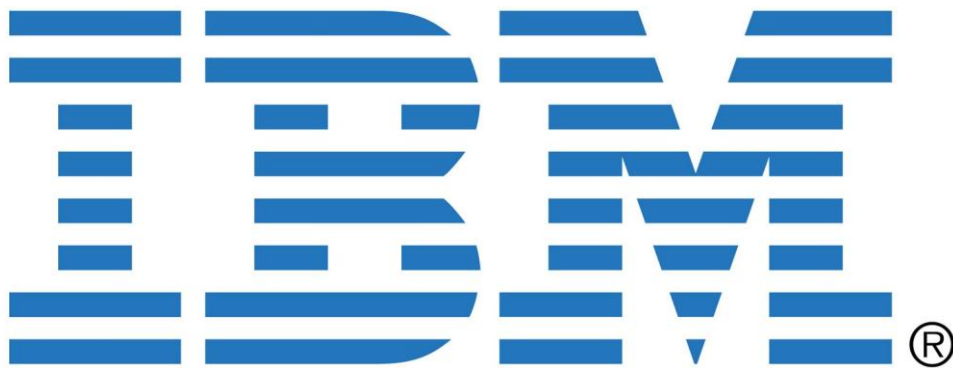


DV-Konzept WebEst

*Portierung des IBM Software Estimation Tool
als interaktive Webanwendung*



Ansprechpartner:

Andre Munzinger (a.munzinger@stud.tu-darmstadt.de)
Florian Friedrichs (friedrichsflorian@googlemail.com)
Dr. Holger Kremmin (holger.kremmin@de.ibm.com)

Stand: Montag, 10. Januar 2011

Inhaltsverzeichnis

1	Ziel des Dokumentes.....	1
2	Projektplanung	2
3	Technische Produktumgebung	3
3.1	Software.....	3
3.1.1	Server.....	3
3.1.2	Client.....	3
3.2	Hardware	3
4	Projektstruktur	4
4.1	Definition der Pakete	4
4.2	Paketstruktur in Java EE	5
4.3	Arten von Eclipse Java EE Projekttypen:	9
5	Datenbankarchitektur.....	12
5.1	Übersicht Datenmodell.....	12
5.2	Beschreibung Datenmodell.....	13
6	Softwarearchitektur.....	21
6.1	Allgemeine Komponenten	21
6.2	Spezifische Komponenten	23
6.2.1	Model: Java Persistence API und Datenbank Mapping	25
6.2.2	Logic: Enterprise Java Beans	27
6.2.3	Frontend: Java Server Faces.....	29
7	Testen und Validieren	31
7.1	White Box Tests.....	32
7.2	Black Box Tests.....	33
7.3	Validierung	34
8	Quellen	V
9	Appendix.....	VI
9.1	Testprotokolle – White Box Tests.....	VI
9.2	Testprotokolle – Black Box Tests	VI
9.3	Testprotokolle – Validierung	XII
9.4	Ausführliche Diagramme	XIV

Abbildungsverzeichnis

Abbildung 1: "Darstellung des Projektplans"	2
Abbildung 3: "Java EE Paketstruktur - Legende"	5
Abbildung 2: "Definition der Projektpakete"	4
Abbildung 4: "Java EE Paketstruktur – webest"	5
Abbildung 5: "Java EE Paketstruktur - webest persistence"	6
Abbildung 6: „Java EE Paketstruktur – webest processing“	7
Abbildung 7: „Java EE Paketstruktur – webest gui“	8
Abbildung 8: „Java EE Paketstruktur – webest webservice“	8
Abbildung 9: „Projekttypen - EAR Projekt“	9
Abbildung 10: „Projekttypen – EJB Projekt“	10
Abbildung 11: „Projekttypen – JPA Projekt“	10
Abbildung 12: „Projekttypen – WAR Projekt“	11
Abbildung 13: „Entity Relationship Model“	12
Abbildung 14: „Von der Datenbank zum Frontend“	21
Abbildung 15: „Model – Logic – Frontend“	24
Abbildung 16: „Model – Java Persistence API“	25
Abbildung 17: „Model – Entity Manager“	26
Abbildung 18: „Logic Enterprise Java Beans“	27
Abbildung 19: „Frontend – Java Server Faces“	30
Abbildung 20: „Testen und Validieren“	31

Tabellenverzeichnis

Tabelle 1: Applicationtype	14
Tabelle 2: Applicationtypegroup.....	14
Tabelle 3: Calculationparameter.....	14
Tabelle 4: Certainty	14
Tabelle 5: Constraint	14
Tabelle 6: Country.....	15
Tabelle 7: Defectcategory	15
Tabelle 8: Division	15
Tabelle 9: Effortunit.....	15
Tabelle 10: Estimate.....	15
Tabelle 11: Granularitylevel.....	16
Tabelle 12: Granularityquestion	16
Tabelle 13: Industrysector	16
Tabelle 14: Industrysectorgroup	16
Tabelle 15: Milestone.....	16
Tabelle 16: Monetaryunit	17
Tabelle 17: MTTDtimeunit.....	17
Tabelle 18: Organisation	17
Tabelle 19: Phasen	17
Tabelle 20: PIHistorycategory.....	17
Tabelle 21: PIHistoryentry.....	18
Tabelle 22: PIPPLookup	18
Tabelle 23: Projectenvironment	18
Tabelle 24: Role	18
Tabelle 25: Beschriftung	19
Tabelle 26: Solution_Granularityquestion.....	19
Tabelle 27: Solution_Staffingshape	19
Tabelle 28: Staffingshape.....	19
Tabelle 29: Template.....	20
Tabelle 30: Usecasecomplexity	20
Tabelle 31: Usecasepack.....	20
Tabelle 32: User.....	20
Tabelle 33: Black Box Test – Login.....	VII
Tabelle 34: Black Box Test - New Estimate	VIII
Tabelle 35: Black Box Test - Modify Estimate.....	VIII
Tabelle 36: Black Box Test - Clone Estimate	VIII
Tabelle 37: Black Box Test – Delete Estimate.....	IX
Tabelle 38: Black Box Test - Show Estimate.....	IX
Tabelle 39: Black Box Test - New Solution	X
Tabelle 40: Black Box Test - Modify Solution.....	X
Tabelle 41: Black Box Test - Clone Solution	XI

Tabelle 42: Black Box Test - Delete Solution	XI
Tabelle 43: Black Box Test - Show Solution	XI
Tabelle 44: Black Box Test - Show Report	XII
Tabelle 45: Validierung - Lastentest	XIII
Tabelle 46: Validierung - Multi User Test	XIII

1 Ziel des Dokumentes

IBM Global Business Services wünscht wie bereits im Pflichtenheft beschrieben, eine Portierung des bereits im Unternehmen verwendeten Softwareschätzungstool als Webanwendung, genannt WebEst (Web Estimation).

Das vorliegende Datenverarbeitungskonzept (DV-Konzept) dient zur Darstellung der relevanten Daten, erläutert deren spezifische Verarbeitung und stellt damit den zweiten Schritt der Projektplanung nach dem Pflichtenheft dar. Ziel ist es, die bereits im Pflichtenheft erwähnten Aufgaben von ihrer technischen Seite zu beleuchten bzw. deren Umsetzung detailliert darzulegen. Außer der technischen Aspekte wird ebenfalls in diesem Dokument die detaillierte Projektplanung sowie das technische Produktumfeld aufgezeigt.

Die geplante Implementierung wird in diesem Dokument in folgende Hauptkapitel unterteilt.

- **Projektplanung**
Im Kapitel über die Projektplanung werden die einzelnen Meilensteine vorgestellt und der Projektplan präsentiert.
- **Technische Produktumgebung**
In diesem Abschnitt werden die technischen Grundlagen und das Umfeld in dem WebEst arbeiten soll.
- **Projektstruktur**
In diesem Kapitel wird die Struktur des Projekts näher erläutert. Es werden die verwendeten Projekttypen und deren Zuordnung zu den einzelnen Paketen beschrieben. Aus der beschriebenen Struktur wird ebenfalls die Unterteilung der Teammitglieder abgeleitet.
- **Datenbankarchitektur**
Hier werden das Datenmodell sowie die zugehörigen Relationen erläutert. Desweiteren findet eine detaillierte Vorstellung der einzelnen Tabellen statt.
- **Softwarearchitektur**
Im Kapitel Softwarearchitektur wird der eigentliche Problemlösungs- bzw. Datenverarbeitungsprozess beschrieben. Außerdem werden hier weitere Module beschrieben, welche zusätzliche Funktionalität anbieten.
- **Testen und Validierung**
Unter dem Punkt "Testen und Validierung" findet eine Beschreibung der einzelnen Testarten und deren Testfälle statt. Dieser Punkt dient zur abschließenden Qualitätssicherung (Qualitätsmanagement) und gewährleistet die Funktionalität der WebEst Anwendung.

2 Projektplanung

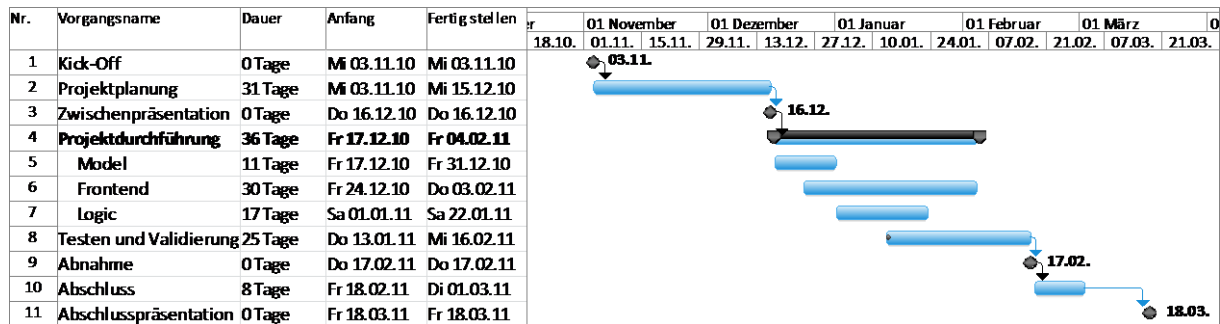


Abbildung 1: "Darstellung des Projektplans"

Nach dem Kick-Off Anfang November 2010 beginnt die Projektplanung, die sich mit der Evaluierung der einzusetzenden Software, Prototyping einzelner Komponenten und der Erstellung von Pflichtenheft und DV-Konzept beschäftigt.

Mit der Vorstellung der Projektplanung bei der Zwischenpräsentation beginnt die Implementierung der ersten Anforderungen. Die Datenbank bildet dabei die Grundlage für alle aufsetzenden Schichten und muss somit zuerst realisiert werden. Kurz darauf folgt das Frontend, welches vorerst mit leeren Feldern ohne Daten entworfen wird, um diese dann später mit Unterstützung der Logic Schicht füllen zu können.

Während der Projektdurchführung werden dem Auftraggeber je nach zeitlicher Verfügbarkeit aktuelle Entwicklungsstände vorgeführt bzw. Rückfragen geklärt, damit die Software den Wünschen des Auftraggebers entspricht.

Bereits während Realisierung folgt die Test- und Validierungsphase, in der Blackbox und Whitebox-Tests durchgeführt werden, um Bugs und Fehler zu beseitigen.

Danach erfolgt in der Mitte vom Februar die Abnahme durch den Auftraggeber und damit der Startschuss zur Erstellung weiterer Dokumente, die am 01.03. abgegeben werden.

Voraussichtlich Mitte März wird die Abschlusspräsentation zur Vorstellung des Projektes gehalten. An diesem Tag wird auch das fertige Software-Produkt dem Auftraggeber übergeben.

Der maximal mögliche Puffer für die Implementierung beträgt 2 Wochen, damit die Qualität der Software nicht leidet.

3 Technische Produktumgebung

In diesem Kapitel wird das technische Umfeld der Anwendung näher erläutert. Da durch den Auftraggeber detaillierte Vorgaben zu der Produktumgebung gemacht hat, sind diese im Folgenden aufgeführt.

3.1 Software

3.1.1 Server

Das Produkt ist lauffähig auf einem IBM WebSphere Application Server Community Edition (IBM WASCE) in der Version 2.1. WASCE basiert auf einem Apache Geronimo Application Server welcher Apache Tomcat als Servlet Container verwendet. Als Datenbank kommt eine integrierte Apache Derby Instanz zum Einsatz.

Grundlage für den Server ist ein installiertes IBM oder Oracle Java 6 SDK.

Obwohl die WebEst¹ Anwendung grundsätzlich unabhängig vom Betriebssystem arbeiten soll, ist sie auf folgenden Betriebssystemen zu testen:

- Windows XP (32 Bit)
- Windows 7 (32/64 Bit)
- Windows Server 2008 R2 (32/64 Bit)
- Debian GNU Linux v5 (32/64 Bit)
- Redhat Linux v5
- Novell SUSE Linux 11

3.1.2 Client

Die grafische Oberfläche ist auf den Browser Mozilla Firefox 3.6 zu optimieren.

3.2 Hardware

Die Hardwarevoraussetzungen entsprechen denen des IBM WASCE. Die WebEst¹ Anwendung selbst benötigt keine darüber hinaus gehenden Hardwarevoraussetzungen². Für optimale Performanz ist eine entsprechend bessere Hardwareausstattung zu wählen.

¹ Web Estimation Software

² <http://www-01.ibm.com/software/webservers/appserv/community/syseq/>

4 Projektstruktur

Der nun vorgestellte Aufbau spiegelt die Projektstruktur in der zu verwendenden Entwicklungsumgebung wieder. Nach einer Auflistung der einzelnen Komponenten erfolgt eine detaillierte Erläuterung, die die genaue Entwicklung und den Problemlösungsprozess schildert.

