielen nicht erwähnt.

Iteration	Dauer	Von	Bis	Ziele
Inception	1 Woche	21.02.2011	27.02.2011	Projektplanung
Elaboration 1	2 Wochen	28.02.2011	13.03.2011	Analyse bestehender Demo-Applikationen, Evaluation von BI-Controls verschiedener Hersteller, Datenbanklösungen und Service-Schnittstellen.
Elaboration 2	2 Wochen	14.03.2011	27.03.2011	Erstellen der Architekturskizze, Fortführen der Evaluation von BI-Controls. Implementierung eines Grundgerüsts für die TestImplementierung der zu evaluierenden Controls.
Elaboration 3	2 Wochen	28.03.2011	10.04.2011	Evaluation weiterer Controls, Testen von komplexeren Funktionen und Interaktion zwischen Controls
Construction 1	3 Wochen	11.04.2011	01.05.2011	Erstellen des GUI Designs, Implementierung der Demo Applikation, Refactoring
Construction 2	3 Wochen	02.05.2011	22.05.2011	Fortführung der Implementierung, Schwerpunkt der letzten Woche auf GUI und Refactoring gesetzt
Transition	2 Wochen	23.05.2011	03.06.2011	Dokumentationen

Tabelle 1: Iterationsplanung

4.2.3 Meilensteine

Meilenstein	Datum	Name	Arbeitsresultat
MS1	28.02.2011	Projektauftrag inkl. Projektplan	Review des Projektauftrages abgeschlossen. Projektauftrag sowie Projektplan von Auftraggeber und Dozent genehmigt
MS2	13.03.2011	1. Teil der Analyse abgeschlossen	Evaluation von Datenbank und Service-Schnittstelle sowie die Analyse bestehender Demo-Applikationen ist abgeschlossen
MS4	27.03.2011	Prototyp	Einfache Client-Server Verbindung mit einer simplen Datendarstellung auf der Clientseite.
MS4	10.04.2011	Analyse und Design abgeschlossen	Analyse abgeschlossen, zu verwendende Architektur ist festgelegt, GUI Design steht
MS5	22.05.2011	Software fertiggestellt	Die Software ist einsatzbereit und die Systemtests erfolgreich abgeschlossen

Tabelle 2: Meilensteine

4.2.4 Besprechungen und Reviews

Es finden wöchentliche Besprechungen von ca. einer Stunde statt. Die besprochenen Punkte werden jeweils in einem Kurzprotokoll zusammengefasst. Für Sitzungen mit Abgaben im Voraus werden Reviewtermine abgemacht.

Datum	Review Inhalt
21.02.2011	Review des Projektauftrags
28.02.2011	Review des Projektplans
14.03.2011	Review der Kommunikationsvarianten Entscheidung
08.04.2011	Review des Prototyps
02.05.2011	Code Review
30.05.2011	Review Kurzfassung/Poster

Tabelle 3: Besprechungen und Reviews

Zeitplan

	Inception		Elaboration1		Elaboration 2		Elaboration 3		Construction 1			Construction 2			Transition	
Woche	1 MS1	2	3 MS2	4	5 MS3	6	7 MS4	8	9	10	11	12	13 MS5	14	15
	21.02.-27.02.	28.02.-06.03.	07.01.-13.03.	14.03.-20.03.	21.03.-27.03.	28.03.-03.04.	04.04.-10.04.	11.04.-17.04.	18.04.-24.04.	25.04.-01.05.	02.05.-08.05.	09.05.-15.05.	16.05.-22.05.	23.05.-29.05	30.05.-03.06.
Projekt Management															
Projektplan															
Requirements															
Anforderungsspezifikation															
Analyse															
Evaluation von BI-Controls															
Evaluation Datenbank															
Eval. Service-Schnittstelle															
Analyse bestehender Demo-Applikationen															
Design															
GUI Design															
Architekturskizze															
Implementierung															
Testimpl. der Controls															
Client Implementierung - GUI mit BI Controls - Clientseitige Logik															
Service-Schnittstelle - Impl. geeigneter Lösung															
Serverseitige Funktionen															
Refactoring															
Tests															
Systemtests															
Dokumentation															
Software Architektur Dokument															
UI Dokumentation															
Technischer Bericht															
Persönlicher Bericht															
Kurzfassung															
Poster															
Projekttagebuch															
Literaturverzeichnis / Glossar															

Tabelle 4: Zeitplan des Projekts

5 Arbeitspakete

5.1 Projektmanagement

5.1.1 Projektplan

Der Projektplan dient der Planung und der Organisation des Projekts. Die erforderlichen Arbeiten werden in Arbeitspakete unterteilt.

5.2 Requirements

5.2.1 Anforderungsspezifikation

Die Anforderungsspezifikation beschreibt die funktionalen und nichtfunktionalen Anforderungen an moderne BI-2.0 Lösungen.

5.3 Analyse

5.3.1 Evaluation von BI-Controls

Dient der Bewertung und dem Schaffen eines Überblicks über bestehende BI-Controls.

5.3.2 Evaluation Datenbank

Als Datenquelle soll eine relationale Datenbank auf einem SQL-Server oder ein Analysis Service Cube sein. Mittels Tutorial wird der Analysis Service Cube kennengelernt. Danach folgt die Entscheidung welche Datenquelle verwendet werden soll.

5.3.3 Evaluation Service-Schnittstelle

Verschiedene Serviceschnittstellen werden evaluiert und miteinander verglichen. Unter Anbetracht der gewählten Datenbank und des Silverlight-Clients wird die beste Variante verwendet.

5.3.4 Untersuchung bestehender Demo-Applikationen

Vergleich der verwendeten UI‘s sowie den Lösungen für die Kommunikation mit der Silverlight-Datenquelle.

5.4 Design

5.4.1 GUI-Design

Aufgrund der Erkenntnisse aus der Analyse wird das GUI Design erstellt und die ausgewählten BI-Controls zusammengestellt.

5.4.2 Architekturskizze

Aufgrund der Ergebnisse von der Analyse von bereits bestehenden Demo-Applikationen, wird die Software Architektur definiert und skizziert. Die Entscheide werden in dem Software Architektur Dokument festgehalten.

5.5 Implementierung

5.5.1 TestImplementierung der Controls

Zur Unterstützung der Evaluation werden die Controls testweise implementiert.

5.5.2 Client Implementierung

Programmierung des Clients basierend auf MVVM. In der Iteration Construction 1 ist der Schwerpunkt vor allem auf die Clientlogik gelegt, während in der Iteration Construction 2 die Logik allmählich abgeschlossen wird und die Ausarbeitung des GUIs im Zentrum steht.

5.5.3 Service-Schnittstelle

Programmierung der Service-Schnittstelle zur Client-Server Kommunikation.

5.5.4 Serverseitige Funktionen

Programmierung von Serverlogik, wie zum Beispiel Datenbankabfragen.

5.5.5 Refactoring

Der Code wird überarbeitet ohne weitere Logik hinzuzufügen.

5.6 Tests

5.6.1 Systemtests

Das gesamte System wird gegen die funktionalen und nichtfunktionalen Anforderungen getestet.

5.7 Dokumentation

5.7.1 Software Architektur Dokument

Das Software Architektur Dokument beinhaltet eine Übersicht über die Architektur, also den logischen und physischen Aufbau des Programmes.

5.7.2 UI Dokumentation

Die UI Dokumentation schafft einen Überblick über die Benutzeroberfläche und beschreibt die Funktionen der einzelnen Komponenten.

5.7.3 Allgemeine Arbeiten

Gemäss Vorgabe der HSR für Studienarbeiten und der Aufgabenstellung werden Titelblatt, Kurzfassung des Projekts, Poster und ein technischer sowie ein persönlicher Bericht erstellt.
Ebenfalls werden laufend ein Projekttagebuch und Kurzprotokolle zu den wöchentlichen Sitzungen geführt.

5.7.4 Literaturverzeichnis / Glossar

Das Literaturverzeichnis enthält Angaben zur verwendeten Literatur, der Glossar listet wichtige Abkürzungen und Begriffe auf, welche im Projekt vorkommen. Beide werden laufend ergänzt.

6 Infrastruktur

6.1 Räumlichkeiten

Grundsätzlich wird der reservierte Raum für die Informatik Semesterarbeiten benutzt. Die weitere Arbeit erfolgt zu Hause.

6.2 Hardware

- Persönlicher Rechner im Studienarbeitsraum (von der HSR zur Verfügung gestellt)
- Privates Notebook
- SVN-Server der HSR
- Privater Drucker / Drucker der HSR

6.3 Software

6.3.1 Betriebssystem

- Microsoft Windows 7 Enterprise / Professional SP1

6.3.2 Testserver

- Microsoft SQL Server 2008 R2

6.3.3 Programmiersprachen / Datenbanksprachen

- C#
- XAML
- SQL

6.3.4 Entwicklungsumgebung und Tools

- Microsoft Visual Studio 2010
- SQL Server Management Studio
- Silverlight Toolkit
- Silverlight SDK
- RIA Services Toolkit
- Telerik RadControls für Silverlight
- MVVM Light
- StyleCop

6.3.5 Versionsverwaltung

- Subversion (SVN-Server der HSR)
- TortoiseSVN

6.3.6 Dokumentation

- Microsoft Word 2010
- Microsoft Excel 2010
- Enterprise Architect
- Adobe Photoshop CS5

7 Qualitätsmassnahmen

7.1 Dokumentation

Um die Erstellung einer hochwertigen Software zu gewährleisten werden wichtige Entscheide in einer Dokumentation festgehalten.

7.2 Zeiterfassung

Die Anzahl geleisteter Stunden wird zusammen mit dem Datum und mit der Beschreibung der Tätigkeit in einer Exceltabelle erfasst.

7.3 Styleguide

7.3.1 Dokumentation

Mit der Verwendung der Formatvorlagen von Microsoft Word wird eine einheitliche Formatierung aller Dokumente ermöglicht.

7.3.2 Programmcode

Um den Programmcode einheitlich und verständlich zu gestalten werden die *.NET Design Guidelines for Class Library Developers*[1] von Microsoft eingehalten. Ebenfalls wird darauf geachtet, dass der gesamte Code in der englischen Sprache geschrieben wird.
Zur Überprüfung der Richtlinien wird das Tool StyleCop verwendet.

7.4 Versionsverwaltung

Alle Dokumente und Programmcodedateien werden über SVN verwaltet und versioniert. Jeder Upload einer neuen Version in das SVN-Repository wird mit einem sinnvollen Kommentar versehen. Somit liegt eine nachvollziehbare Versionierung der Dateien vor.

[1] http://msdn.microsoft.com/en-us/library/czefa0ke(v=VS.71).aspx, 30.05.2011

8 Zeiterfassung

Datum	Tätigkeit	Dauer
Woche 1		**17 h**
21.02.2011	Einrichten des Arbeitsplatzes, Installation der benötigten Software	3 h
22.02.2011	Einlesen und Schaffen eines Überblicks	3 h
24.02.2011	Erstellen des Projektplans	7 h
25.02.2011	Weiterführen des Projektplans, Beschreibung der Artefakte	2 h
27.02.2011	Fertigstellung des Projektplans	2 h
Woche 2		**17 h**
28.02.2011	Erste Analyse der ComponentArt Controls	4 h
28.02.2011	Besprechung des Projektplans und des weiteren Vorgehens	1 h
02.03.2011	Tutorial zu Cube-Datenbank	2 h
03.03.2011	Tutorial zu Cube-Datenbank	6 h
03.03.2011	Analyse der CompoentArt/Telerik Controls	2 h
06.03.2011	Analyse der Telerik Controls	2 h
Woche 3		**24 h**
07.03.2011	Analyse PivotViewer	4 h
07.03.2011	Tutorial zu RIA Services	2 h
07.03.2011	Besprechung der bisherigen Erkenntnisse, des weiteren Vorgehens	1 h
09.03.2011	Weiterführung RIA Service Tutorial	2 h
10.03.2011	Informationssuche über verschiedene Kommunikationsvarianten	8 h
11.03.2011	Informationssuche über verschiedene Kommunikationsvarianten	4 h
13.03.2011	Erstellen einer Übersicht und eines Vergleichs über Kommunikationsvarianten	3 h
Woche 4		**20 h**
14.03.2011	Vergleich der Kommunikationsvarianten	2 h
14.03.2011	Besprechung	1 h
14.03.2011	Erstellen von Projekt mit RIA Service	3 h
15.03.2011	Chart Implementierung	2 h
16.03.2011	Testen der Verbindung mit RIA Service	2 h
17.03.2011	Informationssuche über korrekte RIA Service - Telerik Controls Verbindung	7 h
18.03.2011	Erfolgreiches Verbinden von Chart mit RIA Service Datasource (kein MVVM)	3 h
Woche 5		**21 h**
21.03.2011	Informationssuche über Drilldown mit Telerik Chart	4 h
21.03.2011	Besprechung	1 h
21.03.2011	Erstellen von Stored Procedures	2 h
22.03.2011	Versuch Drilldown-Implementierung	2 h
23.03.2011	Recherche Drilldown in Telerik Demos, Verständnisprobleme	4 h
24.03.2011	Informationssuche über Drilldown mit Telerik Chart	8 h

