



Projekt: wt4u

Software Architektur Dokument

Andreoli Dario (dandreol@hsr.ch)
Schiepek Richard (rschiepe@hsr.ch)
Zahner Tobias (tzahner@hsr.ch)

Änderungsgeschichte

Datum	Version	Änderung	Autor
21.03.14	0.1	Inititaldokument	dandreol
27.03.14	0.2	Erweiterungen	dandreol
28.03.14	0.3	Datenspeicherung, Grössen und Leistungen hinzugefügt	dandreol
01.04.14	0.4	Package Diagramm + Beschreibungen	rschiepe
25.04.14	0.5	Architekturdiagramm & Klassenbeschreibungen	rschiepe
27.04.14	0.6	Controller Methoden	rschiepe
28.04.14	0.7	View, Ordnerstruktur, Klassenmodel (Datenspeicherung)	rschiepe
29.04.14	0.8	Beschreibung BusinessLogic	rschiepe
29.04.14	0.9	Ergänzungen mit Project, Screenshots	dandreol
30.04.14	1.0	SSD CheckInOut eingefügt	dandreol
30.04.14	1.1	Modelldiagramme anpassen (mit Roles)	rschiepe
30.04.14	1.2	Methoden erweitert	tzahner
30.04.14	1.3	Screenshots aktualisiert	dandreol
30.04.14	1.4	Controllers Klassenstruktur aktualisiert	tzahner
30.04.14	1.5	Status View/Controller neu eingetragen	rschiepe
13.05.14	1.6	Neue Controllermethoden ergänzt	dandreol
15.05.14	1.7	Überarbeitung gemäss Feedback aus Review	dandreol
19.05.14	1.8	Sreenshots der Applikation aktualisiert	dandreol

Inhalt

Änderungsgeschichte	2
Inhalt.....	3
1. Einführung	5
1.1 Zweck	5
1.2 Gültigkeitsbereich	5
1.3 Referenzen	5
1.4 Übersicht.....	5
2. Systemübersicht	6
2.1 Client	6
2.2 Server	6
2.2.1 Webserver	6
2.2.2 Datenbank	6
3. Architektonische Ziele & Einschränkungen	7
3.1 Ziele.....	7
3.1.1 Erweiterbarkeit.....	7
3.1.2 Skalierbarkeit.....	7
3.1.3 Privacy und Security	7
3.2 Einschränkungen.....	7
3.2.1 Internetverbindung	7
3.2.2 JavaScript.....	7
3.3 Eingesetzt Technologien	7
3.3.1 ASP.NET	7
3.3.2 MS SQL.....	8
3.3.3 HTML/CSS	8
3.3.4 JavaScript/JQuery	8
3.4 Externes Design.....	9
3.4.1 Mockups	9
3.4.2 Screenshots	9
4. Ordnerstruktur .NET Solution wt4u	11
4.1 Übersicht.....	11
4.2 Packages.....	11
4.3 Project wt4u.....	12
4.4 Project wt4u.Testing.....	13
5. Logische Architektur	14
5.1 Models	15
5.1.1 Klassenstruktur.....	15

5.1.2 Klassen	15
5.1.3 Schnittstellen	15
5.1.4 Abhängigkeiten auf andere Packages.....	15
5.1.5 Wichtige interne Abläufe	15
5.2 Controllers	16
5.2.1 Klassenstruktur	16
5.2.2 Schnittstellen	16
5.2.3 Abhängigkeiten auf andere Packages.....	16
5.2.4 Wichtige interne Abläufe	17
5.2.5 Controller Methoden.....	18
5.3 Views	25
5.3.1 Klassenstruktur	25
5.3.2 View-Files (.cshtml)	25
5.3.3 Schnittstellen	26
5.3.4 Abhängigkeiten auf andere Packages.....	26
5.3.5 Wichtigste interne Abläufe.....	26
5.4 Business Logic	27
5.4.1 Klassenstruktur	27
5.4.2 Klassen	27
5.4.3 Schnittstellen	29
5.4.4 Abhängigkeiten auf andere Packages.....	29
5.4.5 Wichtige interne Abläufe	30
6. Use Cases / Prozesse	31
6.1 Check In/Out	31
6.2 Edit Project.....	32
7. Deployment	33
7.1 Ist	33
7.2 Soll.....	33
8. Datenspeicherung	34
8.1 Klassenmodell	34
8.2 Schnittstelle	34
9. Größen und Leistung	35
9.1 Webserver.....	35
9.2 Datenbank.....	35

1. Einführung

1.1 Zweck

In diesem Dokument wird die gesamte Softwarearchitektur von wt4u beschrieben.

1.2 Gültigkeitsbereich

Dieses Dokument behält für die gesamte Dauer des Projektes wt4u Gültigkeit. Es basiert auf der Analyse und den Requirements, welche festgelegt wurden.

1.3 Referenzen

Anforderungsspezifikation.

Domainanalyse.

Sitzungsprotokoll SW09.

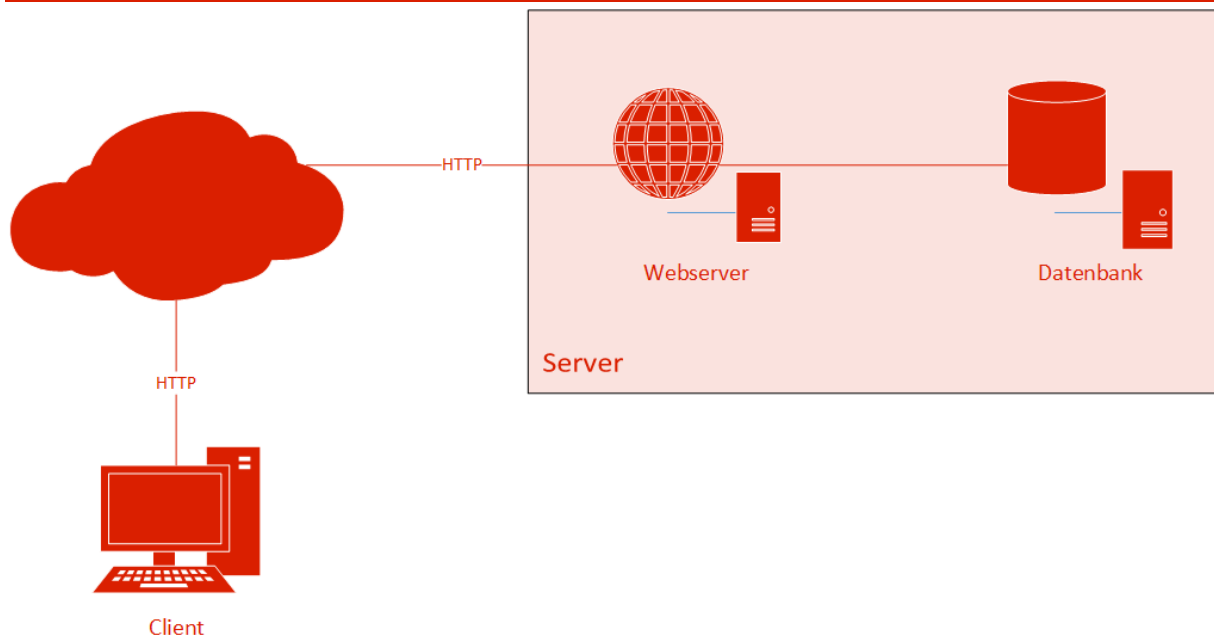
Sitzungsprotokoll SW10.

Sitzungsprotokoll SW11.

1.4 Übersicht

Nachfolgend werden die softwarespezifischen Feinheiten und die Architektur von wt4u erläutert. Dieses Dokument enthält die Struktur über alle verschiedenen Tiers.

2. Systemübersicht



2.1 Client

Wie der Benutzer als Client auf wt4u zugreift, ist ihm selber überlassen. Dies kann per Notebook, sowie auch einem Desktop Computer erfolgen. Auch über mobile Geräte funktioniert der Zugriff. Einzig eine Internetanbindung, sowie ein aktueller Browser, muss vorhanden sein. Kommunikation zwischen Client und Server erfolgt via HTTP und ist somit zustandslos. Dies erlaubt zudem einfache Skalierung sowie Load Balancing.

2.2 Server

Der Server besteht aus Systemsicht aus zwei Komponenten, einem Webserver, sowie einem Datenbankserver.

2.2.1 Webserver

Der Webserver ist eine zentrale Komponente von wt4u. Er nimmt die HTTP Anfragen entgegen und verarbeitet diese. Zudem ist er für die Kommunikation mit der Datenbank zuständig.

2.2.2 Datenbank

Die Datenbank ist für die Persistenz zuständig. Sie beinhaltet sämtliche Daten zu Mitarbeiter, Arbeitszeiten, Projekten, etc.

3. Architektonische Ziele & Einschränkungen

3.1 Ziele

3.1.1 Erweiterbarkeit

Es soll einfach möglich sein, die Software wenn gewünscht zu erweitern. Dies betrifft vor allem die Funktionalität der Software. Da die Systemarchitektur gegeben ist, ist dies auch ohne Probleme möglich.

3.1.2 Skalierbarkeit

Skalierbarkeit hatte einen grossen Einfluss auf Konzeption der Architektur. Bei grosser Last soll es einfach möglich sein, den Server und Datenbank zu skalieren.

3.1.2.1 Webserver

Durch Load Balancing ist es einfach möglich, die Last von einem einzelnen Webserver zu reduzieren und auf mehrere zu verteilen.

3.1.2.2 Datenbank

Sollte die Datenbank überlastet werden, kann ein Datenbank Clustering eingerichtet werden. Damit kann die Last auf mehrere Datenbank Server verteilt werden. Die Datenkonsistenz ist dadurch allerdings nicht gefährdet.

3.1.3 Privacy und Security

3.1.3.1 Sensible Daten

Bei Privacy und Security wird Wert auf die sensiblen Daten gelegt. Dies betrifft vor allem die Daten der Mitarbeiter. In erster Linie ist natürlich das Passwort betroffen. Dies soll gehashed in der Datenbank abgelegt werden. Die Implementierung des Logins mittels SSL wird als optional angesehen und wird somit nur bei genügend Zeit weiter verfolgt.

3.2 Einschränkungen

3.2.1 Internetverbindung

Eine Internetanbindung ist leider, aber auch logischerweise bei wt4u immer Voraussetzung. Natürlich besteht auch die Möglichkeit wt4u in einem internen LAN zu betreiben. Dafür wird nicht zwingend eine Internetanbindung benötigt, die Verbindung zu Webserver und Datenbankserver muss jedoch gewährleistet sein.

3.2.2 JavaScript

Damit alle Funktionen von wt4u genutzt werden können, muss auf dem verwendeten Browser JavaScript aktiviert sein.

3.3 Eingesetzt Technologien

3.3.1 ASP.NET

Trotz der knappen Erfahrung mit ASP.NET im Team, haben wir uns für diese Technologie entschieden. Der Hauptgrund für diese Entscheidung ist, dass ebenso wenig Erfahrung mit

alternativen Technologien (z.B. Spring) vorhanden ist. Zudem bietet es uns die Möglichkeit unser Wissen von .NET zu erweitern.

