Automatisierung der WSUS Auswertung
Ein Projekt mit PHP\MSSQL

Gennaro Piano

8. Juni 2013

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangslage

Wir bieten unseren Kunden die Möglichkeit die Windows Updates über unserem WSUS Server herunterzuladen. Durch dieses Angebot muss sich der Kunde nicht mehr über fehlerhaften Updates kümmern, da wir die Updates zuerst in einer Testumgebung testen und diese danach freigeben. Um die Updateverwaltung einfach zu halten wurden auf dem WSUS Server Produktklassen erstellt. Computer und Server von Kunden die unser Angebot nutzen möchten, werden den entsprechenden Produktklassen zugeteilt. Updates die freigegeben sind werden wiederum den entsprechenden Produktklassen zugeteilt und somit auf den Geräten der Kunden installiert. Dieses System ist gut durchdacht, jedoch ist es sehr kompliziert diese Daten auszuwerten.

Um die Daten auszuwerten werden jedes Quartal alle freigegebenen Updates manuell in einer Excel Datei aufgenommen. Jedes Update wird danach in die entsprechende Produktklasse kopiert. Auf einem weiterem Excel Blatt wird ausgewertet, welcher Benutzer welche Produktklassen benötigt. Dieser Prozess ist sehr zeitintensiv und kostet pro Quartal circa eins bis zwei Arbeitstage.

1.2 Ziel der Arbeit

Um den Aufwand zu kürzen soll der Ablauf des Prozesses so weit wie möglich automatisiert werden. Dies soll mit PHP und einem MS SQL Server realisiert werden. Im ersten Schritt soll die Verbindung zwischen den Updates und den Produktklassen programmiert werden. Als zweiter Schritt werden die Kunden hinzugefügt und die Auswertung automatisiert. Diese Arbeit befasst sich mit dem ersten Schritt des Projekts.

1.3 Sourcecode

Der Sourcecode der Applikation wurde aus Platzgründen nicht in die Dokumentation eingefügt, jedoch kann er direkt aus dem GitHub Repository heruntergeladen werden. Der Link lautet https://github.com/pianogen/auswertung.

1.4 Bemerkung

Die Applikation wird auf einer virtuellen Maschine entwickelt und getestet, sobald alle das Projekt abgeschlossen ist, werden die Dateien auf den produktiven Server kopiert.

2 Requierements

2.1 Einleitung

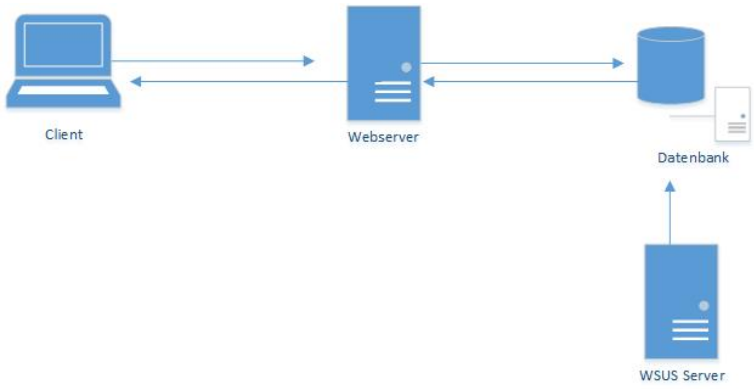
Zweck
Dieser Teil des Dokuments befasst sich mit den Anforderungen, des Projekts. Diese Anforderungen werden mit dem Auftraggeber angeschaut. Erst wenn die Anforderungen vom Auftraggeber bestätigt worden sind und nach seiner Meinung alles abgedeckt ist, wird das Projekt fortgeführt.

Systemumfang
Das System wird für den internen Gebrauch entwickelt und soll von aussen nicht zugänglich sein. Somit besteht keine Gefahr, die Applikation vor unerlaubten Zugriff zu schützen, da dies die Firewall handhabt.

Auftraggeber
Der Auftraggeber dieses Projekts ist zugleich der interne technische Leiter der Firma. Somit wird das Projekt als internes Projekt gehandhabt. Der technische Leiter ist kein Programmierer und stellt deswegen keine technische Anforderungen an das Projekt. Der technische Leiter wird für dieses Projekt als Kunde angesehen.