Woche 6		**21 h**
28.03.2011	Auflistung der Probleme mit Drilldown	4 h
28.03.2011	Besprechung der bisherigen Erkenntnisse, des weiteren Vorgehens	1 h
28.03.2011	Analyse des Drilldowns in der Telerik Demo	1 h
28.03.2011	Beginn Implementierung des Drilldowns	3 h
29.03.2011	Fehlersuche in der Drilldown Implementierung	2 h
30.03.2011	Fertigstellung Drilldown	2.5 h
31.03.2011	Implementierung Zoom and Scroll	7 h
31.03.2011	Erstellen einer optimierten Stored Procedure für Z&S	0.5 h
Woche 7		**20 h**
04.04.2011	Versuch einer Interaktion zwischen einzelnen Controls	3 h
04.04.2011	Besprechung der implementierten Controls, des weiteren Vorgehens	1 h
04.04.2011	Implementierung des Sparkline-Controls	3 h
05.04.2011	Fertigstellen des Sparkline-Controls	2 h
07.04.2011	Implementierung des TimeBar-Controls	4 h
07.04.2011	Verbindung des TimeBar-Controls mit Drill Down Chart	3 h
08.04.2011	Fertigstellung des TimeBar Controls	1.5 h
08.04.2011	Kurze Besprechung der implementierten Controls, weiteres Vorgehen	0.5 h
08.04.2011	Informationssuche zur RadMap-Erstellung	2 h
Woche 8		**18 h**
11.04.2011	Erstellung einer RadMap, Lokalisierung von Ortschaften mit BingGeocodeProvider	7 h
12.04.2011	Einbinden der RadMap in das Dashboard	2 h
13.04.2011	Map-Drilldown mittels RadComboBox Selection	4 h
13.04.2011	Vergrössern der RIA Service Limite, Erstellen von optimierten Stored Procedures	2 h
14.04.2011	Einbinden der neuen Stored Procedures	1 h
15.04.2011	Verbindung zwischen Map, TimeBar, Chart und Pie	2 h
Woche 9		**19 h**
18.04.2011	Fertigstellung Verbindung zwischen Map, TimeBar, Chart und Pie	5 h
18.04.2011	Besprechung	1 h
19.04.2011	Sparkline Drilldown Verbindung	2 h
20.04.2011	Code Refactoring, Aufräumen nicht mehr gebrauchter Stored Procedures	7 h
21.04.2011	Code Refactoring	4 h
Woche 10		**17.5 h**
26.04.2011	Fehlerbehebung Sparkline, Erstellen neuer Stored Procedures Product/Region Drilldown	2 h
26.04.2011	Besprechung	0.5 h
27.04.2011	Refactoring, korrektes Einbinden der DLL's, Beheben von SVN Problemen	4 h
27.04.2011	Erstellung von Bullet Graphs für das Dashboard	1 h
27.04.2011	Dashboard Design	1 h
28.04.2011	Dashboard Design	2 h

29.04.2011	Bullet Graph für Dashboard	2.5 h
30.04.2011	Bullet Graph für Dashboard	1 h
01.05.2011	Bullet Graph für Dashboard	1.5 h
Woche 11		**24 h**
02.05.2011	Bullet Graph für Dashboard	4 h
02.05.2011	Besprechung	1 h
02.05.2011	Erstellung Grundgerüst Sales Dashboard mit RIA Service Projekt	2 h
03.05.2011	Einbindung von TimeBar und Map in Sales Dashboard, Command Binding	3 h
04.05.2011	Einbindung restlicher Komponenten Sales Dashboard	2 h
04.05.2011	Recherche Observer in .NET sowie Recherche MVVM Light	1.5 h
05.05.2011	Recherche Observer in .NET sowie Recherche MVVM Light	2 h
05.05.2011	Implementierung MVVM Light Commands und Messaging	4 h
06.05.2011	Sales Chart Control mit Datasource verbinden, Ladeperformanz optimieren	3.5 h
07.05.2011	Include Fehlerbehebung	0.75 h
07.05.2011	Bullet Graph Fehlersuche	0.25 h
Woche 12		**25.5 h**
09.05.2011	Fertigstellung Bullet Graph	0.5 h
09.05.2011	Falsche Map Darstellung behoben	0.5 h
09.05.2011	Code Refactoring	2 h
09.05.2011	Grid für Dashboard	1.5 h
09.05.2011	Besprechung	0.75 h
09.05.2011	Grid für Dashboard	1.5 h
09.05.2011	Installation MVVM Light Snippets	0.5 h
09.05.2011	Beginn Implementierung ViewModel Locator	1.25 h
10.05.2011	Fortführung Grid Implementierung --> Fehlersuche	1 h
11.05.2011	Arbeit an Grid	2.5 h
12.05.2011	Fertigstellung ViewModel Locator, Fertigstellung Grid Control, Map Verbesserung	7.5 h
13.05.2011	Code Refactoring	1 h
14.05.2011	Code Refactoring und Styleüberprüfungen / -korrekturen	4 h
14.05.2011	Planung Dokumentation	1 h
Woche 13		**26.25 h**
16.05.2011	User Interface Controls Dokumentation	4 h
16.05.2011	Besprechung	0.5 h
16.05.2011	Fehlersuche Designview, Beheben des Fehlers	2 h
16.05.2011	Implementierung von Style Resourcen, Refactoring des XAML Codes	2 h
18.05.2011	Dokumentation Anforderungsspezifikation	4.5 h
19.05.2011	Erstellen der Dokumentgerüste	2.5 h
20.05.2011	Arbeit an Software Architektur Dokument	5 h
21.05.2011	Arbeit an Software Architektur Dokument	5.25 h
22.05.2011	Arbeit an Software Architektur Dokument	0.5 h

Woche 14		**25 h**
23.05.2011	Arbeit an SAD	3 h
23.05.2011	Erstellen von Use Case Diagrammen für SAD	4 h
24.05.2011	Beschreibung der Use Case Diagramme	2 h
25.05.2011	Ausarbeitung SAD, Beschreibung Stored Procedures	4 h
26.05.2011	Ausarbeitung an Anforderungsspezifikation	2.5 h
27.05.2011	Ausarbeitung an Anforderungsspezifikation, Überarbeitung Projektplan	3 h
28.05.2011	Erstellen des Abstracts und des Posters, Korrekturen SAD	5 h
29.05.2011	Erstellen des Hauptdokuments: Struktur, Formatierungen	1.5 h
Woche 15		**20 h**
30.05.2011	Einfügen einzelner Dokumente in Hauptdokument, Strukturierung, Formatierung	6 h
31.05.2011	Ausarbeitung der Dokumentation	3 h
01.06.2011	Ausarbeitung Dokument Evaluation, Schreiben des technischen Berichts	8 h
02.06.2011	Fertigstellung des technischen Berichts, Schreiben des persönlichen Berichts	3 h
02.06.2011	Formatierung, Korrekturlesen	3 h
03.06.2011	Ausdrucken der Arbeit, Brennen der CDs	4 h

Total Stunden	**322.25 h**

Business Intelligence mit Silverlight

Anforderungsspezifikation

Inhaltsverzeichnis

1 Einführung

1.1 Zweck

Die Anforderungsspezifikation beschreibt die Anforderungen für den Showcase des Projekts Business Intelligence mit Silverlight.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments erstreckt sich über die komplette Dauer des Projektes.

1.3 Definitionen und Abkürzungen

- UC: Use Case

2 Allgemeine Beschreibung

2.1 Produkt Perspektive

Der Showcase ist eine Anwendung, die mittels der Darstellung von diversen BI-Controls wie Charts, Maps, KPI's etc. die Entwicklung von interaktiven BI-Clients mit Silverlight veranschaulichen soll.

2.2 Produkt Funktionen

Die Anwendung stellt Daten aus einer Data Warehouse Datenbank graphisch so dar, dass der Benutzer auf einen Blick die ungefähren Zahlen und Trends erkennen kann.
Dem Benutzer soll es möglich sein das Zeitfenster der anzuzeigenden Daten festzulegen und die Detailliertheit der Daten nach seinen Wünschen anzupassen.

2.3 Benutzer Charakteristiken

Die Hauptzielgruppe sind Personen, die regelmässig Kennzahlen des Unternehmens überprüfen, vergleichen sowie unter Umständen präsentieren.
Die Benutzer benötigen für die Anwendung keine speziellen Kenntnisse.

2.4 Einschränkungen

Die Anwendung dient ausschliesslich der Darstellung von Daten. Diese können somit nur gelesen, nicht aber verändert, gelöscht oder neu erstellt werden.

3 Use Cases

3.1 Use Case Diagramm

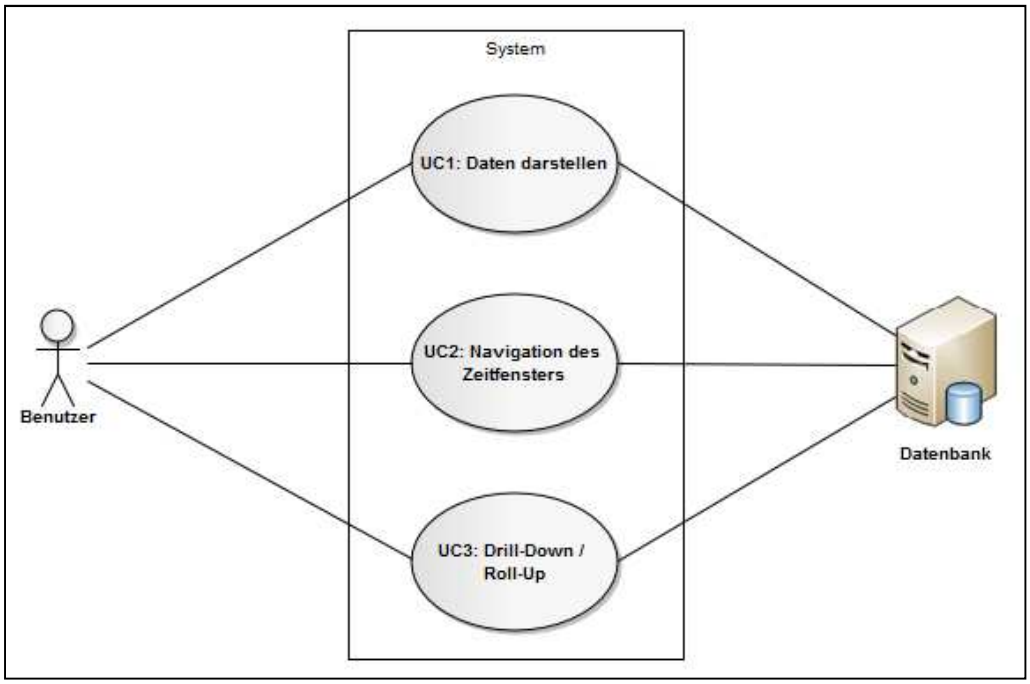


Abbildung 1: Use Case Diagramm

3.2 Aktoren

Die Aktoren werden durch mehrere einzelne Personen, den Benutzern repräsentiert.

3.3 Use Cases Brief

3.3.1 UC01: Daten darstellen

Beim Programmstart werden die Daten geladen und im Show Case dargestellt.

3.3.2 UC02: Navigation des Zeitfensters

Es werden nur Daten aus dem ausgewählten Zeitbereich dargestellt. Dieser Zeitbereich kann beliebig verändert werden.

3.3.3 UC03: Drill-Down / Roll-Up

Die Daten eines Zeitbereichs können zusätzlich nach Region verfeinert dargestellt werden. Es kann schrittweise in die folgenden Regionen ein- oder ausgezoomt werden: Welt, Land, Provinz und letztlich Stadt.

4 Spezifische Anforderungen

4.1 Funktionale Anforderungen

Die funktionalen Anforderungen werden unter Punkt 3 im Kapitel Use Cases beschrieben.

4.2 Nicht funktionale Anforderungen

4.2.1 Bedienbarkeit

Die Anwendung soll ohne Spezialkenntnisse zu bedienen sein. Es soll nicht notwendig sein, im Voraus ein Benutzerhandbuch zu lesen.

4.2.2 Verständlichkeit

Es soll dem Benutzer klar sein, welche Daten dargestellt werden. Komplexere Controls sollen mit Beschriftungen oder Legenden versehen werden.

4.2.3 Richtigkeit

Die Daten müssen korrekt dargestellt werden. Die Berechnungen für allfällige Auswertungen dürfen nicht fehlerhaft sein.

4.2.4 Geschwindigkeit

Die Datendarstellung in den BI-Controls soll effizient sein. Es soll keine Verzögerung aufgrund von Nachladen der Daten oder aufgrund von Erscheinungseffekten oder ähnlichem geben.

4.3 Schnittstellen

4.3.1 Benutzerschnittstelle

Die Schnittstelle zum Benutzer wird mittels einer Silverlight Webapplikation realisiert.

4.3.2 Softwareschnittstelle

Die Anwendung läuft mit jedem Webbrowser, der das Silverlight Plugin installiert hat.

4.3.3 Datenbankschnittstelle

Die in der Anwendung dargestellten Daten werden von einem SQL Server geladen. Bei der verwendeten Datenbank handelt es sich um ein relationales Data Warehouse.

4.4 Lizenzanforderung

Die Rechte an der Software gehören

- der Entwicklerin Anita Hollenstein
- der HSR

Die Software kann von allen Rechteinhabern nach Belieben weiterverwendet werden.

Business Intelligence mit Silverlight

Evaluation

Inhaltsverzeichnis

1 Einführung

1.1 Zweck

Zu Beginn des Projekts wurden Evaluationen zur Service-Schnittstelle und zu BI-Controls durchgeführt. Dieses Dokument dient der Beschreibung der Erkenntnisse dieser Evaluationen und der Begründung von allfälligen Entscheiden.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments erstreckt sich über die komplette Dauer des Projektes.