3.3.1.1 User Identity

Für die Benutzerverwaltung von wt4u wird auf User Identity von .NET zurückgegriffen. Dies erfüllt bereits die Anforderung für die Applikation und kann mit zusätzlichen Feldern (z.B. Adresse) einfach erweitert werden.

3.3.2 MS SQL

Wir haben uns für die Verwendung von MS SQL als Datenbank entschieden. Der Grund dafür ist die simple Anbindung von einer MS SQL DB an eine ASP.NET Applikation.

3.3.3 HTML/CSS

Für alle Views wird HTML und CSS eingesetzt.


3.3.4 JavaScript/JQuery

Um grafische Auswertungen, sowie Manipulationen am GUI durchzuführen wird JavaScript, sowie JQuery verwendet. Zudem werden zur Vereinfachung gesamte Libraries verwendet. Für die tabellarische Darstellung mit Sortierung und Filterung wird JQuery DataTable eingesetzt. Für die grafische Auswertung der Statistiken wird ChartJS von DevExpress verwendet. Dies ermöglicht eine einfache Erzeugung der Diagramme.

3.4 Externes Design

3.4.1 Mockups

13:33



Home

Start Meine Projekte Logout

13:33 Angemeldet als User

Status: working
Current Project: Example

Set Current Project: ☒

Start

Start Meine Projekte Logout

13:33 Angemeldet als User

Projekt-Name	Aufgew. Zeit
Testprojekt	2h 30
Noch ein Projekt	4h 40
Bla Bla Bla	7h 10

Projekte / Meine Projekte

Start Meine Projekte Logout

13:33 Angemeldet als User

Von Bis

Arbeiter	Aufgew. Zeit
Hans Muster	2h 30
Dario Andreoli	4h 40
Richard Schiepek	7h 10
Tobias Zahner	5h 30

Arbeiter

Start Meine Projekte Logout

13:33 Angemeldet als User

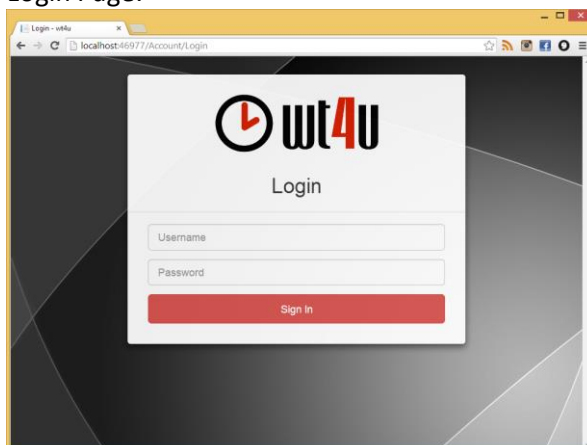
Von Bis

Projekt	Von	Bis	Dauer
Testprojekt	01.04.2014 08:00	01.04.2014 17:00	9h 00
P2	02.04.2014 07:30	02.04.2014 16:00	8h 30
Total			17h 30
Durchschnitt pro Tag			8h 45

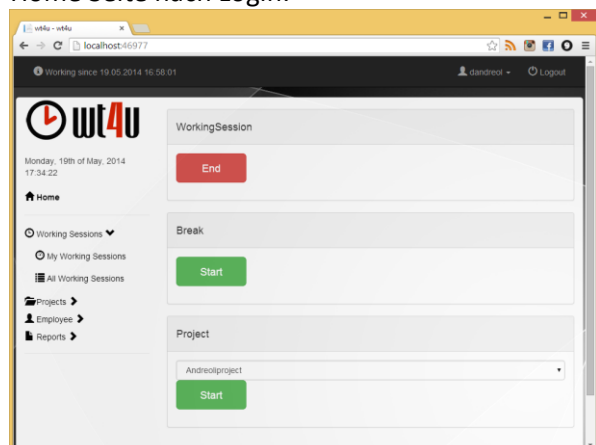
Arbeitszeit

3.4.2 Screenshots

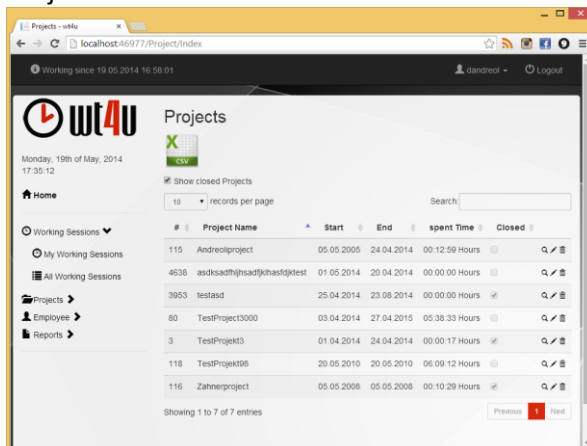
Login Page:



Home Seite nach Login:



Projektübersicht:



Projects

Monday, 19th of May, 2014 17:35:12

Home

Working Sessions

My Working Sessions

All Working Sessions

Projects

Employee

Reports

Show closed Projects

10 records per page

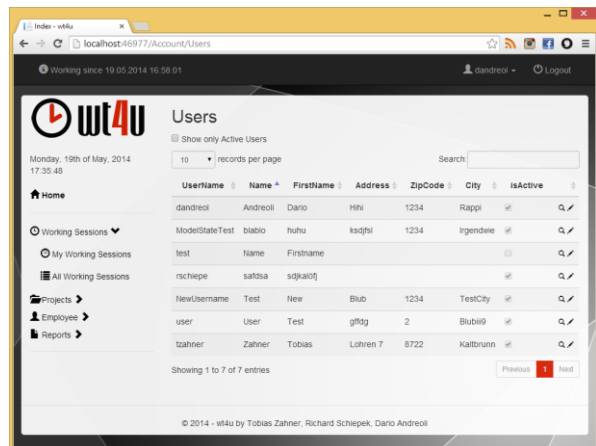
Search

#	Project Name	Start	End	spent Time	Closed
115	AndreoliProject	05.05.2005	24.04.2014	00:12:59 Hours	<input type="checkbox"/>
4638	asdkasdhjhsdfjklhasdfjkltest	01.05.2014	20.04.2014	00:00:00 Hours	<input type="checkbox"/>
3953	testasd	25.04.2014	23.06.2014	00:00:00 Hours	<input checked="" type="checkbox"/>
80	TestProject3000	03.04.2014	27.04.2015	05:58:33 Hours	<input type="checkbox"/>
3	TestProject3	01.04.2014	24.04.2014	00:00:17 Hours	<input checked="" type="checkbox"/>
118	TestProjekt98	20.05.2010	20.05.2010	06:09:12 Hours	<input checked="" type="checkbox"/>
116	Zahnerproject	05.05.2008	05.05.2008	00:10:29 Hours	<input checked="" type="checkbox"/>

Showing 1 to 7 of 7 entries

Previous Next

Benutzerübersicht:



Users

Monday, 19th of May, 2014 17:35:48

Home

Working Sessions

My Working Sessions

All Working Sessions

Projects

Employee

Reports

Show only Active Users

10 records per page

Search

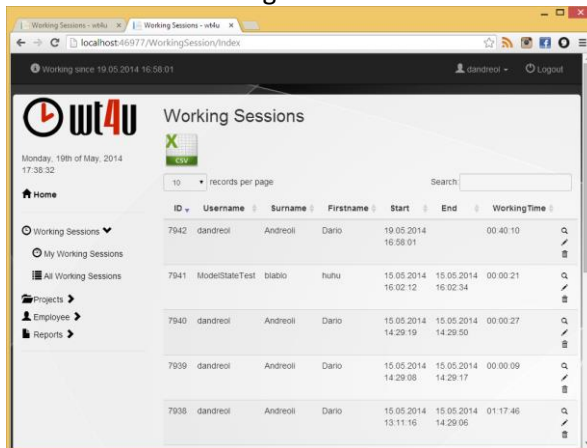
Username	Name	Firstname	Address	ZipCode	City	isActive
dandreei	Andreoli	Dario	Hifi	1234	Rappi	<input checked="" type="checkbox"/>
ModelStateTest	biabio	huhu	ksdjhl	1234	Igendane	<input checked="" type="checkbox"/>
test	Name	Firstname				<input checked="" type="checkbox"/>
rschiepe	safdsa	sdjksdlf				<input checked="" type="checkbox"/>
NewUsername	Test	New	Blub	1234	TestCity	<input checked="" type="checkbox"/>
user	User	Test	gfdg	2	Bubi9	<input checked="" type="checkbox"/>
tzahner	Zahner	Tobias	Lohren 7	8722	Kaltbrunn	<input checked="" type="checkbox"/>

Showing 1 to 7 of 7 entries

Previous Next

© 2014 - wt4u by Tobias Zahner, Richard Schiepek, Dario Andreoli

Übersicht der Working Sessions:



Working Sessions

Monday, 19th of May, 2014 17:35:32

Home

Working Sessions

My Working Sessions

All Working Sessions

Projects

Employee

Reports

10 records per page

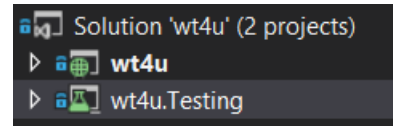
Search

ID	Username	Surname	Firstname	Start	End	WorkingTime
7942	dandreei	Andreoli	Dario	19.05.2014 16:58:01		00:40:10
7941	ModelStateTest	biabio	huhu	15.05.2014 16:02:12	15.05.2014 16:02:34	00:00:21
7940	dandreei	Andreoli	Dario	15.05.2014 14:29:19	15.05.2014 14:29:50	00:00:27
7939	dandreei	Andreoli	Dario	15.05.2014 14:29:08	15.05.2014 14:29:17	00:00:09
7938	dandreei	Andreoli	Dario	15.05.2014 13:11:16	15.05.2014 14:29:06	01:17:46

4. Ordnerstruktur .NET Solution wt4u

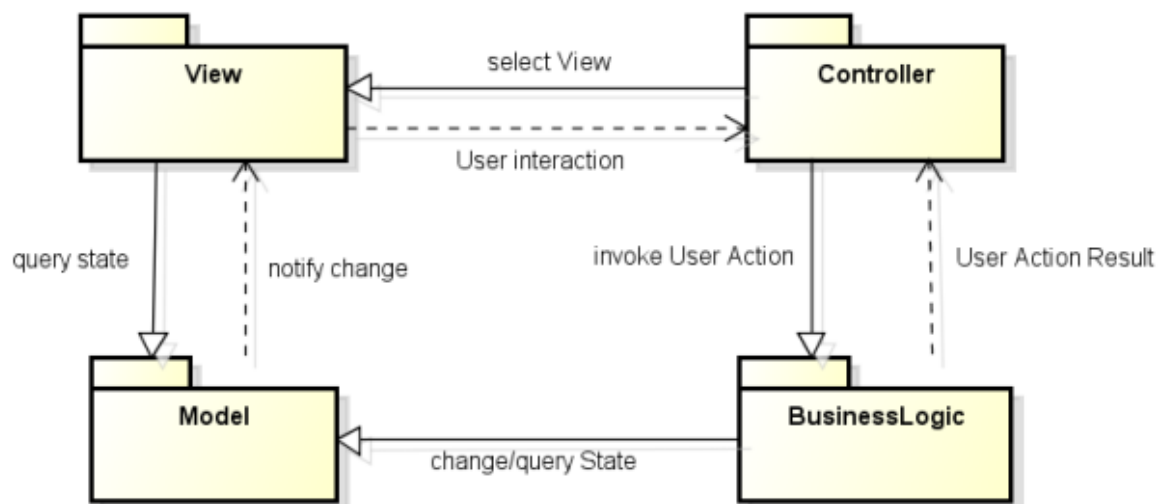
4.1 Übersicht

Die .NET Solution wt4u besteht aus 2 Projekten, wt4u und wt4u.Testing. Das komplette Programm inklusive ASP.NET Standard-Files befindet sich im Projekt wt4u. Alle Testmethoden im Projekt wt4u.Testing. Nachfolgend ein Überblick über die Ordnerstruktur der Solution.

