

***Dokumentation zur betrieblichen Projektarbeit***

***Bugreports***

***Webbasiertes Modul zur Fehlerverwaltung***










***Kevin Hoffmann***

***Fachinformatiker Fachrichtung Anwendungsentwicklung***

***Ausbildungsbetrieb:***  
***ORTEC LOGIPLAN GmbH***  
***Daimlerstr. 7***  
***27793 Wildeshausen***

***Prüfungsbewerber:***  
***Kevin Hoffmann***  
***Am Pumpwerk 23***  
***27801 Dötlingen***  
***Geburtsdatum: 18.02.1986***  
***Geburtsort: Delmenhorst***

## Inhaltsverzeichnis

 <b>Einführung</b> .....	3
 <b>Projektdefinition</b> .....	3
Projektumfeld .....	3
Ist-Situation .....	3
Projektziel .....	4
Projektabgrenzungen.....	5
Schnittstellen.....	5
 <b>Projektplanung</b> .....	5
Zeitplanung der Projektphasen .....	5
Ressourcenplanung .....	6
<b>Kostenplanung</b> .....	6
Einmalige Entwicklungskosten für das neue Modul.....	6
Zeiten für die Gesamte Bearbeitung eines Datensatzes .....	6
Kosten für die Bearbeitung für ein Jahr.....	7
Sonstige Kosten .....	7
Kostenvergleich .....	7
Amortisationsdauer.....	7
Nutzwertanalyse .....	8
 <b>Projektdurchführung</b> .....	8
Analysephase .....	8
Designphase .....	9
Realisierung .....	11
Programmtest .....	12
Dokumentation .....	13
Einführung.....	13
 <b>Projektbewertung</b> .....	13
Zielerreichung .....	13
Soll-/Ist-Abgleich .....	14
Weiterentwicklung .....	14
 <b>Anhänge</b> .....	14
 <b>Abbildungs- und Tabellenverzeichnis</b> .....	15

## Einführung

Das Glossar beinhaltet alle unterstrichenen Wörter sowohl in der Dokumentation als auch in den Anhängen. Einige Erklärungen im Glossar sind mit Hilfe von anderen Quellen, bzw. Hilfsmitteln, entstanden. Diese sind aber mit einer Quellangabe gekennzeichnet. Das Glossar ist als Anhang H gekennzeichnet.

## Projektdefinition

An dieser Stelle werden das Projektumfeld, die Ist-Situation, das Projektziel, die Projektabgrenzungen und die Schnittstellen erläutert.

### ***Projektumfeld***

In der ORTEC Logiplan GmbH werden spezielle Programme zur Optimierung von Prozessen in der Handels- und Distributionslogistik entwickelt. Diese Programme werden vor jedem produktiven Einsatz im Qualitätsmanagement ausführlich getestet, um dem Kunden eine fehlerfreie und optimale Softwarelösung anbieten zu können.

Um eine fehlerfreie Software auszuliefern, müssen Programmfehler gemeldet und behoben werden. Es ist nicht optimal diese direkt an die Entwickler weiterzuleiten, da es so zu Problemen auf der zeitlichen aber auch kommunikativen Ebene führen kann. Stand-Alone Programme sind hier zwar nützlich, doch im produktiven Einsatz nicht die beste Lösung. Es müssen immer bestimmte Systemvoraussetzungen beachtet und eingehalten werden. Außerdem sind bei anfallenden Updates, alle Clients dazu veranlasst diese auf ihren Rechner zu installieren.

### ***Ist-Situation***

Aktuell gibt es eine Stand-Alone Version der Fehlererfassungssoftware. Das bedeutet, dass jeder Mitarbeiter bzw. Benutzer, der auf die Daten zugreifen möchte oder neue Daten hinzufügen muss, eine Kopie der Software auf seinem Arbeitsplatzrechner benötigt.

Die Benutzeroberfläche unterteilt sich in zwei Formulare. Das eine Formular wird zum anzeigen der vorhandenen Daten und das andere zum Anlegen neuer Daten genutzt. Es gibt weder eine Suchoption für die Benutzer, noch eine Möglichkeit, Bugs auszuwerten.