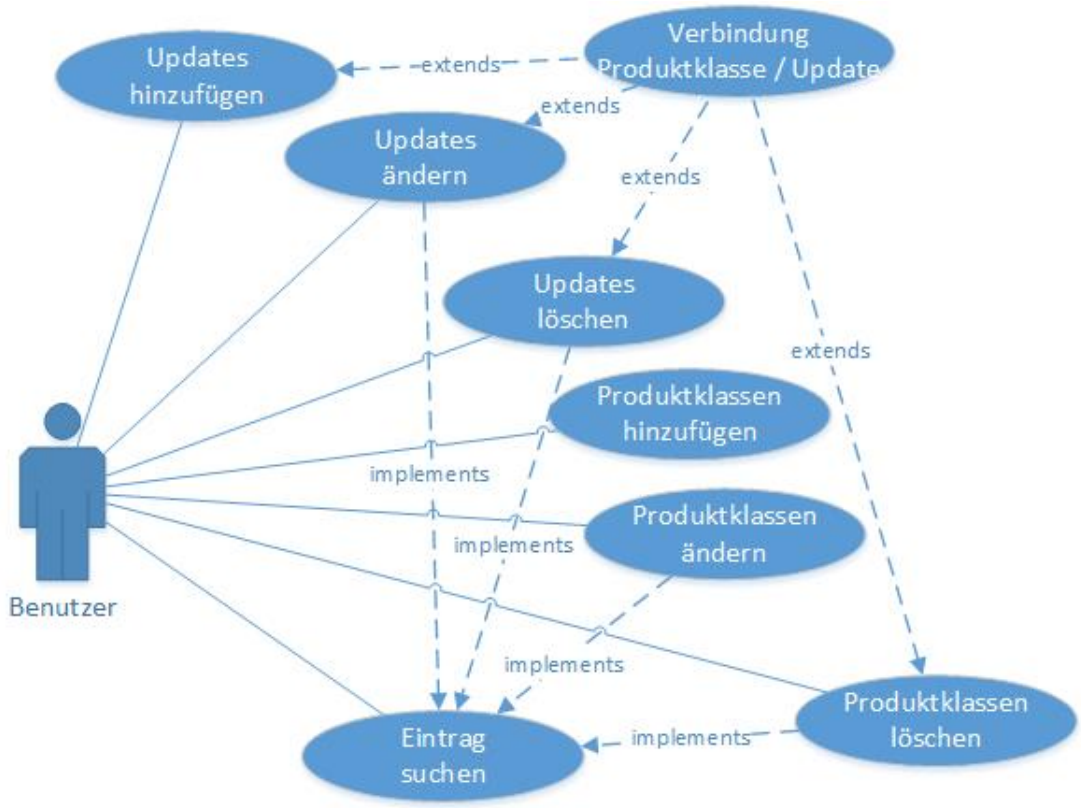
2.2 Allgemeine Übersicht

Systemumfeld
Diese Applikation benötigt einen Webserver und eine Schnittstelle zu einer Microsoft SQL Datenbank. Der WSUS Server gehört auch zum Systemumfeld, jedoch ist die Verbindung zwischen dem WSUS Server und der Applikation nur abstrakt, da die entnommenen Informationen aus dem WSUS Server manuell in der Applikation aufgenommen werden.



Architekturbeschreibung
Die PHP Applikation wird auf dem schon vorhandenen Webserver implementiert. Es handelt sich um einen IIS, dies ist der Webserver von Microsoft. Der IIS unterstützt von sich aus kein PHP. Somit muss der PHP Interpreter nachgerüstet werden. Der PHP Interpreter wird mit dem integrierten Webplattform Installer installiert. Für die Datenbankanbindung wird ebenfalls der vorhandene SQL Server benutzt. Auf diesem Server wird eine neue Datenbank und ein neuer Datenbankbenutzer erstellt. Der neue Benutzer wird nur auf diese Datenbank Zugriff haben, zugleich wird er der einzige Benutzer sein, der auf die neue Datenbank Zugriff hat. Der Webserver und der SQL Server befinden sich physikalisch auf dem gleichen Server, somit ist hardwaremässig nur ein Server von den Veränderungen betroffen. Der PHP Interpreter benötigt eine Schnittstelle um auf die Datenbank zu lesen und zu schreiben, die Schnittstelle zwischen PHP und einem Microsoft SQL Server wird nicht automatisch installiert und muss somit manuell nachinstalliert werden.

Systemfunktionalität
Das Use Case Diagramm soll die Funktionalität des Systems aufzeigen.



Nutzer und Zielgruppen
Der Hauptnutzer ist zugleich der Auftraggeber dieses Projekts. Neben dem Auftraggeber wird die Webapplikation auch von seinem Stellvertreter benutzt werden, dies wird jedoch sehr selten der Fall sein. Der Zugriff auf die Webapplikation wird für alle Benutzer des Unternehmens möglich sein, dies ist bei der momentanen Excel Datei auch der Fall.

Annahmen
Der zweite Teil des Projekts wird wie oben bereits erwähnt erst später entwickelt. In dieser Dokumentation wird auf den zweiten Teil des Projekts bewusst nicht eingegangen, da dies ansonsten den Umfang dieses Dokuments sprengen würde.

2.3 Anforderungen

- Die Applikation muss dem Benutzer ermöglichen, neue Updateeinträge durch eine Maske in die Datenbank einzutragen.
- Die Applikation muss dem Benutzer ermöglichen, über eine Maske nach Updateeinträge auf der Datenbank zu suchen.
- Die Applikation muss dem Benutzer ermöglichen, über eine Maske Updateeinträge auf der Datenbank zu verändern und zu löschen.
- Die Applikation muss dem Benutzer ermöglichen, neue Packageeinträge durch eine Maske in die Datenbank einzutragen.
- Die Applikation muss dem Benutzer ermöglichen, über eine Maske nach Packageeinträge auf der Datenbank zu suchen.
- Die Applikation muss dem Benutzer ermöglichen, über eine Maske Packageeinträge auf der Datenbank zu verändern und zu löschen.
- Die Applikation muss dem Benutzer alle Packagezuweisungen eines Updateeintrags aufzeigen, wenn der Benutzer den Updateeintrag anschaut.
- Die Applikation muss dem Benutzer ermöglichen, über eine Maske ein Packageeintrag, dem entsprechendem Typ zuzuordnen.
- Andersum soll die Applikation dem Benutzer alle Updateeinträge eines Packages aufzeigen, wenn der Benutzer den Packageeintrag anschaut.
- Sobald der Benutzer einen neuen Updateeintrag eröffnen will, soll die Applikation ihm die Packages aufzeigen, damit er sogleich die Verknüpfung erstellen kann.