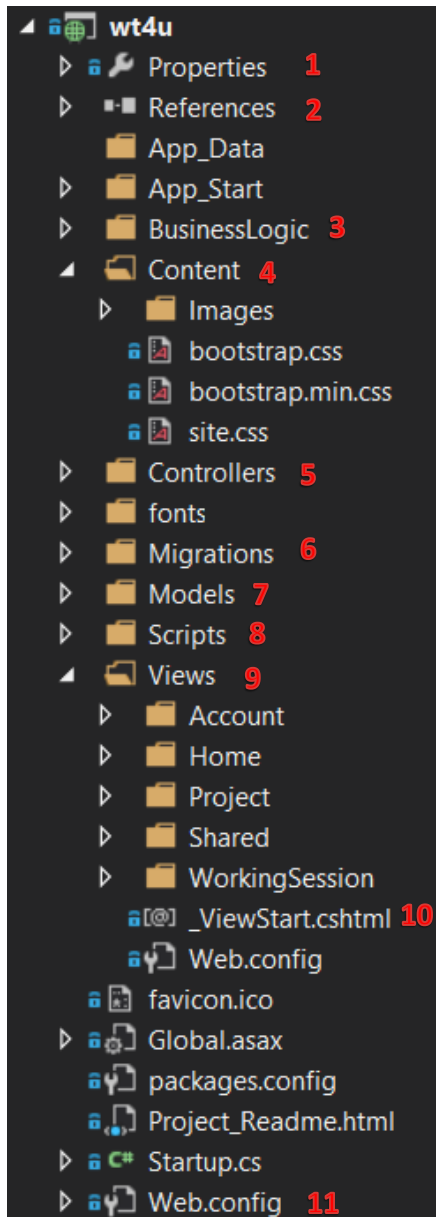


4.2 Packages

Um einen Überblick über wt4u zu erhalten, werden zuerst die verschiedenen Packages, sowie deren Abhängigkeiten untereinander aufgezeigt.

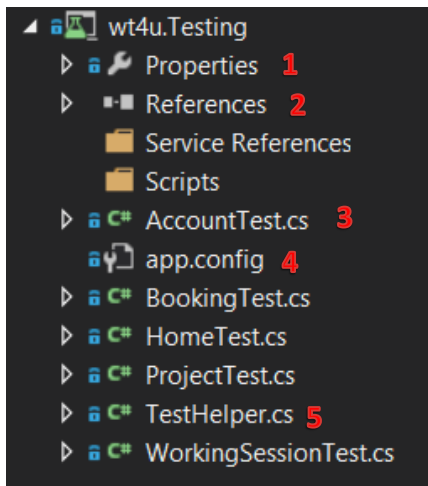


4.3 Project wt4u



Nr.	Beschreibung
1	Allgemeine Projekteigenschaften, sowie Information über das Webserver-Deployment.
2	Alle Referenzen auf NuGET Packages und .NET API's
3	Alle Klassen der BusinessLogic. Nur in der BusinessLogic werden Datenbankzugriffe getätigt. Hier befindet sich der Grossteil der Logik von wt4u. BusinessLogic Klassen werden über den Controller angesteuert (instanciiert).
4	Im Content befinden sich die .css-Files der Applikation. Bootstrap wurde standartmässig integriert. In der site.css werden individuelle Designs gespeichert. Im Images Ordner befinden sich die Bilder für wt4u.
5	Alle Controller Klassen. Controller werden über die Views oder Über die Browserleiste angesteuert.
6	Wt4u wurde nach dem Code-First Prinzip programmiert. Nach Anpassungen der Entities müssen jeweils Datenbank Migrationen getätigt werden. Sämtliche Migrationen (DB-Befehle) sind in diesem Ordner gespeichert.
7	Die Model-Klassen fürs Entity Framework sind hier abgelegt. Werden in einem Model Anpassungen getätigt, müssen Sie auf der Datenbank aktualisiert werden.
8	JavaScript Files werden im Scripts-Ordner abgelegt. Wichtige Scripts für wt4u: JQuery, Bootstrap, Modernizr, respond.
9	Sämtliche Frontendseiten (.cshtml) sind im Views-Ordner abgelegt. Im Shared Ordner sind die Templates gespeichert. Das _Layout.cshtml ist auf jeder weiteren .cshtml Seite sichtbar.
10	Hier wird das Layout-File für die Views festgelegt
11	In dieser Datei sind diverse Einstellungen gespeichert, sowie Bindings zu anderen Assemblies (Packages) eingetragen. Der Wichtigste Punkt ist der ConnectionString. Dort werden die Angaben zur Datenbankverbindung eingetragen.

4.4 Project wt4u.Testing

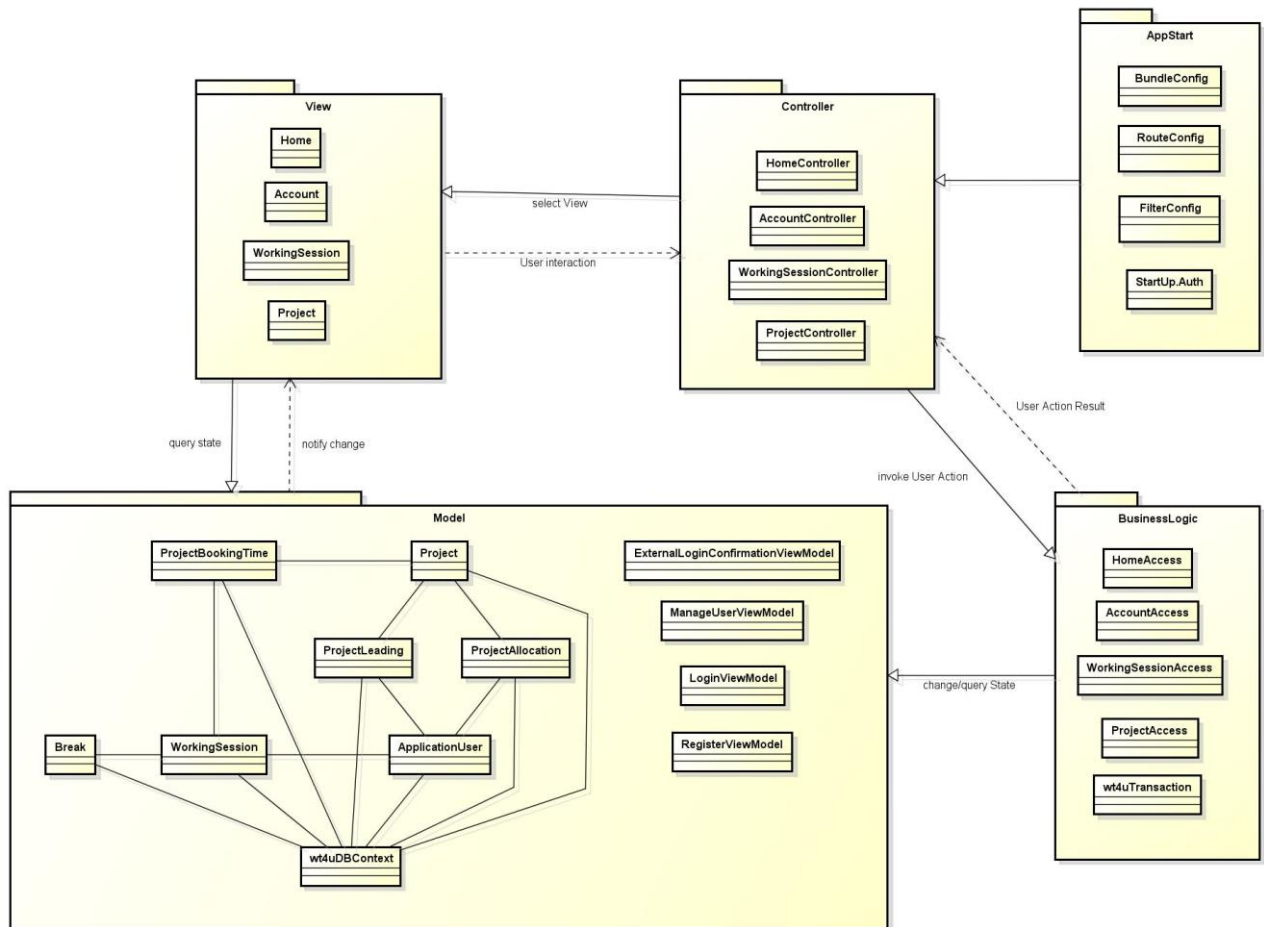


Nr.	Beschreibung
1	Allgemeine Projekteigenschaften, sowie Information über das Webserver-Deployment.
2	Alle Referenzen auf NuGET Packages und .NET API's, die für das Testing benötigt werden. Hier sind nicht alle References aus dem wt4u Project enthalten, die meisten ASP.NET Referenzen sind hier überflüssig.
3	AccountTest, BookingTest, ProjectTest, WorkingSessionTest, AccountTest sind Testklassen. Hier werden diverse Operationen auf die BusinessLogic getestet. Es wird keine ASP.NET Applikation getestet. Dieses Project verfügt über eine Testdatenbankanbindung
4	In dieser Datei sind diverse Einstellungen gespeichert sowie Bindings zu anderen Assemblies (Packages) eingetragen. Der Wichtigste Punkt ist der ConnectionString. Dort werden die Angaben zur Datenbankverbindung eingetragen.
5	In dieser Klasse werden Sample Daten für die Testdatenbank generiert. Im Anschluss an jeden Test werden die Daten wieder sauber abgeräumt. Nachteil: Bei fehlgeschlagenen Tests (Exception) können teilweise Daten übrigbleiben. Vorteil: Es wird direkt mit Datenbank getestet.

5. Logische Architektur

Wir verwenden bei in unserer ASP.NET Applikation das MVC5 Modell, wie es vorgesehen ist. Jede View hat seinen passenden Controller. Die Controller nehmen entsprechende Parameter aus dem View entgegen und leiten sie bei korrektem Modelstate an die Access Klassen (BusinessLogic) weiter. Die Accessklassen sind die einzigen Klassen mit Datenbankzugriff. In diesen Klassen ist die meiste Logik von wt4u implementiert.