Sowohl Datenzugriff als auch Datenspeicherung, werden direkt mit SQL-Anweisungen aus dem Programmcode ausgeführt. Alles wird im firmeninternen Netzwerk übermittelt. Ein Zugriff von außerhalb des Netzwerkes ist nicht möglich. Sobald neue Daten eingetragen wurden, werden keine Nachrichten an zuständige Mitarbeiter weitergeleitet, dies muss manuell erledigt werden (nicht über die Software), genau wie bei Datenänderungen.

Für Softwareupdates gibt es keinen automatisierten Ablauf. Jeder Benutzer, der eine neue Version benötigt, muss das Update manuell installieren. Treten hier Probleme auf, so ist der Programmentwickler als Ansprechpartner vorgesehen.

## Projektziel

Im Laufe der Projektabwicklung ist eine Anwendung zu entwickeln, die sowohl für das Qualitätsmanagement als auch für andere Benutzer (externe) einen schnelleren und übersichtlicheren Weg zur Meldung von Bugs (Softwarefehlern) bietet. Diese Anwendung ist nicht mehr als Stand-Alone Version vorgesehen, sondern als Webanwendung. Diese Webanwendung ist ein Modul, welches Teil eines größeren Systems ist. Hier können die Benutzer von überall zugreifen, wo ein Internetanschluss zur Verfügung steht und ein Browser installiert ist. Außerdem muss keine zusätzliche Software auf die Arbeitsplatzrechner installiert werden. Mit Hilfe der neuen ASP.NET AJAX Technologie werden Ladezeiten verkürzt und die Bedienung für die Benutzer wird mit Hilfe neuer Komponenten für die Webentwicklung wesentlich vereinfacht.

Durch Zugriff auf zentral gespeicherte Daten, werden Datenzugriffszeiten verringert und die Darstellung zur Anzeige von gespeicherten Daten übersichtlicher gestaltet. Jeder Datensatz soll gesammelt und geordnet werden, damit jederzeit alle Änderungen nachvollziehbar sind. Außerdem soll die Anwendung in zwei Sprachen übersetzt werden. Um SQL Anweisungen nicht mehr im Programmcode zu speichern, werden SQL-Prozeduren angelegt, die als Teil der Datenbank anzusehen sind. Somit sind Änderungen an SQL-Abfragen nicht mit einer Neukompilierung der Webanwendung verbunden.

Vorhandene Daten sollen für die Benutzer exportierbar sein, damit eine Liste mit allen Bugs und Anhängen als lokale Version zur Verfügung steht.

Ein neues Benachrichtigungskonzept sorgt dafür, dass sämtliche Änderungen jederzeit an alle beteiligten Benutzern/Mitarbeitern weitergeleitet werden.

Ein weiteres Ziel ist es, den Projektzeitplan einzuhalten und alle Arbeiten bis zum Projektabschluss zu beenden.

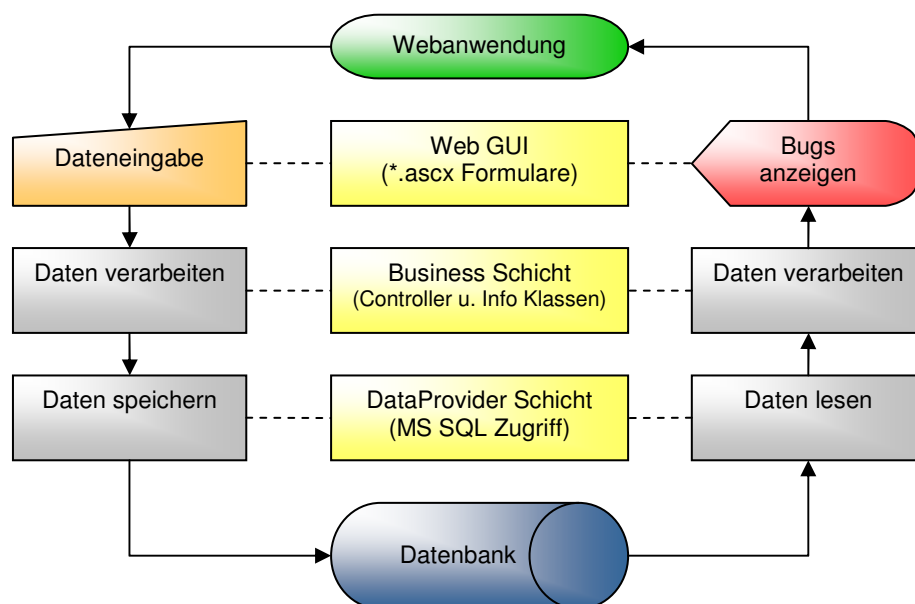


Abbildung 1 – Modul Datenverarbeitung

## Projektabgrenzungen

Die Entwicklung einer Benutzerverwaltung ist aus zeitlichen Gründen kein Teil des Projektes. Außerdem sind die Routinen für die Installation dieser Anwendung auf einem Webserver nicht zu entwickeln und zu realisieren.