4.1 Definition der Pakete

Bei der Definition der Pakete ist darauf zu achten, die bereits erläuterten Software-Architekturrichtlinien einzuhalten. Es erfolgt also eine Einteilung in drei Schichten.

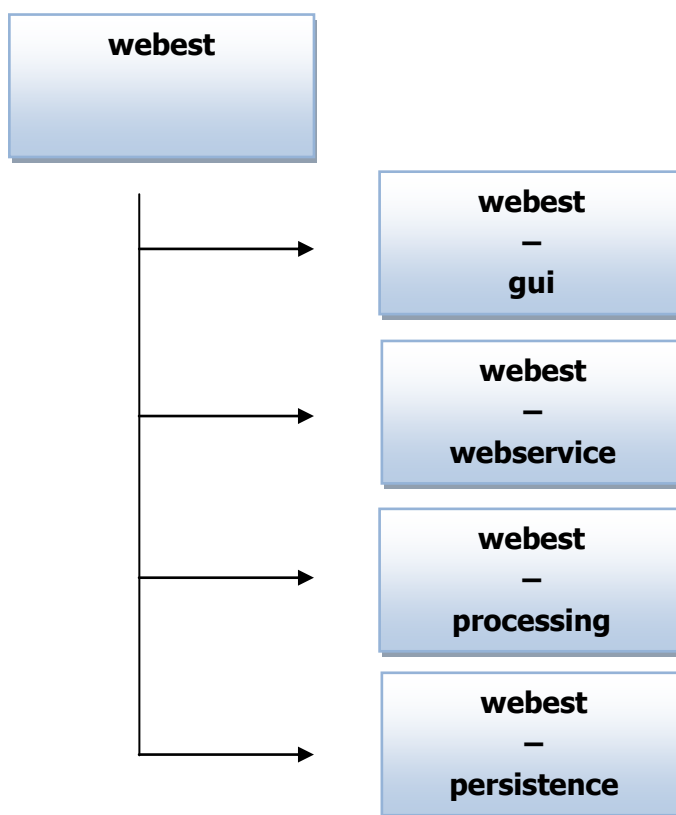


Abbildung 2: "Definition der Projektpakete"

Zusätzlich zum 3-Schichten-Modell gibt es ein Package "webservice", welches eine Funktionserweiterung der Software darstellt. In der Hierarchie sind die Webservices zwischen Logic und Frontend einzuordnen.

4.2 Paketstruktur in Java EE

Die Paketstruktur bietet einen modularen Aufbau der zu entwickelnden Anwendung in Java EE. Wie unter Punkt 4.1 bereits definiert, wird hier die genaue Paketstruktur genauer beleuchtet.

Die vollständige grafische Illustration der Paketstruktur ist im Anhang unter 0 „Ausführliche Diagramme“ zu finden.

Legende:

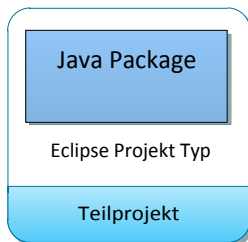


Abbildung 3: "Java EE Paketstruktur - Legende"

- **webest**
Dieses Projekt ist das Hauptprojekt der Anwendung. Es bündelt alle zugehörigen Ressourcen zu einem Enterprise Archive.
Als Projekttyp dieses Packages findet ein Enterprise Application Project (EAR Project) Verwendung. Das EAR Project beinhaltet den "Deployment Descriptor" und dient zur Publikation des Gesamtprojektes auf dem WASCE.



Abbildung 4: "Java EE Paketstruktur – webest"

- **webest-persistence**
Das Projekt "webest-persistence" ist für die persistente Datenhaltung zuständig. Die Unterteilung dieser Schicht beläuft sich auf folgende Einheiten (siehe Abb. 5):
 - **webest-persistence-model**
Das Projekt "webest-persistence-model" stellt die gewünschten Entitäten der Datenbank zur Verfügung. Als Projekttyp dient ein Java Persistence API Project.

- webest-persistence-service

Das Projekt "webest-persistence-service" dient zur Weiterverarbeitung der in "webest-persistence-model" implementierten Entitäten. Hier werden die sog. Facade Beans implementiert. Als Projekttyp ist ein Enterprise Java Beans Project zu verwenden.

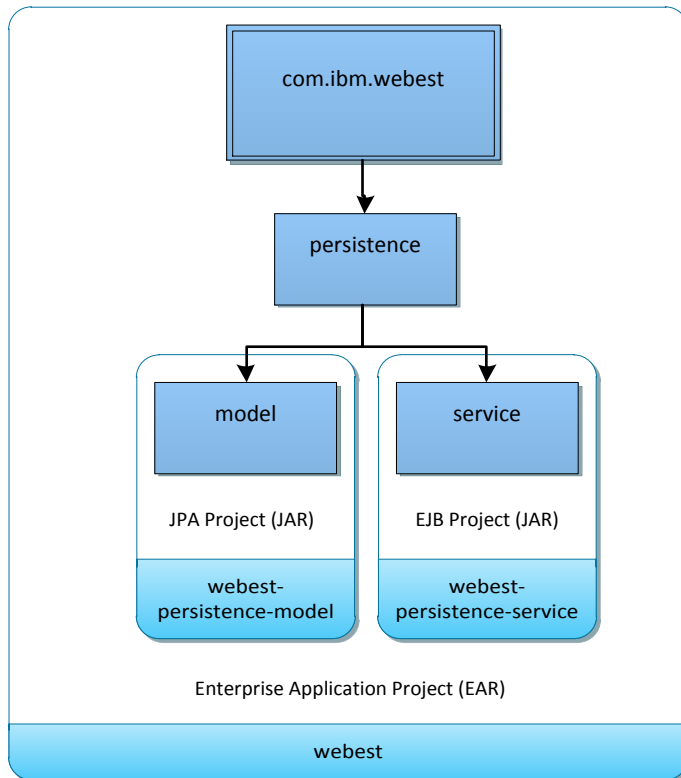


Abbildung 5: "Java EE Paketstruktur - webest persistence"

- webest-processing

Das Projekt "webest-processing" repräsentiert nun die zweite Schicht des 3-Schichten-Modells, die Logic-Schicht. Sie dient als Schnittstelle zwischen Model (webest-persistence-*) und Frontend (webest-gui-*).

Die Unterteilung in dieser Schicht beläuft sich auf folgende Einheiten (siehe Abb. 6):

- webest-processing-administration

Das Projekt "webest-processing-administration" dient zur Verwaltung von Estimates und Solutions. Es fungiert als Schnittstelle zwischen Model und Frontend. Als Projekttyp ist ein Enterprise Java Bean Project zu wählen.

- webest-processing-calculation

Das Projekt "webest-processing-calculation" stellt den eigentlichen Berechnungsprozess zur Verfügung. Hier erfolgt die Kalkulation der Solution mit Berücksichtigung der vom Benutzer eingegebenen Use Cases und Projektumgebungsvariablen. Die erstellte Kalkulation ist identisch mit dem Schätzprozess, welcher im Research Paper definiert wurde.

Als Projekttyp ist ein Enterprise Java Bean Project zu wählen.

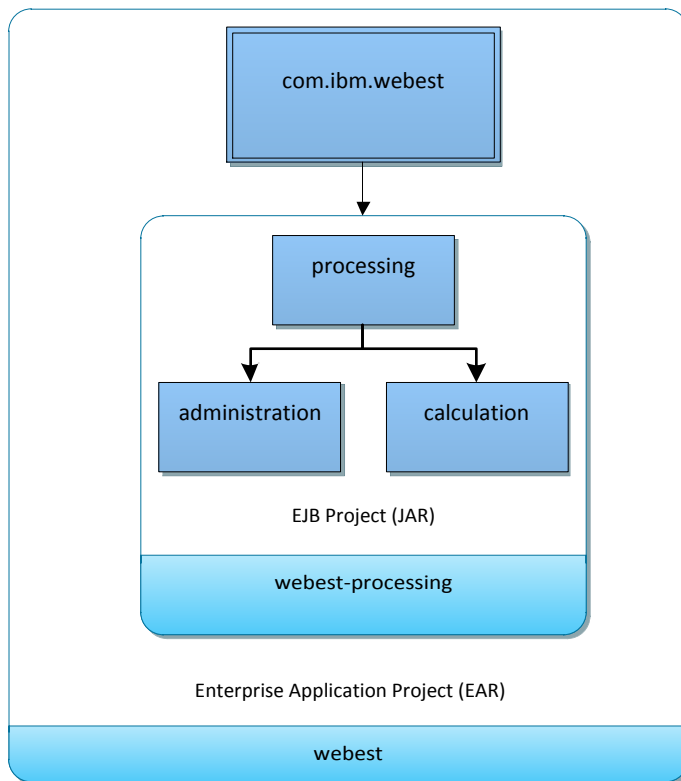


Abbildung 6: „Java EE Paketstruktur – webest processing“

- webest-gui

Das Projekt "webest-gui" repräsentiert die letzte und dritte Schicht unserer Anwendung, das Frontend (Java Server Faces).
Die Unterteilung in dieser Schicht beläuft sich auf folgende Einheiten (siehe Abb. 7):

 - webest-gui-action

Das Projekt „webest-gui-action“ dient zur Steuerung der einzelnen GUI Elemente und Webseiten. Die hier definierten Klassen dienen als sogenannte Backing Beans für die JSF Seiten. Sie versorgen die Ein- und Ausgabefelder mit Werten, stellen die Validität der Eingaben sicher und leiten Befehle an die Business Logic EJBs weiter.
Als Projekttyp ist ein Dynamic Web Project zu wählen.
 - webest-gui-view

Das Projekt "webest-gui-view" stellt das eigentliche Frontend zur Verfügung. Hier erfolgt die Gestaltung des Webseiten mittels CSS (Cascading Style Sheets) und den zu zusätzlichen Komponenten Java Server Faces, Apache MyFaces).
Als Projekttyp ist auch hier ein Dynamic Web Project zu wählen.

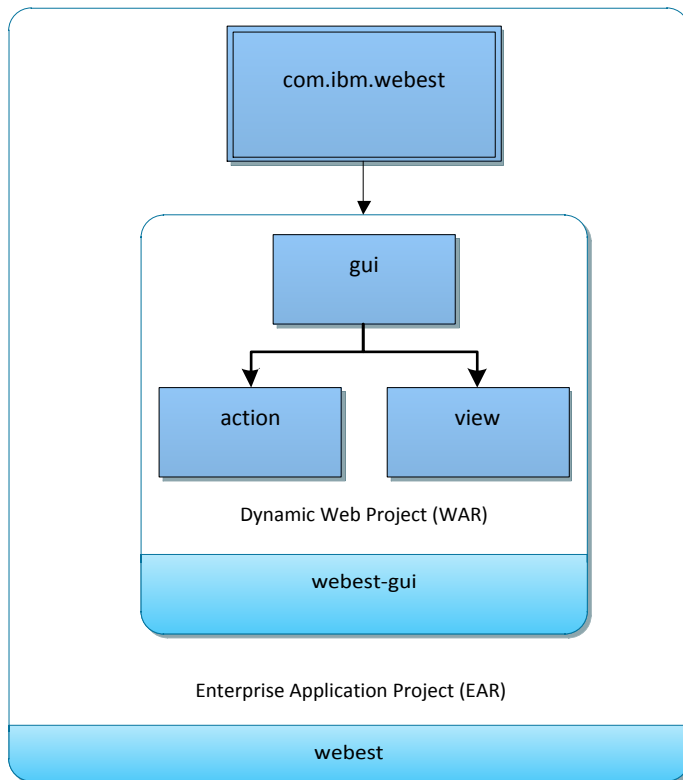


Abbildung 7: „Java EE Paketstruktur – webest gui“

- webest-webservice

Das Projekt "webest-webservice" dient zur Bereitstellung von Webservices. Diese ermöglichen einen einfachen Zugriff von externen Applikationen.

Als Projekttyp ist ein Enterprise Java Bean Project zu wählen.

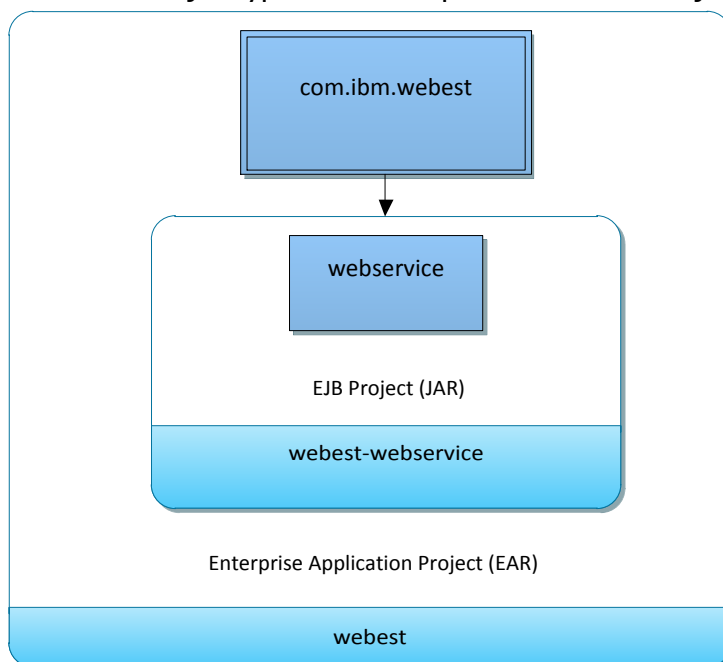


Abbildung 8: „Java EE Paketstruktur – webest webservices“

Die Packages der Testklassen sind unter Punkt 4.2 explizit nicht berücksichtigt worden. Die genaue Definition und Verwendung des Testens mittels JUnit ist unter Punkt 9.1 „Testen und Validieren – White Box Tests“ zu finden.