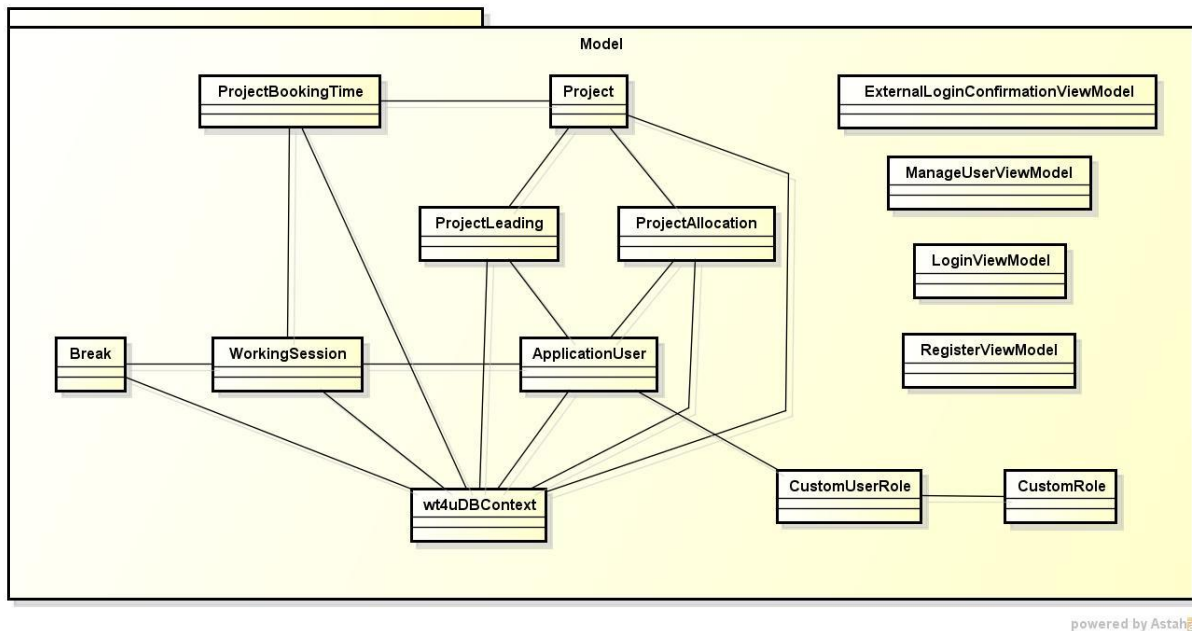
Das AppStart Package ist für die Architektur nicht weiter relevant, da es automatisch vom ASP.NET MVC5 Framework erzeugt wurde. Nur die RouteConfig ist hier von Belangen, dort wird festgelegt, über welche URL ein Controller aufgerufen wird und welche die Standard-Home-Seite ist.



powered by Astah

5.1 Models

5.1.1 Klassenstruktur



5.1.2 Klassen

Die Klassen wurden genau so übernommen, wie sie in der Domainanalyse festgelegt wurden. Um die Beschreibung der Klassen sowie deren Attribute einzusehen, siehe Punkt Datenspeicherung.

5.1.3 Schnittstellen

Wt4uDBContext ist die Schnittstelle des Models zur Datenbank. Über eine Instanz von wt4uDBContext kann direkt (z.B. über Linq) auf die Datenbank zugegriffen werden. Während der Entwicklung wird hier eine Testdatenbank im Einsatz sein, welche einfach im Connectionstring angepasst werden kann.

5.1.4 Abhängigkeiten auf andere Packages

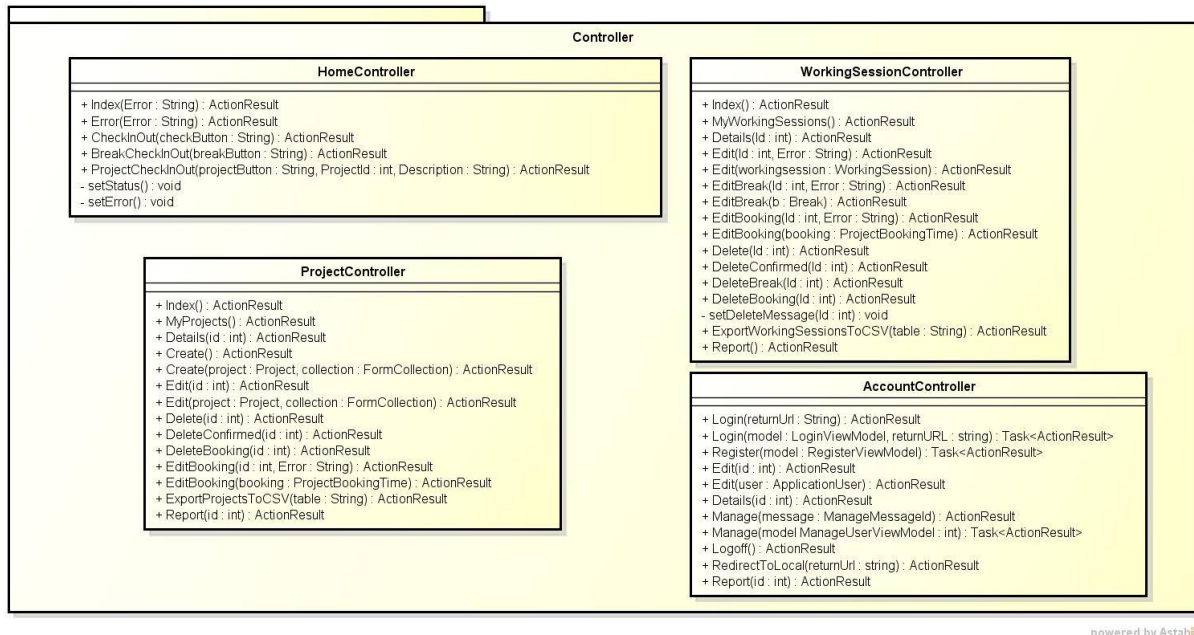
- View kann auf sämtliche Modelklassen(ausser wt4uContext) zugreifen und gewünschte Queries abfragen (query state).
- Methoden aus Klassen der BusinessLogic können Modeldaten abfragen sowie verändern.

5.1.5 Wichtige interne Abläufe

Interne Abläufe gibt es im Model keine. Die Modelklassen sind direkt an die DB gebunden. Hier wird das Entity Framework von .NET eingesetzt. Die Modelklassen enthalten keine Logik, sie sind nur als Datenhaltungsklassen konzipiert. Die Logik der Applikation befindet sich im Controller Package und grösstenteils im BusinessLogic Package.

5.2 Controllers

5.2.1 Klassenstruktur



powered by Astah

5.2.2 Schnittstellen

Jeder Controller kann diverse Views (.cshtml-Seiten) zurückgeben, somit ist jeder Controller eine Schnittstelle für die Views.

Methoden ohne Annotations (GET oder POST) sind HTTP-GET Methoden. POST-Methoden müssen extra deklariert werden. Dies wird mittels der Annotation [HttpPost] gemacht.

Der Controller befindet sich Serverseitig und nimmt HTTP-Requests vom Client(View) entgegen. Im Package AppStart befindet sich die Datei RouteConfig.cs. In dieser externen Schnittstelle ist festgelegt welcher Controller auf eine spezifisch URL hört. Hier ist insbesondere die Standard-Homepage anzugeben. Falls in der RouteConfig sonst nichts angegeben ist, hören die Controller jeweils auf den Klassennamen:

Beispiel: www.wt4u.ch/Project/Index = Hier wird die View angezeigt, welche die Index Methode im Project Controller zurückgibt. Im Abschnitt Controller Methoden wird erwähnt über welche Controller Methoden wie zugegriffen werden kann.

5.2.3 Abhängigkeiten auf andere Packages

- Der Controller nimmt Anfragen von der Benutzeroberfläche (View-Package) entgegen. Entsprechende Methoden sind im ViewCode folgendermassen sichtbar:

Über Forms

```
@using (Html.BeginForm("ProjectCheckInOut", "Home", FormMethod.Post))
{<input type="submit" name="projectButton" value="Start" class="btn btn-lg btn-success" /> }
```

Über ActionLinks

```
@Html.ActionLink("Edit", "EditBooking", new { id = item.BookingId })
```

- Der Controller gibt Resultate die vom User angefragt werden über eine View zurück. Entsprechende Resultate können auf zwei Varianten auf die View gebracht werden:

Über ViewBag

Im Controller: ViewBag.foo = „bar“

Im View: Zugriff via @ViewBag.foo

Über ViewModel

Im Controller: return View(workingSession)

Im View: @model wt4u.Models.WorkingSession >> Zugriff via model.Variable

In wt4u wird bei simplen Oberflächen das Model-Binding benutzt, sonst aber der ViewBag.

- Der Controller fungiert vor Allem als Weiterleitung auf die BusinessLogic. User Requests werden dementsprechend auf die passende BusinessLogic Klasse weitergeleitet. Dort wird das passende Resultat geholt.

5.2.4 Wichtige interne Abläufe

Controller kommunizieren nicht untereinander sondern direkt mit der BusinessLogic und den Views. Darum ergeben sich im Package keine internen Abläufe.

5.2.5 Controller Methoden

5.2.5.1 Home

Adresse	HTTP	Beschreibung	Parameter	Rechte
Home/Index	GET	Nach dem Einloggen ist dies die Startseite der wt4u Application. In dieser Übersicht kann man folgende Operationen Starten oder Beenden: Arbeitszeit, Pause oder Projektarbeitszeit	- <u>String Error</u> Ein allfälliger Error (z.B. bei fehlgeschlagener Transaction oder bei fehlender Eingabe des Descriptionfelds) kann hier mitgegeben werden	Employer Employee
Home/Error	GET	Taucht aus diversen Gründen ein Fehler auf (z.B. fehlende Berechtigung für eine Controller Methode wird auf die Errorseite weitergeleitet.	- <u>String Error</u> Falls noch gewünscht können individuelle Errornachrichten auf der Errorpage ausgegeben werden.	Employer Employee
Home/CheckInOut	POST	Mit dieser Controllerfunktion kann eine Arbeitszeit entweder gestartet oder beendet werden.	- <u>String checkButton</u> Ob eine Arbeitszeit gestartet oder beendet wird hängt vom gedrückten Button ab. Diese Angabe ob Start oder End wird hier als Parameter mitgegeben.	Employer Employee
Home/BreakCheckInOut	POST	Mit dieser Controllerfunktion kann eine Pause entweder gestartet oder beendet werden.	- <u>String breakButton</u> Ob eine Pause gestartet oder beendet wird hängt vom gedrückten Button ab. Diese Angabe ob Start oder End wird hier als Parameter mitgegeben.	Employer Employee
Home/ProjectCheckInOut	POST	Mit dieser Controllerfunktion kann ein Projekt entweder gestartet oder beendet werden.	- <u>String breakButton</u> Ob ein Projekt gestartet oder beendet wird hängt vom gedrückten Button ab. Diese Angabe ob Start oder End wird hier als Parameter mitgegeben. - <u>Int? ProjectId</u> Hier wird die Id mitgegeben, die im View aus der Liste der verfügbaren Projekte ausgewählt wurde. - <u>String Description</u> Beendet man eine Projektbuchung bewusst, muss eine Beschreibung mitgegeben werden, was in dieser Zeit erledigt wurde.	Employer Employee

Home/Status	GET	Hier kann der Status des aktuell eingeloggten Users etwas genauer betrachtet werden mit einigen Detailinformationen.	-	Employer Employee
-------------	-----	----------------------------------------------------------------------------------------------------------------------	---	----------------------