Beim Speichervorgang des neuen Eintrag soll das System folgende Einträge die durch den Benutzer erfolgt sind überprüfen:

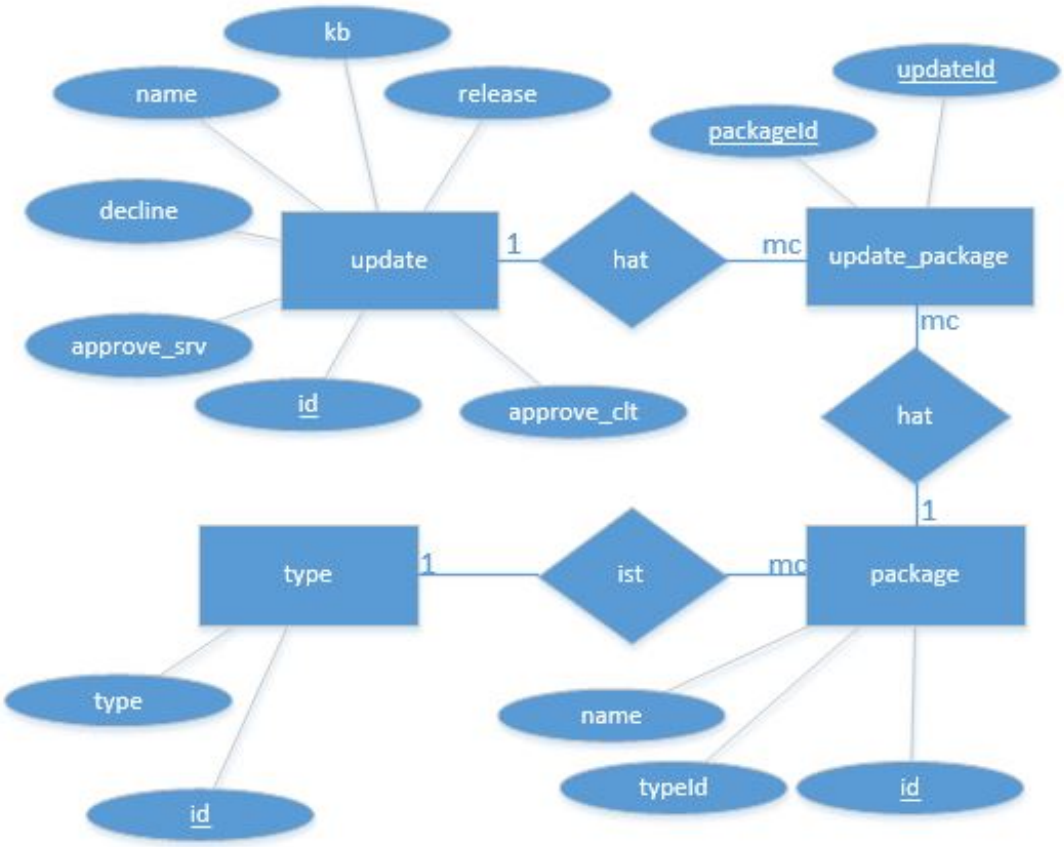
- Der Titel eines Updateeintrag darf nicht leer sein.
- Die KB Nummer eines Updateeintrags darf nicht leer sein und muss einen numerischen Wert haben.
- Das Veröffentlichungsdatum eines Updateeintrags darf nicht leer sein.
- Ein Updateeintrag kann mehreren Packages zugeteilt werden.
- Datumsfelder werden nur als dd.mm.JJJJ angenommen
- Ein Updateeintrag kann keinem Package zugeteilt werden.
- Ein Update kann von Microsoft auch zurückgezogen werden, dies sollte ebenfalls im Datensatz aufgenommen werden können. Dieser kann auch leer sein.
- Sobald der ein Update mindestens einem Package zugeteilt worden ist, muss der Benutzer auch ein Approvement Daten setzen.
- Falls das Update keinem Package zugeteilt worden ist, darf kein Approvement Datum gesetzt sein.
- Der Titel eines Packageeintrag darf nicht leer sein.
- Der Typ eines Packageeintrag darf nicht leer sein.

Falls eine dieser Überprüfungen fehlschlägt, darf der Eintrag nicht gespeichert werden. Die Anforderungen des Speicher- und Änderungsvorgangs sind identisch. Das Löschen eines Updateeintrag muss die Verknüpfung zu den zugeteilten Packages ebenfalls löschen. Das Löschen eines Packageeintrag muss wiederum die Verknüpfungen zu den implementierten Updates löschen.

3 Entwicklung

3.1 Datenbank

Um die aufgenommenen Updates zu speichern, wird eine Datenbank benötigt. Vor der Erstellung der Datenbank wurde ein Entity Relationship Diagram erstellt. Auf der nächsten Seite befindet sich ein genaue Beschreibung der Tabellen und Attributen. Der Text in Klammern beschreibt die Eigenschaften der Attributen.



3.1.1 Tabelle update

In dieser Tabelle befinden sich alle Informationen, die ein Update benötigt.

- **id** (Primärschlüssel, int, auto increment, NOT NULL)
  Wird für die Verbindung zur Tabelle update_package benötigt.
- **name** (varchar(400), NOT NULL)
  Das ist der Titel des Updates, da der Titel recht lange sein kann, wird der String auf 400 Zeichen begrenzt. Diese Attribut ist ein Pflichtfeld.
- **kb** (int, NOT NULL)
  Das ist die KB Nummer[1] des Updates. Dieses Attribut wurde nicht als Primary Key benutzt, da es schon vorgekommen ist, das zwei Updates die selbe KB Nummer haben. Dieses Attribut ist ein Pflichtfeld, da jedes Update eine KB Nummer hat.
- **release** (date, NOT NULL)
  Dieses Attribut beinhaltet das Erscheinungsdatum des Updates. Es ist ein Pflichtfeld, da jedes Update dass eingetragen wird, schon erschienen ist.
- **decline** (date)
  Falls ein Update von Microsoft wegen einem Bug zurückgenommen wird, wird dieses Datum in diesem Feld eingefügt.
- **approve_clt** (date)
  Dieses Attribut beinhaltet das Datum, an dem das Update für die Clients freigegeben wurde. Dieses Feld darf leer sein, da nicht jedes Update zwingend für Clients ist.
- **approve_srv** (date)
  Dieses Attribut beinhaltet das Datum, an dem das Update für die Server freigegeben wurde. Dieses Feld darf leer sein, da nicht jedes Update zwingend für Server ist.

