HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT
FÜR
SOFTWARE

# Design of a .NET Parallelization as a Service Portal

## Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2014


Autoren: Jan Balmer und Adrian Rieser
Betreuer: Prof. Dr. Luc Bläser

# Inhaltsverzeichnis

# Abstract

Das Projekt „.NET Parallelization as a Service“ verfolgt den Ansatz, parallele Rechenleistung über einen Web-Service zur Verfügung zu stellen, so dass diese nahtlos in .NET-Programmen genutzt werden kann. Dafür existiert bereits ein Laufzeitsystem, welches gewöhnliche .NET Parallel Tasks automatisch über diesen Service auf einem Micorosoft HPC Cluster verteilt und so mit einer hohen Anzahl Cores beschleunigt. Dafür soll nun in dieser Arbeit ein Web-Portal entwickelt werden, welches Benutzern auf einfache Weise den Zugang und die Verwendung dieses Cloud Dienstes ermöglicht.

Dafür haben wir das „Parallelization as a Service Portal" entwickelt. Neben der automatischen Registrierung für neue Benutzer regelt dieses Portal den Zugriff auf den Web Dienst und kontrolliert die benutzte parallele Rechenleistung pro Benutzer. Mittels eines Kostenmodells (Kosten pro Nutzer = Summe der Rechendauer aller benutzten Cores für Task-Ausführung eines Nutzers) wird zudem die Beanspruchung gemessen und kann mit Quotas pro Benutzer limitiert werden. Die Architektur besteht aus drei Komponenten: (1) Web Portal für die Benutzerverwaltung, Service Client Download und interaktive Task-Verwaltung als ASP.NET MVC, (2) Erweiterung des Web Services in WCF für das Parallel Task Dispatching von Client Libraries mit Quota-Kontrolle, (3) System-Monitoring Systemdienst (Windows Service) zur asynchronen Überwachung verwaister Tasks und Quoteneinhaltung.

Das realisierte System ist funktionsfähig und bildet die Basis für ein mögliches Angebot des Parallelisierungsdienstes an einen offenen Benutzerkreis. Die Funktionstüchtigkeit und Robustheit wurde mit dem HSR Cluster im Backend validiert. Im Hinblick einer kommerziellen Vermarktung der „Cloud Task Parallelisierung“ müsste unsere Lösung in Zukunft nur noch um einen Bezahlmechanismus ergänzt werden.

# Aufgabenstellung

Unsere Aufgabe war es, eine Webapplikation für die Verwendung des „CloudTaskParallelism“ zu entwickeln. Die Benutzer sollen imstande sein, sich über den Service zu informieren und ihnen eine einfache Verwendung des Client.Runtime zu ermöglichen. Ausserdem soll die Applikation eine Benutzerverwaltung zur Verfügung stellen.

Es soll den Benutzern ermöglichen, leicht eine Übersicht über ihre Tasks zu erhalten und eine einfache Verwaltung von diesen Tasks zu ermöglichen.

Es sollen gewisse Beschränkungen und Kontrollen der Benutzer eingerichtet werden, um eine Ausnutzung der HPC Cluster-Infrastruktur durch die Benutzer zu verhindern.

Es sollte ausserdem darauf geachtet werden, eine mögliche spätere Kommerzialisierung zu ermöglichen. Die verbrauchte Rechenleistung der Benutzer sollte deshalb erfasst werden und durch eine Formel in Kosten bzw. Credits umgewandelt werden.

**Folgende spezifische Ziele wurden vorgegeben:**

- Aufnahme der Anforderungen für das „Parallelization as a Service Portal“ und das zugehörige Laufzeitsystem
- Architektur des Service Portal mit den oben genannten Aspekten.
- Entwurf und Implementierung des Web Portals (basierend auf C#, ASP.NET MVC) und Ausbau des Web Service (WCF).
- Validierung mit automatischen Unit und Integrationstest bzw. allenfalls manuellen Systemtests.
- Optional: Ausbau des unterstützten Feature Sets der .NET Programmierkonstrukte für den bestehenden Laufzeitprototypen.

# Eigenständigkeitserklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- das ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum: Rapperswil, 30.5.2014

Name, Unterschrift: Jan Balmer,

Ort, Datum: Rapperswil, 30.5.2014

Name, Unterschrift: Adrian Rieser,

# Anforderungsspezifikationen

## Aktorenbeschreibung

| Akteur | Akteur-Typ | Ziele |
|---|---|---|
| Benutzer | Primary | • Der Benutzer will sich registrieren können.<br>• Der Benutzer will sich anmelden können.<br>• Der Benutzer will mit möglichst wenig Aufwand seine Tasks verteilt und parallel ausführen lassen. |
| Administrator | Primary | • Der Administrator will die anderen Benutzer des Parallelization as a Service Portal überwachen um sicherstellen, dass niemand den Service zu stark in Beanspruchung nimmt. |

## Verwendete Systeme

| System | Verwendung |
|---|---|
| Microsoft HPC Cluster | • Wir verwenden die Verteilung der Tasks sowie die grosse Rechenleistung des Microsoft HPC Clusters. |

## Use Cases im “brief”-Format

| | |
|---|---|
| **UC 01** | **Registrieren** |
| | Ein neuer Benutzer gibt seine Kundeninformationen ein. Die Informationen werden vom System überprüft und der Kunde wird gegebenenfalls auf fehlende Informationen hingewiesen. Nach erfolgreicher Überprüfung wird der Benutzer freigeschalten. Der Kunde erhält vom System daraufhin eine Registrationsbestätigung. |
| **UC 02** | **Login** |
| | Nach erfolgreichem registrieren können sich Benutzer beim Parallelization as a Service Portal anmelden. |
| **UC 03** | **Herunterladen des Serviceclients** |
| | Nach erfolgreichem anmelden kann der Serviceclient heruntergeladen werden. Der Benutzer erhält Informationen über die Verwendung und Konfiguration des Serviceclients. |
| **UC 04** | **Verwalten der eigenen Tasks** |
| | Nach erfolgreichem anmelden kann der Benutzer seine bisherigen und aktuellen Jobs betrachten. Die eigenen laufenden Tasks können vom Benutzer beendet werden. |
| **UC 05** | **Verwalten aller Tasks** |
| | Der Administrator erhält Informationen über alle laufenden und bisherigen Tasks die von Benutzern des Parallelization as a Service Portals ausgeführt wurden. Die aktuell laufenden Tasks können beendet werden. |
| **UC 06** | **Sperren/ entsperren von Benutzern** |
| | Der Administrator kann die Benutzer verwalten und falls nötig sperren bzw. entsperren. Es können zudem Infos über Rechenzeiten der einzelnen Benutzer eingesehen werden. |

## Change Cases

| CC 01 | Payment |
|---|---|
| | Es soll zusätzlich eine Verrechnungs- und Bezahlmöglichkeit von den benützten Ressourcen eingeführt werden. |