5.2.5.2 WorkingSession

Adresse	HTTP	Beschreibung	Parameter	Rechte
WorkingSession/Index	GET	Diese Seite wird durch den Menüpunkt workingSessions > All WorkingSessions erreicht. Hier sind Arbeitszeiten aller Mitarbeiter einsehbar inklusive Links zu Details, Delete, Edit.	-	Employer
WorkingSession /MyWorkingSessions	GET	Diese Seite wird durch den Menüpunkt workingSessions > My WorkingSessions erreicht. Hier sind die eigenen Arbeitszeiten einsehbar inklusive Links zu Details, Delete, Edit.	-	Employer Employee
WorkingSession/Details/{id}	GET	In den Details sind diverse Zeitdaten zur Arbeitszeit einsehbar sowie alle dazugehörigen Projektbuchungen und Pausen. Employees dürfen nur Details zu eigenen Sessions einsehen.	- Int Id Die Id der gewünschten WorkingSession	Employer Employee (nur eigene Sessions)
WorkingSession/Edit/{id}	GET	Als Employer hat man das Recht Arbeitszeiten zu bearbeiten. Die Start- und Endzeit kann im Editview angepasst werden.	- Int Id Die Id der gewünschten WorkingSession - String Error Bei Zeitangaben, die Überschneidungen mit der Arbeitszeit, der Pause oder der Buchungszeit ergeben, erscheint eine Errormessage, die im Controller als String mitgegeben wird	Employer
WorkingSession/Edit/{id}	POST	Durch den Save-Button in der Edit-View wird die Postmethode ausgelöst. Bei korrekten Eingaben wird auf der Datenbank gespeichert. Bei Falschangaben wird wieder auf die Edit-View inklusive Errormessage weitergeleitet	- WorkingSession workingsession Die Arbeitszeit aus dem Formular im Edit-View. Geänderte Daten direkt mit Modelbinding übergeben.	Employer
WorkingSession	GET	Als Employer hat man das Recht Pausen zu	- Int Id	Employer

/EditBreak/{id}		bearbeiten. Die Start- und Endzeit kann im EditBreak-View angepasst werden.	<p>Die Id der gewünschten Break</p> <ul style="list-style-type: none"> - <u>String Error</u> Bei Zeitangaben, die Überschneidungen mit der Arbeitszeit, der Pause oder der Buchungszeit ergeben, erscheint eine Errormessage, die im Controller als String mitgegeben wird 	
WorkingSession /EditBreak/{id}	POST	Durch den Save-Button in der EditBreak-View wird die Postmethode ausgelöst. Bei korrekten Eingaben wird auf der Datenbank gespeichert. Bei Falschangaben wird wieder auf die EditBreak-View inklusive Errormessage weitergeleitet	<ul style="list-style-type: none"> - <u>Break b</u> Die Break aus dem Formular im EditBreak-View. Geänderte Daten direkt mit Modelbinding übergeben. 	Employer
WorkingSession /EditBooking/{id}	GET	Als Employer hat man das Recht Projektarbeitszeiten zu bearbeiten. Die Start- und Endzeit kann im EditBooking-View angepasst werden.	<ul style="list-style-type: none"> - <u>Int Id</u> Die Id der gewünschten Projektbuchung - <u>String Error</u> Bei Zeitangaben, die Überschneidungen mit der Arbeitszeit, der Pause oder der Buchungszeit ergeben, erscheint eine Errormessage, die im Controller als String mitgegeben wird 	Employer
WorkingSession /EditBooking/{id}	POST	Durch den Save-Button in der EditBooking-View wird die Postmethode ausgelöst. Bei korrekten Eingaben wird auf der Datenbank gespeichert. Bei Falschangaben wird wieder auf die EditBooking-View inklusive Errormessage weitergeleitet	<ul style="list-style-type: none"> - <u>ProjectBookingTime booking</u> Die ProjectBookingTime aus dem Formular im EditBooking-View. Geänderte Daten direkt mit Modelbinding übergeben. 	Employer
WorkingSession/Delete/{id}	GET	Durch den Delete-Link wird man auf eine Seite weitergeleitet, die eine Bestätigung verlangt ob man die Arbeitszeit löschen will. Es erscheint eine Warnung falls allfällige Projektbuchungen und Pausen die zur WorkingSession gehören auch gelöscht werden.	<ul style="list-style-type: none"> - <u>Int Id</u> Die Id der gewünschten Arbeitszeit 	Employer
WorkingSession/Delete/{id}	POST	Bestätigt man die Löschung der WorkingSession auf der Delete-View, wird die Arbeitszeit &	<ul style="list-style-type: none"> - <u>Int Id</u> Die Id der gewünschten Arbeitszeit 	Employer

		Abhängige Bookings und Breaks gelöscht		
WorkingSession/DeleteBreak/{id}	POST	Auf dem Detail-View der WorkingSession kann man direkt eine Pause löschen.	- <u>Int Id</u> Die Id der gewünschten Pause	Employer
WorkingSession/DeleteBooking/{id}	POST	Auf dem Detail-View der WorkingSession kann man direkt eine Projektbuchungszeit löschen.	- <u>Int Id</u> Die Id der gewünschten Projektbuchungszeit	Employer
WorkingSession/ExportWorkingSessionsToCSV/{table}	GET	In der Übersicht der Arbeitszeiten kann man sich die Tabelle als CSV-Datei exportieren lassen indem man auf den entsprechenden Button clickt.	- <u>String Table</u> Hier wird mitgegeben aus welcher View man kommt (MyWorkingSessions oder AllWorkingSessions(index))	Employer Employee(nur „my“)
WorkingSession/Report	GET	Methode gibt die Report View für eine Gesamtübersicht der Arbeitszeiten über Projekte und Wochentage.	-	Employer

5.2.5.3 Project

Adresse	HTTP	Beschreibung	Parameter	Rechte
Project/Index	GET	Diese Methode wird durch den Menüpunkt Projects > All Projects aufgerufen. Hier sind alle Projekte tabellarisch aufgelistet. Jedes Projekt besitzt zudem einen Link zu Details, Edit und Delete.	-	Employer
Project/MyProjects	GET	Diese Methode wird durch den Menüpunkt Projects > My Projects aufgerufen. Hier sind die eigenen Projekte einsehbar inklusive Links zu Details, Edit und Delete.	-	Employer Employee
Project/Details/{id}	GET	Diese Methode wird aufgerufen, sobald bei einem Project der Edit Link aufgerufen wird. Sie gibt im Erfolgsfall die Detail View des jeweiligen Projektes zurück.	- <u>Int id</u> Die id des gewünschten Projektes	Employer, Employee
Project/Create	GET	Die Methode wird aufgerufen, wenn ein neues Projekt erstellt werden soll und gibt die View zum create.cshtml zurück.	-	Employer

Project/Create	POST	Wird beim Absenden des Formulars aus create.cshtml aufgerufen und erstellt das gewünschte Projekt in der DB. Am Ende wird die Index View aus Project zurückgegeben.	<ul style="list-style-type: none"> - <u>Project project</u> Die Daten des Formulars werden mittels Binding auf die Variable project gebunden - <u>FormCollection collection</u> Um auf den gewählten Projektführer und die Liste des Projektteams zugreifen zu können, werden die Eingaben des Formulars zusätzlich über eine FormCollection übergeben (wird über das Formular übergeben). Dies ist notwendig, da die genannten Felder nicht im Model des Projects sind. 	Employer
Project/Edit/{id}	GET	Methode wird aufgerufen, wenn Projekt bearbeitet werden soll. Am Ende wird im Erfolgsfall die entsprechende Edit View zurückgegeben.	<ul style="list-style-type: none"> - <u>Int id</u> Id des gewünschten Projektes 	Employer, Employee (nur Projekt-leiter)
Project/Edit	POST	Änderungen am Project werden in der DB gespeichert.	<ul style="list-style-type: none"> - <u>Project project</u> Das Project aus dem Formular in der Edit View. Daten werden per Binding übergeben - <u>FormCollection collection</u> Da auch hier auf Projektleiter und das Projektteam zugegriffen werden muss, können diese Daten aus der Variable collection ausgelesen werden (wird über das Formular übergeben). 	Employer, Employee (nur Projekt-leiter)
Project/Delete/{id}	GET	Methode wird beim Betätigen des Links zum Löschen eines Projektes aufgerufen und gibt die Delete View für das entsprechende Projekt zurück.	<ul style="list-style-type: none"> - <u>Int id</u> Id des zu löschenden Projektes 	Employer, Employee (nur Projekt-leiter)
Project/DeleteConfirmed/{id}	POST	Methode wird aufgerufen, um das Projekt aus der Datenbank zu löschen.	<ul style="list-style-type: none"> - <u>int id</u> Id des zu löschenden Projektes 	Employer, Employee (nur Projekt-

				leiter)
Project/DeleteBooking	POST	Methode wird aufgerufen, wenn aus der Detail View eines Projektes ein Datensatz vom Typ ProjectBookingTime gelöscht werden soll.	- Int id Id der gewünschten ProjectBookingTime	Employer
Project/EditBooking/{id}	GET	Methode wird aufgerufen, wenn aus der Detail View eines Projektes ein Datensatz vom Typ ProjectBookingTime editiert werden soll.	- Int id Id der gewünschten ProjectBookingTime - String Error Bei Zeitangaben, die Überschneidungen mit der Arbeitszeit, der Pause oder der Buchungszeit ergeben, erscheint eine Error Message, die im Controller als String mitgegeben wird	Employer
Project/EditBooking	POST	Methode speichert die Daten des Bookings in der Datenbank.	- ProjectBookingTime booking Daten der ProjectBookingTime werden per Binding aus dem Formular der Methode übergeben	Employer
Project/ExportProjectsToCSV/{table}	GET	In der Übersicht der Projekte kann man sich die Tabelle als CSV-Datei exportieren lassen indem man auf den entsprechenden Button clickt.	- String Table Hier wird mitgegeben aus welcher View man kommt (MyProjects oder AllProjects(index))	Employer Employee(nur „my“)
Project/Report/{id}	GET	Methode gibt die Report View für Projekte zurück.	- Int id Id des gewünschten Projektes	Employer, Employee (nur Projekt-leiter)

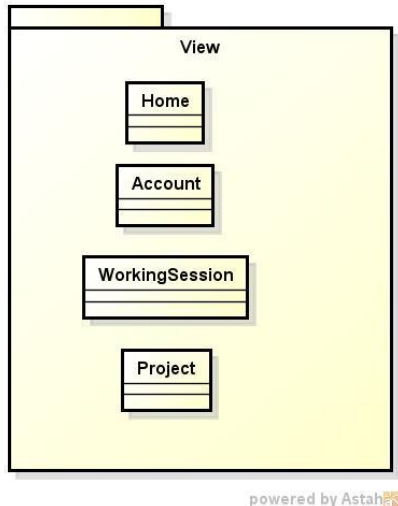
5.2.5.4 Account

Adresse	HTTP	Beschreibung	Parameter	Rechte
Account/Login	GET	Gibt die Login-View zurück	String returnUrl URL auf die man nach erfolgreichem Login weitergeleitet wird.	Öffentlich
Account/Login	POST	Führt den Login durch, bei erfolgreichem Login kommt man auf die returnUrl-Seite, ansonsten erneut auf die Login-Seite	LoginViewModel model Model des Login-Formulars String returnUrl	Öffentlich