3.1.2 Tabelle update_package

Diese Tabelle ist eine Zwischentabelle. Durch diese Tabelle wird die Beziehung zwischen den Tabellen update und package ermöglicht. In dieser Tabelle besteht der Primärschlüssel aus zwei Attributen.

- **updateId** (Primary Key, int, NOT NULL)
  Dieses Attribut enthält von jedem Update, welches mit einem Package verknüpft wurde, den entsprechenden Fremdschlüssel.
- **packageId** (Primary Key, int, NOT NULL)
  Dieses Attribut enthält von jedem Package, welches mit einem Update verknüpft wurde, den entsprechenden Fremdschlüssel.

[1] Knowlegde Base Number: Wird für Microsoft genutzt, um Updates eindeutig zu kennzeichnen

3.1.3 Tabelle package

Diese Tabelle beinhaltet alle Informationen die ein Package benötigt.

- **id** (Primary Key, Auto Increment, int, NOT NULL)
  Wird für die Beziehung zur Tabelle update benötigt.
- **name** (varchar(50), NOT NULL)
  Das ist der Titel des Package, die Stringgrösse von diesem Attribut wurde auf 50 Zeichen gesetzt.
- **typeId** (int, NOT NULL)
  Dies ist der Fremdschlüssel der Tabelle type, er wird für die Beziehung zwischen den Tabellen package und type benutzt.

3.1.4 Tabelle type

Diese Tabelle beinhaltet alle Typen der Packages. Sie wird benötigt um Redundanz und somit Dateninkonsistenz zu vermeiden.

- **id** (Primary Key, int, Auto Increment, NOT NULL)
  Wird für die Beziehung zwischen den Tabellen type und package benötigt.
- **type** (varchar(10), NOT NULL)
  Dieses Attribut beinhaltet den Namen des Typs. Die Stringgrösse wird auf 10 gesetzt, da die momentan benötigten Typen relativ kurz sind

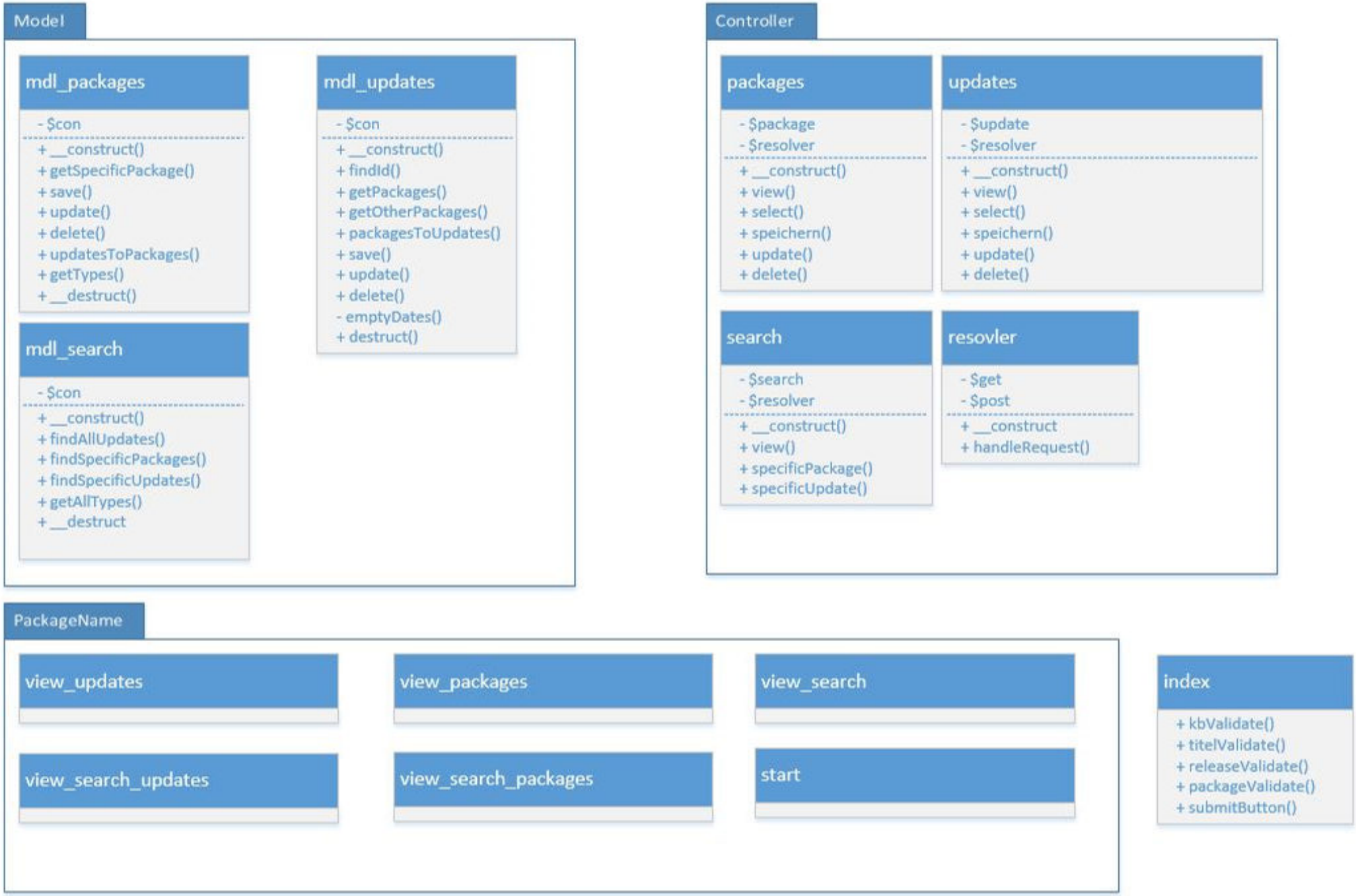
3.2 Sofwarearchitektur

Das Programm wird mit der Sprache PHP entwickelt. Das ganze Programm wird nach dem MVC Konzept[2] konzipiert. Das Programm muss folgende Funktionen anbieten.