1.3 Definitionen und Abkürzungen

- RIA: Rich Internet Application
- MVVM: Model-View-ViewModel
- BI: Business Intelligence
- KPI: Key Performance Indicator

1.4 Referenzen

- MSDN Übersicht WCF RIA Services
  http://msdn.microsoft.com/de-de/library/ee707344%28v=VS.91%29.aspx, letzter Zugriff 11.03.2011
- MSDN Übersicht WCF Data Services
  http://msdn.microsoft.com/de-de/library/cc668792.aspx, letzter Zugriff 11.03.2011
- Vergleich RIA Services / Data Services
  http://jack.ukleja.com/wcf-data-services-vs-wcf-ria-services/, letzter Zugriff 11.03.2011
- MSDN Übersicht ADOMD.NET
  http://msdn.microsoft.com/en-us/library/ms123483.aspx, letzter Zugriff 13.03.2011
- Dokumentation zu TelerikRad Controls für Silverlight
  http://www.telerik.com/help/silverlight/introduction.html, letzter Zugriff 21.03.2011
- Demo zu TelerikRad Controls für Silverlight
  http://demos.telerik.com/silverlight/#Controls, letzter Zugriff 21.03.2011
- Microsoft Pivot Viewer
  http://en.wikipedia.org/wiki/Live_Labs, letzter Zugriff 11.03.2011
  http://www.silverlight.net/learn/pivotviewer/collection-xml-schema/, letzter Zugriff 11.03.2011

2 Evaluation Service-Schnittstelle

2.1 WCF RIA Services

2.1.1 Kurzbeschreib

Geeignet für End-to-End Business Applikationen, basierend auf existierenden Daten.

2.1.2 Detailliert

WCF RIA Services vereinfacht die Entwicklung von n-Tier Rich Internet Applikationen (RIA). RIAs sind Internetanwendungen wie z.B. Silverlight, die eine vielfaltige Menge an Interaktionsmöglichkeiten mit ihrer Benutzeroberfläche bieten.
RIA Services liefert Framework Komponenten, Tools und Services, welche die Applikationslogik des Servers dem Client so zur Verfügung stellt, dass dieser die Logik nicht duplizieren muss. Somit wird sichergestellt, dass die Applikationslogik nicht sowohl auf der Präsentationsebene, als auch auf der mittleren Ebene vorhanden ist.
Für den Zugriff auf die Daten kann jeder beliebige Typ von Datenzugriffsebene verwendet werden. Dazu gehören zum Beispiel:

- Enitity Data Model
- LINQ to SQL-Objektmodel
- Common Language Runtime-Objekt

2.2 WCF Data Services

2.2.1 Kurzbeschreib

Geeignet für den Zugriff auf Daten über einen Service, der OData Feeds zur Verfügung stellt.

2.2.2 Detailliert

WCF Data Services ermöglicht mithilfe des Open Data Protocol das Erstellen und Verwenden von Datendiensten für das Internet. Mit OData können Daten als Ressourcen, welche über URIs adressierbar sind, verfügbar gemacht werden. Somit können mit der RESTful-Architektur und mit den standardmässigen HTTP-Methoden GET, PUT, POST und DELETE auf Daten zugegriffen und Änderungen gemacht werden.

2.3 Direktvergleich RIA / Data Services

WCF RIA Services	WCF Data Services
Präskriptiver Ansatz zu N-Tier Applikationsentwicklung	Datenmodel wird als RESTful Webservice veröffentlicht
Speziell für End-to-End Silverlight & ASP.NET Anwendungen entwickelt	Ziel ist eine Zusammenarbeit über verschiedene MS Produkte wie SQL 2008 R2, Azure, Excel 2010 und SharePoint 2010
Client und Server sind zusammen implementiert	Client und Server sind lose gekoppelt
Der Service Layer kann mit Logik ergänzt werden	Der Service Layer veröffentlicht „rohe“ Datenquellen

Tabelle 5: Vergleich WCF RIA Services und WCF Data Services

Zu den Gemeinsamkeiten der beiden Services gehören:

- WCF-basierte Architektur
- Als WCF Service gehostet
- RESTful Architektur
- Clientseitige Bibliotheken ermögliche LINQ-Abfragen

2.4 ADOMD.NET

ADOMD.NET ist ein Microsoft .NET Framework Data Provider, welcher der Kommunikation mit den Microsoft SWL Server Analysis Services dient. ADOMD.NET verwendet XML für Analysis um mit den analytischen Datenquellen zu kommunizieren. Dabei werden über TCP/IP oder HTTP Verbindungen SOAP-Anfragen und Antworten übermittelt. Die Kommandos können in Multidimensional Expressions (MDX), Data Mining Extensions (DMX) oder Analysis Services Scripting Language (ASSL) gesendet werden.
Mit dem ADOMD.NET Model können analytische Daten, Key Performance Indicators und Mining Models abgefragt und modifiziert werden.

2.5 Entscheid

Aufgrund des Vergleichs der verschiedenen Kommunikationsvarianten fällt der Entscheid für die Kommunikationsvariante auf WCF RIA Services. RIA Services wurde speziell für N-Tier RIAs wie z.B. Silverlight entwickelt und liefert somit ein optimales Framework für die Business Intelligence Applikation dieses Projekts.
Die zusätzlich mögliche Logik auf dem Service Layer erlaubt es die Daten so weit zu filtern, dass nur die wirklich benötigten Daten an den Client gesendet werden. Die Ladezeiten für den Client sind somit kürzer und effizienter.

3 Evaluation Controls

Für die Demo-Applikation werden die Telerik RadControls für Silverlight verwendet. RadControls stellt eine grosse Menge an unterschiedlichen Controls für Silverlight Applikationen zur Verfügung. Controls, welche sich für eine Business Intelligence Applikation eignen wurden evaluiert. Dabei wurden sie auf die Datenschnittstelle und zusätzliche Funktionen wie zum Beispiel Drill-Down untersucht.
Des Weiteren wurde auch das Silverlight Control PivotViewer von Microsoft analysiert.

3.1 RadGridView

Das RadGridView Control dient der tabellarischen Darstellung von Daten. Als Datenquelle dient eine einfache Liste von Entitäten, deren Attribute in der Tabelle dargestellt werden. Jedes einzelne Attribut kann nach Bedarf angezeigt oder verborgen werden. Es ist möglich, die Daten direkt aus dem RIA Service Context zu laden. Leider kann dabei das MVVM Pattern nicht eingehalten werden. Aus diesem Grund werden die Daten vom Context in das ViewModel der GridView geladen und per Property Binding der View zur Verfügung gestellt.

3.1.1 Hierarchical GridView

Die GridView erlaubt eine verschachtelte Darstellung von Tabellen. Es können also einzelne Reihen ausgeklappt werden und eine Liste mit weiteren Details dazu angezeigt werden. Dies bedingt, dass die Entität der Datenquelle ein Attribut enthält, das eine weitere Liste von einer Entität enthält.
Enthält eine Entität zum Beispiel eine Liste von Produktkategorien, kann diese ausgeklappt werden und die einzelnen Produkte anzeigen, sofern jeder Produktkategorie eine Liste mit Produkten zugehört.

3.1.2 RadSparkline

In den Tabellenfeldern lassen sich auch Sparklines darstellen. Dies sind Linien, die mittels x- und y-Werten eine Kurve, wie zum Beispiel der Verlauf der Verkaufszahlen eines Produkts, darstellen.
Dazu muss jedem Produkt ein Attribut mit einer Liste zugehören. Diese Liste enthält Einträge mit dem Datum für den x-Wert und der Verkaufszahl dieses Datums für den y-Wert.

3.2 RadChart

Das RadChart Control dient der Darstellung von diversen verschiedenen Diagrammen und Graphen. Daten können einzig mittels Data Binding hinzugefügt werden.
Mittels Anpassungen im XAML Code kann die Erscheinung und die Art der Diagramme und Graphen nach Belieben angepasst werden. Das Grundgerüst und die Datenanbindung bleibt jedoch immer dieselbe.

3.2.1 Drill-Down

RadChart ermöglicht die Funktion eines Drill-Downs. Dazu muss, wie bereits bei der Hierarchical GridView und der RadSparkline, ein Attribut mit einer weiteren Liste vorhanden sein. Diese zusätzlichen Daten können nicht bei einem Drill-Down-Kommando nachgeladen werden, sondern müssen bereits zusammen mit den Basisdaten geladen werden. Dies führt dazu, dass unter Umständen eine sehr grosse Menge an Daten geladen werden müsste. Daraus folgen lange Ladezeiten und Verzögerungen der Applikation.

3.2.2 Zoom & Scroll

Die Zoom & Scroll Funktion ermöglicht bei der Darstellung einer grossen Datenmenge in die x-Achse hinein zu zoomen und mittels Scrollbar auf der x-Achse zu navigieren. Die Funktion verhindert, dass x-Werte zu nahe aufeinanderliegen und nicht mehr gelesen werden können.

3.3 RadTimeBar

Das RadTimeBar Control stellt eine ihm zugewiesene Zeitspanne dar. Über diese Zeitspanne kann gescrollt und navigiert werden. Des Weiteren ist die Selektion eines Zeitbereichs in dieser Zeitspanne möglich. Alle an die TimeBar gebundenen Controls filtern ihre dargestellten Daten auf diesen selektierten Zeitbereich.
Mittels Zuweisung einer Datenquelle, welche neben dem Datum als x-Wert auch einen y-Wert enthält, kann die Darstellung der TimeBar mit einer Sparkline ergänzt werden.

3.4 RadMap

Das RadMapControl dient der Visualisierung von geographischen Daten und erlaubt auf Karten zu navigieren. Mit einem zusätzlichen Geocode Provider können einzelne Punkte auf einer Karte in eine Adresse umgewandelt oder umgekehrt eine Adresse in einen Punkt auf der Karte umgewandelt werden.

3.5 RadBulletGraph

Das RadBulletGraph Control erlaubt den direkten Vergleich von zwei Metriken, wie z.B. KPIs. Es ist ein wichtiger Bestandteil von modernen BI-Lösungen.
Die Metriken werden per Data Binding an Properties des ViewModels gebunden.

3.6 Microsoft PivotViewer

Der PivotViewer wurde von den Microsoft Live Labs hergestellt. Dieses Unternehmen wurde im Oktober 2010 von Microsoft aufgelöst. Folglich funktionieren auch die Webseiten der Live Labs nicht mehr. Dies hat zur Folge, dass über den PivotViewer nur noch wenige Informationen und Hilfen zur Verfügung stehen. Ebenfalls funktionieren auch Demoversionen nicht mehr vollständig, weil sie auf Daten zugreifen, die über die ehemaligen Webserver von Live Labs zugänglich waren.
Obwohl der PivotViewer weiterhin als Silverlight Control zur Verfügung gestellt wird, kann davon ausgegangen werden, dass dieser nicht mehr weiterentwickelt wird.

Die Daten, welche im PivotViewer dargestellt werden sollen, werden diesem mittels einem CXML File übergeben. Eine andere Möglichkeit um die Daten zur Verfügung zu stellen, wie zum Beispiel der Zugriff auf einen SQL Server mittels WCF-Service, gibt es nicht.
Aus der Beschreibung des CXMLs im folgenden Unterkapitel wird ersichtlich, dass dieses nur für kleinere Datenmengen geeignet ist. Grundsätzlich enthalten Datenbanken für Business Intelligence Anwendungen sehr viele komplexe Datensätze. Dies macht es praktisch unmöglich die BI-Daten mit dem PivotViewer darzustellen.

3.6.1 CXML

Collection XML (CXML) ist das Schema, welches strukturierte Daten beschreibt um diese im PivotViewer darzustellen. Ein CXML besteht aus den folgenden Elementen:

Collection	Beinhaltet alle Elemente der Collection
FacetCategories	Gruppiert alle Kategorien
FacetCategory	Gruppiert mehrere Kategorien des gleichen Typs
Facet	Kategorie, dient dem Filtern, Sortieren oder Darstellen detaillierter Informationen eines Items
Items	Gruppiert die einzelnen Einträge
Item	Individueller Eintrag in einer Collection, enthält eine Beschreibung, ein zugehöriges Bild, sowie die zugehörigen Kategorien
Description	Beschreibungstext eines Eintrages

Tabelle 6: CXML Elemente

In der untenstehenden Abbildung ist dargestellt, wie ein einzelner Hello World Eintrag im CXML-File so beschrieben wird, damit er im PivotViewer mit den oben erwähnten Elementen dargestellt werden kann. Man stellt fest, dass das CXML File bereits mit einem einzelnen Eintrag und einer einzelnen Kategorie schon sehr umfangreich ist.
Das CXML muss entweder von Hand erstellt oder mit dem PivotViewer Collection Tool aus einer Excel-Liste generiert werden. Ein Export aus SQL Datenbanken ist nicht möglich.

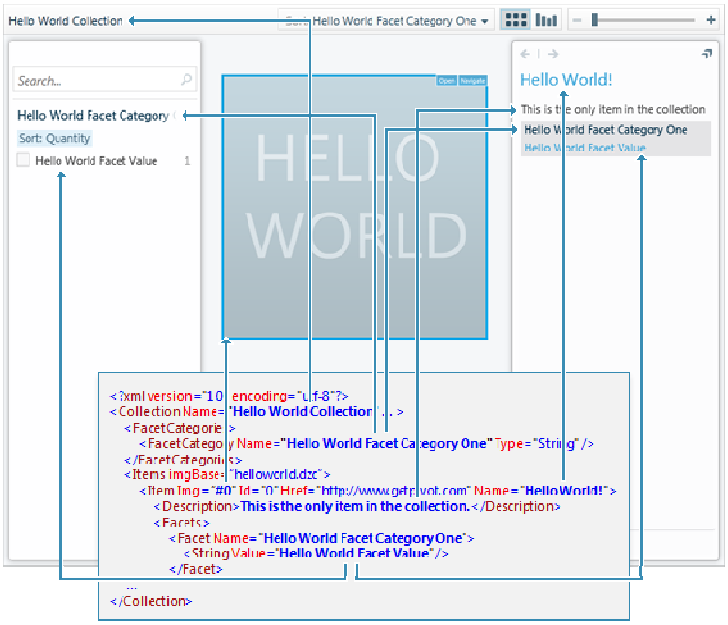


Abbildung 2: Darstellung eines CXML-Files

Business Intelligence mit Silverlight

Software Architektur Dokument

Inhaltsverzeichnis

1 Einführung

1.1 Zweck

Dieses Dokument dient der Beschreibung der Softwarearchitektur des Projekts.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments erstreckt sich über die komplette Dauer des Projekts.