			URL auf die man nach erfolgreichem Login weitergeleitet wird.	
Account/Register	GET	Gibt die View zur Erfassung eines neuen Benutzers zurück.	-	Employer
Account/Register	POST	Legt einen neuen Benutzer an.	RegisterViewModel model Model des RegisterViews, welches die Userfelder enthält.	Employer
Account/Edit	GET	Gibt die Edit-View eines bestimmten Users zurück	Int id userId des zu editierenden Benutzers	Employer
Account/Edit	POST	Speichert die Formular-Felder in das Userobjekt und dadurch auch in die Datenbank.	ApplicationUser user Der zu bearbeitende User	Employer
Account/Details	GET	Greift auf ProjectAccess und AccountAccess zu und gibt die Detail-View eines Users zurück	Int id Id des anzuzeigenden Users	Employer
Account/Manage	GET	Gibt die View zur Änderung des Passworts zurück	ManageMessageId message MessageObjekt um Meldungen anzuzeigen, z.B. Passwort wurde erfolgreich geändert	Employer, Employee
Account/Manage	POST	Ändert das Passwort des angemeldeten Users, sofern die Angaben (altes Passwort, PasswordConfirmation) gültig sind.	ManageUserViewModel model Model der ManageView	Employer, Employee
Account/LogOff	POST	Der aktuell angemeldete Benutzer wird ausgeloggt	-	Employer, Employee
Account/Report/{id}	GET	Methode gibt die Report View für die Benutzer zurück.	- Int id Id des gewünschten Employee	Employer, Employee (nur Projekt-leiter)

5.3 Views

5.3.1 Klassenstruktur



5.3.2 View-Files (.cshtml)

Home	
Index.cshtml	Startseite > Hier werden Arbeitszeiten, Projektarbeitszeiten und Pausen gestartet und beendet.
Error.cshtml	Falls man eine Operation(Controller) ausführt, zu der man keine Berechtigung hat, landet man auf dieser Errorseite. Dies passiert wenn man in der Browserleiste einen Controller manuell ansteuert. Z.B. /WorkingSession/Edit/1234
Status.cshtml	Hier wird der Status des aktuellen Nutzers beschrieben. Informationen über die aktuelle Arbeitszeit, Pause und Projektbuchungen. Dies dient als erweiterte Statusanzeige zum Status in der schwarzen Leiste oben.
WorkingSession	
Index.cshtml	Hier werden alle Arbeitszeiten angezeigt inklusive Links für Edit, Details und Delete.
Delete.cshtml	Bestätigungsseite zur Löschung einer Arbeitszeit
Details.cshtml	Detailanzeige zu einer Arbeitszeit. Es werden ebenfalls alle Bookings und Breaks der Arbeitszeit angezeigt
Edit.cshtml	Editieren der Start- und Endzeit einer WorkingSession
EditBooking.cshtml	Editieren der Start- und Endzeit einer Projektbuchungszeit
EditBreak.cshtml	Editieren der Start- und Endzeit einer Pause
Report.cshtml	Generelle Übersicht über die Arbeitszeiten der Projekte und eine Statistik der Auslastung über die Wochentage
Project	
Index.cshtml	Hier werden alle Projekte angezeigt mit Links für Edit, Details und Delete.
Create.chhtml	Ein neues Projekt erstellen
Delete.cshtml	Bestätigungsseite zur Löschung eines Projekts

Edit.cshtml	Editieren von Angaben über das Projekt, inklusive Mitarbeiterzuweisungen
EditBooking.cshtml	Editieren der Start- und Endzeit einer Projektbuchungszeit
Report.cshtml	Report über das gewünschte Projekt

Account	
_ChangePasswordPartial.cshtml	HTML-Part welcher für das Ändern des Passwortes zuständig ist.
_SetPasswortPartial.cshtml	HTML-Part welcher für das Setzen des Passwortes zuständig ist, falls noch keines vorhanden ist.
Details.cshtml	Detail-View eines bestimmten Benutzers
Edit.cshtml	Mit dieser View können die Angaben eines Benutzers editiert werden.
Login.cshtml	Startseite, auf welcher sich der Benutzer einloggen kann.
Manage.cshtml	Bindet je nach Bedarf _ChangePasswordPartial oder SetPasswortPartial ein. Wird verwendet um das Passwort des Benutzers zu ändern.
Register.cshtml	Wird benötigt um einen neuen Benutzer zu erfassen.
Users.cshtml	Enthält eine Liste aller User, welche bei Bedarf auch gleich im Detail betrachtet oder editiert werden können.
Report.cshtml	Report über den gewünschten Employee

5.3.3 Schnittstellen

Der Client mit seinem Browser ist hier die Schnittstelle für das User-Interface. Jedes ViewPackage enthält mehrere .cshtml Seiten die durch den Controller zur Verfügung gestellt werden.

5.3.4 Abhängigkeiten auf andere Packages

- Leitet User Interaktionen mittels HTTP-Requests an den entsprechenden Controller weiter.
- Fragt Daten von EntityModels ab um diese auf der cshtml-Seite darzustellen.

5.3.5 Wichtigste interne Abläufe

Unter den Viewklassen gibt es keine interne Abläufe. Erwähnenswert ist noch das File _Layout.cshtml. Der Inhalt dieser Datei ist auf allen anderen Views zu sehen und ist das Grundlayout. Der Bereich für das Rendering des Contents kann ebenfalls in diesem Dokument gewählt werden.

5.4 Business Logic

Im BusinessLogic Layer befindet sich der Grossteil der Logik der Applikation wt4u. Beim ASP.NET MVC-Framework werden häufigerweise nur die Packages Model-View-Controller verwendet. Doch gemäss der Besprechung vom „Review 3 Architekturprototyp“ entschieden wir uns dafür ein neues Package BusinessLogic hinzuzufügen. Der Hauptgrund für die Erstellung eines neuen Packages war die Testbarkeit vom Controller-Package. Wenn sich alle Operationen mit Datenbankzugriff im Controllerpackage befinden, können diese Operationen nicht ohne Mocks getestet werden. Der Grund ist, um einen Controller anzusteuern, muss die ganze ASP.NET MVC-Applikation gestartet werden und dann „künstliche“ HTTP-Requests auf den Controller aufgerufen werden. Dies ist allerdings in einem Test-Projekt nicht ohne Weiteres möglich.

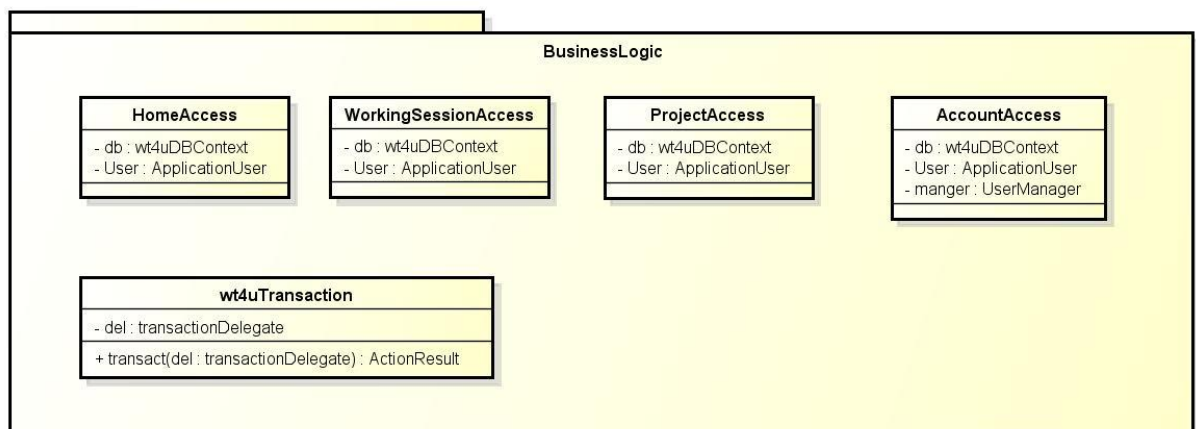
Mit der BusinessLogic können wir direkt Testen indem man einfach eine neue Instanz einer Access-Klasse erstellt und dort den gewünschten User erstellt. Hier ein kleines Beispiel für einen Methodenaufruf aus dem Controller, wie eine WorkingSession aus der Datenbank ausgelesen werden kann:

```
WorkingSession workingSession = new WorkingSessionAccess(User.Identity.Name).getWorkingSession(id);
```

Hier ein einfaches Codebeispiel wie Daten aus der Datenbank gelesen werden können:

```
18 references | 0/8 passing
public WorkingSession getWorkingSession(int? WorkingSessionId)
{
    return db.WorkingSessions.Find(WorkingSessionId);
}
```

5.4.1 Klassenstruktur



powered by Astah

5.4.2 Klassen

HomeAccess	
Beschreibung	Im HomeAccess sind Methoden für den Home-Controller implementiert. Die Hauptfunktionen dieser Klasse sind: <ul style="list-style-type: none"> Starten und Beenden von Arbeitszeiten, Pausen und Projektarbeitszeiten Abfrage des Status von Arbeitszeit, Pause und Projektarbeitszeit.
Klassenvariablen	<ul style="list-style-type: none"> <u>Wt4uDBContext db</u>; Über diese Variable können Datenbankzugriffe getätigt werden. Dies bedeutet Felder abfragen, Felder updaten oder erstellen. <u>ApplicationUser User</u>; Repräsentiert den User, der die Instanz der Klasse erstellt hat.
Methoden	Siehe Methoden in der .NET Solution. Die Methodennamen erklären den Sinn

der Funktion vollständig.

WorkingSessionAccess

Beschreibung	<p>Im WorkingSessionAccess sind Methoden für den WorkingSession-Controller implementiert.</p> <p>Die Hauptfunktionen dieser Klasse sind:</p> <ul style="list-style-type: none"> • Abfrage aller WorkingSessions oder aller WorkingSession des Users. • Berechnung der Arbeitszeit, Pausenzeit und Buchungszeit einer WorkingSession. • Abfrage von Pausen und Projektbuchungen pro WorkingSession. • Löschen von Arbeitszeiten, Pausen. • Editieren von Arbeitszeiten, Pausen. • Überprüfungen ob Edit & Deletes korrekt sind.
Klassenvariablen	<ul style="list-style-type: none"> • <u>Wt4uDBContext db;</u> Über diese Variable können Datenbankzugriffe getätigt werden. Dies bedeutet Felder abfragen, Felder updaten oder erstellen. • <u>ApplicationUser User;</u> Repräsentiert den User, der die Instanz der Klasse erstellt hat.
Methoden	Siehe Methoden in der .NET Solution. Die Methodennamen erklären den Sinn der Funktion vollständig.