- Suchen von Updates und Packages
- Erstellen, ändern und löschen von Updates
- Erstellen, ändern und löschen von Packages
- Anzeigen der Packages und den beinhalteten Updates

Durch diese gewünschten Funktionen entsteht folgendes Architektur.

[2]MVC nach Vorlesung Webprogrammieren mit PHP\MySQL


Model
mdl_packages
- $con
+ __construct()
+ getSpecificPackage()
+ save()
+ update()
+ delete()
+ updatesToPackages()
+ getTypes()
+ __destruct()
mdl_search
- $con
+ __construct()
+ findAllUpdates()
+ findSpecificPackages()
+ findSpecificUpdates()
+ getAllTypes()
+ __destruct
mdl_updates
- $con
+ __construct()
+ findId()
+ getPackages()
+ getOtherPackages()
+ packagesToUpdates()
+ save()
+ update()
+ delete()
- emptyDates()
+ destruct()
Controller
packages
- $package
- $resolver
+ __construct()
+ view()
+ select()
+ speichern()
+ update()
+ delete()
updates
- $update
- $resolver
+ __construct()
+ view()
+ select()
+ speichern()
+ update()
+ delete()
search
- $search
- $resolver
+ __construct()
+ view()
+ specificPackage()
+ specificUpdate()
resovler
- $get
- $post
+ __construct
+ handleRequest()
PackageName
view_updates
view_packages
view_search
view_search_updates
view_search_packages
start
index
+ kbValidate()
+ titelValidate()
+ releaseValidate()
+ packageValidate()
+ submitButton()

3.2.1 Model

Das Model ist für die SQL Abfragen zuständig. Jede Klasse hat einen Konstruktor und einen Dekonstruktor. Der Konstruktor baut jeweils die Verbindung zur SQL Datenbank auf und der Dekonstruktor baut sie wieder ab.

Die anderen Methoden sind jeweils zuständig um die Einträge zu speichern, ändern und zu löschen, sowie benutzerspezifische Suchabfragen durchzuführen und dementsprechend die gefundenen Datensätze zurückzugeben.

3.2.2 Controller

Der Controller ist für die Logik des Programm zuständig, der Controller arbeitet mit den POST und GET Werte und weiss dadurch welche Funktion im Model aufgerufen werden muss. Nach der Abfrage der Datenbank durch das Model, hat der Controller Zugriff auf die gewünschten Daten, nun wird anhand der GET und POST Werte entschieden, welche Datei des Packages View aufgerufen werden soll.

Die Verbindung der Controllerklassen mit den Modelklassen wird einfach implementiert sein. Eine Controllerklasse hat nur Zugriff auf die entsprechende Modelklasse, dies bedeutet das die Controllerklasse *packages* hat nur Zugriff auf die Modelklasse *class.packages*.

Die einzige Klasse, die diese Regel nicht befolgt ist die Klasse *class.resolver*.

3.2.3 class.resolver

Die Klasse class.resolver hat die wichtigste Funktion in der Logik. Sie überprüft, welche Werte sich im POST und GET befinden, und führt somit die entsprechende Funktion in den entsprechenden Klassen aus. Die Klasse class.resolver wurde aus der Vorlesung Webprogrammieren mit PHP\MySQL entnommen.

3.2.4 View

In der View befinden sich alle Dateien, die für das Visualisieren der Daten zuständig sind. Sie werden im Controller implementiert und stellen die entsprechenden abgefragten Daten grafisch dar.

3.2.5 index.php und JavaScript

In dieser Datei befindet sich das visuelle Grundgerüst der Web Applikation und ein paar JavaScript Funktionen. Die JavaScript Funktionen sind für die Validierung der Benutzereingabe zuständig. Durch die Implementation der JavaScript Funktionen werden die Eingabefelder vor dem Absenden validiert. Folgende Eingaben werden von JavaScript überprüft.

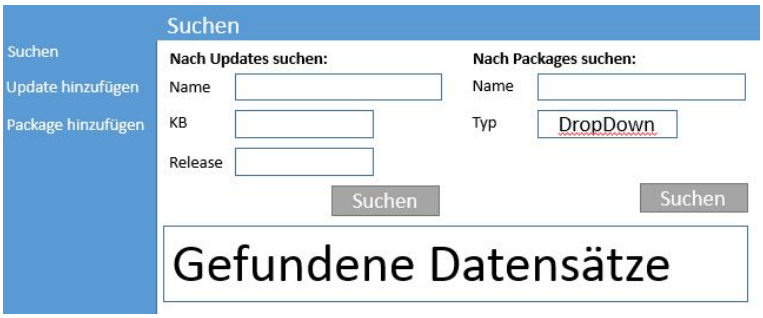
- **Suchfelder**
  Es darf nur ein Suchfeld, sobald der Benutzer ein Suchfeld ausgefüllt hat, werden die anderen Felder gräulich hinterlegt und der Benutzer kann die Suche ausführen. Das Suchfeld kb darf nur Zahlen beinhalten, falls im Suchfeld andere Zeichen vorhanden sind, wird der Suchknopf deaktiviert und eine Fehlermeldung erscheint. Das Erscheinungsdatum darf nur ein Datum haben im Format dd.mm.YYYY haben, ansonsten wird der Suchknopf deaktiviert und eine Fehlermeldung erscheint.