Diese beiden Abschnitte übernimmt ein Open Source Projekt namens DotNetNuke®. DotNetNuke® bietet hier eine einfache Lösung Module zu installieren und anzuzeigen. Über die Benutzerverwaltung sind für den Systemverwalter jederzeit alle Daten zu jedem Benutzer übersichtlich dargestellt. Jeder Benutzer erhält einen Benutzernamen und Passwort damit dieser den Zugriff auf das Modul erhält.

## Schnittstellen

Die Anwendung steht in Verbindung mit dem firmeninternen Portal, denn hier soll die Anwendung nach Fertigstellung für jeden Benutzer verfügbar sein.

Vorhandene Daten, wie Kundendaten, Projektdaten und Mitarbeiterdaten sind in der bereits vorhandenen Portaldatenbank enthalten.

Der Prozess zur Speicherung von Bugs ist ein eigener Prozess mit eigenen Datenbankzugriffen. Es werden lediglich Inhalte aus anderen Tabellen angezeigt und als Verweis mit in den neuen Tabellen abgespeichert.

Eine andere und sehr wichtige Schnittstelle ist die der Benutzerverwaltung. An dieser Stelle tritt DotNetNuke® in Kraft und wird dazu verwendet, sämtliche Benutzerverwaltungen und -zugriffe zu regeln.

## Projektplanung

In diesem Abschnitt ist die Zeitplanung, die Ressourcenplanung und die Kostenplanung zu finden.

### Zeitplanung der Projektphasen

Projektphase	Aufgabenbeschreibung	Soll-Stunden
Analysephase	Ist-Analyse	3
	Soll-Konzept definieren	5
	Pflichtenheft erstellen	4
Entwicklung	Datenbankdesign	4
	Datenbankerstellung	3
	Moduldesign	4
	Modulerstellung	21
Modultest	Modulfunktionstest	2
Dokumentation	Inhalte für Projektbericht erstellen	12
	Dokumentation zusammenstellen	2
	Benutzerhandbuch erstellen	8
Einführung	Installation	2
<b>Summe</b>		<b>70</b>

Tabelle 1 – Zeitplanung der Projektphasen

## Ressourcenplanung

Für die Datenbank wird ein Microsoft® SQL Server 2005 benötigt, der aber schon mit den im Abschnitt Schnittstellen benötigten Projektdaten, Mitarbeiterdaten und Kundendaten zur Verfügung steht.

Zur Entwicklung und Umsetzung der Webanwendung steht ein Arbeitsplatzrechner mit Microsoft® Windows XP bereit. Auf diesem Rechner ist die Entwicklungsumgebung Microsoft® Visual Studio 2005 Enterprise Edition installiert, mit dem die nötigen Quellcode Dateien erstellt und kompiliert werden.

Um die Quelltextdateien zu sichern, steht ein CVS-Server bereit. Der dazugehörige Client ist auf dem Arbeitsplatzrechner installiert.

Weitere Voraussetzungen sind das .NET Framework 2.0 und das ASP.NET AJAX Framework 1.0 sowie die neuste Version des Open Source Frameworks DotNetNuke®.

## Kostenplanung

Die Kostenplanung beschreibt die Wirtschaftlichkeit des entwickelten Moduls und führt daher genauere Kostendetails auf. Die hier anfallenden Kosten sind keinem Kunden in Rechnung zu stellen, da es sich in diesem Fall um ein firmeninternes Projekt handelt. Alle Berechnungen beziehen sich auf den Standard Stundensatz.