ProjectAccess

Beschreibung	<p>Im ProjectAccess sind Methoden für den Project-Controller implementiert.</p> <p>Die Hauptfunktionen dieser Klasse sind:</p> <ul style="list-style-type: none"> • Abfrage aller Projekte oder alle Projekte des Users. • User einem Projekt zuweisen oder gleich zum Leader machen. • Erstellen eines neuen Projekts. • Löschen von Projekten, Projektbuchungen. • Editieren von Projekten, Projektbuchungen. • Überprüfungen ob Edit & Deletes korrekt sind. • Teammember eines Projekts auslesen. • Projektbuchungen eins Projekts auslesen.
Klassenvariablen	<ul style="list-style-type: none"> • <u>Wt4uDBContext db;</u> Über diese Variable können Datenbankzugriffe getätigt werden. Dies bedeutet Felder abfragen, Felder updaten oder erstellen. • <u>ApplicationUser User;</u> Repräsentiert den User, der die Instanz der Klasse erstellt hat.
Methoden	Siehe Methoden in der .NET Solution. Die Methodennamen erklären den Sinn der Funktion vollständig.

AccountAccess

Beschreibung	<p>Im AccountAccess sind Methoden für den Account-Controller implementiert.</p> <p>Die Hauptfunktionen dieser Klasse sind:</p> <ul style="list-style-type: none"> • Abfrage aller Users der Applikation. • User editieren inklusive Role. • User erstellen inklusive Role. • Ändern des User-Passworts.
Klassenvariablen	<ul style="list-style-type: none"> • <u>UserManager<ApplicationUser, int> manager;</u> Der UserManager ist vom ASP.NET Framework bereitgestelltes Objekt,

	<p>mit dem Operationen für den Account vollzogen werden können. Z.B. Speichern eines Passworthashs auf der Datenbank oder Zuweisung einer Rolle für einen User.</p> <ul style="list-style-type: none"> • <u>Wt4uDBContext db;</u> Über diese Variable können Datenbankzugriffe getätigt werden. Dies bedeutet Felder abfragen, Felder updaten oder erstellen. • <u>ApplicationUser User;</u> Repräsentiert den User, der die Instanz der Klasse erstellt hat.
Methoden	Siehe Methoden in der .NET Solution. Die Methodennamen erklären den Sinn der Funktion vollständig.

Wt4uTransaction	
Beschreibung	<p>Die Transaction wurde erstellt, damit ein User nicht zweimal die gleiche Operation ausführen kann. Dies wäre der Fall wenn der gleiche User zweimal eingeloggt wäre und dieselbe Operation ausführt. Dies würde zu Fehlern während des Datenbankzugriffs kommen.</p> <p>Alle schreibenden Operationen werden mittels der wt4uTransaction ausgeführt. Diese Programmierung ist in allen Controllern sichtbar.</p>
Klassenvariablen	<ul style="list-style-type: none"> • <u>Public delegate ActionResult transactionResult();</u> DerTransactionmethode wird ein Delegate vom Typ transactionResult() übergeben. Diese ist als Klassenvariable festgelegt.
Methoden	<ul style="list-style-type: none"> • <u>Public ActionResult transact(transactionDelegate del)</u> Mittels TransactionScope serializable (.NET) wird hier die Ausführung des Delegates umgeben. Somit kann die Ausführung nur einmal vom gleichen User getätigt werden.

5.4.3 Schnittstellen

Die Access-Klassen in der BusinessLogic haben alle eine Datenbankschnittstelle. Alle DB-Zugriffe werden in diesem Layer getätigt. Somit ist der DB-Zugriff abgekoppelt vom Controller. Folgende Objekte repräsentieren den Datenbankzugriff

- Wt4uDBContext db;
Über dieses Objekt kann auf die Datenbank zugegriffen werden, hier einige Befehle

Objekt aus der DB holen	Db.Tabelle.Find(id)
Query	Db.Tabelle.Where(x => x.Id = Id)
Objekt in DB speichern	Db.Tabelle.Add(Object) Db.SaveChanges()
Objekt in DB ändern	Db.Entry(Object).State = EntityState.Modified Db.SaveChanges
Objekt in DB löschen	Db.Tabelle.Remove(Object) Db.SaveChanges
- userManager<ApplicationUser, int> manager
Mit dem userManager können Operationen was den Account (Username, Passwort, Rolle ...) betrifft direkt mit Datenbankzugriff ausgeführt werden.

5.4.4 Abhängigkeiten auf andere Packages

- Eine Controllermethode erstelle eine Instanz einer AccessMethode und ruft dann die gewünschte Funktion auf. Das gewünschte Resultat wird dann vom Access an den Controller per return-Wert übergeben.

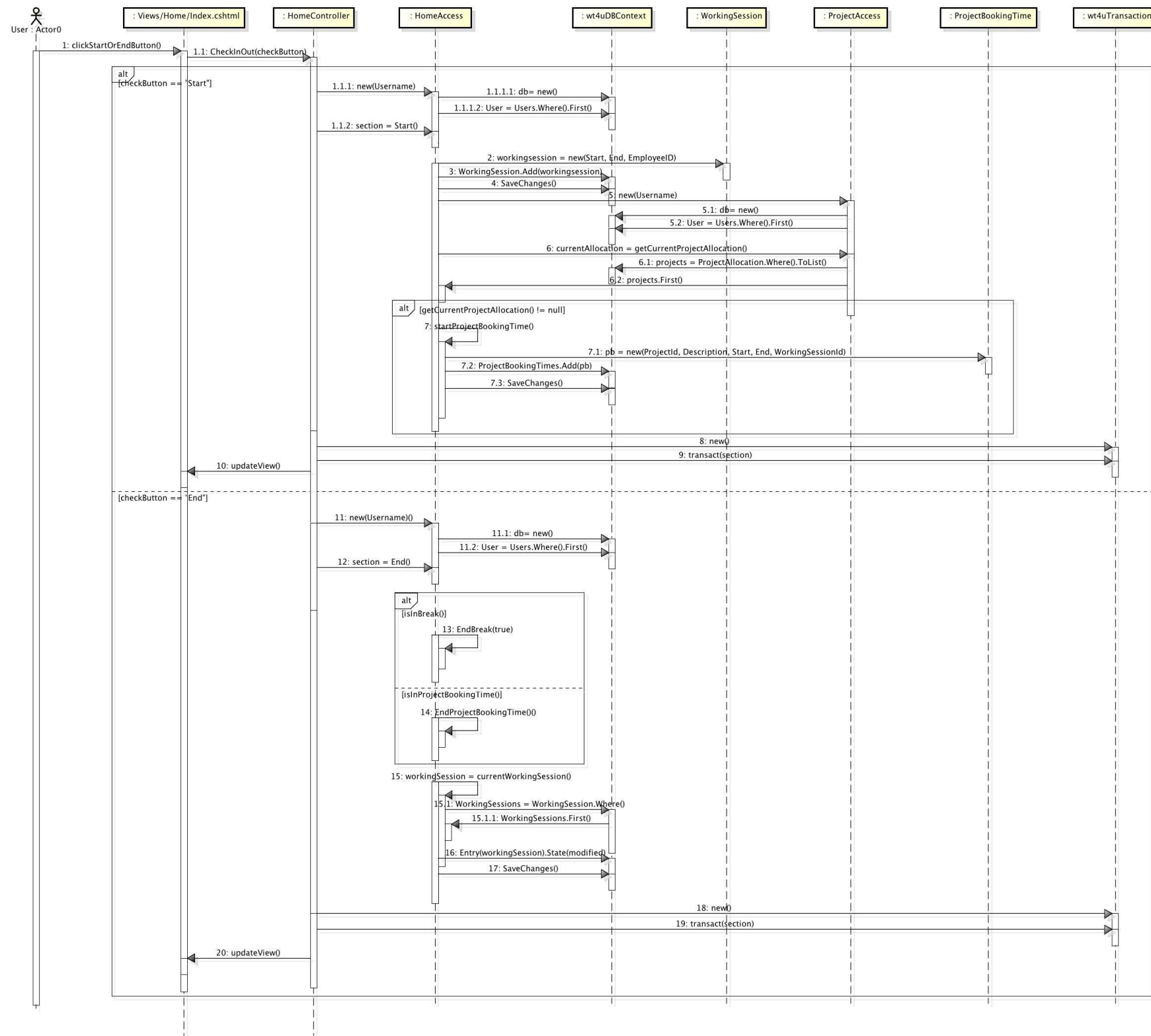
- Die Access-Klassen haben Abhängigkeiten auf die Models, dort werden entweder Werte der Entities ausgelesen oder geändert (change/query).

5.4.5 Wichtige interne Abläufe

Innerhalb des Package BusinessLogic gibt es keine internen Abläufe. Es existieren nur sehr wenige Querverbindungen zwischen den Klassen, diese wurden bewusst minimal gehalten.