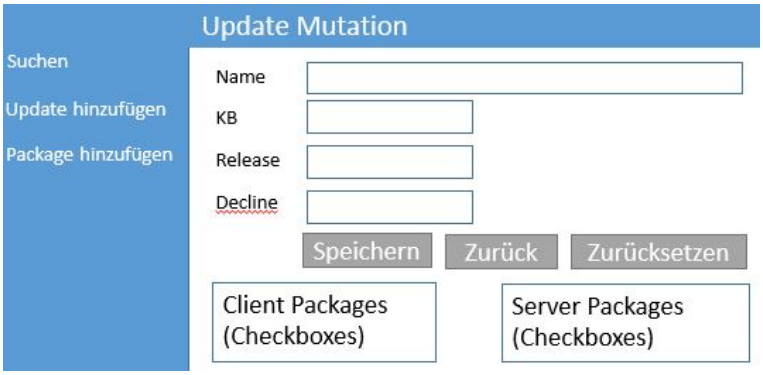
- **Update Management** Der Knopf um das Update zu speichern wird erst aktiviert, wenn in den Feldern Name, KB und Erscheinungsdatum gültige Werte vorhanden sind. Sobald ein Package für das Update ausgewählt wird, wird der Suchknopf deaktiviert solange der Benutzer das Package wieder abwählt oder ein gültiges Freigabedatum für den Typ dieses Packages eingibt. Diese Validierung gilt für das Eröffnen oder Ändern eines Eintrags.

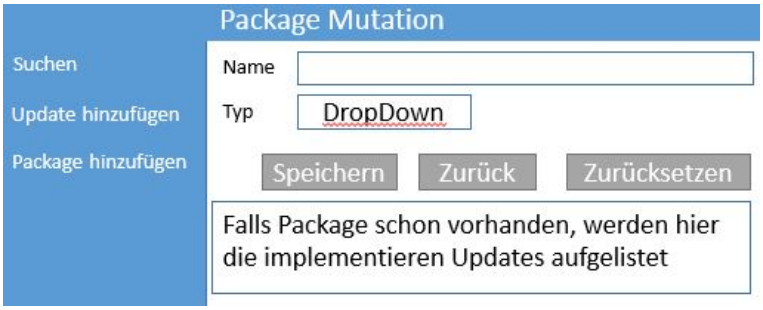
- **Package Management** Der Knopf um das Package zu speichern wird erst aktiviert, wenn in dem Feld Name ein gültiger Wert eingegeben wurde. Diese Validierung gilt für das Eröffnen oder Ändern eines Eintrags.

4 Entwurf

Die Webseite besteht aus drei verschiedenen Ansichten. Eine für die Suche, eine um Updates zu mutieren und eine um Packages zu mutieren.







5 Testkonzept

5.1 Teststratgie

Aus Zeitgründen werden nur Modelklassen als Units getestet. Die Modelklasse sind hauptsächlich für die Datenbankanbindung und den Datenbankabfragen zuständig. Der Controller wird erst bei dem Intergrationstest herangezogen. Und die View wird zum Schluss bei den Systemtest getestet. Diese Teststrategie mach durchaus Sinn, da um die Controllerklassen komplett zu testen, die Resultate aus den Datenbankabfragen benötigt werden. Die View wird nur visuell getestet, da die Viewdateien relativ klein und übersichtlich sind. Der Systemtest wird überprüfen, ob die Anforderungen erfüllt wurden. Zum Schluss wird es noch einen Abnahmetest durch den Auftraggeber geben, dieser Test entscheidet, ob die Software freigegeben werden kann.

5.2 Testwerkzeug

Es wurde versucht, das PHPUnit Modul als Testwerkzeug in die Applikation zu implementieren, jedoch hat das nicht wie gewünscht geklappt. Somit wurde eine PHP Klasse mit manuellen assert-Methoden erstellt. Es wurde ein eigenes Testwerkzeug erstellt, welches durch die cmd aufgerufen werden kann, und die Testresultate in der cmd ausgibt. Dieses Testwerkzeug wird für die Unit und Integrationstests benutzt.

5.3 Unit Tests der Modelklassen

In der Modelklasse werden alle Methoden vereinzelt getestet, die benötigten GET- und POST-Werte werden mit Dummy Werte ausgeführten. In der untenstehenden Tabelle werden alle benötigten Tests aufgelistet. Die Resultatet werden im Nachhinein in der Tabelle eingetragen.

Name	**Wo**	**Was wird getestet**	**Resultat**
constructTest()	Allen Klassen	Ob die Datenbankverbindung aufgebaut werden kann	Erwartet: True Erfolgreich
findIdTest()	mdl_updates	Mit Hilfe eines GET-Werts wird ein Updates gesucht.	Erwartet: NotNull Erfolgreich
gpTest()	mdl_updates	Es wird nach allen Packages gesucht	Erwartet: NotNull Erfolgreich
gopTest()	mdl_updates	Mit Hilfe eines GET-Werts werden alle Packages, die nicht für das Update sind gesucht	Erwartet: NotNull Erfolgreich