**Kosten Stunden-Satz: 60 €**

### Einmalige Entwicklungskosten für das neue Modul

Projektphase	Zeitaufwand	Kosten
Ist-Analyse	2 Stunden	120,00 €
Soll-Konzept	5 Stunden	300,00 €
Programmierung/Codierung	40 Stunden	2.400,00 €
Modultest	6 Stunden	360,00 €
Moduldokumentation	13 Stunden	780,00 €
Einführung/Installation	4 Stunden	240,00 €
<b>Summe</b>	<b>70 Stunden</b>	<b>4.200,00 €</b>

*Tabelle 2 – Einmalige Entwicklungskosten*

### Zeiten für die Gesamte Bearbeitung eines Datensatzes

	Altes Verfahren	Neues Verfahren
Datensatz eintragen:	3 Min.	2 Min.
Benachrichtigung an Mitarbeiter:	3 Min.	1,5 Min.
Lösung melden:	4 Min.	2 Min.
Suche nach Datensatz:	1 Min.	0,5 Min.
Bearbeitung des Datensatzes:	3 Min.	2 Min.
<b>Summe:</b>	<b>14 Min.</b>	<b>8 Min.</b>

*Tabelle 3 – Zeiten für die Bearbeitung eines Datensatzes*

Die Zeiten wurden bei Testergebnissen erzielt und sind daher die Richtlinie für diese Berechnung. Natürlich können im produktiven Einsatz Schwankungen bei den einzelnen Bearbeitungsschritten auftreten.

#### Kosten für die Bearbeitung für ein Jahr

	<b>Altes Verfahren</b>	<b>Neues Verfahren</b>
Datensätze monatlich:	75	75
Datensätze jährlich:	900	900
Zeitaufwand jährlich: in Min.	12.600	7.200
in Std.	210	120
<b>Summe:</b>	<b><u>12.600,00 €</u></b>	<b><u>7.200,00 €</u></b>

**Tabelle 4 – Kosten für die Bearbeitung der Datensätze**

Die Anzahl der Datensätze ist nicht immer konstant. Bei diesen Werten handelt es sich um Durchschnittswerte, die aus vorhandenen Daten errechnet wurden.

#### Sonstige Kosten

	<b>Altes Verfahren</b>	<b>Neues Verfahren</b>
Wartung des Programms:	120 Min	90 Min
Installieren eines Updates:	pro Benutzer 10 Min	einmalig 12 Min
Wartungen jährlich:	12	12
Updates jährlich:	4	4
Anzahl Mitarbeiter:	20	20
Zeitaufwand jährlich: in Min.	2.240	1.128
in Std.	37,3	18,8
<b>Summe:</b>	<b><u>2.238,00 €</u></b>	<b><u>1.128,00 €</u></b>

**Tabelle 5 – Sonstige Kosten**

Updates werden Quartalsweise durchgeführt.

#### Kostenvergleich

	<b>Altes Verfahren</b>	<b>Neues Verfahren</b>
Summe Bearbeitungskosten:	12.600,00 €	7.200,00 €
+ Summe sonstige Kosten:	2.238,00 €	1.128,00 €
= Gesamtkosten	14.838,00 €	8.328,00 €
<b>Ersparnis jährlich:</b>		<b><u>+ 6.510,00 €</u></b>

**Tabelle 6 - Kostenvergleich**

#### Amortisationsdauer

Entwicklungskosten : Ersparnis = Dauer  
 $4.200,00 \text{ €} : 6.510,00 \text{ €} \approx 0,65 \text{ Jahre}$

Die Entwicklungskosten würden, bei einer Datenmenge von 75 Bugs pro Monat, nach etwa acht Monaten gedeckt sein.

## Nutzwertanalyse

Dieser Vergleich soll prüfen, ob die Entwicklung eines neuen Modules wirtschaftlicher ist, als das bisherige System. Es werden das alte und neue Verfahren seperat verglichen.

Kriterium	Gewichtung	Bewertung (alt)	Bewertung (neu)
Zuverlässigkeit	3	5	8
Performance	2	3	5
Funktionalitäten	2	4	7
Benutzerfreundlichkeit	1	3	4
Korrektheit	2	4	5
Erweiterbarkeit	1	2	2
<b>Summe</b>	<b>11</b>	<b>42</b>	<b>64</b>
<b>Wirtschaftlichkeitskoeffizient</b>		<b>3,8</b>	<b>5,8</b>

Tabelle 7 – Nutzwertanalyse

Anhand des Ergebnisses (Wirtschaftlichkeitskoeffizient) ist zu sehen, dass mit dem neue Modul 35% wirtschaftlicher, als mit dem alten Verfahren gearbeitet werden kann.