4.3 Arten von Eclipse Java EE Projekttypen:

Die folgenden Projekttypen werden in der Entwicklung des Problemlösungsprozess benannt und finden unter Punkt 4.2 Verwendung. In den Grafiken sind die, in den jeweiligen Projekten zu verwendenden Konfigurationsdateien dargestellt. In den Kreisen sind die jeweiligen Komponenten (Datenbank, JPA, Sicherheit etc.) aufgeführt, die in diesen Konfigurationsdateien zu konfigurieren sind.

- Enterprise Application Project (EAR)
Das Enterprise Application Project dient, wie in Punkt 4.2 bereits erläutert, zur Publikation der gesamten Java EE Applikation. Es führt die erstellten *.WAR und *.JAR Projekte zu einem Enterprise Application-Project (EAR) Projekt zusammen, welches im Anschluß auf dem WASCE deployed werden kann.

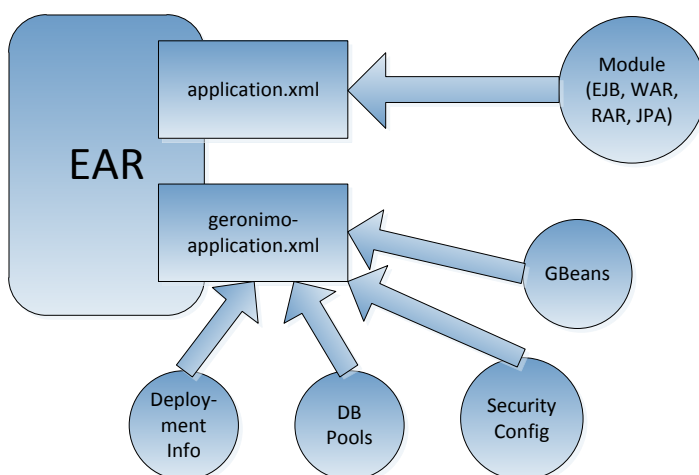


Abbildung 9: „Projekttypen - EAR Project“

- Enterprise Java Beans Project (JAR)
Der Projekttyp „Enterprise Java Beans Project“ dient zur Definition der einzelnen Enterprise Java Beans (EJBs) definiert. Eine Erläuterung dieser Beans ist unter Punkt **Fehler! Verweisquelle konnte nicht gefunden werden.** zu finden.

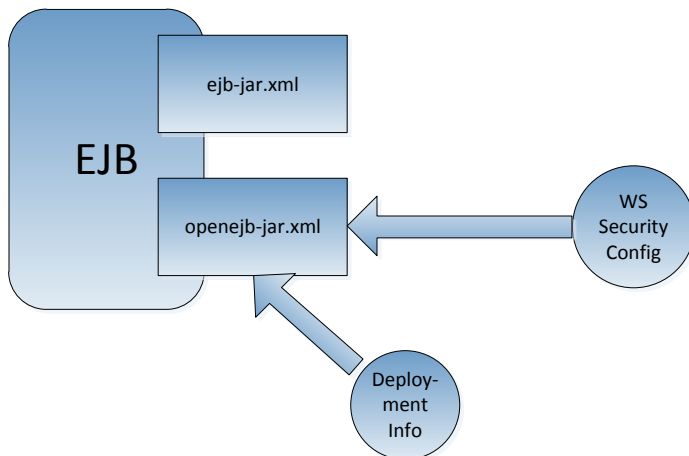


Abbildung 10: „Projekttypen – EJB Projekt“

- Java Persistence API Project (JAR)
In dem Java Persistence API Project findet die Definition der Datentabellen und deren Relationen zueinander statt. Eine Erläuterung dieser ist unter Punkt **Fehler! Verweisquelle konnte nicht gefunden werden.** zu finden.

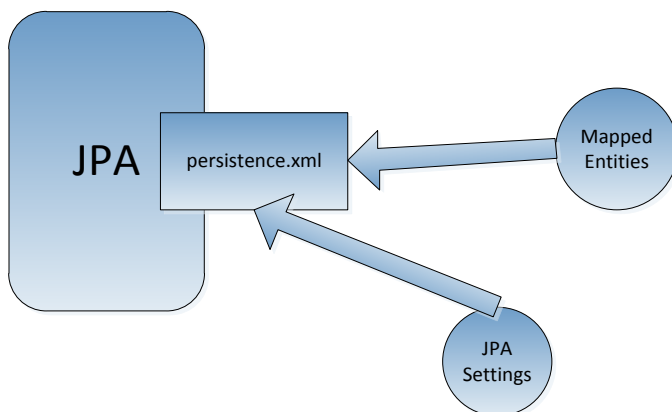


Abbildung 11: „Projekttypen – JPA Projekt“

- Dynamic Web Project (WAR)
Das Dynamic Web Project dient zum Erstellen des Frontends, der JSF-Seiten mit all ihren Komponenten. Die Erläuterung der hier zu verwendenden Komponenten ist unter Punkt **Fehler! Verweisquelle konnte nicht gefunden werden.** zu finden.

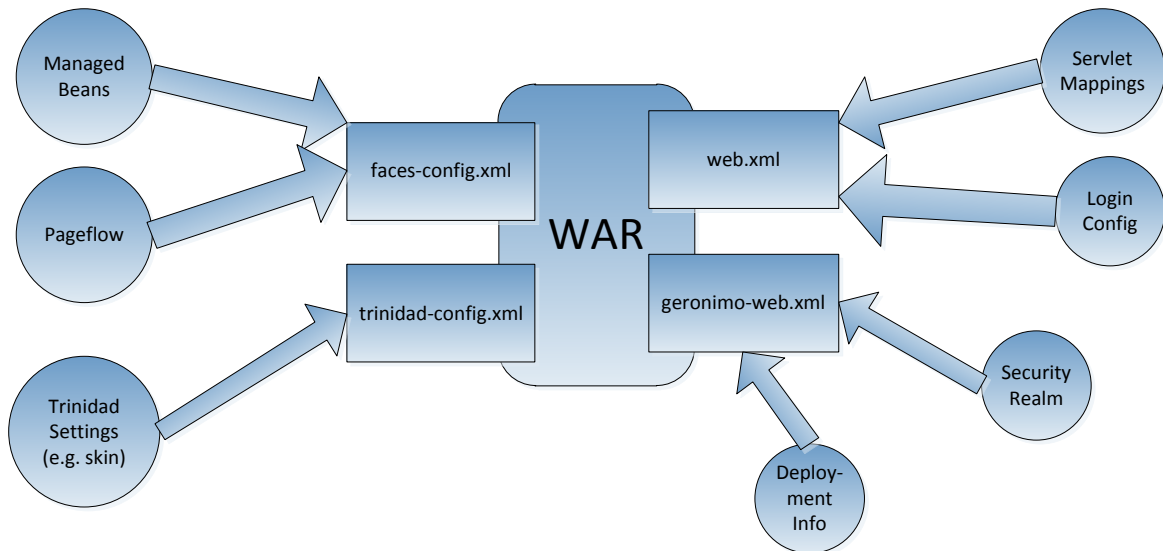


Abbildung 12: „Projekttypen – WAR Projekt“

5 Datenbankarchitektur

5.1 Übersicht Datenmodell

Das nachfolgende Entity Relationship Model (ERM) dient als Grundlage der Datenmodellierung der zu entwickelnden WebEst¹ Applikation. Es stellt neben den benötigten Daten auch die Relationen, also die Beziehungen und Verknüpfungen zwischen den einzelnen Datenpaaren illustrativ zur Verfügung und gilt im Folgenden zu verwenden.

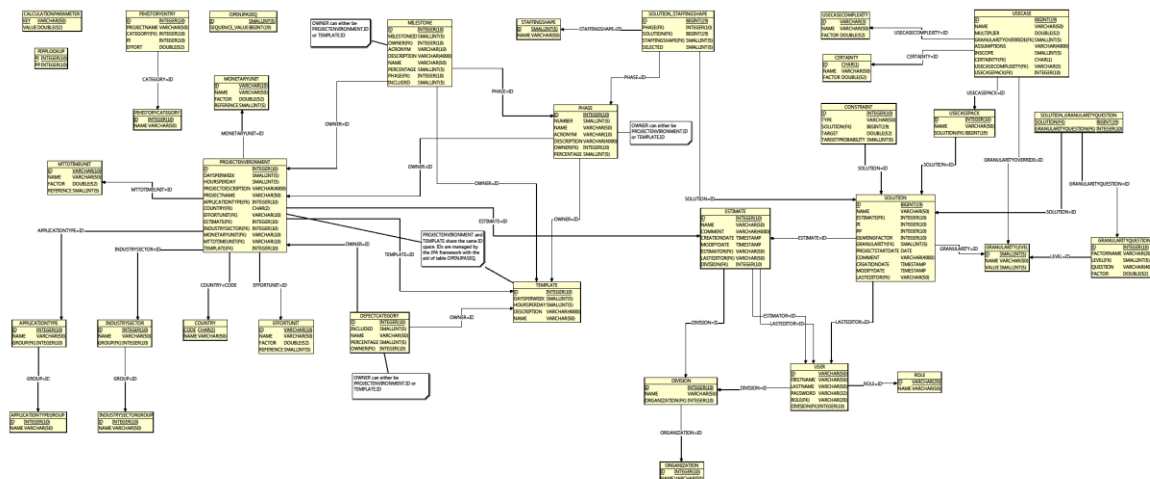


Abbildung 13: „Entity Relationship Model“

Das vollständige Entity Relationship Model kann im Anhang unter Punkt 0 „Ausführliche Diagramme“ eingesehen werden.

¹ Web Estimation Software

5.2 Beschreibung Datenmodell

Das hier geschilderte Datenmodell beschreibt die genaue Ausgestaltung der einzelnen Datentabellen. Neben dem Tabellennamen und des Datenfeldes wird auf die individuelle Beschränkung des Typs hingewiesen (data type, maxrange etc.).

Die wesentlichen Tabellen aus dem Datenmodell werden unter diesem Punkt kurz vorgestellt.

Die folgenden Tabellen sind als zentral anzusehen.

- **Tabelle USER**
Die Tabelle User regelt die Zugangsberechtigung und Rollenverteilung der zu entwickelnden Anwendung.
- **Tabelle ESTIMATE**
In der Tabelle Estimate findet die genaue Zuordnung des Estimate Namens, Autors und Erfassungs-/ Bearbeitungsdatums statt.
- **Tabelle PROJECTENVIRONMENT**
Zu jedem Estimate werden im ProjectEnvironment individuelle Projektdaten gespeichert.
- **Tabelle PHASE**
Ein Projekt ist in verschiedene Phasen gegliedert. Diese werden hier definiert.
- **Tabelle MILESTONE**
Zu jedem Projekt werden hier verschiedene zu erreichende Meilensteine festgelegt.
- **Tabelle SOLUTION**
Zu jedem Estimate können verschiedene Solutions angelegt werden, die eine individuelle Schätzung enthalten.
- **Tabelle USECASE**
Die Tabelle USECASE definiert die Use-Cases als Basis des Schätzprozess. Die Dokumentation des vollständigen Schätzprozesses ist im Research Paper zu finden.
- **Tabelle TEMPLATE**
Das Template dient als Schablone zur Vordefinierung einiger Pflichtfelder.

Darstellung der kompletten Datentabellen¹:

-  Primärschlüssel
 Fremdschlüssel

APPLICATIONTYPE



No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	
 3	GROUP	INTEGER(10)	N	

Tabelle 1: Applicationtype

APPLICATIONTYPEGROUP


No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	

Tabelle 2: Applicationtypegroup

CALCULATIONPARAMETER


No.	Logical Name	Type	Null	Description
 1	KEY	VARCHAR(50)	N	
2	VALUE	DOUBLE(52)	N	

Tabelle 3: Calculationparameter

CERTAINTY


No.	Logical Name	Type	Null	Description
 1	ID	CHAR(1)	N	
2	NAME	VARCHAR(50)	N	
3	FACTOR	DOUBLE(52)	N	

Tabelle 4: Certainty

CONSTRAINT



No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	TYPE	VARCHAR(50)	N	
 3	SOLUTION	BIGINT(19)	N	
4	TARGET	DOUBLE(52)	N	
5	TARGETPROBABILITY	SMALLINT(5)	Y	

Tabelle 5: Constraint

¹

Nummer	Feldname	Datentyp (Größe)	Null-Werte erlaubt (Ja/Nein)	Beschreibung
--------	----------	------------------	------------------------------	--------------

COUNTRY


No.	Logical Name	Type	Null	Description
 1	CODE	CHAR(2)	N	
2	NAME	VARCHAR(50)	N	

Tabelle 6: Country

DEFECTCATEGORY


No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	INCLUDED	SMALLINT(5)	N	
3	NAME	VARCHAR(50)	Y	
4	PERCENTAGE	SMALLINT(5)	Y	
5	OWNER	INTEGER(10)	N	TEMPLATE.ID or PROJECTENVIRONMENT.ID

Tabelle 7: Defectcategory

DIVISION



No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	
 3	ORGANIZATION	INTEGER(10)	N	

Tabelle 8: Division

EFFORTUNIT


No.	Logical Name	Type	Null	Description
 1	ID	VARCHAR(10)	N	
2	NAME	VARCHAR(50)	N	
3	FACTOR	DOUBLE(52)	N	
4	REFERENCE	SMALLINT(5)	N	

Tabelle 9: Effortunit

ESTIMATE





No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	
3	COMMENT	VARCHAR(4000)	Y	
4	CREATIONDATE	TIMESTAMP	N	
5	MODIFYDATE	TIMESTAMP	N	
 6	ESTIMATOR	VARCHAR(50)	N	
 7	LASTEDITOR	VARCHAR(50)	N	
 8	DIVISION	INTEGER(10)	N	