Name	**Wo**	**Was wird getestet**	**Resultat**
pTuTest()	mdl_updates	Mit Hilfe eines GET-Werts werden alle Packages für das Update gesucht	Erwartet: NotNull Erfolgreich
saveTest()	mdl_updates	Es wird ein neuer Datensatz gespeichert	Erwartet: True Erfolgreich
updateTest()	mdl_updates	Es wird ein vorhandener Datensatz geändert	Erwartet: True Erfolgreich
deleteTest()	mdl_updates	Es wird ein vorhandener Datensatz gelöscht	Erwartet: True Erfolgreich
gspTest()	mdl_packages	Mit Hilfe eines GET-Werts wird ein Package gesucht	Erwartet: NotNull Erfolgreich
saveTest() updateTest() deleteTest()	mdl_packages	Gleiches Tests wie bei den Methoden der Update-Klasse	Erwartet: True Erfolgreich
uTpTest()	mdl_packages	Mit Hilfe eines GET-Werts werden alle Updates für das Package gesucht	Erwartet: NotNull Erfolgreich
getTypesTest()	mdl_packages	Mit Hilfe eines GET-Werts wird der Typ des Packages ermittelt	Erwartet: NotNull Erfolgreich
fAuTest()	mdl_search	Es sollen alle Updates zurückgegeben werden	Erwartet: NotNull Erfolgreich
fspTest()	mdl_search	Mit Hilfe eines Suchbegriffes alle betroffenen Packages zurückgeben	Erwartet: NotNull Erfolgreich
gatTest()	mdl_search	Alle Typen zurückgegeben	Erwartet: NotNull Erfolgreich

Das sind die wichtigsten Test die für diese Applikation benötigt werden, man könnte noch Tests implementieren, die falsche Werte zurückgeben müssen, jedoch wurde wegen Zeitgründen auf diese Tests verzichtet.

5.4 Integrationstests

Die Integrationstest koppeln das Model und den Controller zusammen und überprüfen, ob das Zusammenspiel der beiden Komponenten auf Eingaben richtig reagiert. Der Aufbau ist bei jedem Integrationstest der Gleiche. Die Klasse class.resolver wird mit verschiedenen Eingaben aufgerufen und überprüft, ob Sie die richtigen Klassen und Methoden im Controller und im Model aufrufen. Zum Schluss wird überprüft, ob der Controller die richtigen Werte der View übergibt.

Diese Tests sind sehr aufwendig und nur teilweise implementiert worden. Auf die einzelnen Tests wird nicht weiter eingegangen, da diese wie schon vorhin gesagt, im Aufbau sehr ähnlich sind.

5.5 Systemtest

Der Systemtest wird genutzt um zu überprüfen, ob alle Anforderungen erfüllt wurden. Bei dem Systemtest wird die Applikation aus den Augen des Auftraggebers betrachtet und dadurch entschieden, ob die Anforderung erfüllt wurde oder nicht.

Speichern, Ändern und Löschen von Updates und Packages
Der Benutzer kann in jeweils zwei verschiedenen Masken Updates und Packages hinzufügen, ändern oder löschen. Eingabefelder die ein bestimmtes Format benötigen, werden vor dem Absenden überprüft.

Suchen von Updates und Packages
Diese Anforderung wurde erfüllt, der Benutzer kann nach Updates und Packages suchen. Die Sucheingaben werden vor dem Absenden auf Richtigkeit überprüft. Somit kann der Benutzer keine Suchanfrage starten, wenn er nichts eingegeben hat oder unzulässige Zeichen benutzt.

Dem Package einen Typ zuordnen
Der Benutzer muss jedem aufgenommenen Package einen Typ zuordnen, da jedes Packages einem Typ zugeordnet werden muss. Diese Aktion wird in der Package Management Maske gemacht.

Updates einen bestimmten Package aufzeigen
Sobald der Benutzer ein Package öffnet werden auf der unteren Hälfe der Seite die Updates, die sich in diesem Package befinden, aufgezeigt.

Verknüpfung Updates Packages
Diese Anforderung wurde erfüllt. Im Update Management kann der Benutzer das Update jederzeit einem Package hinzufügen oder wieder entfernen, ganz egal ob das Update schon vorhanden ist oder nicht.

Der Titel eines Updateeintrags darf nicht leer sein
Diese Anforderung wurde mit Hilfe von JavaScript erfüllt, solange der Titel leer ist, ist der Knopf um den Eintrag zu speichern oder zu ändern deaktiviert.

KB Nummer darf nicht leer und muss numerisch sein
Diese Anforderung wurde mit Hilfe von JavaScript erfüllt, solange der Wert der KB Nummer nicht numerisch ist, ist der Knopf um den Eintrag zu speichern oder zu ändern deaktiviert.

Veröffentlichungsdatum darf nicht leer sein
Diese Anforderung wurde mit Hilfe von JavaScript erfüllt, solange das Feld des Veröffentlichungsdatum leer ist, ist der Knopf um den Eintrag zu speichern oder zu ändern deaktiviert.

Update kann mehreren Packages zugeteilt werden
Diese Anforderung wurde erfüllt. Im Update Management werden alle Packages, egal ob das Update dem Package zugeteilt wurde oder nicht, angezeigt. Der Benutzer kann jeweils durch Auswahlboxen auswählen, welchem Package das Update zugeteilt ist und welchem nicht.

Update kann keinem Package zugeteilt sein Diese Anforderung wurde erfüllt. Ein Update ohne Packagezuteilung gespeichert werden.

Rückzug eines Updates
Diese Anforderung wurde erfüllt. Für diese Anforderung wurde ein eigenes Feld definiert. Sobald ein Update von Microsoft zurückgezogen wurde, kann man das Datum in diesem definierten Feld eingeben.

Datumsformat

Packagezuteilung und Freigabedatum
Diese Anforderung wurde teilweise erfüllt. Sobald der Benutzer ein Freigabedatum eines Updates setzt und keine Packages auswählt, wird der Knopf um den Datensatz zu speichern deaktiviert.

Korrektes Speichern eines Packages
Diese Anforderung wurde erfüllt. Ein Package kann gespeichert werden, sobald ein Titel eingetragen worden ist und ein Typ ausgewählt wurde, ansonsten ist der Knopf um das Package zu speichern oder zu ändern deaktiviert.