## Projektdurchführung

Hier werden alle erledigten Tätigkeiten beschrieben. Es werden jeweils die Tätigkeit, der Zeitaufwand und eine Beschreibung aufgeführt.

### Analysephase

Tätigkeit: Analysieren des Ist-Zustandes  
Definition des Soll-Zustandes  
Lastenheft erstellt

Zeitaufwand: 5 Stunden

Besprechung der aktuellen Situation mit meinem Teamleiter. Danach werden alle Projektziele und Anforderungen des neuen Moduls geklärt. Am Ende werden alle besprochenen Punkte in ein Lastenheft zusammengefasst. Status-, Benachrichtigungs- und Berechtigungskonzept wurden festgehalten. Das Benachrichtigungskonzept wird in Anhang E durch ein Use Case Diagramm dargestellt. Zudem wurde ein EPK erstellt, welches das Anlegen eines neuen Bugs verdeutlichen soll. Zu finden ist dies in Anhang D.

Tätigkeit: Erstellen eines Pflichtenheftes

Zeitaufwand: 4 Stunden

Mit Hilfe der im Lastenheft zusammengefassten Informationen wird ein detailliertes Pflichtenheft erstellt. Das Pflichtenheft beschreibt den Moduleinsatz, Modulumbgebungen, Funktionen und weitere Informationen zu Moduldaten und -leistungen. Trotz des nur internen Einsatzes des Moduls sollte ein Pflichtenheft erstellt werden. Das erstellte Pflichtenheft ist als Anhang A angefügt.



Tätigkeit: Programmablaufplan für das Statuskonzept erstellt  
Zeitaufwand: 1 Stunde

Im Hinblick auf das Statuskonzept habe ich einen Programmablaufplan erstellt, der den Prozess der Statusvergabe verdeutlichen soll. Jeder Bug ist mit einem bestimmten Status gekennzeichnet, um den Bearbeitungsfortschritt einsehen zu können. Welche Status es gibt und was jeder Status im Einzelnen bedeutet, wird im Pflichtenheft erläutert. Der Programmablaufplan ist in Anhang F zu finden. An dieser Stelle kommt auch das Benachrichtigungskonzept ins Spiel, bei jeder Statusänderung wird eine Nachricht an alle beteiligten Personen versendet.

## Designphase

Tätigkeit: Datenbankdesign erarbeitet  
Datenbankmodell erstellt  
Zeitaufwand: 2 Stunden

Anforderungen an die Datenbank wurden definiert und festgelegt. Anzahl der Tabellen mit Beziehungen und Datentypen wurden bestimmt. Danach wurde ein Datenbankmodell erstellt, welches in Anhang B zu finden ist. Hier eine grobe Übersicht des Datenbankdesigns:

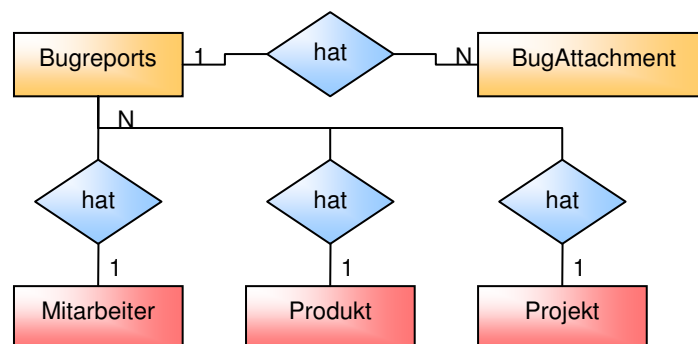


Abbildung 2 – Übersicht Datenbankdesign

Tätigkeit: Moduldesign (GUI – Grafische Benutzeroberfläche)  
Zeitaufwand: 2 Stunden

Um eine übersichtliche und geordnete Ansicht der einzelnen Komponenten zu bieten, werden die \*.ascx Seiten im Tabellenformat aufgebaut. Für Farb- und Texteingenschaften wurden CSS-Dateien angelegt, die dann die entsprechenden Styles beinhalten. Zusätzlich werden DotNetNuke® Textlabels eingesetzt, denen ein Hilfetext mitgegeben werden kann, der dann den Benutzern als Hilfe zur Verfügung steht. So ist es z.B. möglich, Eingabefehler direkt zu vermeiden.