Tabelle 10: Estimate

GRANULARITYLEVEL


No.	Logical Name	Type	Null	Description
 1	ID	SMALLINT(5)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	
3	VALUE	SMALLINT(5)	N	

Tabelle 11: Granularitylevel

GRANULARITYQUESTION



No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	FACTORNAME	VARCHAR(20)	N	
 3	LEVEL	SMALLINT(5)	N	
4	QUESTION	VARCHAR(4000)	N	
5	FACTOR	DOUBLE(52)	N	

Tabelle 12: Granularityquestion

INDUSTRYSECTOR



No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	
 3	GROUP	INTEGER(10)	N	

Tabelle 13: Industrysector

INDUSTRYSECTORGROUP


No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	

Tabelle 14: Industrysectorgroup

MILESTONE



No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	MILESTONEID	SMALLINT(5)	N	
3	OWNER	INTEGER(10)	N	TEMPLATE.ID or PROJECTENVIRONMENT.ID
4	ACRONYM	VARCHAR(10)	N	
5	DESCRIPTION	VARCHAR(4000)	Y	
6	NAME	VARCHAR(50)	N	
7	PERCENTAGE	SMALLINT(5)	N	
 8	PHASE	INTEGER(10)	Y	
9	INCLUDED	SMALLINT(5)	N	

Tabelle 15: Milestone

MONETARYUNIT


No.	Logical Name	Type	Null	Description
 1	ID	VARCHAR(10)	N	
2	NAME	VARCHAR(50)	N	
3	FACTOR	DOUBLE(52)	N	
4	REFERENCE	SMALLINT(5)	N	

Tabelle 16: Monetaryunit

MTTDTIMEUNIT


No.	Logical Name	Type	Null	Description
 1	ID	VARCHAR(10)	N	
2	NAME	VARCHAR(50)	N	
3	FACTOR	DOUBLE(52)	N	
4	REFERENCE	SMALLINT(5)	N	

Tabelle 17: MTTDtimeunit

ORGANIZATION


No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	

Tabelle 18: Organisation

PHASE


No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	NUMBER	SMALLINT(5)	N	
3	NAME	VARCHAR(50)	N	
4	ACRONYM	VARCHAR(10)	N	
5	DESCRIPTION	VARCHAR(4000)	Y	
6	OWNER	INTEGER(10)	N	TEMPLATE.ID or PROJECTENVIRONMENT.ID
7	PERCENTAGE	SMALLINT(5)	Y	

Tabelle 19: Phasen

PIHISTORYCATEGORY


No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	

Tabelle 20: PIHistorycategory

PIHISTORYENTRY



No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	PROJECTNAME	VARCHAR(50)	N	
 3	CATEGORY	INTEGER(10)	N	
4	PI	INTEGER(10)	N	
5	EFFORT	DOUBLE(52)	N	

Tabelle 21: PIHistoryentry

PIPPLOOKUP


No.	Logical Name	Type	Null	Description
 1	PI	INTEGER(10)	N	
2	PP	INTEGER(10)	N	

Tabelle 22: PIPPLookup

PROJECTENVIRONMENT










No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	
2	DAYSPERWEEK	SMALLINT(5)	N	
3	HOURSPERDAY	SMALLINT(5)	N	
4	PROJECTDESCRIPTION	VARCHAR(4000)	Y	
5	PROJECTNAME	VARCHAR(50)	N	
 6	APPLICATIONTYPE	INTEGER(10)	N	
 7	COUNTRY	CHAR(2)	N	
 8	EFFORTUNIT	VARCHAR(10)	N	
 9	ESTIMATE	INTEGER(10)	N	
 10	INDUSTRYSECTOR	INTEGER(10)	N	
 11	MONETARYUNIT	VARCHAR(10)	N	
 12	MTTDTIMEUNIT	VARCHAR(10)	N	
 13	TEMPLATE	INTEGER(10)	Y	

Tabelle 23: Projectenvironment

ROLE


No.	Logical Name	Type	Null	Description
 1	ID	VARCHAR(20)	N	
2	NAME	VARCHAR(50)	N	

Tabelle 24: Role

SOLUTION





No.	Logical Name	Type	Null	Description
 1	ID	BIGINT(19)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	
 3	ESTIMATE	INTEGER(10)	N	
4	PI	INTEGER(10)	Y	
5	PP	INTEGER(10)	Y	
6	GEARINGFACTOR	INTEGER(10)	N	
 7	GRANULARITY	SMALLINT(5)	N	
8	PROJECTSTARTDATE	DATE	N	
9	COMMENT	VARCHAR(4000)	Y	
10	CREATIONDATE	TIMESTAMP	N	
11	MODIFYDATE	TIMESTAMP	N	
 12	LASTEDITOR	VARCHAR(50)	N	

Tabelle 25: Beschriftung

SOLUTION_GRANULARITYQUESTION





No.	Logical Name	Type	Null	Description
  1	SOLUTION	BIGINT(19)	N	
  2	GRANULARITYQUESTION	INTEGER(10)	N	

Tabelle 26: Solution_Granularityquestion

SOLUTION_STAFFINGSHAPE





No.	Logical Name	Type	Null	Description
 1	ID	BIGINT(19)	N	AUTO INCREMENT
 2	PHASE	INTEGER(10)	N	
 3	SOLUTION	BIGINT(19)	N	
 4	STAFFINGSHAPE	SMALLINT(5)	N	
5	SELECTED	SMALLINT(5)	N	

Tabelle 27: Solution_Staffingshape

STAFFINGSHAPE


No.	Logical Name	Type	Null	Description
 1	ID	SMALLINT(5)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	

Tabelle 28: Staffingshape

TEMPLATE


No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	
2	DAYSPERWEEK	SMALLINT(5)	N	
3	HOURSPERDAY	SMALLINT(5)	N	
4	DESCRIPTION	VARCHAR(4000)	Y	
5	NAME	VARCHAR(50)	N	

Tabelle 29: Template

USECASECOMPLEXITY


No.	Logical Name	Type	Null	Description
 1	ID	VARCHAR(3)	N	
2	NAME	VARCHAR(50)	N	
3	FACTOR	DOUBLE(52)	N	

Tabelle 30: Usecasecomplexity

USECASEPACK



No.	Logical Name	Type	Null	Description
 1	ID	INTEGER(10)	N	AUTO INCREMENT
2	NAME	VARCHAR(50)	N	
 3	SOLUTION	BIGINT(19)	N	

Tabelle 31: Usecasepack

USER




No.	Logical Name	Type	Null	Description
 1	ID	VARCHAR(50)	N	
2	FIRSTNAME	VARCHAR(50)	N	
3	LASTNAME	VARCHAR(50)	N	
4	PASSWORD	VARCHAR(32)	N	MD5 Hex String
 5	ROLE	VARCHAR(20)	Y	
 6	DIVISION	INTEGER(10)	Y	

Tabelle 32: User

6 Softwarearchitektur

6.1 Allgemeine Komponenten

Im Folgenden Kapitel werden die allgemeinen Komponenten der Softwarearchitektur definiert, welche den vollständigen Weg der Datenverarbeitung beschreiben. Nach Vorstellung und Erläuterung der allgemeinen Komponenten wird das spezifische Datenkonzept vorgestellt. Hierbei handelt es sich um eine an das MVC-Pattern³ angelehnte Architektur welche die Software-Entwicklung in drei verschiedene Schichten unterteilt. Wir sprechen von den Schichten Frontend, Logic und dem Model.

Abbildung 14 veranschaulicht den gewünschten Weg der Datenverarbeitung.

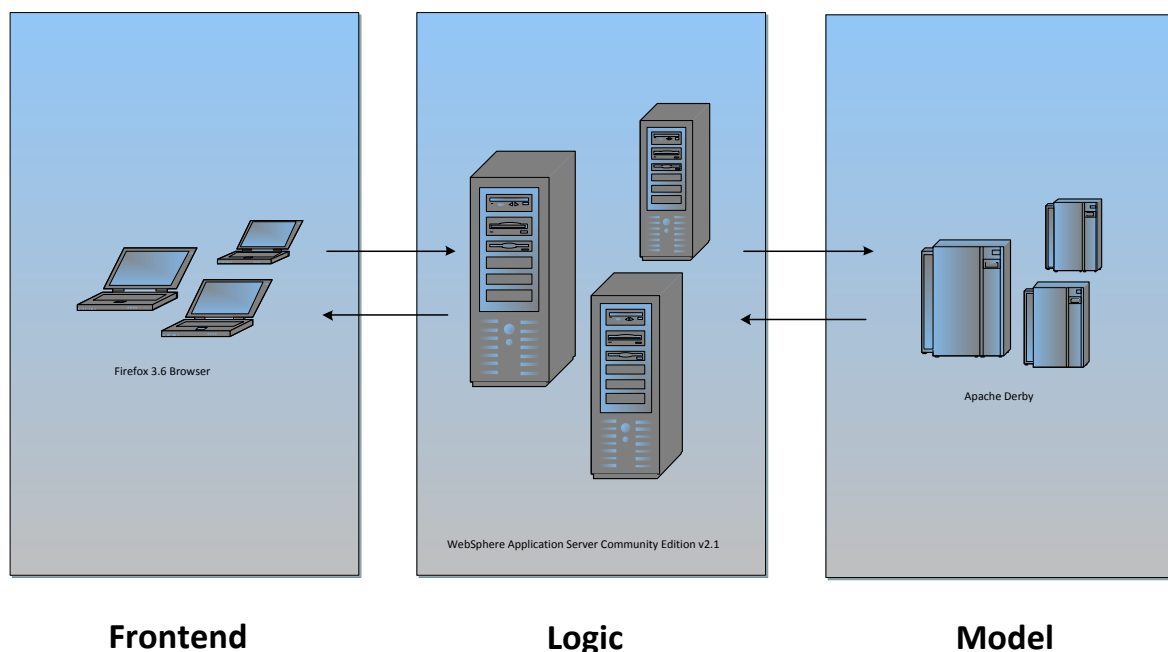


Abbildung 14: „Von der Datenbank zum Frontend“

Für die Entwicklung der WebEst¹ Applikation dient Apache Derby als Datenbank. Desweiteren dient WebSphere Application Server Community Edition (WASCE) als Anwendungsserver, respektive Application Server. Für das Frontend ist, wie im Pflichtenheft beschrieben, JavaServer Faces zu verwenden. Hierbei wird auf die Implementierung Apache MyFaces Trinidad zurückgegriffen.

¹ Web Estimation Software

³ http://www.wikipedia.de/Model_View_Controller

- Datenbank:
Die Datenbank wird mit Hilfe des Apache Derby implementiert. Apache Derby ist ein Java basiertes Datenbank Management System, das bereits in WASCE integriert ist. Es ermöglicht rationale Datenbanken für Java Anwendungen zu nutzen.⁴
- Application Server:
Der zu verwendende Open Source Java EE 5 Application Server ist "WebSphere Application Server Community Edition v2.1", dieser stellt mit seinen integrierten Komponenten die Plattform für das Ausführen von Java Enterprise Anwendungen zur Verfügung.⁵
- Browser/ Frontend:
Firefox 3.6
- Entwicklungsumgebung:
Eclipse Java EE
- Zusätzliche Komponenten:
IBM Java SDK v1.6

Durch die Implementierung des verwendeten 3-Schichten-Modell besteht die Möglichkeit der einfachen Erweiterbarkeit der Anwendung. Desweiteren ist im Vergleich zu anderen Programmmentwurfsmustern die Wiederverwendbarkeit der Applikation erheblich vereinfacht und verbessert.

⁴ <http://www.db.apache.org/derby/manuals/index.html>

⁵ <http://www-01.ibm.com/software/webservers/appserv/community>

6.2 Spezifische Komponenten

Die spezifischen Komponenten beschreiben das umzusetzende Entwicklungskonzept für die Implementierung der Software. Nach einer allgemeinen Erläuterung des Prozesses wird auf die gewählte Schichtenarchitektur eingegangen. Diese setzen sich zusammen aus:

- Model: Java Persistence API und Datenbank Mapping
- Logic: Enterprise Java Beans
- Frontend: Java Server Faces

Neben den drei Schichten wird das WebServices-Feature näher beschrieben. Diese WebServices stellen eine einfache Möglichkeit für externe Anwendungen dar, um auf Daten der WebEst¹ Applikation zuzugreifen. Die WebServices sind auf der Ebene der Logic-Schicht angesiedelt.

Die Kommunikation zwischen den einzelnen Schichten verläuft bidirektional zwischen den Schichten. Es soll nicht möglich sein, eine Schicht bei der Kommunikation zwischen den Schichten zu überspringen.

Im Folgenden wird der geplante Verlauf vom Frontend zur Datenbank beschrieben (GUI -> Datenbank). Der inverse Datenverarbeitungsprozess (Datenbank <- GUI) ist als entsprechend äquivalent anzusehen.

Die vom Endanwender im Frontend eingegeben Informationen (beispielsweise Auswahl eines Optionsfeldes) werden in der Logic-Schicht verarbeitet und die Ergebnisse in die Datenbank gespeichert. In den folgenden Abschnitten findet eine genaue Erläuterung der einzelnen Schichten (Model, Logic und Frontend) statt.

¹ Web Estimation Software

Die nachfolgende Grafik (Abb. 15) zeigt auf anschauliche Weise den detaillierten Prozess der Datenverarbeitung.