1.3 Definitionen und Abkürzungen

- WCF: Windows Communication Foundation
- RIA: Rich Internet Application
- MS: Microsoft
- MVVM: Model-View-ViewModel
- DTO: Data Transfer Object
- UC: Use Case
- SP: Stored Procedure

1.4 Referenzen

- Silverlight Architektur Übersicht http://sandrinodimattia.net/blog/post/Making-WCF-RIA-Services-work-in-a-DMZMultitier-architecture-using-Application-Request-Routing.aspx, letzter Zugriff 19.05.2011
- MSDN Übersicht WCF RIA Services http://msdn.microsoft.com/de-de/library/ee707344%28v=VS.91%29.aspx, letzter Zugriff 20.05.2011
- Übersicht WCF RIA Services http://www.silverlight.net/getstarted/riaservices/, letzter Zugriff 20.05.2011
- Einführung WCF RIA Services http://www.silverlightshow.net/items/WCF-RIA-Services-Part-1-Getting-Started.aspx, letzter Zugriff 20.05.2011

2 Architektonische Darstellung

2.1 Architekturübersicht

Die Business Intelligence Silverlight-Applikation basiert auf einer Client Server Architektur mit WCF RIA Services.

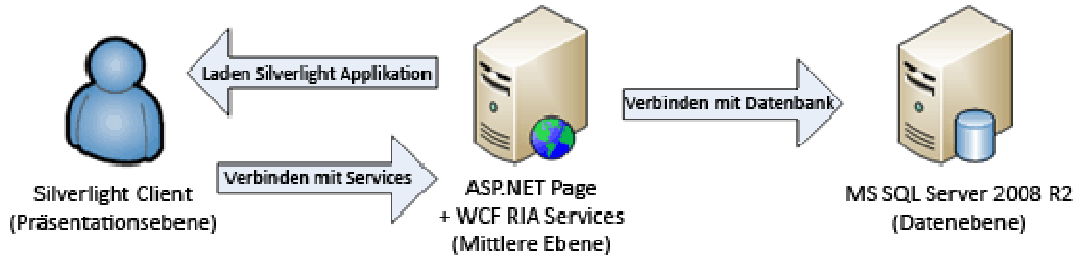


Abbildung 3: Übersicht Architektur WCF RIA Services

2.2 WCF RIA Services

WCF RIA Services vereinfacht die Entwicklung von N-Tier Projekten für Rich Internet Applikationen (RIA), wie z.B. Silverlight Applikationen. RIA Services stellt Framework-Komponenten, Tools und Dienste zur Entwicklung der Anwendungslogik auf der mittleren Ebene zur Verfügung. Somit wird die gleichzeitige Entwicklung der Anwendungslogik auf der Präsentations- und der mittleren Ebene verhindert.

Auf der Serverseite werden Domain Services, Domain Entities und die zugehörige Logik implementiert. Auf der Clientseite generiert RIA Services korrespondierende Klassen, die ein einfaches Aufrufen auf diese Domain Services ermöglichen.

2.2.1 Mittlere Ebene

In der mittleren Ebene werden Domänendienste definiert, welche die Entitäten und Vorgänge für die Domänengeschäftslogik enthalten.

Der Zugriff auf den MS SQL Server 2008 R2 erfolgt über das ADO.NET Entity Framework.

2.2.2 Präsentationsebene

Für jeden Domänendienst der in der mittleren Ebene definiert ist, wird automatisch eine Domänenkontextklasse generiert. Diese Klasse enthält Abfrage- und Änderungsmethoden, die mit der entsprechenden Methode des Domänendienstes kommuniziert. Dabei werden die Aufrufe sowie die Antworten zwischen den beiden Parteien weitergeleitet.

3 Architektonische Ziele & Einschränkungen

3.1 Ziele

3.1.1 N-Tier

Die Software soll in eine Präsentationsebene, eine mittlere Ebene und eine Datenebene aufgeteilt werden. Somit entsteht eine N-Tier Anwendung mit einfacher Verwaltbarkeit.

3.1.2 MVVM

Die Präsentationsebene soll die Regeln des Model-View-ViewModel-Musters einhalten. Somit wird also die View keine Logik enthalten, sondern einzig für die Darstellung der Benutzeroberfläche zuständig sein. Das ViewModel stellt der View Properties für die Datendarstellung zur Verfügung und nimmt von der View Commands entgegen. Das Model besteht aus dem DomainContext und den Entities, welche von RIA Services automatisch anhand der DomainServices und der DTO's der mittleren Ebene generiert werden.

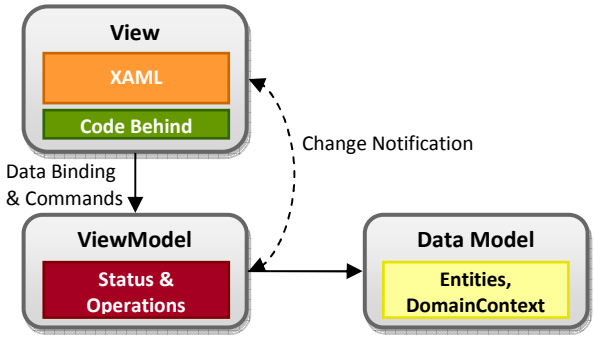


Abbildung 4: MVVM mit RIA Services

3.2 Einschränkungen

Die Anwendung ist nicht für die Darstellung von sensitiven Daten gedacht. Aus diesem Grund werden keine Sicherheitsfunktionen, wie zum Beispiel das Authentisieren eines Benutzers, implementiert.

4 Logische Architektur

4.1 Übersicht

Die Klassen werden in zwei Hauptpackages aufgeteilt. Das SalesDashboard Package repräsentiert dabei die Präsentationsebene und das Web Package die mittlere Ebene. Innerhalb dieser zwei Packages gibt es weitere Subpackages, welche das System weiter gliedern.

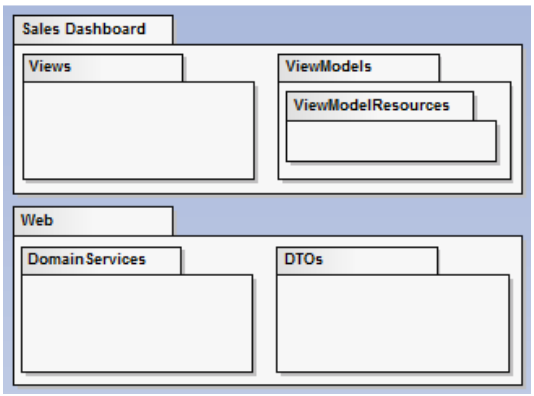


Abbildung 5: Package Übersicht

4.2 Client

4.2.1 Aufbau

Der Client ist nach dem unter Punkt 3.1.2 beschriebenen MVVM Muster aufgebaut. Weil das Model von den RIA Services automatisch im Hintergrund generiert wird, ist es in der untenstehenden Abbildung nicht ersichtlich.

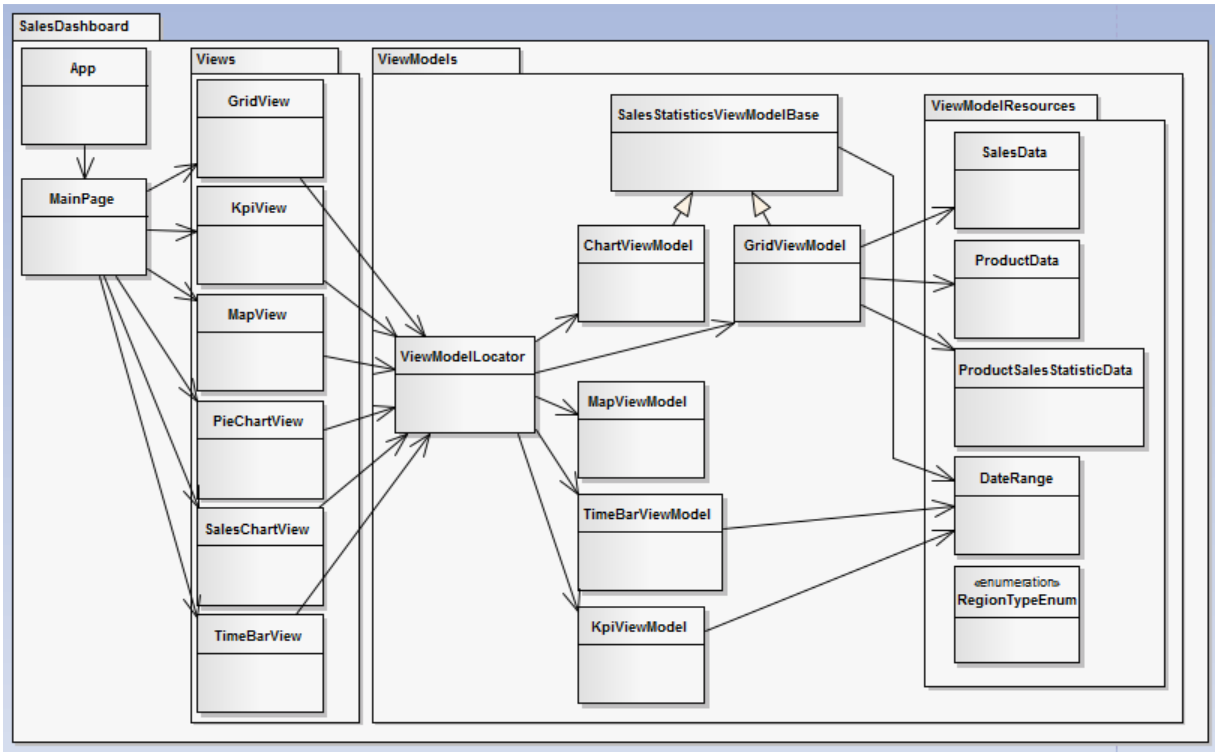


Abbildung 6: Übersicht Clientarchitektur

4.2.2 Views

Jedes Rad Control Element, das im Sales Dashboard dargestellt wird, ist in einem eigenen UserControl definiert. Die Definition geschieht dabei mittels XAML Code. Der Code Behind sollte grundsätzlich nur die Standardmethode InitializeComponent() im Konstruktor enthalten und keine weiteren Funktionen. Eine Ausnahme wird bei Rad Controls gemacht, für welche zwingend zusätzliche Logik im Code Behind benötigt wird.
Mittels Command Binding und der EventToCommand Funktion des MVVM Light Toolkits werden auch Commands und Events an das ViewModel weitergeleitet.

4.2.3 ViewModels

Jede View besitzt ihr eigenes ViewModel. Einzig die SalesChartView und die PieChartView teilen sich das ViewModel, da sie praktisch gleich aufgebaut sind. Des Weiteren gibt es die Klasse SalesStatisticsViewModelBase. Diese dient als Basis für die teilweise identisch aufgebauten ChartViewModel und GridViewModel.
Als Schnittstelle zwischen den ViewModels und den Views dient der ViewModelLocator. Dieser wird im App.xaml als globale Ressource für die einzelnen Views deklariert und dient sozusagen als ViewModel Factory.

4.2.4 ViewModelResources

Das Package ViewModelResources enthält verschiedene Hilfsklassen für die ViewModels. Beispielsweise verlangen einige Rad Controls die Daten in einer bestimmten Gruppierung, damit diese korrekt dargestellt werden können. Für diese Gruppierungen werden zusätzliche Klassen benötigt.

4.2.5 Verwendete Libraries

Telerik Rad Controls

Zur Einbindung der Rad Controls in den Programmcode werden für die jeweils verwendeten Controls die entsprechenden Libraries verwendet.

MVVM Light Toolkit

Vom MVVM Light Toolkit werden folgende Funktionen verwendet:

- *ViewModelBase*
  Durch Ableitung der ViewModel Klassen von ViewModelBase steht für die Properties der ViewModels die Methode RaisePropertyChanged() zur Verfügung. Diese ermöglicht es die Views über Änderungen der Properties in Kenntnis zu setzen.
- *ViewModelLocator*
  Der ViewModelLocator dient als Schnittstelle zwischen Views und ViewModels. Er erlaubt eine einfachere Verwaltung der ViewModels und ermöglicht es zwischen Designzeit- und Laufzeit-ViewModels zu unterscheiden.
- *Messages*
  Manche Properties sind nicht nur für die View eines ViewModels sondern auch für weitere ViewModels von Bedeutung. Um diese über die Änderung eines Properties in Kenntnis zu setzen, werden Messages an die Klasse Messenger gesendet. Die Messages enthalten die Identifikation des Properties und dessen neuen Wert. Die ViewModels registrieren sich beim Messenger für den Empfang jener Messages, welche für sie relevant sind.

- *Command Binding / EventToCommand*
  Um Commands von den Views in die ViewModels weiterzuleiten wird das von MVVM Light zur Verfügung gestellte Command Binding verwendet.
  Grundsätzlich werden keine Event Bindings unterstützt. Die EventToCommand Funktion ermöglicht jedoch im XAML Code beliebige Events über EventTrigger an Commands weiterzuleiten.