6 Fazit

6.1 Review

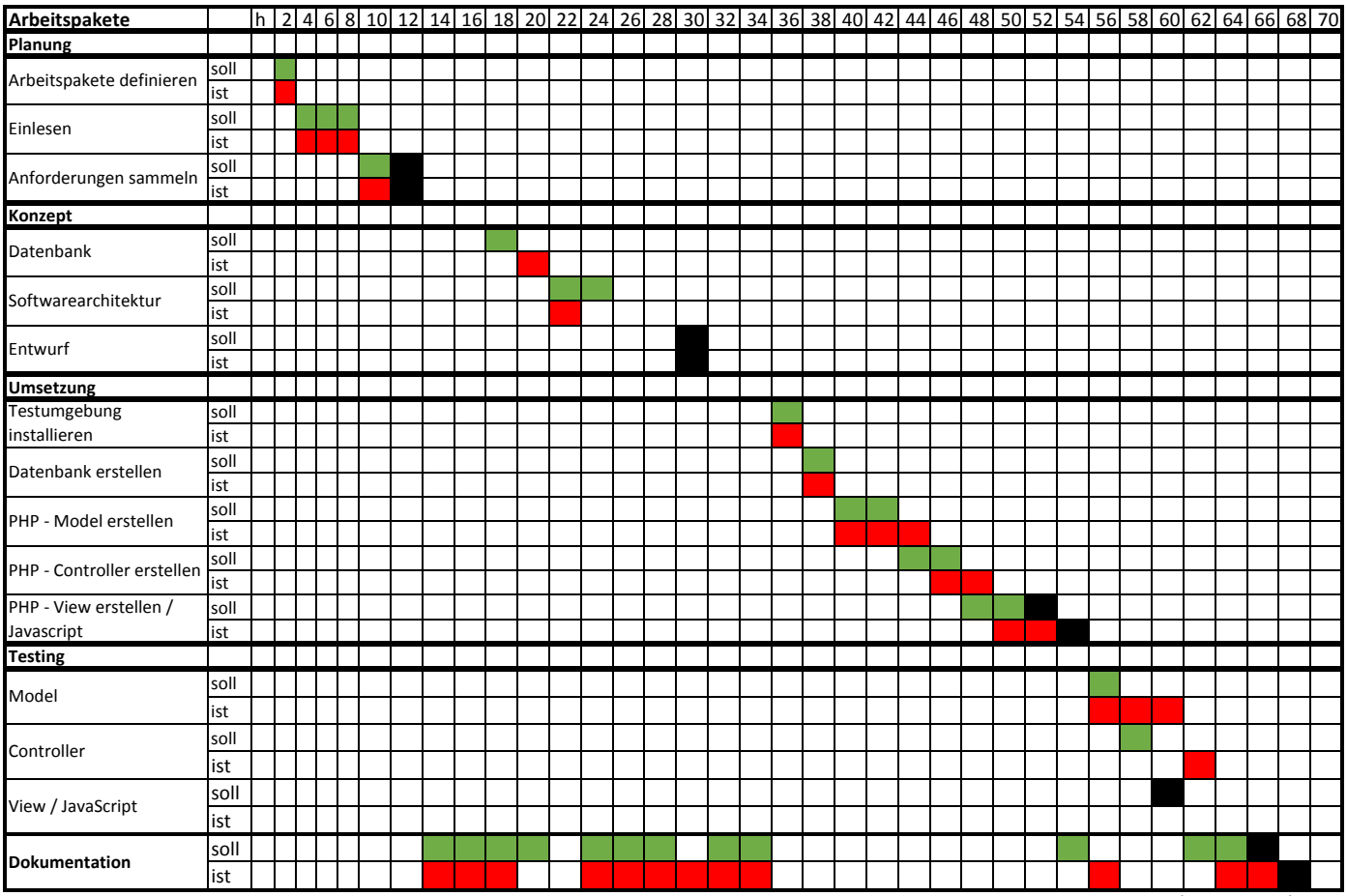
Der erste Teil des Projekts wurde vollständig implementiert und die Applikation, kann sobald die Dateien auf die produktive Umgebung verschoben werden genutzt werden. Das ganze Projekt war relativ einfach einzuschätzen, da nur Anforderungen und keine Wünsche vom Auftraggeber gegeben wurden. Jedoch habe ich die Zeit ein wenig unterschätzt, da ich nicht so gewandt bin mit PHP. Zusätzlich musste ich mich noch Leider wurde nicht das gesamte Testkonzept implementiert, da ich zwar PHPUnit installieren konnte, jedoch PHPUnit keine Teste durchführte. Als das Suchen einer Lösung zu zeitintensiv wurde, habe ich mich entschieden manuelle Testklassen zu schreiben, die in der CMD das Resultat der Tests ausgeben. Der fertige erste Teil des Projekt wurde vom Auftragsgeber kontrolliert und angenommen.

6.2 Ausblick

Es stehen noch folgende offene Arbeiten aus:

- Das Kopieren der Dateien auf den produktiven Server
- Die Implementation des zweiten Teil mit Planung, Entwicklung, Realisierung und Testen

7 Projetkplan



Meileinstein = schwarz

Literatur

[1] Klaus Pohl / Chris Rupp (Juni 2011): *Basiswissen Requirements Engineering*

[2] Mahbouba Gharbi / Arne Koschel / Andreas Rausch / Gernot Starke (Dezember 2012): *Basiswissen für Softwarearchitekten*

[3] Andreas Spillner / Tilo Linz (September 2012): *Basiswissen Softwaretest*

[4] Microsoft SQL Server Driver for PHP: http://php.net/manual/en/book.sqlsrv.php