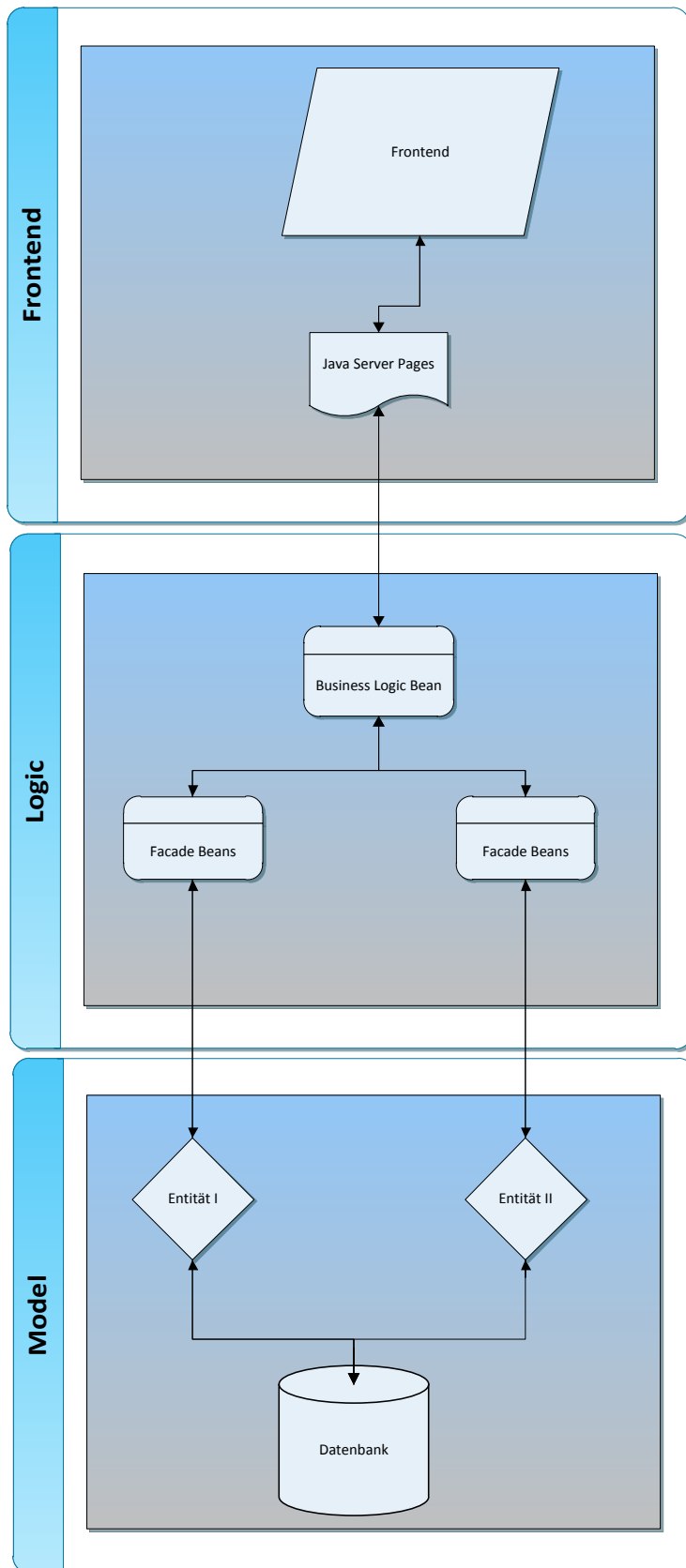


Abbildung 15: „Model – Logic – Frontend“

6.2.1 Model: Java Persistence API und Datenbank Mapping

Die Datenbank-Schicht speichert alle die Daten persistent und stellt die in der Anwendung benötigten Daten bereit. Als Datenbank wird wie oben beschrieben die in WASCE integrierte Datenbank Apache Derby verwendet. Die initiale Erstellung der Tabellen soll beim ersten Starten der Anwendung automatisch vorgenommen werden. Die in der Datenbank erstellten Relationen (siehe Punkt 5.2), werden über die Java Persistence API mittels „Object Relational Mapping“ (ORM) auf Klassen, die sog. Entitäten abgebildet, welche reine Datencontainer darstellen und ergänzend datenbankspezifische Funktionalitäten beinhalten. Dabei ist für jede Tabelle eine Klasse zu erstellen, dessen Attribute die Spaltenwerte aufnehmen.

Das Mapping zwischen den Entitäten und den Tabellen wird durch OpenJPA Annotationen im Java Code realisiert. Diese legen auch Primärschlüssel-Attribute fest und spezifizieren die Beziehungen (1:1, 1:n, n:1 oder n:m) zwischen mehreren Entitäten (Fremdschlüsselbeziehungen). Diese Beziehung können uni- oder bidirektional modelliert werden.⁶

Das unten dargestellte Beispiel illustriert auf anschauliche Weise die Datenverarbeitung von der Entität zur Datenbank.

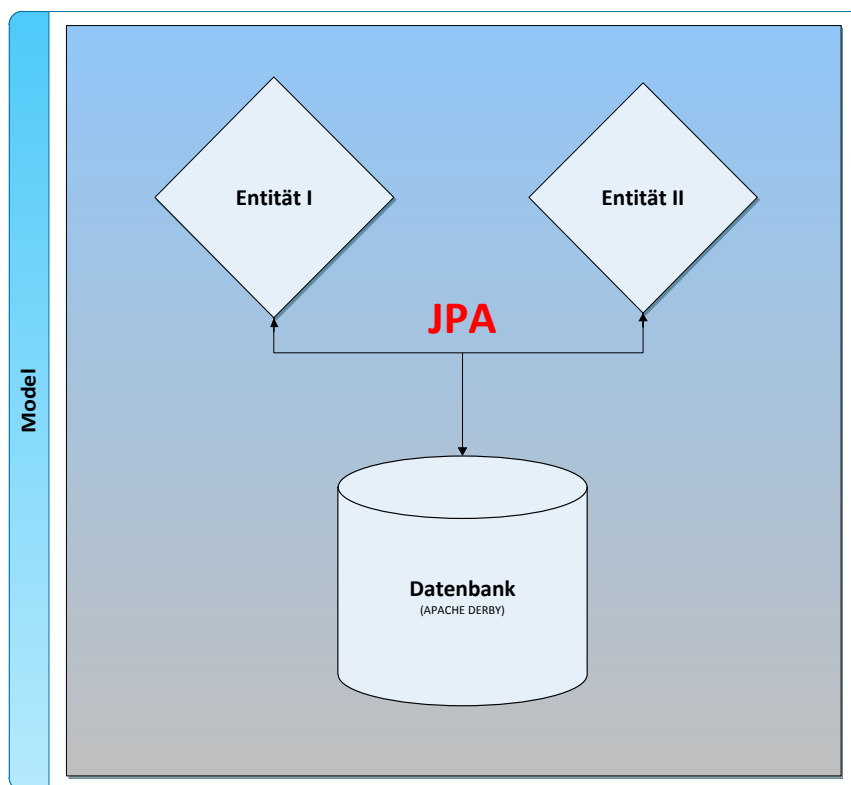


Abbildung 16: „Model – Java Persistence API“

⁶ IBM Redbook: Experience JavaEE! Using WebSphere Application Server Community Edition 2.1

Die Validierungskriterien für die zu speichernden Werte sollen bereits im Model angegeben werden, um sicher zu stellen, dass die Datenbank in einem konsistenten Zustand bleibt. Dies ist über Annotation der Java Bean Validation (JSR-303) zu realisieren, da diese direkt von den JSF Komponenten ausgewertet werden können.

Es ist das in WASCE integriertes JPA Framework OpenJPA zu verwenden.

Die direkten SQL-Anfragen für das Abfragen, Einfügen, Aktualisierten oder Löschen werden durch dieses Framework erledigt.

Der Entity Manager (siehe Abb. 17) ist ein zentraler Bestandteil der Java Persistence API. Er steuert die Verknüpfung zwischen Facade Beans und Entitäten. Über Ihn werden die einzelnen Entitäten angesteuert und können nun in den Facade Beans weiter verwendet werden.

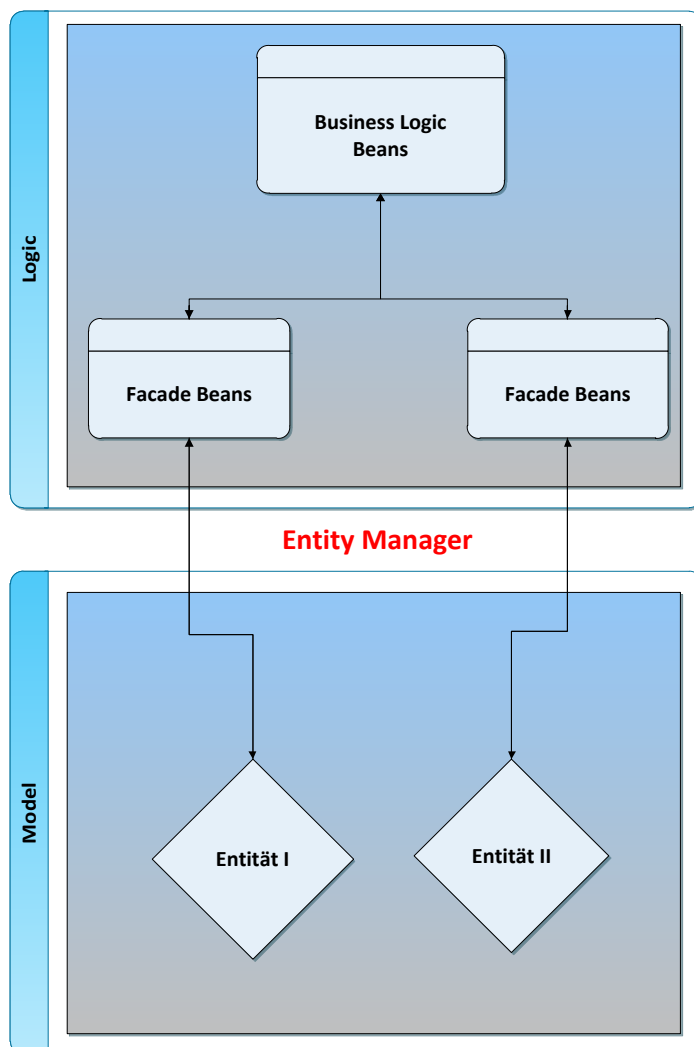


Abbildung 17: „Model – Entity Manager“

Eine abstrakte Abfrage der Entitäten findet in den Facade Beans statt. Diese sind des Weiteren auch dafür zuständig, in der Business Logic modifizierte Entities zur dauerhaften Speicherung an das Framework zu übergeben. Die Facades sind getrennt von den Entities als EJBs zu implementieren und versorgen die EJBs der Business Logic mit den gewünschten Daten.

Die Dokumentation zu der von uns zu verwendenden Version 1.2.2 des Frameworks, ist unter dem folgenden Link zu finden:⁷

6.2.2 Logic: Enterprise Java Beans

Der nächste Schritt beschreibt den Übergang von der Logic-Schicht in das Frontend. Die Logic-Schicht, welche die gesamte Business Logic beinhalten wird, erhält die zu bearbeitenden Daten von der darunter liegenden Datenbank-Schicht oder aber von der GUI. Diese Daten werden nun mittels der Business Logic verarbeitet und ausgewertet. Die Ergebnisse sollen im Anschluss in Form von zuvor definierten Objekttypen, zu Speicherung an die Datenbank, oder aber zur Darstellung auf der Website an das Frontend weitergereicht werden.

Die zu implementierenden Beans, bei welchen es sich um Enterprise Java Beans (EJB 3.0) handelt, sind ausschließlich als stateless (zustandslos) zu deklarieren. Dies hat den Vorteil, dass es nur eine Instanz des Business Logic Bean geben kann und somit alle Anfragen im gleichen Zustandskontext beantwortet werden.

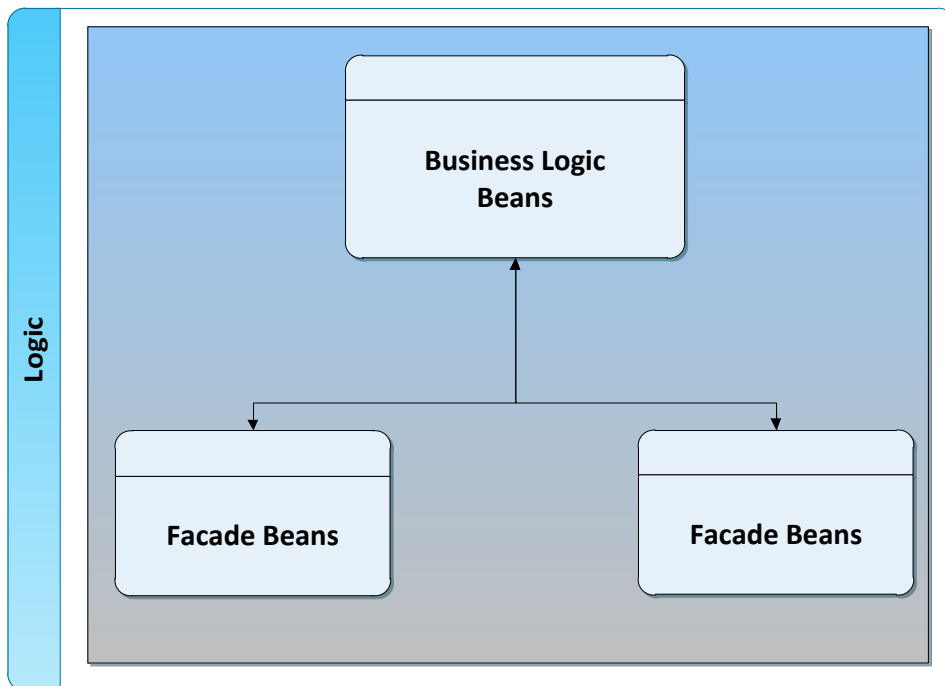


Abbildung 18: „Logic Enterprise Java Beans“

⁷ <http://openjpa.apache.org/builds/1.2.2/apache-openjpa-1.2.2/docs/manual/index.html>

Jedoch müssen so dem Bean bei jedem Aufruf, alle zur Berechnung notwendigen Information als Parameter übergeben werden. Mittels der Business-Logic-Beans sollen nun alle für das Software Estimation Projekt benötigten Berechnungs- und Administrations Methoden implementiert werden.