4.3 Web

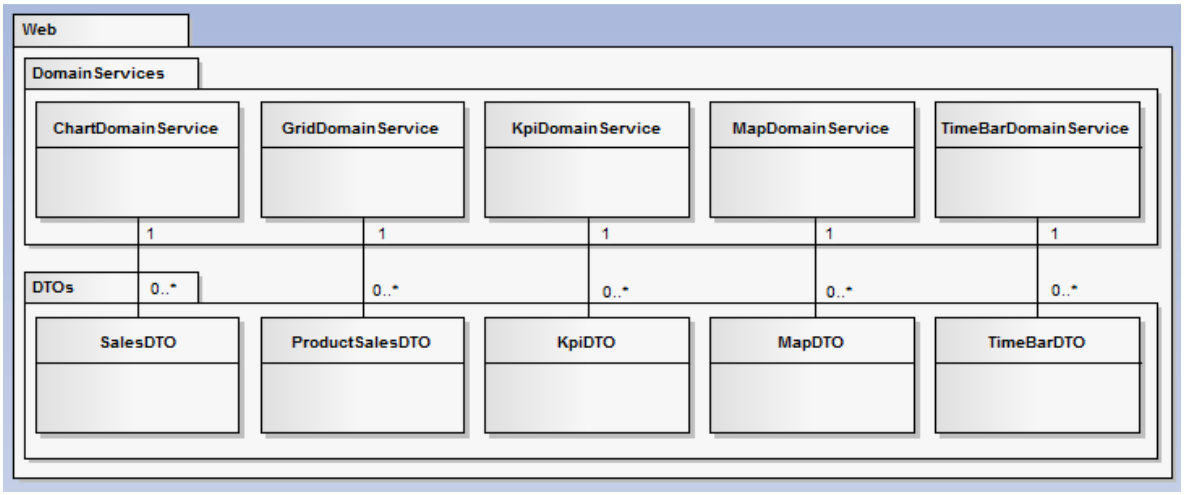


Abbildung 7: Übersicht Webarchitektur

4.3.1 DomainServices

Die Domain Service Klassen werden durch die Ableitung der Klasse LinqToEntitiesDomainService an das ADO.NET-Entitätsmodell gebunden. Jede Domain Service Klasse implementiert eine oder mehrere Abfragemethoden, welche ein IQueryable<Entity> zurückgeben. Für jede Entität, die von einem DomainService zurückgegeben wird, generiert das RIA Services-Framework eine Entität in der Präsentationsebene.
Über den DomainContext kann die Präsentationsebene die Abfragemethoden der DomainServices abfragen.

4.3.2 DTOs

Die DTO-Klassen dienen als Entitäten, welche von den Abfragemethoden der Domain Service Klassen zurückgegeben werden. Ein DTO kann nicht für mehrere Domain Services gleichzeitig verwendet werden.
Jedes DTO benötigt mindestens ein Property welches mit einem Key Attribut versehen ist, um jede einzelne Entität eindeutig identifizieren zu können.

5 Use Case Darstellung

5.1 UC1: Daten darstellen

Beim Start der Applikation werden alle Controls, also die einzelnen Views, in die Page geladen. Dazu werden auch erste Daten zur Darstellung geladen.
Da der Ablauf der Initialisierung für jedes Control gleich abläuft wurde der Ablauf zusammengefasst und die einzelnen Komponenten mit dem Platzhalter X für jedes Control ersetzt.

Zu Beginn wird über die App Klasse die globale Ressource ViewModelLocator erstellt sowie die Komponenten der einzelnen Views initialisiert. Der ViewModelLocator instanziiert alle ViewModels. Jedes ViewModel initialisiert danach seinen Context indem es über den DomainContext die Query zum Laden seiner Daten aufruft. Der vom RIA Service generierte DomainContext leitet die Query an die entsprechende Methode des zugehörigen DomainServices weiter. Sobald das Laden der Daten fertiggestellt ist, werden diese dem Data-Property des ViewModels zugewiesen. Per RaisePropertyChanged() wird die View über die Änderung in Kenntnis gesetzt und aktualisiert seine externe Darstellung.

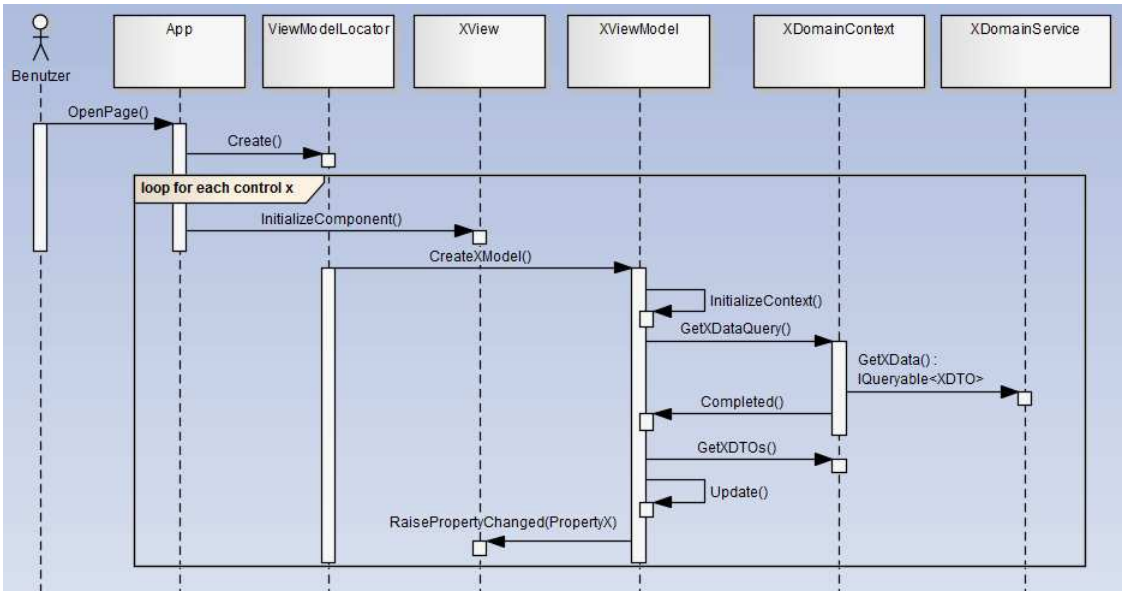


Abbildung 8: Ablaufdiagramm UC1: Daten darstellen

5.2 UC2: Navigation des Zeitfensters

Die Daten für das TimeBar Control werden einmalig zu Applikationsstart geladen. Das zu Beginn selektierte Zeitfenster ist per Default auf die letzten beiden Monate des Zeitbereichs der geladenen Daten gesetzt. Sobald diese Daten geladen wurden, wird diese Selektion gesetzt und der View mitgeteilt.

Wird die Selektion vom Benutzer geändert, dann wird dies mit einem Command dem ViewModel mitgeteilt. Dieses setzt den DateRange neu und sendet eine Notification Message mit dem Inhalt des neuen DateRanges an die Messenger-Klasse. Das Start- und Enddatum werden in eine DateRange zusammengefasst um das Senden von zwei Messages nacheinander zu verhindern.

Die beiden ViewModels SalesStatisticsViewModelBase und KpiViewModel werden nun aufgrund ihrer früheren Registrierung beim Messenger über die Änderung des Zeitfensters informiert und aktualisieren ihre Daten.

Auf die Darstellung des Updates, sowie aller anderen Funktionen von SalesStatisticsViewModelBase KpiViewModel und Messenger wird in diesem Sequenzdiagramm verzichtet. Das Update-Verhalten kann im Sequenzdiagramm zu UC3 nachvollzogen werden.

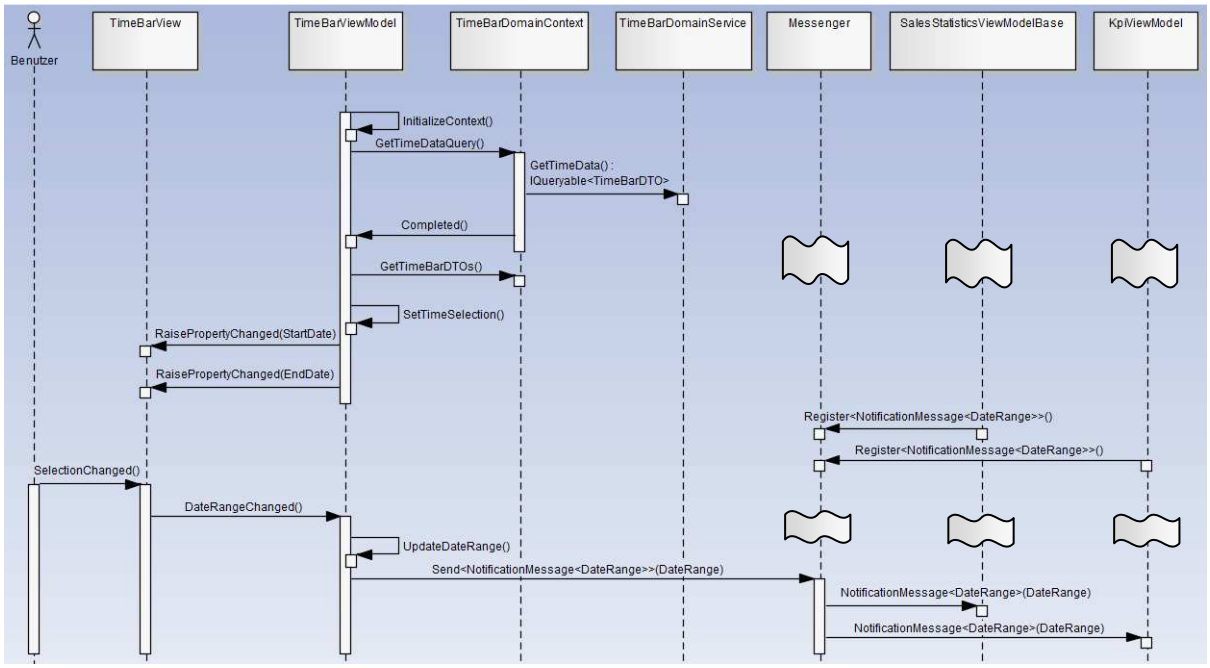


Abbildung 9: Ablaufdiagramm UC2: Navigation des Zeitfensters

5.3 UC3: Drill-Down / Roll-Up

Bei Änderungen der Regionsselektion in der MapView wird dies mittels Command Binding dem MapViewModel mitgeteilt. Dieses sendet dem Messenger ein Notification Message mit dem die neue Regionsselektion, in diesem Fall das Land, als Parameter mitgegeben wird.
Die SalesStatisticsViewModelBase, welche sich zuvor bei der Messenger Klasse für Messages dieser Art registriert hat, erhält den Typ der selektierten Region (Welt, Land, Provinz oder Stadt) und deren neuen Wert zugesendet. Die Subklassen der SalesStatisticsViewModelBase, also ChartViewModel und GridViewModel, rufen darauf folgend ihre UpdateData() Methode auf.
Im Sequenzdiagramm wird nur das weitere Vorgehen des ChartViewModels aufgezeigt, da sich das GridViewModel grundsätzlich gleich verhält.
Beim Update der Daten werden als erstes die vorhandenen Daten im ChartDomainContext gelöscht. Danach wird die Query, welche der selektierten Region entspricht, aufgerufen und das selektierte Land als Parameter mitgegeben. In diesem Beispiel handelt es sich also um die GetCountrySalesStatisticsQuery.
Der DomainContext leitet die Query an den ChartDomainService weiter, welcher per Stored Procedure Aufruf die nach Land gefilterten Datensätze von der Datenbank lädt. Sobald der DomainContext die Datensätze vollständig geladen hat, werden diese dem ChartData Property zugewiesen und die ChartView per RaisePropertyChanged über die Änderung in Kenntnis gesetzt. Die Daten werden nun nach dem selektierten Land gefiltert dargestellt.

Im Zusammenhang mit der Query ist zu beachten, dass normalerweise auch die selektierte Zeitspanne als Parameter mitgegeben wird. Diese wurde jedoch weggelassen um die Darstellung zu vereinfachen.