Abbildung 3 – Ansicht DotNetNuke® Textlabel (“Rückfrage”)

Die Navigation erfolgt über Links im Textformat, zusätzlich werden kleine Icons angezeigt, um das Design zu verbessern. Die Daten (Bugs) werden mit einer Tabellenkomponente dargestellt und aufgelistet.

Tätigkeit: Moduldesign (Quellcode)  
Zeitaufwand: 2 Stunden

Aufbau und Inhalt des Quelltextes wurden festgelegt und Funktionen definiert, die das Modul bereitstellen muss. Für den Aufbau soll eine klar geordnete Struktur in mehreren Quellcodedateien benutzt werden, um die Übersicht zu verbessern. Der Quelltext muss gut kommentiert werden, damit bei späteren Änderungen oder bei Übergabe des Quelltextes an andere Entwickler keine oder nur wenige Fragen – bezüglich der Methoden – offen bleiben.

Die Übersetzung in andere Sprachen erfolgt in mehrere Sprachdateien (\*.resx Dateien). Für jede Seite (\*.ascx Datei) ist eine eigene Sprachdatei anzulegen.

Hier ist der Aufbau der Projektmappe aus dem Visual Studio 2005. Alle Dateien in den roten Markierungen gehören zur Webanwendung (Modul).

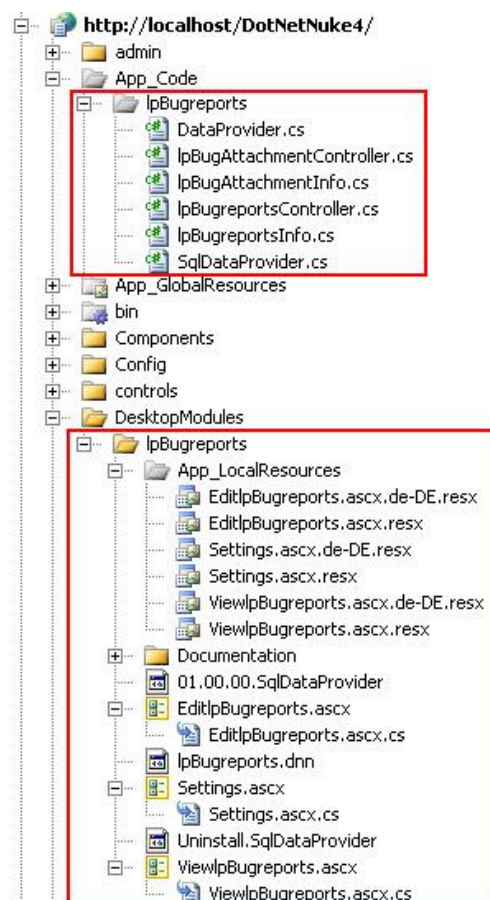


Abbildung 4 – Übersicht Projektmappe im Visual Studio

## **Realisierung**

Tätigkeit: Datenbank erstellt  
Prozeduren erstellt

Zeitaufwand: 3 Stunden

Mit Hilfe des erstellten Datenbankmodells werden Tabellen und Beziehungen angelegt. Das ganze geschieht mit Hilfe des SQL Server Management Studio 2005. Anschließend werden alle Prozeduren zur Datenspeicherung sowie zum Datenauslesen angelegt und getestet.

Tätigkeit: DataProvider Klassenmethoden erstellt  
SqlDataProvider Klassenmethoden erstellt  
Bugreports Controllerklasse und Infoklasse erstellt  
BugAttachment Controllerklasse und Infoklasse erstellt

Zeitaufwand: 8 Stunden

Datenzugriffsklassen und Methoden erstellt. Die DataProvider Klasse ist die Basisklasse, in der alle Funktionen für die Datenspeicherung und des Datenauslesens zur Verfügung stellt. Folgende Standardmethoden sind hier zu sehen:

- AddBugreport (Zum Anlegen eines neuen Bugs)
- GetBugreport (Zum Auslesen eines Bugs mit Hilfe der BugID)
- ListBugreport (Selektiert alle Bugs aus der Datenbanktabelle)
- UpdateBugreport (Speichert Änderungen eines vorhandenen Bugs)
- DeleteBugreport (Löscht den Datensatz der übergebenen BugID)

In der SqlDataProvider Klasse werden die Methoden aus der abgeleiteten Basisklasse benutzt. Diese Methoden bieten den direkten Zugriff auf die Prozeduren der Datenbank um hier Daten aus- bzw. einzulesen (siehe Anhang I – Quellcode).