Die in Abbildung 18 abgebildete Business Logic Bean fungiert bidirektional. Die erste Funktionalität besteht darin, bereits bestehende Facade Beans zu zusammenzuführen und die so erhaltenen Daten mittels der Business Logic aufzubereiten. Als weitere Funktion soll das Bean nun die zuvor bearbeiteten Daten als spezifische Objekte an die nächst höhere Schicht, dem Frontend, zur Verfügung zu stellen. Der Zugriff auf die so verfügbar gemachten Objekte, soll von dem Frontend aus, mittels JSF Seiten erfolgen. Zudem soll es möglich sein mittels der Business-Logic Beans Daten aus dem Frontend aufzubereiten und sie zur Speicherung an das jeweilige Facade Bean weiterzureichen.

- Web Services

Die in der WebEst Anwendung zu entwickelnden Web Services dienen als Schnittstelle für den Zugriff von externen Anwendungen auf WebEst interne Daten. Der Zugriff auf diese Web Services erfolgt durch eine URI (Uniform Ressource Identifier). Diese URI dient zur eindeutigen Identifikation des spezifischen WebServices. Die Kommunikation mit der externen Applikation erfolgt über standardisierte XML-Aufrufe, das SOAP (Simple Object Access Protocol).

Folgende Web Services sollen zur Verfügung gestellt werden:

- Abrufen aller Estimates
Beinhaltet alle in der Datenbank vorhandenen Estimates mit allen Solutions und Abhängigkeiten
- Generieren eines Reports
Beinhaltet das Generieren eines Reports für eine Solution anhand ihrer ID

Aus Sicherheitsgründen ist für die Verwendung der Web Services, ebenso wie für die Nutzung der WebEst Anwendung über die Weboberfläche, ist hier eine Authentifizierung erforderlich. Hierzu ist die HTTP Basic Authentifizierung zu nutzen. Der Authentifizierungs- und Autorisierungsprozess wird von WASCE verwaltet (siehe unten).

- Authentifizierung und Autorisierung

Um ausschließlich berechtigten Nutzern Zugriff auf die Anwendung zu gewähren, muss ein Sicherheitskonzept implementiert werden. Aus den Anforderungen ergibt sich eine rollenbasierte Berechtigungsstruktur. Benutzerdaten (Benutzername und gehashtes Passwort) werden mit ihrer Rolle in der Datenbank (Tabellen USER und ROLE) abgelegt. Die Sicherstellung der Authentifizierung gegen die Benutzerdaten aus der Datenbank und der Autorisierung auf Web- und EJB-Basis wird von WASCE übernommen, der eine Implementierung des Java Authentication and Authorization

Service (JAAS) enthält. Über Konfigurationsdateien (für Webseiten), Annotationen auf EJB-Ebene oder direkt im Code können die Restriktion für einzelne Rollen angegeben werden. Es muss ein applikationsweiter Security Realm definiert werden, anhand dessen der Application Server prüft, ob der aktuell angemeldete Benutzer die Berechtigungen zum Zugriff auf die einzelnen Objekte (Webseiten oder EJBs) hat. Ist dies nicht der Fall wird der Zugriff verweigert. Ist ein Benutzer nicht angemeldet, soll ihm automatisch eine Login-Seite präsentiert werden. Weitere Informationen zum Sicherheitskonzept in WASCE.⁸

- Konstanten im Quellcode

Konstante Werte wie Pfade zu externen Dateien oder Standardwerte beim Erstellen von Entitäten werden nach Möglichkeit nicht direkt im Quellcode definiert, sondern in standardisierte Java-Properties-Dateien ausgelagert. Damit wird gewährleistet, dass man diese Werte ohne Neukompilierung der Anwendung ändern kann. Ausgenommen sind hiervon bewusst Fehlermeldungen (Exception-Messages), da in der Anwendung keine Lokalisierung stattfindet und diese Meldungen in der Benutzeroberfläche durch passende Texte ersetzt werden können. Auch Beschriftungstexte in der Benutzeroberfläche sind nur in englischer Sprache zu halten und somit direkt in den HTML-Code zu integrieren. Desweiteren werden diese Dateien nicht kompiliert und bleiben somit bearbeitbar.

6.2.3 Frontend: Java Server Faces

Die dritte Schicht repräsentiert die Frontendschicht beziehungsweise Graphical User Interface. Das Graphical User Interface dient als Schnittstelle für den Endanwender, um Eingaben zu tätigen und umfassend mit der Benutzeroberfläche zu interagieren.

Die Frontendschicht im Allgemeinen repräsentiert die grafische Darstellung der WebEst¹ Applikation im Web Browser. Dies wird durch die Anwendung von verschiedenen Technologien ermöglicht. Die Seiten werden mit JavaServer Faces (JSF) erstellt, während die Styles (Layout, Grafiken, Schriftgrößen etc.) in zentralen CSS Dateien definiert werden. Für die Generierung der einzelnen Seiten ist das JSF - Java Server Faces Framework (siehe Abb. 18) zu verwenden, welches Bestand der JSP Technologie ist. Die einzelnen GUI Elemente (Button, DropDown etc.), die in den JSF Seiten definiert werden, greifen auf sogenannte Managed Beans zu. Dies sind Java Klassen, die die Logik und Funktionen der Elemente implementieren.

⁸ <http://publib.boulder.ibm.com/wasce/V2.1.0/en/security-realms.html>

¹ Web Estimation Software

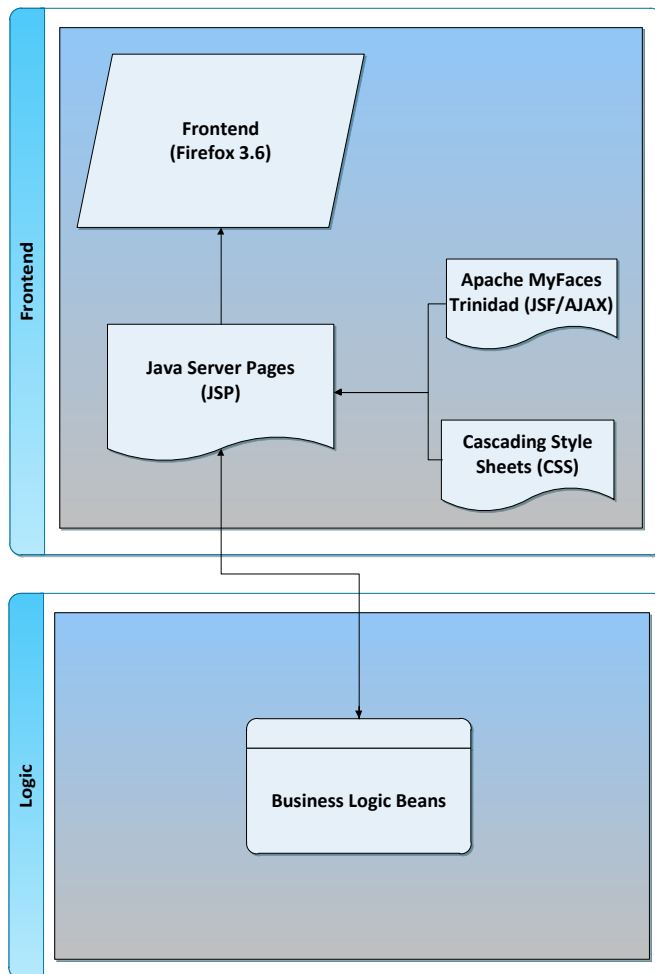


Abbildung 19: „Frontend – Java Server Faces“

Das JSF Framework wird dem MVC (Model – View – Controller) Konzept gerecht und ist somit für die zu entwickelnde Webanwendung geeignet („Best Practice“). Zudem garantiert JSF eine einfache Erstellung von Komponenten und Benutzerschnittstellen in Webseiten (Drag & Drop Prinzip).

Das hier speziell einzusetzende JSF Framework ist Apache MyFaces Trinidad. Apache MyFaces Trinidad ist eine Open Source JSF Implementierung, welche zusätzliche Komponenten und Features bietet. Das wichtigste Feature ist die Einbindung von AJAX, so dass „partial page rendering“⁹ und der Einsatz dynamischer Komponenten ermöglicht wird. Ein weiteres Feature ist z.B. vereinfachtes Skinning der Oberfläche.

Das Styling ist mittels zentraler Cascading Style Sheets (CSS) zu realisieren. Cascading Style Sheets ermöglichen es, Änderungen an Layout, Schriften, Grafiken, Hintergründen etc. schnell und problemlos an einer zentralen Stelle durchzuführen, ohne die einzelnen .xhtml Seiten zu editieren. Die Verwendung von JSF, CSS sowie AJAX garantiert, dass die im Pflichtenheft definierten Vorgaben von IBM eingehalten und erfolgreich umgesetzt werden können.

⁹ <http://myfaces.apache.org/trinidad/devguide/ppr.html>

7 Testen und Validieren

Im Folgenden wird das verwendete Testkonzept zur Evaluierung der WebEst¹ Anwendung näher beleuchtet.

Im ersten Schritt findet eine Validierung anhand von White Box Tests statt, d.h. die Testfälle werden aus dem Quellcode abgeleitet. Im nächsten Schritt findet eine Evaluierung durch Black Box Tests statt. Die dort verwendeten Testfälle ergeben sich aus den jeweils vorher definierten funktionalen Anforderungen der WebEst¹ Anwendung (siehe Pflichtenheft).

Abschließend erfolgt eine Validierung des Gesamtprojektes. Dieser Abschlusstest dient als qualitätssichernde Maßnahme für die Datenmodelle sowie des Quellcodes und stellt somit eine endgültige Validierung der Software Applikation sicher.

100 % (Grad der Fertigstellung)

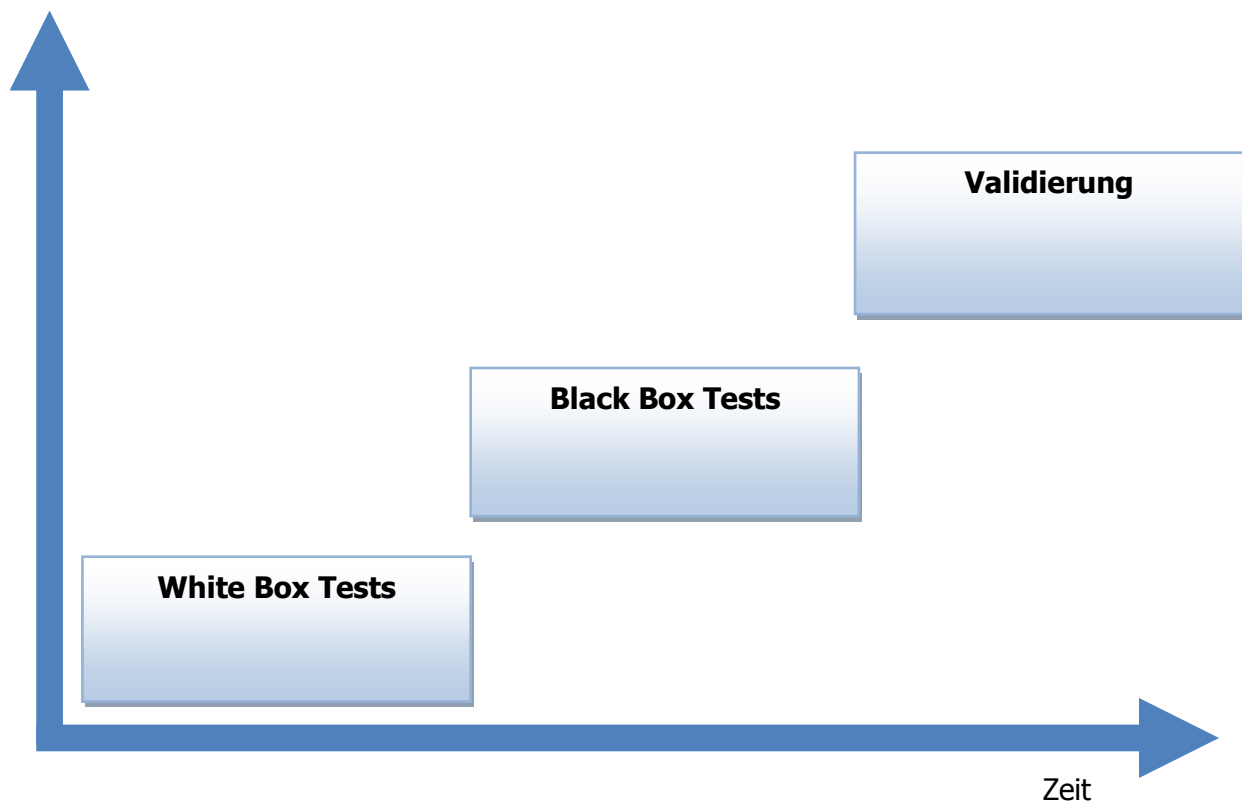


Abbildung 20: „Testen und Validieren“

¹ Web Estimation Software

7.1 White Box Tests

Unter White Box Tests sind Testfälle zu verstehen, die den Quellcode testen sollen. Dies beinhaltet das Testen der Funktionalität von Methoden und Klassen, sowie das Auslösen von individuellen Exceptions.

Folgende Guidelines sind im Folgenden zu beachten:

- Zu allen public-Methoden sind einzelne Testfälle zu definieren
- Alle private/protected-Methoden werden implizit durch Aufruf der public-Methoden mitgetestet.
- Es werden neben den Basiswerten auch Randwerte (0, null etc.) getestet.
- Das Abfangen und Behandeln von Exceptions ist ebenfalls Bestandteil der Tests.