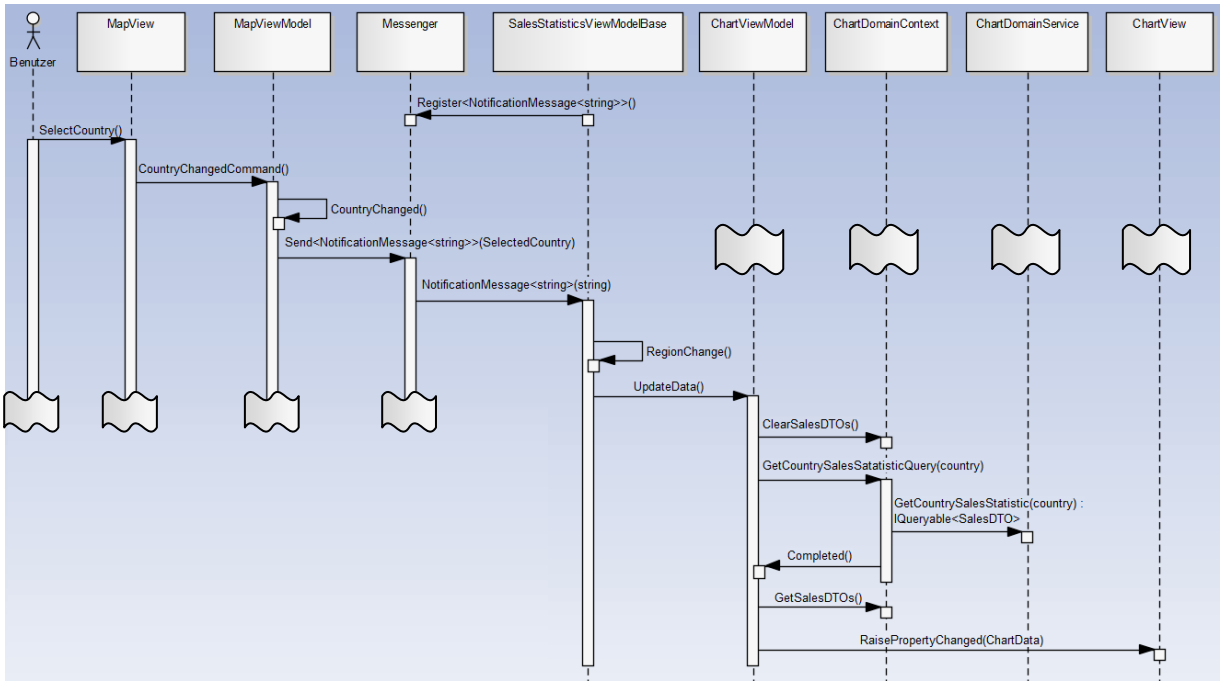


Abbildung 10: Ablaufdiagramm UC3: Drill-Down / Roll-Up

6 Daten

Als Datenquelle wird eine Beispieldatenbank von Microsoft verwendet. Dabei handelt es sich um das AdventureWorks Data Warehouse, Version 2008 R2.
Die Datenauswahl des AdventureWorks DWs wurde auf die in der Abbildung 11 ersichtliche Struktur begrenzt. Trotz der Begrenzung handelt es sich bei der Auswahl um eine sehr grosse Menge an Datensätzen.

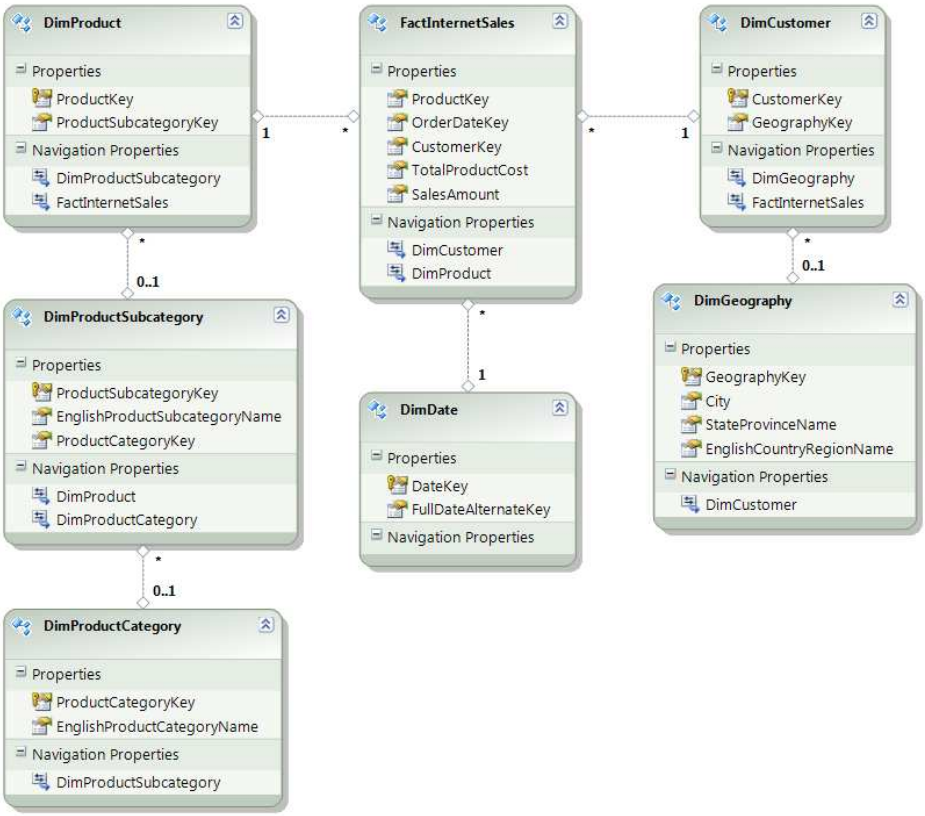


Abbildung 11: Struktur der Datenauswahl aus AdventureWorks DW

6.1 Stored Procedures

Im Normalfall wird jede einzelne Tabelle von der Datenbank zum Server geschickt um dann dort verbunden und ausgewertet zu werden. Dies ist vor allem ein Nachteil, wenn sehr viele Tabellen miteinander in Verbindung gebracht werden müssen.
Eine Abhilfe leistet hier der Einsatz von Stored Procedures. Diese erlauben es, die Tabellen bereits auf der Datenbank zu vereinen und die Daten mit vordefinierten Werten oder vom Server gelieferten Parametern zu filtern. An den Server werden anschliessend nur jene Daten und Tabellenattribute geliefert, welche er wirklich benötigt.
Anstelle von Entitätstypen der einzelnen Tabellen arbeitet der Server nun mit komplexen Typen, welche dem Aufbau des Resultats der Stored Procedures entsprechen.

6.1.1 SP: GetTimeData

Diese Stored Procedure wird vom TimeBarDomainService aufgerufen. Sie verbindet die FactInternetSales Tabelle mit der DimDate Tabelle und liefert für jeden Tag, an dem mindestens ein Verkauf stattgefunden hat, die Summe der Verkaufsdaten. Der FullDateAlternateKey liefert dabei die benötigte Zeitspanne des TimeBars und die SalesAmount-Summe dient der Darstellung der Sparkline im TimeBar.

6.1.2 SP: GetMapData

GetMapData liefert alle Länder-, Provinz- und Städtenamen die in der Tabelle DimGeography abgelegt sind. Die Daten werden dabei mit der Rollup-Funktion gruppiert.
Enthält die Tabelle zum Beispiel die Einträge Schweiz, St. Gallen, Wil und Sargans, dann sähe eine Rollup-Gruppierung wie folgt aus:

	Country	Province	City
1	NULL	NULL	NULL
2	Schweiz	NULL	NULL
3	Schweiz	St. Gallen	NULL
4	Schweiz	St. Gallen	Wil
5	Schweiz	St. Gallen	Sargans

Tabelle 7: Group by Rollup Resultat

Die 1. Reihe liefert nun Werte der ganzen Welt, die 2. Reihe Werte der ganzen Schweiz usw. Dies bietet eine optimale Vorlage für das Abfüllen der Daten in die ComboBoxen der Benutzeroberfläche und ermöglicht einen einfachen Drill-Down und Roll-Up.

Eine normale Gruppierung würde zu folgendem Ergebnis führen:

	Country	Province	City
1	Schweiz	St. Gallen	Wil
2	Schweiz	St. Gallen	Sargans

Tabelle 8: Group Resultat

Hier werden nur die Daten der tiefsten Drill-Down Stufe geliefert. Für die höheren Stufen müsste die Tabelle „auseinander“ genommen werden.

6.1.3 SP: GetSalesStatistics

Um die Daten des Sales Charts und des Pie Charts darzustellen werden folgende Elemente benötigt: FullDateAlternateKey für das Datum, EnglishProductCategoryName für die Produktkategorie und die Summe der Verkaufskosten. Dazu müssen die Tabellen FactInternetSales, DimDate, DimProductCategory, DimProductSubcategory und DimProduct miteinander verbunden werden.

6.1.4 SP: GetCountrySalesStatistics

GetCountrySalesStatistics basiert auf der GetSalesStatistics Stored Procedure, nimmt jedoch zusätzlich einen varchar-Parameter entgegen um die Daten auf das im Parameter angegebene Land zu filtern.
Da nun zusätzlich auch die DimGeography Tabelle benötigt wird, müssen alle Tabellen der in Abbildung 11 ersichtlichen Struktur miteinander vereint werden.

6.1.5 SP: GetProvinceSalesStatistics

Praktisch identisch mit GetCountrySalesStatistics, nimmt jedoch als Parameter einen Provinznamen entgegen und filtert folglich nach Provinzen.

6.1.6 SP: GetCitySalesStatistics

Praktisch identisch mit GetCountrySalesStatistics, nimmt jedoch als Parameter einen Städtenamen entgegen und filtert folglich nach Städten.

6.1.7 SP: GetProductSalesStatistics

Funktioniert gleich wie GetSalesStatistics, gibt jedoch für die Darstellung im GridView zusätzlich auch die Produktsubkategorie zurück.

6.1.8 SP: GetProductCountrySalesStatistics

Diese Stored Procedure hat die gleiche Funktion wie GetCountrySalesStatistics, basiert jedoch auf GetProductSalesStatistics.

6.1.9 SP: GetProductProvinceSalesStatistics

Praktisch identisch mit GetProductCountrySalesStatistics, nimmt jedoch als Parameter einen Provinznamen entgegen und filtert folglich nach Provinzen.

6.1.10 SP: GetProductCitySalesStatistics

Praktisch identisch mit GetProductCountrySalesStatistics, nimmt jedoch als Parameter einen Städtenamen entgegen und filtert folglich nach Städten.

6.1.11 SP: GetKPIs

GetKPIs liefert die Daten für die BulletGraphs zur Darstellung der Key Performance Indicators. Als KPI werden der Vergleich der durchschnittlichen Verkaufs- und Produktionskosten sowie deren Gesamtsumme verwendet.
Als Basis des Vergleichs dienen dabei die selektierte Zeitspanne sowie die gleiche Zeitspanne des Vorjahrs.
Der SP werden die Angaben für diese beiden Zeitspannen als Parameter mitgeliefert. Diese gibt dann, gefiltert nach den Zeitspannen, die benötigten Vergleichszahlen zurück.

Business Intelligence mit Silverlight

UI Dokumentation

Inhaltsverzeichnis

1 Übersicht Sales Dashboard

Das User Interface ist eine Microsoft Silverlight Applikation. Es besteht aus einer Zusammenstellung von verschiedenen BI-Controls. Verwendet wurden dabei die Telerik RadControls für Silverlight.
Die Zusammenstellung der Controls stellt ein Sales Dashboard dar. Das heisst, Verkaufszahlen eines Unternehmens werden auf einen Blick in verständlicher Art und Weise dargestellt. Das Sales Dashboard ermöglicht es die Daten wunschgemäss detailliert oder eher übersichtsmässig darzustellen.
Es bestehen folgende Möglichkeiten um die Daten einzuschränken oder zu erweitern:

- Anpassung der Zeitspanne der darzustellenden Verkaufszahlen
- Anpassung der geographischen Region
- Zusammenfassen der Produkte in Produktkategorien

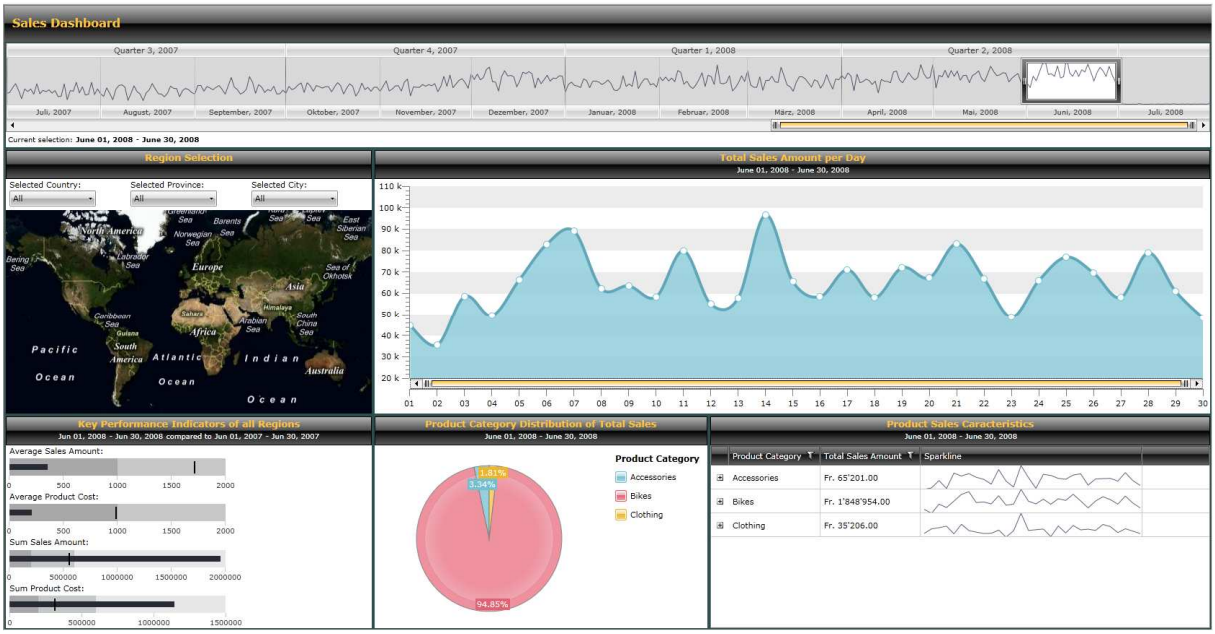


Abbildung 12: Sales Dashboard User Interface

2 Übersicht Controls

Die einzelnen Komponenten des Sales Dashboards – die sogenannten Controls – stellen die Verkaufszahlen dar oder dienen der Einstellung der darzustellenden Datenbereiche. Dies erfordert, dass die Controls eng miteinander verbunden sind und zum Beispiel über eine Änderung der Zeitspanne sofort in Kenntnis gesetzt werden.