Die Controllerklassen sind als Schnittstelle zwischen Benutzeroberfläche und Datenzugriff zu sehen. Mit einer Instanz der Controllerklasse können im Anschluss Methoden aufgerufen werden, die dann die Add-, Get-, List-, Update- und Delete-Funktionen ausführen (siehe Anhang I – Quellcode).

In den Infoklassen sind die Objekte für Bugreports und BugAttachments zu finden. Konstruktor und Eigenschaften für jedes Objekt erstellt. Die Objekteigenschaften setzen sich Anhand der Datenbanktabellen zusammen.

Tätigkeit: ViewlpBugreports.ascx GUI und Übersetzungsdateien erstellt  
EditlpBugreports.ascx GUI und Übersetzungsdateien erstellt  
Settings.ascx GUI und Übersetzungsdateien erstellt

Zeitaufwand: 5 Stunden

Benutzeroberfläche für die Übersichtsseite (ViewlpBugreports.ascx), Bearbeitungsseite (EditlpBugreports.ascx) und Einstellungsseite (Settings.ascx) angelegt. Dabei Funktionalitäten und Steuerelemente aus dem AJAX for ASP.NET Framework, ASP.NET Framework und DotNetNuke® Framework benutzt.

Für jede Benutzeroberfläche alle Textausgaben ins Deutsche und Englische übersetzt und in die Textdateien (\*ascx.de-DE.resx/\*ascx.resx) gespeichert (siehe Anhang I – Quellcode).

Die nachfolgende Abbildung zeigt die ViewIpBugreports.aspx Seite (Bug - Eigenschaften).

**Bug - Eigenschaften**

Bug ID: -1    von: Kevin Hoffmann    am: Montag, 23. April 2007  
Letzte Änderung: ----

Status:     Priorität: ☒ Niedrig ☐ Mittel ☐ Hoch  
Behoben am: ---    Test OK am: ---  
Kunde:   
Projekt:   
Produkt:     Versionsnr.:   
Zuständig:   
Kurztext:   
Fehlertext:   
Rückfrage:   
Lösung:   
[Hinzufügen](#) [Abbrechen](#)

Abbildung 5 – Bug-Eigenschaften GUI (Ansicht beim Anlegen eines neuen Bugs)

## Programmtest

Tätigkeit: Quellcode Analyse  
Anwendungstest  
Zeitaufwand: 4 Stunden

Die hier ermittelten Testergebnisse sind in der Testdokumentation, in Anhang C, zusammengefasst. Sämtliche Tests wurden nach Abschluss der Entwicklungsarbeiten durchgeführt. Die Testumgebung ist in der Testdokumentation beschrieben.

## **Dokumentation**

Tätigkeit: Inhalte für die Dokumentation erstellt  
Zeitaufwand: 14 Stunden

Um den großen Zeitaufwand zu begründen, habe ich hier alle Bearbeitungsschritte für die Erstellung der Dokumentation eingerechnet.  
Der Zeitaufwand für die Erstellung der Dokumentation sowie die Erstellung des Anhangs beläuft sich über den gesamten Zeitraum des Projektes.

Tätigkeit: Dokumentation zusammenstellen  
Benutzerhandbuch erstellen  
Zeitaufwand: 6 Stunden

Die gesamte Dokumentation zusammengestellt, die optische Darstellung erstellt und anschließend ins PDF Format konvertiert.  
Außerdem eine Benutzerdokumentation erstellt, die sämtliche Funktionen und Abläufe für den Anwender beschreibt. Dieses Benutzerhandbuch ist in Anhang G zu finden.

## **Einführung**

Tätigkeit: Implementierung des Moduls ins interne Portalsystem  
Zeitaufwand: 1 Stunde

Nach Beendigung aller Programmieraufgaben und lokalen Tests, werden das neue Modul sowie die neuen Datenbanktabellen, in das vorhandene Portalsystem implementiert.