Folgende Testpakete sind in Java EE zu erstellen:

- webest-persistence-test
Das Projekt "webest-persistence-test" dient zum Testen der Model Schicht.
- webest-processing-test
Das Projekt "webest-processing-test" dient zum Testen der Logic Schicht.
- webest-webservice-test
Das Projekt "webest-processing-test" dient zum Testen der WebServices.

Als Grundlage zum Erstellen von Testfällen dient JUnit. Zum Einbinden der Enterprise Java Beans in die Unit Testklassen dienen die jeweiligen Remote Interfaces.

7.2 Black Box Tests

Als Black Box Tests sind jene Testfälle zu verstehen, welche aus den funktionalen Anforderungen der zu entwickelnden Applikation generiert werden. Für die Black Box Tests dienen die im Pflichtenheft erstellten Use Cases als Grundlage.

- Login
Authentifizierung mit dem Endsystem
- New Estimate
Erstellung eines neuen Estimates
- Modify Estimate
Editierung eines bestehenden Estimates
- Clone Estimate
Duplizieren eines bestehenden Estimates
- Delete Estimate
Löschen eines bestehenden Estimates
- Show Estimate
Anzeige eines bestehenden Estimates
- New Solution
Erstellung einer neuen Solution
- Modify Solution
Bearbeiten einer bestehenden Solution
- Clone Solution
Duplizieren einer bestehenden Solution
- Delete Solution
Löschen einer bestehenden Solution
- Show Solution
Anzeige einer bestehenden Solution
- Show Report
Erstellung des Reports aus einer bestehenden Solution

Die detaillierte Dokumentation der Black Box Tests ist unter Punkt 7.29.2 „Anhang – Testprotokolle“ zu finden.

7.3 Validierung

Die Validierung beinhaltet abschließende Schritte, um die Korrektheit, Robustheit und Zuverlässigkeit der zu entwickelnden Softwareapplikation zu gewährleisten. Nachfolgend sind folgende Qualitätsanforderungen zu überprüfen, welche als maßgebend anzusehen sind:

- Robustheit
- Zuverlässigkeit
- Korrektheit

Zur Evaluation dieser drei maßgebenden Qualitätsanforderungen werden zwei Testszenarien verwendet:

- Lastentest
Der Lastentest beinhaltet das Erstellen von mehreren Estimates und Solutions. Zu diesem Zweck werden in der zu entwickelnden Anwendung zehn Estimates mit jeweils drei Solutions angelegt. Aus diesen werden darauf folgend die „on-the-fly“ Reporte generiert.
- Multi-User-Test
Der Multi User Test beinhaltet die zeitgleiche Nutzung der zu entwickelnden WebEst Plattform. Zu diesem Zweck werden sieben Clients gleichzeitig auf die Anwendung geschaltet. Danach sind folgende Schritte auszuführen:
 - Authentifizierung (user/password)
 - Erstellung Estimate (estimate_client-x*)
 - Erstellung Solution (solution_client -x*)
 - Erstellung Report (report_client-x*)

*x-client bezeichnet die Nummer des jeweiligen Clients

8 Quellen

Der aktuelle Stand der Verlinkungen von Webseiten bezieht sich auf den 27.02.2011.

- [1] Web Estimation Software – zu entwickelnde Applikation
- [2] <http://www-01.ibm.com/software/webervers/appserv/community/syseq/>
- [3] http://www.wikipedia.de/Model_View_Controller
- [4] <http://www.db.apache.org/derby/manuals/index.html>
- [5] <http://www-01.ibm.com/software/webervers/appserv/community>
- [6] IBM Redbook: Experience JavaEE! Using WebSphere App. Server Community Edition
- [7] <http://openjpa.apache.org/builds/1.2.2/apache-openjpa-1.2.2/docs/manual/index.html>
- [8] <http://publib.boulder.ibm.com/wasce/V2.1.0/en/security-realms.html>
- [9] <http://myfaces.apache.org/trinidad/devguide/ppr.html>

9 Appendix

9.1 Testprotokolle – White Box Tests

Die entsprechenden JUnit Testklassen sind im Source Code zu finden. Eine Aufzählung der einzelnen Testklassen wird aus Platzgründen nicht vorgenommen. Für die genaue Spezifikation der Tests siehe Punkt 7.1 „Testen und Validierung – White Box Tests“.

9.2 Testprotokolle – Black Box Tests

Die nun gezeigten Testprotokolle beinhalten die Umsetzungen der definierten Use Cases aus dem WebEst Pflichtenheft. Die Generierung der Testfälle bezieht sich auf die drei verwendbaren Rollen (admin, estimator und projectmanager). Aus Redundanzgründen wird hier nur die Testsuite des Administrators vorgestellt. Die beiden andern Testsuites (estimator und projectmanager) sind bis auf kleine Änderungen, wie beispielsweise die Logindaten, äquivalent.

Die nachfolgenden Tests sind aufeinander aufbauend. Dies bedeutet, dass für den Testfall „NewEstimate“ der Testfall „Login“ bereits korrekt evaluiert wurde. Zum Testen Verwenden Sie bitte nicht den einzelnen Testfall, sondern die ganze Testsuite.

Testsuite:

- admin_TESTSUITE

Testcases:

- Login
- NewEstimate
- ModifyEstimate
- CloneEstimate
- DeleteEstimate
- ShowEstimate
- NewSolution
- ModifySolution
- CloneSolution
- DeleteSolution
- ShowSolution
- ShowReport

Login		
Schritt	ToDo	Erwartetes Ergebnis
1	WebEst Anwendung öffnen	WebEst Anwendung geöffnet.
2	{User = admin}	User auf admin gesetzt
3	{Password = 1234}	Password auf 1234 gesetzt
4	Button {Login} tätigen	User {admin, 1234} wurde erfolgreich eingeloggt. Screen {Home} erscheint. //Testfall beendet

Tabelle 33: Black Box Test – Login

New Estimate		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Button New Estimate	New Estimate - Erstellvorgang gestartet
3	Template {No Template} wählen	Screen {New Est. – Project Description} erscheint
4	Reiter {Project Description} Eingaben tätigen: {Project Name = TestEstimate} {Organisation = TestOrganisation} {Division = TestDivision} {Preparer Name = Administrator} {Country = Germany} {Application Type = Value} {Industry Sector = Value} {Monetary Unit = EUR} {Effort Unit = MDAY} {MTTD Time Unit = Days }	Eingaben getätigt
5	Reiter {Reliability} wählen	Screen {New Est. - Reliability} erscheint
6	Reiter {Reliability} Eingaben tätigen: {Days per week = 7} {Hours per day = 5}	Eingaben getätigt
7	Reiter {Phases} wählen	Screen {New Est. – Phases} erscheint
8	Reiter {Phases} Eingaben tätigen: {Phase1 = 45 %} {Phase2 = 25 %} {Phase3 = 15 %} {Phase4 = 15 %}	Eingaben getätigt
9	Reiter {Milestones} wählen	Screen {New Est. – Milestones} erscheint
10	Reiter {Milestones} Button {New Milestone} wählen {Included = True}	Eingaben getätigt

	{ID = 1} { Acronym = MS1} {Name = MilestoneOne} { % = 100 %} {of Phase = Phase1} {Description = TestDescription	
11	Button {Save} wählen	Estimate gespeichert. //Testfall beendet

Tabelle 34: Black Box Test - New Estimate

Modify Estimate		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Estimate {TestEstimate} auswählen	Estimate {TestEstimate} geöffnet
3	Reiter {Phase1} auswählen	Reiter {TestEst. - Milestones} geöffnet
4	Folgende Änderungen vornehmen {Phase1 = Phase_Eins, 25 %} {Phase2 = Phase_Zwei, 25 %} {Phase3 = Phase_Drei, 25 %} {Phase4 = Phase_Vier, 25 %}	Eingaben getätigt
5	Button {Save} tätigen	Estimate erfolgreich geändert und gespeichert //Testfall beendet

Tabelle 35: Black Box Test - Modify Estimate

Clone Estimate		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Estimate {TestEstimate} auswählen	Estimate {TestEstimate} geöffnet
3	Button {Clone Estimate} auswählen	Screen {TestEst. - Clone Estimate} erscheint
4	Reiter {Project Description} auswählen	Screen {TestEst. - Project Description erscheint
5	Folgende Änderungen vornehmen {Project Name = TestEstimate_Clone} {Monetary Unit = DOLLAR}	Eingaben getätigt
6	Button {Save} tätigen	Estimate erfolgreich geklont und unter neuem Namen gespeichert //Testfall beendet

Tabelle 36: Black Box Test - Clone Estimate

Delete Estimate		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Estimate {TestEstimate_Clone} auswählen	Estimate {TestEstimate_Clone} geöffnet
3	Button {Delete Estimate} auswählen	Screen {TestEst. - Delete Estimate} erscheint
4	Mit Button {OK} das Löschen von Estimate {TestEstimate_Clone} bestätigen.	Estimate {TestEstimate_Clone} gelöscht. Screen { Home} erscheint. //Testfall beendet

Tabelle 37: Black Box Test – Delete Estimate

Show Estimate		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Estimate {TestEstimate} auswählen	Estimate {TestEstimate} geöffnet //Testfall beendet

Tabelle 38: Black Box Test - Show Estimate

New Solution		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Estimate {TestEstimate} auswählen	Estimate {TestEstimate} geöffnet
3	Button New Solution	New Solution {TestEstimate} - Erstellvorgang gestartet
4	Reiter {Solution Assumptions} wählen	Reiter {Solution Assumptions} geöffnet
5	Reiter {Solution Assumptions} Eingaben tätigen: {Solution = Solution_TestEstimate} {GearingFactor = 100} {PI = 12}	Eingaben getätigt
6	Reiter {UseCase Granularity} wählen	Reiter {UseCase Granularity} geöffnet
7	Reiter { UseCase Granularity } Eingaben tätigen: {Granularity = Medium}	Eingaben getätigt
8	Reiter {Sizing} wählen	Reiter {Sizing} geöffnet

9	Button New Use Case wählen und folgende Eingaben tätigen {Name = UseCase1} {Complexity = Medium} {Certainty = Medium} Button New Use Case wählen {Name = UseCase2} {Complexity = High} {Certainty = High} Button New Use Case wählen {Name = UseCase3} {Complexity = Low} {Certainty = Low}	Eingaben getätigt
11	Button {Save} wählen	Solution gespeichert. //Testfall beendet

Tabelle 39: Black Box Test - New Solution

Modify Solution		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Estimate {TestEstimate} auswählen	Estimate { TestEstimate } geöffnet
3	Solution {Solution_TestEstimate} auswählen	Solution {Solution_TestEstimate} geöffnet
4	Reiter {Solution Assumptions} wählen	Reiter {Solution Assumptions} geöffnet
5	Reiter {Solution Assumptions} Eingaben tätigen: {GearingFactor = 125} {PI = 12}	Eingaben getätigt
5	Button {Save} tätigen	Solution erfolgreich geändert und gespeichert //Testfall beendet

Tabelle 40: Black Box Test - Modify Solution

Clone Solution		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Estimate {TestEstimate} auswählen	Estimate {TestEstimate} geöffnet
3	Solution {Solution_TestEstimate} auswählen	Solution {Solution_TestEstimate} geöffnet
3	Button {Clone Solution} auswählen	Screen {TestEst. - Clone Solution} erscheint.
4	Reiter {Solutions Assumptions} auswählen	Reiter {Solution Assumptions} geöffnet
5	Reiter {Solution Assumptions} Eingaben tätigen: {Solution = Solution_TestEstimateClone}	Eingaben getätigt

6	Button {Save} tätigen	Solution erfolgreich geklont und unter neuem Namen gespeichert //Testfall beendet
---	-----------------------	--

Tabelle 41: Black Box Test - Clone Solution

Delete Solution		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Estimate {TestEstimate} auswählen	Estimate {TestEstimate} geöffnet
3	Solution {Solution_TestEstimate} auswählen	Solution {Solution_TestEstimate} geöffnet
3	Button {Delete Solution} auswählen	Screen {Delete Solution} erscheint
4	Mit Button {OK} das Löschen von Estimate {Solution_TestEstimateClone} bestätigen.	Solution {Solution_TestEstimateClone} gelöscht. Screen {Home} erscheint. //Testfall beendet

Tabelle 42: Black Box Test - Delete Solution

Show Solution		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Estimate {TestEstimate} auswählen	Estimate {TestEstimate} geöffnet
3	Solution {Solution_TestEstimate} auswählen	Solution {Solution_TestEstimate} geöffnet. Solution wird angezeigt //Testfall beendet

Tabelle 43: Black Box Test - Show Solution

Show Report		
Schritt	ToDo	Erwartetes Ergebnis
1	Login – {user: admin, pw: 1234}	User {admin} wurde eingeloggt. Screen {Home} erscheint
2	Estimate {TestEstimate} auswählen	Estimate {TestEstimate} geöffnet
3	Solution {Solution_TestEstimate} auswählen	Solution {Solution_TestEstimate} geöffnet. Solution wird angezeigt
4	Reiter {Sizing} auswählen	Reiter {Sizing} geöffnet
5	Button {CalculateUCP} wählen	Sizing Werte werden „on-the-fly“ berechnet Erwartetes Ergebnis {Solution_TestEstimate}: {UseCase1: UCP 11}

		{UseCase2: UCP 20} {UseCase3: UCP 5}
6	Button {Generate Report} wählen	Report wird generiert: {UCP: 36} {Gearing Factor: 125} {PI: 12} {SLOC 4500} //Testfall beendet

Tabelle 44: Black Box Test - Show Report

9.3 Testprotokolle – Validierung

Um wie in Punkt 7.3 beschrieben die Qualitätsanforderungen der Korrektheit, Robustheit und Zuverlässigkeit zu gewährleisten, sind im Folgenden zwei Testfälle näher eruiert worden.