2.1 Time Bar

Das Time Bar Control ermöglicht es dem Benutzer über die Zeitspanne der darzustellenden Daten zu navigieren. Die Zeitspanne kann beliebig erweitert und/oder verschoben werden. Zu beachten gilt, dass je grösser die Zeitspanne ist, umso grösser ist auch die Menge der darzustellenden Daten. Dies könnte zu eventuellen Verzögerungen beim Datenladen führen.
Bei einer Änderung der Zeitspanne müssen folgende Controls darüber in Kenntnis gesetzt werden: Sales Chart, Pie Chart, Product Grid und die Bullet Graphs.

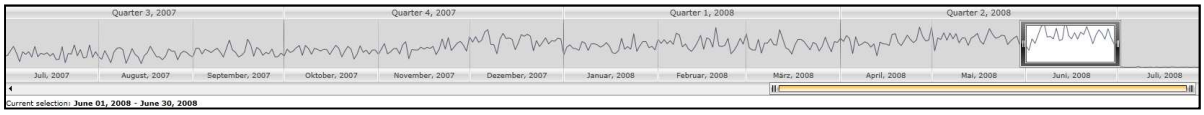


Abbildung 13: Time Bar Control

2.2 Map

Das Map Control bietet die Möglichkeit auf den Daten Drill-Down- oder Roll-Up-Operationen in Bezug zur geographischen Region durchzuführen. Zur Auswahl stehen dabei alle Länder, Provinzen und Städte, welche in der Datenbank eingetragen sind.
Die Auswahl wird mittels Selektion in den Combo Boxen getroffen. Die Karte dient der visuellen Unterstützung, indem in die selektierten Regionen herein gezoomt wird.
Wird die Regionsselektion geändert, dann werden folgende Controls darüber in Kenntnis gesetzt: Sales Chart, Pie Chart und Grid.

Abbildung 14: Map Control

2.3 Sales Chart

Das Sales Chart Control stellt die totale Verkaufssumme pro Tag dar. Dabei wird die x-Achse, also die Darstellung der einzelnen Tage, entsprechend der Selektion in der Time Bar angepasst. Die Scrollbar im Chart erlaubt es selbst bei einer grossen Zeitspanne die Daten übersichtlich und nicht übereinanderliegend darzustellen.
Wird im Map Control die Regionsauswahl verändert, so wird das Sales Chart seine Datenquelle auf die entsprechende Region eingrenzen beziehungsweise erweitern.

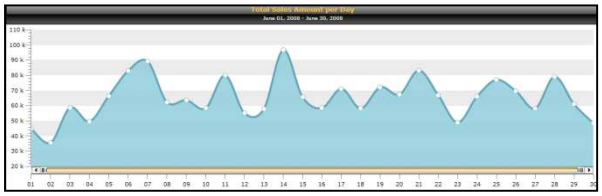


Abbildung 15: Sales Chart Control

2.4 Pie Chart

Das Pie Chart Control zeigt die Verteilung der Produktkategorien in Bezug zu den Verkaufszahlen der momentanen Datenselektion. In Bezug zu Änderungen der Datenquelle verhält es sich gleich wie das Sales Chart.

Abbildung 16: Pie Chart Control

2.5 Grid

Das Grid Control erweitert die Prozentuale Übersicht des Pie Charts. Dabei werden zu den einzelnen Produktkategorien die totalen Verkaufszahlen sowie eine Sparkline dargestellt. Die Sparkline erlaubt es auf einen Blick das ungefähre Verkaufsverhalten der Produktkategorie zu einer Zeitspanne festzustellen.
Zusätzlich können die einzelnen Ansichten der Produktkategorien erweitert werden um das Verkaufsverhalten der zugehörigen Produkte mit weiteren Sparklines darstellen zu können.
Auch das Grid passt seine Daten entsprechend der aktuell gewählten Zeitspanne und Region an.

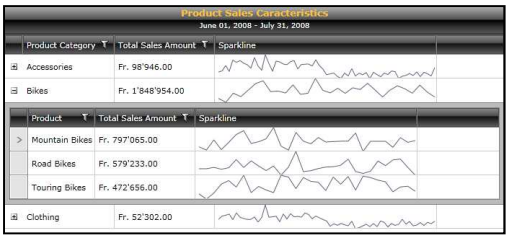


Abbildung 17: Grid Control

2.6 Bullet Graph

Das Bullet Graph Control repräsentiert die Key Performance Indicators (KPIs) der Verkaufs- und Produktionszahlen des Unternehmens.
Jeder Bullet Graph stellt mit einem horizontalen schwarzen Balken die Kennzahl der aktuell gewählten Zeitspanne sowie mit dem senkrechten schwarzen Strich die Kennzahl der gleichen Zeitspanne ein Jahr zuvor dar. Somit lässt sich sofort erkennen, ob im Vergleich zum Vorjahr eine Verkaufssteigerung oder ein Verkaufsrückgang stattgefunden hat.
Die Bullet Graphs passen sich der aktuell selektierten Zeitspanne an.

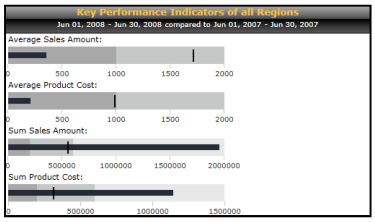


Abbildung 18: Bullet Graph Control

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] CZERNICKI, Bart: *Silverlight 4 Business Intelligence Software*
Apress, 2010 – ISBN 978-1-4302-3060-1

[2] LARMAN, Craig: *Applying UML and Patterns*
Prentice Hall, 2004 – ISBN 978-0-13-14906-2

[3] Silverlight Architektur Übersicht
http://sandrinodimattia.net/blog/post/Making-WCF-RIA-Services-work-in-a-DMZMultitier-architecture-using-Application-Request-Routing.aspx, letzter Zugriff 19.05.2011

[4] Übersicht WCF RIA Services
http://www.silverlight.net/getstarted/riaservices, letzter Zugriff 20.05.2011

[5] Einführung WCF RIA Services
http://www.silverlightshow.net/items/WCF-RIA-Services-Part-1-Getting-Started.aspx, letzter Zugriff 20.05.2011

[6] MSDN Übersicht WCF RIA Services
http://msdn.microsoft.com/de-de/library/ee707344%28v=VS.91%29.aspx, letzter Zugriff 11.03.2011

[7] MSDN Übersicht WCF Data Services
http://msdn.microsoft.com/de-de/library/cc668792.aspx, letzter Zugriff 11.03.2011

[8] Vergleich RIA Services / Data Services
http://jack.ukleja.com/wcf-data-services-vs-wcf-ria-services, letzter Zugriff 11.03.2011

[9] MSDN Übersicht ADOMD.NET
http://msdn.microsoft.com/en-us/library/ms123483.aspx, letzter Zugriff 13.03.2011

[10] TelerikRad Controls für Silverlight Dokumentation
http://www.telerik.com/help/silverlight/introduction.html, letzter Zugriff 21.03.2011

[11] TelerikRad Controls für Silverlight Demo
http://demos.telerik.com/silverlight/#Controls, letzter Zugriff 21.03.2011

[12] Microsoft Pivot Viewer
http://en.wikipedia.org/wiki/Live_Labs, letzter Zugriff 11.03.2011
http://www.silverlight.net/learn/pivotviewer/collection-xml-schema, letzter Zugriff 11.03.2011

Glossar

BI	Business Intelligence
DTO	Data Transfer Object
KPI	Key Performance Indicator
MS	Meilenstein
MVVM	Model-View-ViewModel
RIA	Rich Internet Application
SP	Stored Procedure
UC	Use Case
UI	User Interface
WCF	Windows Communication Foundation

Technischer Bericht

1 Einleitung und Übersicht

Das Ziel war, mit Hilfe einer Demo-Applikation zu zeigen, wie mit Silverlight moderne BI-Clients entwickelt werden können. Als erstes war dabei wichtig, die verschiedenen Varianten von Service-Schnittstellen zu vergleichen sowie zu untersuchen welche Controls sich für eine BI-Lösung eignen.
Anschliessend wurde aufgrund dieser Erkenntnisse ein Silverlight basierter Showcase implementiert. Der Client sollte dabei auf dem MVVM Pattern basieren.

2 Ergebnisse

In den folgenden Unterkapiteln werden die erreichten Ziele aufgeführt und zusätzliche Erkenntnisse und Vorgehen zum Lösen von Problemen beschrieben. Ebenfalls werden die nicht erreichten Ziele erwähnt und die Ursachen erläutert.

2.1 Erreichte Ziele

- Sales Dashboard als Showcase mit Drill-Down, Roll-Up und Navigations-Operationen
- MVVM basierter Client
- WCF RIA Services basierte Service-Schnittstelle

2.1.1 Sales Dashboard

Das Sales Dashboard ist eine Zusammensetzung von verschiedenen BI-Controls. Die Controls wurden dabei in einzelne Silverlight User Control eingebunden, welche dann geordnet in die Main Page integriert wurden. Das Map Control bietet Drill-Down- und Roll-Up-Operationen nach geografischer Region. Die TimeBar erlaubt die Navigation in einem bestimmten Zeitbereich.

Die Drill-Down- und Roll-Up-Operationen des MapControls geschehen mit zusätzlichen Comboboxen, welche die zur Auswahl stehenden Regionen enthalten. Bei der Selektion einer neuen Region wird per Command Binding das ViewModel des MapContols darüber in Kenntnis gesetzt.
Zusätzlich ist beim MapControl ausnahmsweise auch im Code Behind Logik vorhanden. Dies aus dem Grund, weil das MVVM Pattern es nicht erlaubt, vom ViewModel aus auf ein Control der View zuzugreifen. Die Map muss jedoch mit einem Geocode Provider und dessen Logik ergänzt werden, dies geschieht somit im Code Behind. Der Geocode Provider erlaubt es, die selektierten Regionen von der Adresse in einen Lokalisierungspunkt umzuwandeln.
Bei einer Selektionsänderung wird also neben dem Command Binding auch ein SelectionChanged Event an den Code Behind weitergegeben. Im Code Behind wird der Lokalisierungspunkt der selektierten Region vom GeocodeProvider abgefragt und danach in der Karte auf den entsprechenden Punkt gezoomt. Dies ermöglicht die visuelle Unterstützung mit der Kartengrafik.

Das TimeBar Control löst bei einer Änderung der selektierten Zeitspanne einen Event aus. Weil keine Events, sondern nur Commands mittels Binding an ein ViewModel weitergeleitet werden können, wird hier die EventToCommand Funktion des MVVM Light Toolkits verwendet um den Event an einen Command weiterzuleiten.

2.1.2 MVVM Client

Aufgrund des MVVM Patterns geschieht keine direkte Kommunikation zwischen den einzelnen Controls. Dafür sind die jeweiligen ViewModels zuständig. Um den Programmcode so locker gekoppelt wie möglich zu halten, wird die Message-Funktion des MVVM Light Toolkits verwendet. Per Messages teilen die ViewModels allfällige Änderungen ihrer Properties den anderen ViewModels mit.

2.2 Nicht erreichte Ziele

- Userspezifische Darstellung im Showcase
- Analysis Service Server als Datenquelle
- Systemtests

2.2.1 Ursachen

Die Ursachen für die nicht erreichten Ziele liegen hauptsächlich an mangelnder Zeit. Die Kenntnisse für die Anwendung der Telerik RadControls sind nun jedoch vorhanden. Es wäre also möglich, den Showcase mit nur wenig zusätzlichen Kenntnissen zu erweitern, damit es, zum Beispiel durch Dependency Injection, dem Benutzer möglich wäre die Benutzeroberfläche nach seinen Wünschen anzupassen.
Die Einarbeitung in die Analysis Service mit Cube Technologie war zu umfangreich um sie ohne Vorkenntnisse in die kurze Zeit des Projektes zu integrieren.
Auch für die Tests war die Zeit zu knapp um diese seriös zu implementieren. Da die logischen Teile des Programms vor allem dazu dienen Daten von der Datenbank abzurufen, zu gruppieren, zu filtern usw. müsste für die Tests eine Mock-Datenbank erstellt werden. Dies wäre auf jeden Fall machbar gewesen, hätte jedoch weitere Zeit benötigt, die nicht vorhanden war. Folglich wurde bei einer Besprechung mit dem Betreuer beschlossen auf die Tests zu verzichten.

3 Schlussfolgerung

Das Resultat, also das Sales Dashboard als Showcase, veranschaulicht die Entwicklung eines modernen und interaktiven BI-Clients mit Silverlight. Die Hauptziele wurden somit erreicht. Obwohl noch weitere Funktionen und Anpassungsmöglichkeiten der Benutzeroberfläche möglich wären, zeigt das Sales Dashboard die wichtigsten BI-Controls und unterstützt diverse BI-Operation wie zum Beispiel Drill-Down.
Neben der Funktion als Demo-Applikation liefert das Sales Dashboard auch eine gute Basis für Weiterentwicklungen. Es wurde besonders auf eine saubere Strukturierung des Codes geachtet. Die einzelnen Komponenten können ohne grossen Aufwand problemlos ausgetauscht oder verändert werden.

Der Entscheid WCF RIA Services als Service-Schnittstelle zu verwenden hat sich sehr positiv auf das Projekt ausgewirkt. Obwohl RIA Services aufgrund des automatisch erzeugten Codes im Hintergrund am Anfang etwas schwer nachzuvollziehen war, erwies es sich als äussert effizient und pflegeleicht während der Entwicklung.