## Use-Case Diagramm

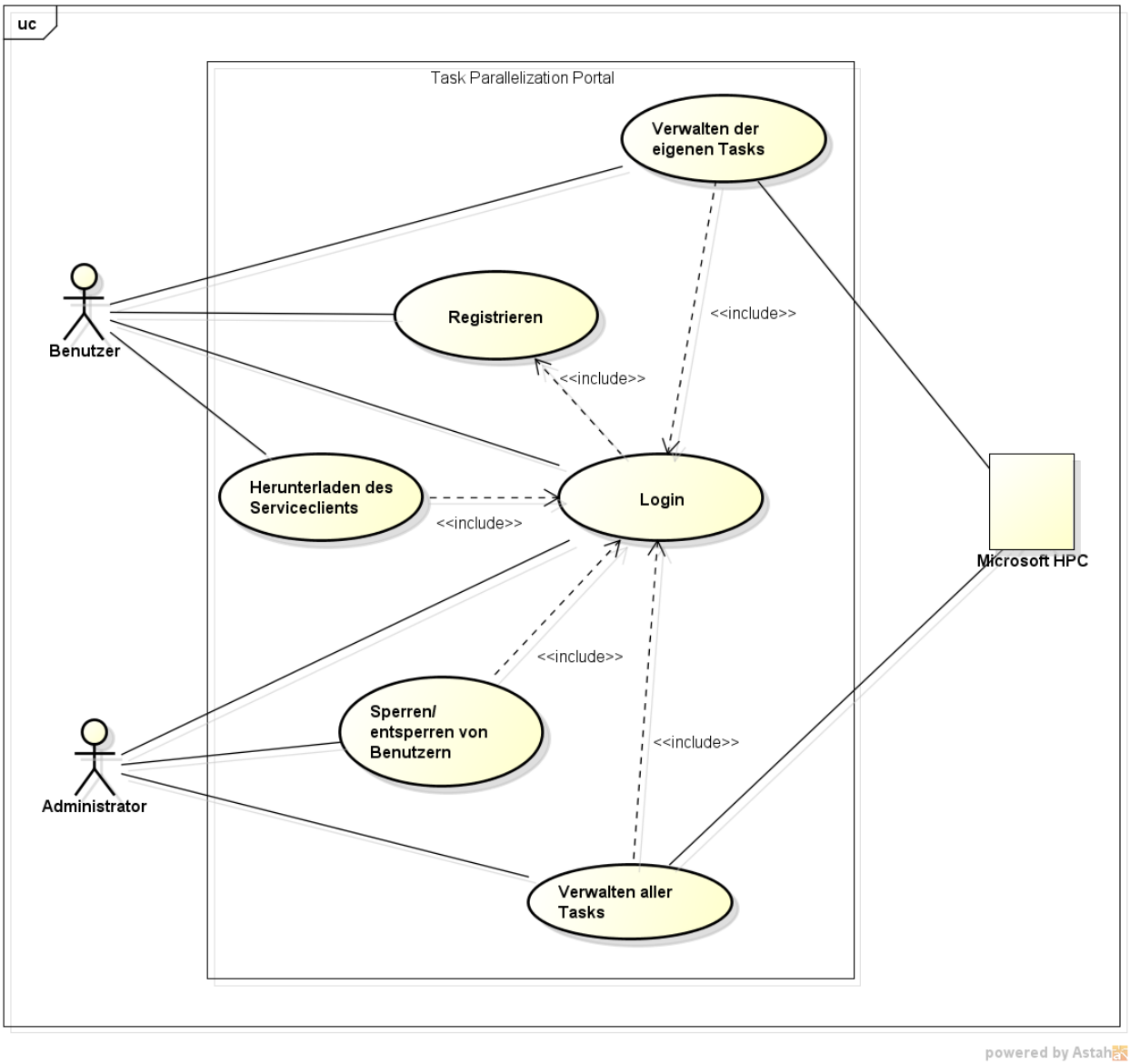



*Abbildung 1: Use-Case Diagramm*

## Use-Cases im “fully dressed”-Format

### UC 01 Registrieren

| | |
|---|---|
| **Use Case ID** | **01** |
| **Use Case Name** | **Registrieren** |
| **Overview** | Ein neuer Benutzer gibt seine Kundeninformationen ein. Die Informationen werden vom System überprüft und der Kunde wird gegebenenfalls auf fehlende oder fehlerhaft eingegebene Informationen hingewiesen. Der Kunde erhält vom System daraufhin eine Registrationsbestätigung. Nach erfolgreicher Freischaltung wird der Benutzer aktiviert. |
| **Stakeholder and Interests** | Benutzer: Schnelle Reaktionszeit |
| **Preconditions** | Der Benutzer ist noch nicht für das Parallelization as a Service Portal registriert. |
| **Postconditions** | Der Benutzer ist für das Parallelization as a Service Portal registriert und aktiviert. |
| **Format** | Fully dressed |
| **Main Success Scenario** | 1. Der Benutzer gibt seine Kontaktdaten (Name, Adresse und E-Mail-Adresse) ein.<br>2. Das System validiert die Eingaben.<br>3. Das System sendet eine Aktivierungs-E-Mail an die angegebene E-Mail-Adresse.<br>4. Der Benutzer benutzt den Link im Aktivierungs-E-Mail.<br>5. Das System aktiviert den Benutzer. |
| **Extensions (or Alternative Flows):** | 2.a) Das System findet fehlerhafte Eingaben: Es wird eine Fehlermeldung dargestellt welche dem Benutzer zeigt, wo er einen Fehler gemacht hat. |
| **Special Requirements:** | |
| **Technology and Data Variations List:** | |
| **Frequency of Occurence:** | 1 Mal pro Benutzer |
| **Open Issues:** | |
| **Bemerkungen** | |