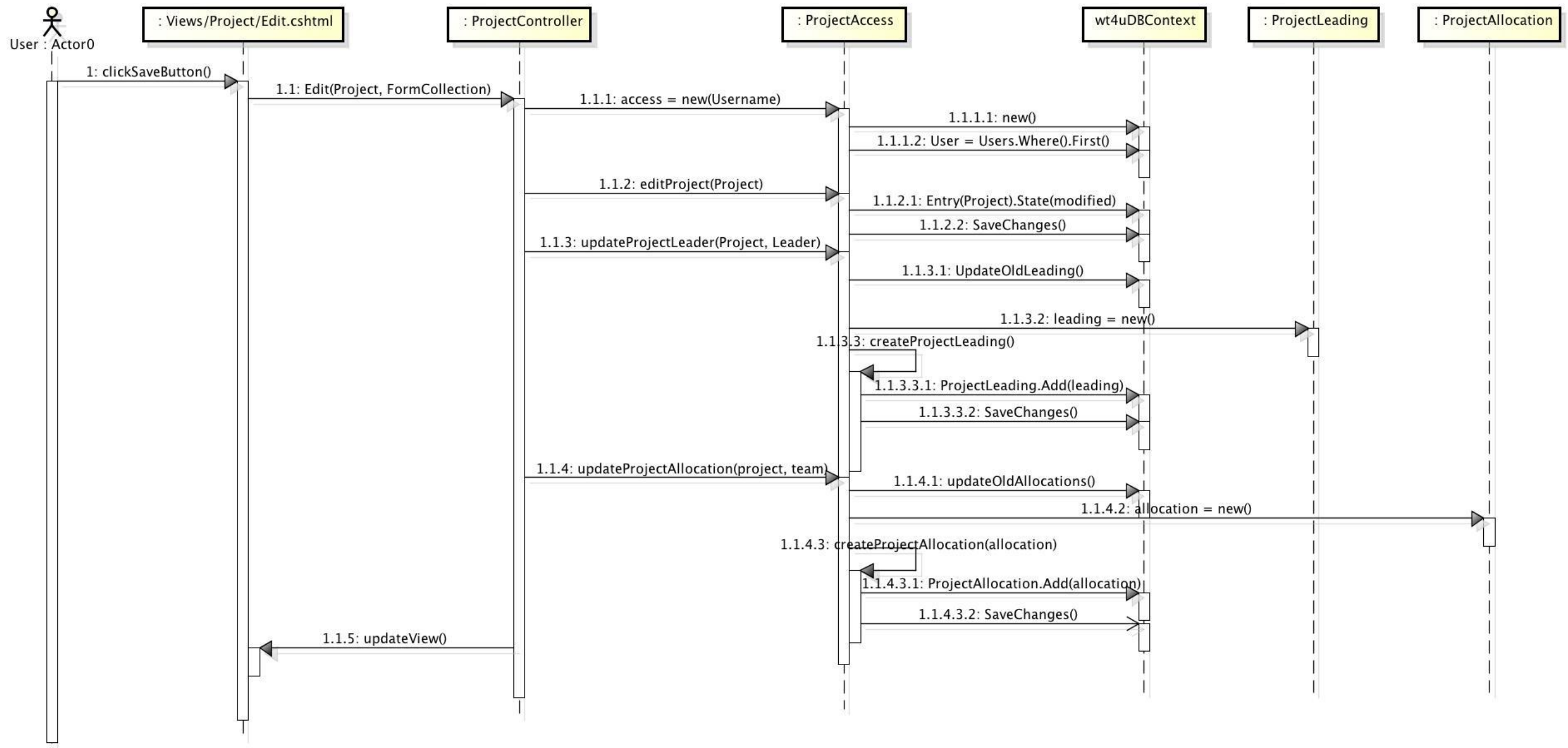
6. Use Cases / Prozesse

6.1 Check In/Out



powered by Astah

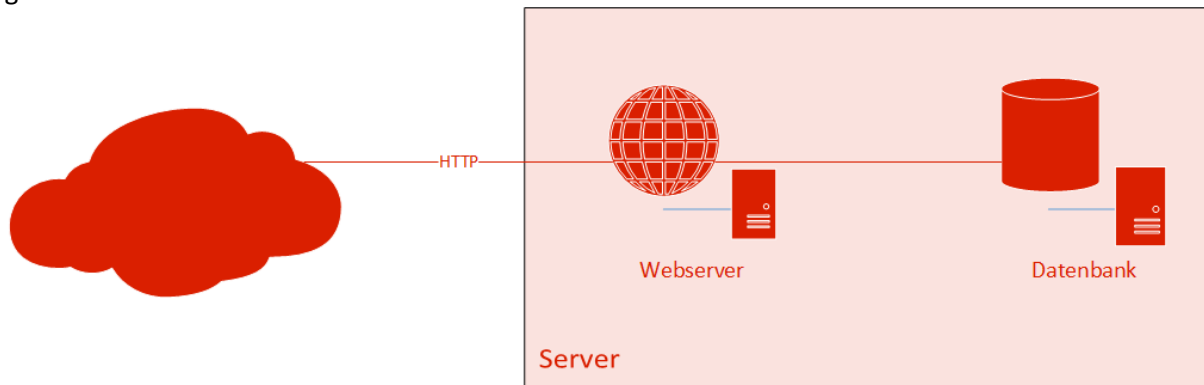
6.2 Edit Project



7. Deployment

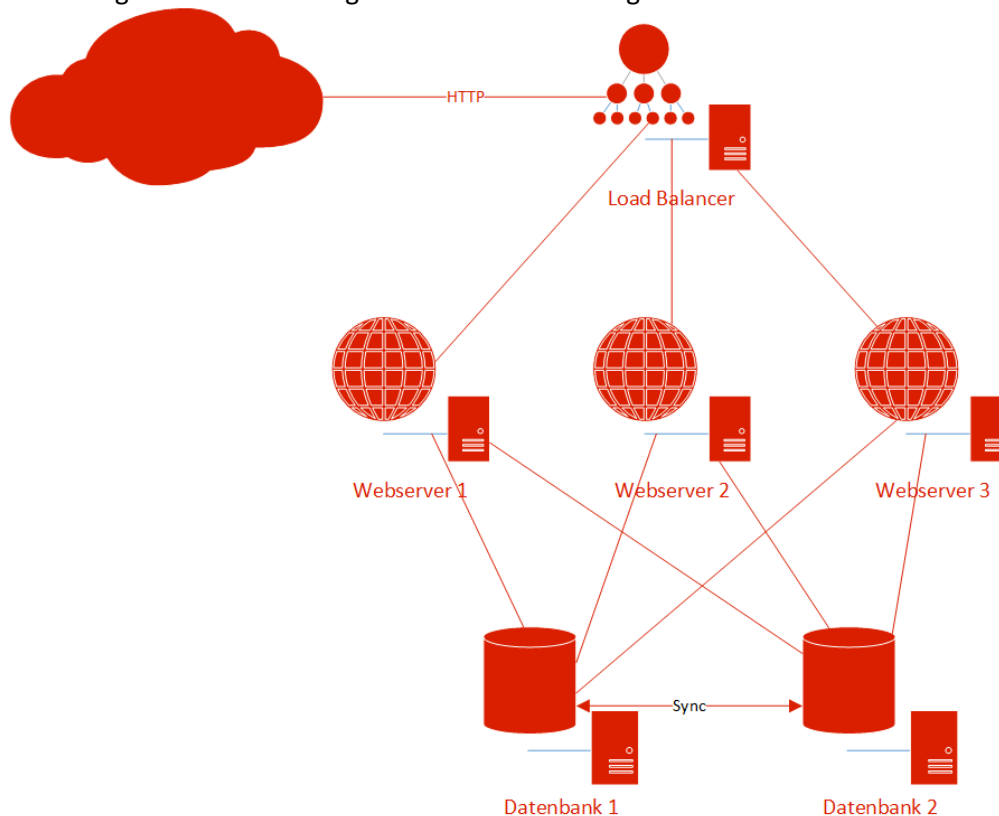
7.1 Ist

Für dieses Projekt laufen die Datenbank und der Webserver auf einem Server. Dies stellt die Testumgebung dar und soll bei einem produktiven Betrieb abgelöst werden. Für die Dauer des Projektes ist die aktuelle Umgebung ausreichend. Wird wt4u jedoch vollumfänglich eingesetzt, so muss die Infrastruktur des Soll-Szenarios eingesetzt werden, damit die Verfügbarkeit von wt4u gewährleistet ist.



7.2 Soll

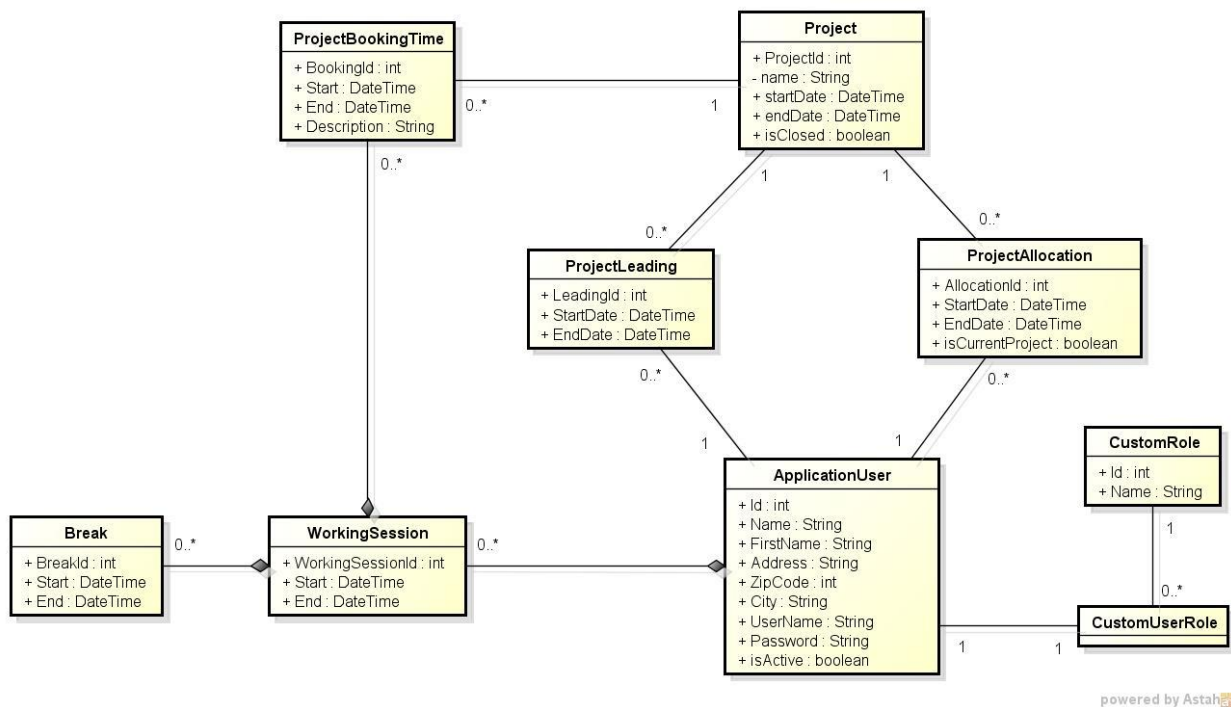
Es soll mit einem vorgeschalteten Load Balancer die Last auf verschiedene Server verteilt werden. Auf diesen verteilten Server läuft jeweils ein Webserver, welcher die Anfragen entgegennimmt und verarbeitet. Jeder Webserver greift entsprechend auf die Datenbank zu. Mittels Datenbank Clustering soll zudem der Zugriff auf die Datenbank gewährleistet werden.



8. Datenspeicherung

Die Daten (z.B. Projekte, Arbeitszeiten, Mitarbeiter) werden in einer zentralen MS SQL Datenbank gehalten. Für die Entwicklung wurde eine separate Test Datenbank eingerichtet. Diese ist ein Abbild der produktiven Datenbank und besitzt somit dasselbe Schema. Damit ist gewährleistet, dass an der produktiven Datenbank keine Schäden entstehen.

8.1 Klassenmodell



8.2 Schnittstelle

Der Zugriff auf die Datenbank erfolgt direkt aus der in ASP.NET geschriebenen Applikation wt4u. Dabei kann der entsprechende Context in der Datei Web.config angepasst werden. Über diese kann direkt auf die Daten der Datenbank zugegriffen werden. Der Context hat den Namen wt4uDBContext und alle benötigten Informationen (DB, User, PW, etc.) sind im Connection String hinterlegt.

Connection String:

```

add name="wt4uDBContext" connectionString="Data Source=152.96.56.77\SQLExpress,40001;
Database=wt4uTest;User Id=wt4u;Password=***;" providerName="System.Data.SqlClient" />

```

9. Grössen und Leistung

Bei der Planung von wt4u wurde geachtet, dass eine Skalierung schnell und einfach durch weitere Webserver möglich ist. Zudem besteht die Möglichkeit für die Datenbank ein Clustering einzurichten.

9.1 Webserver

Für den Kunden das Wichtigste ist die Antwortzeit bei der Bedienung der Applikation. Dabei muss der Webserver innert nützlicher Frist eine Response auf einen HTTP Request zurückgeben. Damit dies immer gewährleistet wird, können leistungsstärkere Server oder Load Balancing eingesetzt werden. Dies ist dem Kunden jeweils selber überlassen und hängt von der Anzahl der Mitarbeiter ab.

9.2 Datenbank

Da aktuell MS SQL Server Express eingesetzt wird, bringt dies einige Einschränkungen mit sich. Darin können maximal 10 GB an Daten gespeichert werden. Zudem bringt die Express Version einige Einschränkungen in Sachen Performance mit sich (nur 1 Prozessor und 1GB RAM verwendet). Es kann allerdings ohne Weiteres eine andere Datenbank eingesetzt werden, damit diese Einschränkungen nicht mehr vorhanden sind. In diesem Fall kann eine Verbesserung der Hardware Sinn machen, damit der Zugriff auf die Datenbank beschleunigt wird.