Persönlicher Bericht

1 Allgemein

Mit den Modulen Microsoft Technologien und Internet Technologien im letzten Semester habe ich .NET und Silverlight kennengelernt. Ich habe mich sofort dafür begeistert und wagte mich an die Studienarbeit „Business Intelligence mit Silverlight“, obwohl ich damals nur Grundkenntnisse und keine Projekterfahrungen zu .NET und Silverlight besass.

2 Ablauf des Projekts

Weil meine Arbeit ursprünglich für zwei Personen geplant war, fühlte ich mich zu Beginn des Projekts etwas verloren und wusste nicht so richtig wie und wo ich starten soll. Mit dem Projekt des Software Engineering 2 Moduls habe ich meine ersten Erfahrungen mit Software-Projekten gemacht und versuchte mich deshalb so gut wie möglich an diesem Projekt zu orientieren.
Die Erstellung des Projektplans gab mir dann eine grobe Übersicht zum Projekt und schaffte eine gute Orientierung.
Im ersten Teil des Projekts befasste ich mich vor allem mit der Recherche über die möglichen Service-Schnittstellen, die verschiedenen Datenbanken und der Evaluation von BI-Controls.
Ich musste feststellen, dass sehr viele neue Technologien auf mich zukommen. Auch das Verstehen des Codes der Demo-Applikationen zu den BI-Controls verschiedener Anbieter machte mir zu Beginn etwas Mühe.
Dies führte unter anderem dazu, dass ich mich trotz grossem Interesse gegen die Verwendung von Analysis Service Cube Datenbanken entschied.
Beim Verstehen der BI-Controls waren die Sitzungen mit meinem Betreuer Prof. Hansjörg Huser sehr hilfreich. Mit seiner Hilfe und seinen Tipps verstand ich den C#-Code immer besser. Somit verstand ich die Dokumentation zu den Controls auch besser und konnte mit dem Austesten und der Testimplementierung der verschiedenen Controls beginnen.
Anschliessend befasste ich mich mit der Implementierung des Sales Dashboards als Demo-Applikation. Weil ich zu diesem Zeitpunkt schon viel bessere Kenntnisse in der C#- und XAML-Programmierung hatte, kam ich mit der Implementierung sehr gut voran. Es machte Spass, neue Logik zu implementieren, den geschriebenen Code zu optimieren und das GUI zu designen.
Es lag mir sehr viel daran, dass mein Programmcode sauber strukturiert und geschrieben ist. Aus diesem Grund verzichtete ich auf die Implementierung einiger zusätzlicher Funktionen und nützte diese Zeit für das Refactoring des Codes.
Die restlichen zwei Wochen befasste ich mich mit dem Schreiben der Dokumentation. Auch hier lag mir viel daran genügend Zeit dafür zu investieren, damit diese am Schluss korrekt ist.

3 Fazit

Mit der Studienarbeit habe ich sehr viel Erfahrung im Zusammenhang mit Software-Projekten, wie auch mit .NET und Silverlight machen können. Ich kann mir gut vorstellen, mich in der Zukunft auf die Programmentwicklung mit .NET zu spezialisieren.
Die Arbeit alleine zu machen hatte sowohl Vor- als auch Nachteile. Es lag an mir zu bestimmen, wann, wie und woran ich arbeite. Ich konnte Arbeiten die mir weniger zusprachen nicht abschieben, sondern musste mich einfach durchbeissen. Schlussendlich stellte ich fest, dass diese Arbeiten gar nicht so arg waren.
Der Nachteil an dieser Einzelarbeit war, dass mir jemand fehlte, mit dem ich mich rücksprechen konnte. Es lag an mir zu bestimmen, ob ich auf dem richtigen Weg war oder nicht.
Letztendlich konnte ich mich jedoch immer mit Herrn Huser oder mit meinen Mitstudenten über ein Problem unterhalten und somit gemeinsam auf einen Lösungsansatz kommen. Hiermit möchte ich mich bei allen Beteiligten für die erhaltene Unterstützung bedanken.

Mit dieser Studienarbeit habe ich sehr viel Neues gelernt. Seien es nun neue Technologien oder neue Erfahrungen im Zusammenhang mit Software-Projekten. Ich bin nun gewappnet für die Bachelorarbeit und hoffe auch mit dieser eine weitere spannende Arbeit schreiben zu können.

Sitzungsprotokolle

1 Kurzprotokoll vom 28.02.2011

1.1 Was wurde erreicht?

- Projektplan wurde erstellt
  Feedback:
  - Risikoanalyse und Anforderungsspezifikation werden vorerst nicht durchgeführt. Aufgrund der neu beschlossenen Verwendung von Cube-Datenbanken wird neu Zeit für das entsprechende Einarbeiten eingeplant.
  - Die Elaboration 1 begrenzt sich auf die Analyse der bereits bestehenden Applikations-Demos und das Einarbeiten in die Cube-Datenbanken.
  - Spezifikation wird erst nach der Erstellung des Prototyps durchgeführt
- Erste grobe Evaluation der BI-Controls wurde durchgeführt

1.2 Was ist geplant?

- Detaillierte Evaluation von ComponentArt- und Telerik-Controls anhand der Demos
  - Technische Funktionen: Wie werden Daten geladen (speziell Drill-Down)?
  - Welche Architektur benutzen die Demos für Anbindungen?
- Evaluation des PivotViewer → Nachlesen im Buch
- Verwendung von Cube-Datenbanken: Cube-Tutorial durchspielen

2 Kurzprotokoll vom 07.03.2011

2.1 Was wurde erreicht?

- Cube-Tutorial durchgespielt
- Evaluation der Demos
- Grobe Evaluation des PivotViewers

2.2 Was ist geplant?

- Vergleich der drei Kommunikationsvarianten WCF RIA, WCF Data Services und WCF Custom (MDX-Abfragen auf Cube-Datenbanken)
- Analyse Pivot Collection Design: CXML

→ Analyse soll soweit abgeschlossen sein, dass bei der Besprechung am 14.03. die zu verwendenden Technologien festgelegt werden können. Danach folgt das Erstellen eines PivotViewers mit Kommunikationsschnittstellen, der als Cockpit dienen soll.

2.3 Welche Probleme stehen an?

- Die Demoversionen von ComponentArt und Telerik sind nur erschwert analysierbar. Der Code wurde aufgrund des Demonstrationszwecks stark vereinfacht und durch „Bastellösungen“ ersetzt.

3 Kurzprotokoll vom 14.03.2011

3.1 Was wurde erreicht?

- Aufgrund des Vergleichs der verschiedenen Kommunikationsvarianten, fällt der Entscheid für die Kommunikationsvariante auf die WCF RIA Services. Diese wurden speziell für N-Tier RIAs wie z.B. Silverlight entwickelt. Somit liefern sie ein optimales Framework für unsere Business Intelligence Applikation. Zusätzlich bieten die Telerik Rad Controls für Silverlight Unterstützung für das Binding zu RIA Services.
- Der Microsoft PivotViewer hat sich aufgrund seiner eingeschränkten Möglichkeiten als nicht geeignet herausgestellt

3.2 Was ist geplant?

- Telerik Chart mit WCF RIA Service erstellen → möglichst generisch
- Komplexerer Chart mit Zeitrange und Drilldown

4 Kurzprotokoll vom 21.03.2011

4.1 Was wurde erreicht?

- Einfacher statischer Chart, der seine Daten über einen WCR RIA Service bezieht.

4.2 Was ist geplant?

- Chart mit Drilldown
- Einsatz von Stored Procedures für vereinfachte und optimierte Datenbankabfragen

4.3 Welche Probleme stehen an?

- Die Verbindung zwischen den Telerik Rad Controls und den Daten aus dem WCF RIA Service Context erweist sich als doch nicht so einfach. Die zugehörige Dokumentation von Telerik ist eher dürftig und die dort dokumentierte Lösung führte erst nach diversen Anpassungen zu einem zufriedenstellenden Ergebnis.
- Die Datenbankabfragen sollten so optimiert werden, dass die Daten bereits auf der Serverseite möglichst gut gefiltert werden. Erhält ein Rad Control mehr als ca. 2000 bis 4000 Datensätze zugewiesen, lassen sich diese nicht mehr darstellen.

5 Kurzprotokoll vom 28.03.2011

5.1 Was wurde erreicht?

- Nach zahlreichen Versuchen mit verschiedenen Lösungsansätzen wurde kein Resultat erreicht, dass zu einem zufriedenstellenden Drill Down Chart führt.

5.2 Was ist geplant?

- Drill Down
- Zoom and Scroll

→ beide so gut wie möglich von der Telerik Demoversion übernehmen und anpassen wo nötig.

6 Kurzprotokoll vom 04.04.2011

6.1 Was wurde erreicht?

- Drill Down sowie Zoom and Scroll Chart wurden implementiert

6.2 Was ist geplant?

- TimeBar
- Sparkline

→ Implementierung beider Controls, in Verbindung mit anderen Controls

- Danach sollen die Anforderungsspezifikationen erstellt und der Inhalt für den zu erstellenden Showcase detailliert geplant werden.

7 Kurzprotokoll vom 08.04.2011

7.1 Was wurde erreicht?

- TimeBar und Sparkline wurden implementiert

7.2 Was ist geplant?

- Implementierung eines Showcase, ausgehend von einer Map mit Drill Down
- Suchen nach einer Möglichkeit die Daten zu cachen

8 Kurzprotokoll vom 18.04.2011

8.1 Was wurde erreicht?

- Dashboard mit Zeit- und Map-Drill Down

8.2 Was ist geplant?

- Bestehendes Dashboard zu Sales Dashboard erweitern und mit einem Product-Drill Down ergänzen
- Lösungen für Data Caching suchen um das Laden von Daten zu optimieren
- Ist RIA Service Paging zusammen mit den Telerik Controls einsetzbar?

8.3 Welche Probleme stehen an?

- Daten laden/nachladen dauert zu lange und ist zu komplex

9 Kurzprotokoll vom 26.04.2011

9.1 Was wurde erreicht?

- Product-Detaildarstellung in DataGrid

9.2 Was ist geplant?

- Fertigstellung des GUI's
- Dokumentation:
  - Dashboard GUI: Anforderungen, Gestaltung
  - Evaluation der BI-Controls und zugehörigen Demos
- Nächste Woche Code Review: MVVM, Datenstrukturen (Clientseitig, Schnittstelle zu Server)

10 Kurzprotokoll vom 02.05.2011

10.1 Was wurde erreicht?

- Dashboard Beispiel wurde fertiggestellt

10.2 Was ist geplant?

- Refactoring des Codes für Sales Dashboard
  - Komponenten aufteilen (View, ViewModel)
  - RIA Service mit mehreren Service Interfaces (nach Komponenten aufgeteilt)
  - Evtl. Einsatz von MVVM Light für Messages zwischen den ViewModels
  - Guid wenn möglich ersetzen (evtl. Identity)
- Dokumentation der verwendeten Controls und des GUIs

11 Kurzprotokoll vom 09.05.2011

11.1 Was wurde erreicht?

- Refactoring des Codes für Sales Dashboard
  - Komponenten aufgeteilt (View, ViewModel)
  - RIA Service mit mehreren Service Interfaces (nach Komponenten aufgeteilt)
  - Einsatz von MVVM Light für Messages zwischen den ViewModels sowie für Command Binding

11.2 Was ist geplant?

- Sales Dashboard fertigstellen
  - Grid View fertig einbinden
  - Default TimeBar Datum auf letztes Quartal o.ä. setzen
  - Einsatz des View Model Locators von MVVM Light um korrekte Darstellung in der Designview sicherzustellen
- Planung der Dokumentation
- Wenn genügend Zeit:
  - Einsatz von Style Templates um mehrfach wiederholenden XAML-Code zu vermeiden
  - Tests
  - Einsatz von Dependency Injection

12 Kurzprotokoll vom 16.05.2011

12.1 Was wurde erreicht?

- Sales Dashboard:
  - Grid View fertig eingebunden
  - Default ausgewählte TimeBar Zeitspanne auf letzte 2 Monate gesetzt
  - Einsatz des View Model Locators von MVVM Light um korrekte Darstellung in der Designview sicherzustellen und Verbindung zwischen View und ViewModel an einem Punkt zu verwalten
- Inhalt Dokumentation geplant

12.2 Was ist geplant?

- Einsatz von Style Templates im XAML Code
- Suche Design View Fehler
- Dokumentation

12.3 Welche Probleme stehen an?

- Design View funktioniert immer noch nicht einwandfrei

13 Kurzprotokoll vom 23.05.2011

13.1 Was wurde erreicht?

- Style Templates im XAML Code implementiert
- Design View Fehler in TimeBar Control behoben
- Dokumentation wird fortlaufend erweitert

13.2 Was ist geplant?

- Fortführung der Dokumentation
- Review des Abstracts am kommenden Montag, 30. Mai

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Business Intelligence mit Silverlight

Anita Hollenstein

Betreuer
Prof. Hansjörg Huser

Institut
Institut für vernetzte Systeme (INS)

Studienarbeit FS 2011
Internet-Technologien und -Anwendungen / Software

Ziel:

- Evaluation Kommunikationsvarianten
- Evaluation BI-Controls
- Implementation Demo-Applikation

Webclient

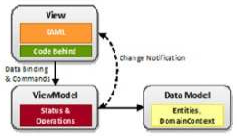
- Sales Dashboard als Showcase
- Navigation des Zeitfensters
- Drill-Down / Roll-Up nach Region

Microsoft Silverlight

Architektur:

- Client Server Architektur mit WCF RIA Services

- Client basierend auf MVVM-Pattern



- Relationales Data Warehouse auf MS SQL Server 2008 R2
  → Einsatz von Stored Procedures