Aus den GUI-Dateien, DLLs und Datenbank-Skriptdateien (Tabellen und Prozeduren) wird eine Archivdatei erstellt, damit diese dann über das DotNetNuke® System installiert werden kann. Hierzu bietet DotNetNuke® eine einfache Installationsroutine, die es ermöglicht, neue Module über die Archivdatei anzulegen.

Nach erfolgreicher Installation ist das Modul für den produktiven Einsatz freigegeben.

## **Projektbewertung**

Die Projektbewertung beschreibt, ob alle Zielvorgaben und Zeitplanungen eingehalten wurden und welche Erweiterungen für die Zukunft geplant sind.

## **Zielerreichung**

Nach der erfolgreichen Einführung des Moduls kann ich sagen, dass das Ziel, welches in der Sollkonzeption definiert wurde, erreicht ist. Der Soll-/Ist-Abgleich soll verdeutlichen, in wie fern ich mit meiner Zeitplanung richtig gelegen habe. Dazu kann ich sagen, dass der Zeitaufwand für die Entwicklung doch höher war als geplant.

Denn hier sind an bestimmten Stellen doch einige Schwierigkeiten aufgetreten, die ich vorher nicht absehen konnte.

### **Soll-/Ist-Abgleich**

Der im Antrag vorgestellte Zeitplan von 70 Stunden wurde eingehalten. Die nachfolgende Tabelle zeigt die Abweichungen im Soll-/Ist-Abgleich.

<b>Projektphase</b>	<b>Aufgabenbeschreibung</b>	<b>Soll-Stunden</b>	<b>Ist-Stunden</b>
Analysephase	Ist-Analyse	3	2
	Soll-Konzept definieren	5	3
	Pflichtenheft erstellen	4	4
Entwicklung	Datenbankdesign	4	2
	Datenbankerstellung	3	3
	Moduldesign	4	4
	Modulerstellung	21	27
Modultest	Modulfunktionstest	2	4
Dokumentation	Inhalte für Dokumentation erstellen	12	14
	Dokumentation zusammenstellen	2	2
	Benutzerhandbuch erstellen	8	4
Einführung	Installation	2	1
<b>Summe</b>		<b>70</b>	<b>70</b>

*Tabelle 8 – Soll-/Ist-Abgleich*

### **Weiterentwicklung**

In zukünftigen Versionen ist es geplant, eine Aufgabenverwaltung mit in das Bugreport Modul zu integrieren. Dies resultiert daraus, dass aus Bugs neue Aufgaben oder Anforderungen entstehen können, so wäre es vorteilhaft, diese direkt mit einem Bug zu verknüpfen. Auch bugunabhängige Aufgaben sollten dann pflegbar sein. Die Benachrichtigungsfunktion könnte als Zusatz, angehängte Dateien mit versenden.

### **Anhänge**

<b>Anhang A</b>	Pflichtenheft
<b>Anhang B</b>	Datenbankmodell
<b>Anhang C</b>	Testdokumentation
<b>Anhang D</b>	Ereignisgesteuerte Prozesskette (EPK)
<b>Anhang E</b>	Use Case Diagramm
<b>Anhang F</b>	Programmablaufplan (Statusvergabe)
<b>Anhang G</b>	Anwenderdokumentation
<b>Anhang H</b>	Glossar
<b>Anhang I</b>	Quellcodeauszüge

## **Abbildungs- und Tabellenverzeichnis**

<b>Abbildung 1</b>	Modul Datenverarbeitung
<b>Abbildung 2</b>	Übersicht Datenbankdesign
<b>Abbildung 3</b>	Ansicht DotNetNuke® Textlabel ("Rückfrage")
<b>Abbildung 4</b>	Übersicht Projektmappe im Visual Studio
<b>Abbildung 5</b>	Bug-Eigenschaften GUI (Ansicht beim Anlegen eines neuen Bugs)
<b>Tabelle 1</b>	Zeitplanung der Projektphasen
<b>Tabelle 2</b>	Einmalige Entwicklungskosten
<b>Tabelle 3</b>	Zeiten für die Bearbeitung eines Datensatzes
<b>Tabelle 4</b>	Kosten für die Bearbeitung der Datensätze
<b>Tabelle 5</b>	Sonstige Kosten
<b>Tabelle 6</b>	Kostenvergleich
<b>Tabelle 7</b>	Nutzwertanalyse
<b>Tabelle 8</b>	Soll-/Ist-Abgleich