## UC 02 Login

| | |
|---|---|
| **Use Case ID** | **02** |
| **Use Case Name** | **Login** |
| **Overview** | Nach erfolgreichem registrieren können sich Benutzer beim Parallelization as a Service Portal anmelden. |
| **Stakeholder and Interests** | Benutzer: Schnelle Reaktionszeit |
| **Preconditions** | Der Benutzer ist registriert und aktiviert. |
| **Postconditions** | Der Benutzer hat den Serviceclient heruntergeladen. |
| **Format** | Fully dressed |
| **Main Success Scenario** | 1. Der Benutzer gibt seine Logindaten (E-Mail-Adresse und Passwort) ein.<br>2. Das System validiert die Eingaben.<br>3. Das System zeigt dem Benutzer die nächsten möglichen Schritte an. |
| **Extensions (or Alternative Flows):** | 2.a) Das System findet fehlerhafte Eingaben: Es wird eine Fehlermeldung dargestellt welche dem Benutzer zeigt, wo er einen Fehler gemacht hat.<br>3.a) Das System hat keinen Benutzer der mit der eingegebenen E-Mail-Adresse oder dem eingegeben Passwort übereinstimmt. |
| **Special Requirements:** | |
| **Technology and Data Variations List:** | |
| **Frequency of Occurence:** | Je nach Benutzung des jeweiligen Benutzers:<br>Schätzungsweise maximal 5x pro Tag pro Benutzer |
| **Open Issues:** | |
| **Bemerkungen** | |

## UC 03 Herunterladen des Serviceclients

| | |
|---|---|
| **Use Case ID** | **03** |
| **Use Case Name** | **Herunterladen des Serviceclients** |
| **Overview** | Nach erfolgreichem anmelden kann der Serviceclient heruntergeladen werden. Der Benutzer erhält Informationen über die Verwendung und Konfiguration des Serviceclients. |
| **Stakeholder and Interests** | Benutzer: Schnelles herunterladen des Serviceclients |
| **Preconditions** | Der Benutzer ist eingeloggt. |
| **Postconditions** | Der Benutzer hat das Serviceclient heruntergeladen. |
| **Format** | Fully dressed |
| **Main Success Scenario** | 1. Der Benutzer wählt den Link im Parallelization as a Service Portal.<br>2. Der Benutzer lädt den Serviceclient herunter. |
| **Extensions (or Alternative Flows):** | |
| **Special Requirements:** | |
| **Technology and Data Variations List:** | |
| **Frequency of Occurence:** | Schätzungsweise 1 Mal pro Benutzer |
| **Open Issues:** | |
| **Bemerkungen** | |

## UC 04 Verwalten der eigenen Tasks

| | |
|---|---|
| **Use Case ID** | **04** |
| **Use Case Name** | **Verwalten der eigenen Tasks** |
| **Overview** | Nach erfolgreichem anmelden kann der Benutzer seine bisherigen und aktuellen Jobs betrachten. Die eigenen laufenden Tasks können vom Benutzer beendet werden. |
| **Stakeholder and Interests** | Benutzer: Schnelle Reaktionszeit |
| **Preconditions** | Der Benutzer ist eingeloggt. |
| **Postconditions** | Falls der Benutzer seine Tasks abgebrochen hat: Tasks sind abgebrochen. |
| **Format** | Fully dressed |
| **Main Success Scenario** | 1. Der Benutzer wählt den Link im Parallelization as a Service Portal.<br>2. Der Benutzer sieht eine Übersicht über seine eigenen Tasks.<br>3. Der Benutzer kann seine eigenen Tasks abbrechen. |
| **Extensions (or Alternative Flows):** | |
| **Special Requirements:** | |
| **Technology and Data Variations List:** | |
| **Frequency of Occurence:** | Je nach Benutzung des jeweiligen Benutzers:<br>Schätzungsweise maximal 5x pro Tag pro Benutzer |
| **Open Issues:** | |
| **Bemerkungen** | |