- Lastentest
- Multiusertest

Lastentest		
Schritt	ToDo	Erwartes Ergebnis
Login	Username, Password eingeben	User ist eingeloggt
NewEstimate	Estimate_EINS erstellen	Estimate_EINS erstellt
NewSolution	Solution_EINS (Estimate_Eins) erstellen	Solution_EINS erstellt
NewSolution	Solution_ZWEI (Estimate_Eins) erstellen	Solution_ZWEI erstellt
NewSolution	Solution_DREI (Estimate_Eins) erstellen	Solution_DREI erstellt
NewEstimate	Estimate_ZWEI erstellen	Estimate_ZWEI erstellt
NewSolution	Solution_EINS (Estimate_ZWEI) erstellen	Solution_EINS erstellt
NewSolution	Solution_ZWEI (Estimate_ZWEI) erstellen	Solution_ZWEI erstellt
NewSolution	Solution_DREI (Estimate_ZWEI) erstellen	Solution_DREI erstellt
NewEstimate	Estimate_DREI erstellen	Estimate_DREI erstellt
NewSolution	Solution_EINS (Estimate_DREI) erstellen	Solution_EINS erstellt
NewSolution	Solution_ZWEI (Estimate_DREI) erstellen	Solution_ZWEI erstellt
NewSolution	Solution_DREI (Estimate_DREI) erstellen	Solution_DREI erstellt
NewEstimate	Estimate_VIER erstellen	Estimate_VIER erstellt
NewSolution	Solution_EINS (Estimate_VIER) erstellen	Solution_EINS erstellt
NewSolution	Solution_ZWEI (Estimate_VIER) erstellen	Solution_ZWEI erstellt
NewSolution	Solution_DREI (Estimate_VIER) erstellen	Solution_DREI erstellt
NewEstimate	Estimate_FUENF erstellen	Estimate_FUENF erstellt
NewSolution	Solution_EINS (Estimate_FUENF) erstellen	Solution_EINS erstellt
NewSolution	Solution_ZWEI (Estimate_FUENF) erstellen	Solution_ZWEI erstellt
NewSolution	Solution_DREI (Estimate_FUENF) erstellen	Solution_DREI erstellt
NewEstimate	Estimate_SECHS erstellen	Estimate_SECHS erstellt
NewSolution	Solution_EINS (Estimate_SECHS) erstellen	Solution_EINS erstellt
NewSolution	Solution_ZWEI (Estimate_SECHS) erstellen	Solution_ZWEI erstellt
NewSolution	Solution_DREI (Estimate_SECHS) erstellen	Solution_DREI erstellt
NewEstimate	Estimate_SIEBEN erstellen	Estimate_SIEBEN erstellt
NewSolution	Solution_EINS (Estimate_SIEBEN) erstellen	Solution_EINS erstellt
NewSolution	Solution_ZWEI (Estimate_SIEBEN) erstellen	Solution_ZWEI erstellt
NewSolution	Solution_DREI (Estimate_SIEBEN) erstellen	Solution_DREI erstellt
NewEstimate	Estimate_ACHT erstellen	Estimate_ACHT erstellt
NewSolution	Solution_EINS (Estimate_ACHT) erstellen	Solution_EINS erstellt
NewSolution	Solution_ZWEI (Estimate_ACHT) erstellen	Solution_ZWEI erstellt
NewSolution	Solution_DREI (Estimate_ACHT) erstellen	Solution_DREI erstellt
NewEstimate	Estimate_NEUN erstellen	Estimate_NEUN erstellt

NewSolution	Solution_EINS (Estimate_NEUN) erstellen	Solution_EINS erstellt
NewSolution	Solution_ZWEI (Estimate_NEUN) erstellen	Solution_ZWEI erstellt
NewSolution	Solution_DREI (Estimate_NEUN) erstellen	Solution_DREI erstellt
NewEstimate	Estimate_ZEHN erstellen	Estimate_ZEHN erstellt
NewSolution	Solution_EINS (Estimate_ZEHN) erstellen	Solution_EINS erstellt
NewSolution	Solution_ZWEI (Estimate_ZEHN) erstellen	Solution_ZWEI erstellt
NewSolution	Solution_DREI (Estimate_ZEHN) erstellen	Solution_DREI erstellt
		//TESTFALL BEENDET

Tabelle 45: Validierung - Lastentest

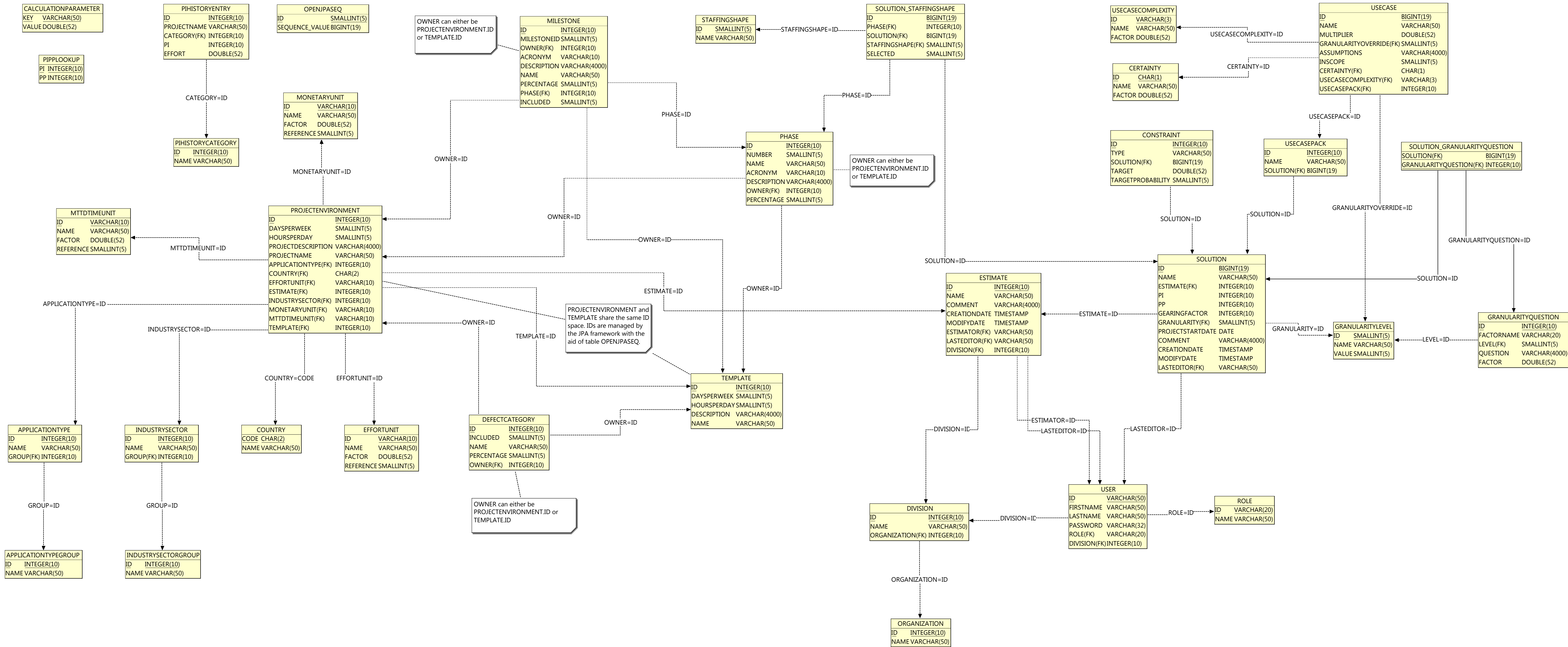
Multi User Test		
Schritt	ToDo	Erwartetes Ergebnis
Login	Username, Password eingeben	Client-1 loggt sich ein
Login	Username, Password eingeben	Client-2 loggt sich ein
Login	Username, Password eingeben	Client-3 loggt sich ein
Login	Username, Password eingeben	Client-4 loggt sich ein
Login	Username, Password eingeben	Client-5 loggt sich ein
Login	Username, Password eingeben	Client-6 loggt sich ein
Login	Username, Password eingeben	Client-7 loggt sich ein
NewEstimate	Estimate_Client-1 erstellen	Estimate_Client-1 erstellt
NewEstimate	Estimate_Client-2 erstellen	Estimate_Client-2 erstellt
NewEstimate	Estimate_Client-3 erstellen	Estimate_Client-3 erstellt
NewEstimate	Estimate_Client-4 erstellen	Estimate_Client-4 erstellt
NewEstimate	Estimate_Client-5 erstellen	Estimate_Client-5 erstellt
NewEstimate	Estimate_Client-6 erstellen	Estimate_Client-6 erstellt
NewEstimate	Estimate_Client-7 erstellen	Estimate_Client-7 erstellt
NewSolution	Solution_Client-1 erstellen	Solution_Client-1 erstellt
NewSolution	Solution_Client-2 erstellen	Solution_Client-2 erstellt
NewSolution	Solution_Client-3 erstellen	Solution_Client-3 erstellt
NewSolution	Solution_Client-4 erstellen	Solution_Client-4 erstellt
NewSolution	Solution_Client-5 erstellen	Solution_Client-5 erstellt
NewSolution	Solution_Client-6 erstellen	Solution_Client-6 erstellt
NewSolution	Solution_Client-7 erstellen	Solution_Client-7 erstellt
NewReport	Report_Client-1 erstellen	Report_Client-1 erstellt
NewReport	Report_Client-2 erstellen	Report_Client-2 erstellt
NewReport	Report_Client-3 erstellen	Report_Client-3 erstellt
NewReport	Report_Client-4 erstellen	Report_Client-4 erstellt
NewReport	Report_Client-5 erstellen	Report_Client-5 erstellt
NewReport	Report_Client-6 erstellen	Report_Client-6 erstellt
NewReport	Report_Client-7 erstellen	Report_Client-7 erstellt
		//TESTFALL BEENDET

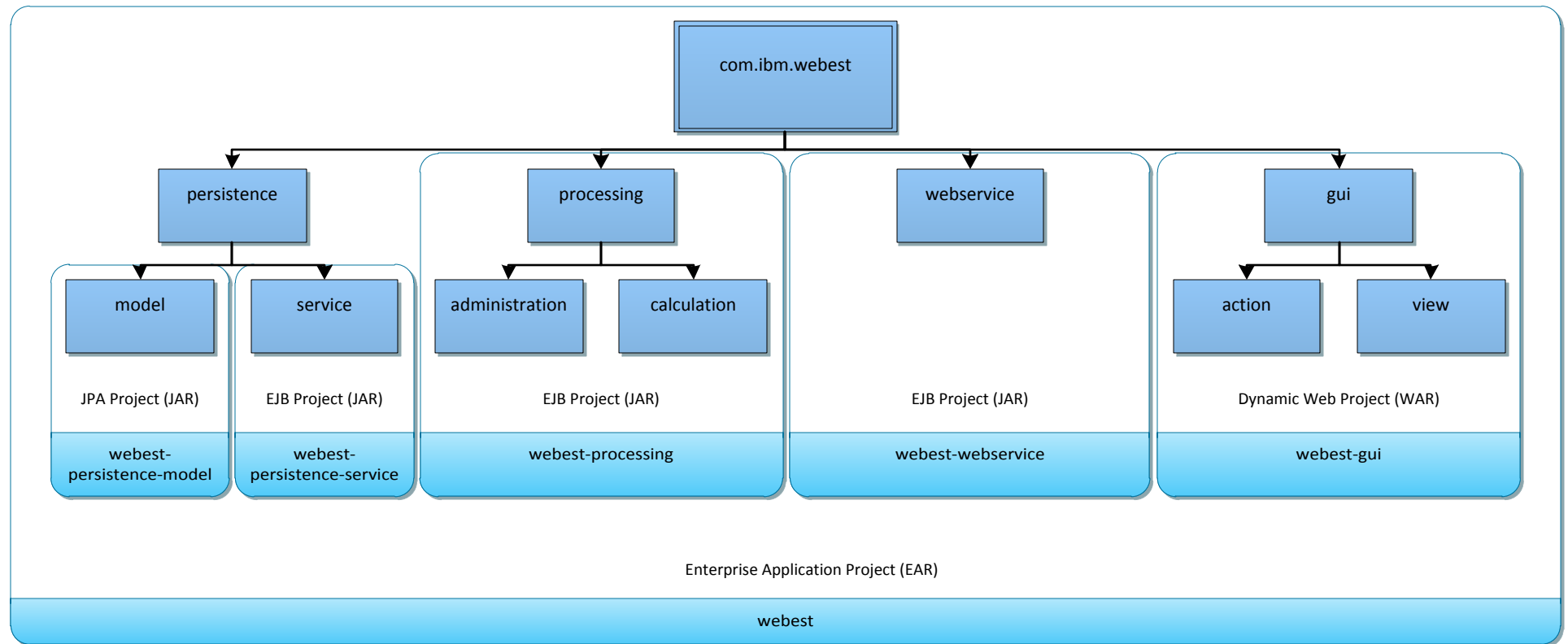
Tabelle 46: Validierung - Multi User Test

9.4 Ausführliche Diagramme

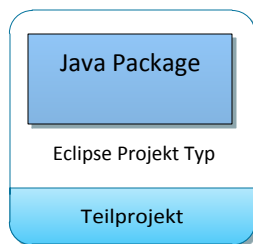
Auf den folgenden Seiten finden Sie die im DV-Konzept vorgestellten vollständigen und ausführlichen Diagramme:

- Entity Relationship Diagramm
- Java Paketstruktur
- UML Klassendiagramm
Das vollständige UML Klassendiagramm ist aus Gründen der Komplexität (Größe DIN A0) lediglich in elektronischer Form verfügbar.





Legende:



Projektbeschreibungen in QuickInfos