## UC 05 Verwalten der eigenen Tasks

| | |
|---|---|
| **Use Case ID** | **05** |
| **Use Case Name** | **Verwalten aller Tasks** |
| **Overview** | Der Administrator erhält Informationen über alle laufenden und bisherigen Tasks die von Benutzern des Parallelization as a Service Portal ausgeführt wurden. Die aktuell laufenden Tasks können beendet werden. |
| **Stakeholder and Interests** | Benutzer: Schnelle Reaktionszeit |
| **Preconditions** | Der Benutzer ist als Administrator eingeloggt. |
| **Postconditions** | Falls der Administrator Tasks abgebrochen hat: Tasks sind abgebrochen. |
| **Format** | Fully dressed |
| **Main Success Scenario** | 1. Der Administrator wählt den Link im Parallelization as a Service Portal.<br>2. Der Administrator sieht eine Übersicht über alle Tasks die vom Parallelization as a Service Portal übermittelt worden sind.<br>3. Der Administrator kann Tasks abbrechen. |
| **Extensions (or Alternative Flows):** | |
| **Special Requirements:** | |
| **Technology and Data Variations List:** | |
| **Frequency of Occurence:** | Je nach Benutzung des jeweiligen Administrators:<br>Schätzungsweise maximal 5x pro Tag pro Administrator |
| **Open Issues:** | |
| **Bemerkungen** | |

## UC 06 Verwalten der eigenen Tasks

| | |
|---|---|
| **Use Case ID** | **06** |
| **Use Case Name** | **Sperren/ entsperren von Benutzern** |
| **Overview** | Der Administrator kann die Benutzer verwalten und falls nötig sperren bzw. entsperren. Es können zudem Infos über Rechenzeiten der einzelnen Benutzer eingesehen werden. |
| **Stakeholder and Interests** | Administrator: Schnelle Reaktionszeit |
| **Preconditions** | Der Benutzer ist als Administrator eingeloggt. |
| **Postconditions** | Falls Benutzer gesperrt/entsperrt wurden: Benutzer sind gesperrt/entsperrt. |
| **Format** | Fully dressed |
| **Main Success Scenario** | 1. Der Administrator wählt den Link im Parallelization as a Service Portal.<br>2. Der Administrator sieht eine Übersicht über alle Benutzer.<br>3. Der Administrator kann Benutzer sperren und gesperrte Benutzer wieder entsperren. |
| **Extensions (or Alternative Flows):** | |
| **Special Requirements:** | |
| **Technology and Data Variations List:** | |
| **Frequency of Occurence:** | Je nach Benutzung des jeweiligen Administrators:<br>Schätzungsweise maximal 5x pro Tag pro Administrator |
| **Open Issues:** | |
| **Bemerkungen** | |

## Non-functional requirements

| NFR Nr. | Beschreibung |
|---|---|
| NFR 01 | Der Zugriff auf die Benutzerverwaltung und die Festlegung der Quotas hat nur der Administrator und können nur von diesem geändert werden. |
| NFR 02 | Der Monitor, der die Einhaltung der Quotas überwacht, überprüft diese in einer konfigurierbaren Periode. |
| NFR 03 | Die Quotas verhindern eine Überlastung der HPC Cluster-Infrastruktur durch einen Benutzer. |
| NFR 04 | Alle übermittelten Tasks der Benutzer sollen in einer Separaten vom HPC Cluster unabhängigen Datenbank gespeichert werden, um bei Datenverlust des HPC Clusters eine Nachvollziehung der Kosten zu ermöglichen. |
| NFR 05 | Der ConnectionString für die Verbindung zur Datenbank, sowie die Credentials für die Verbindung zum HPC Cluster sind konfigurierbar. |
| NFR 06 | Beim Download des ServiceClients erhält der User eine für ihn konfigurierte Version des Clients. |
| NFR 07 | Das Passwort der User kann nicht in Klartext aus der Datenbank ausgelesen werden. |
| NFR 08 | SQL injection und Cross Site Scripting wird verhindert |
| NFR 09 | Es wird überprüft, ob der User eine gültige E-Mail Adresse angegeben hat und auf diese zugreifen kann. |
| NFR 10 | Das Design ist einheitlich und ist einfach zu erweitern und abzuändern. |

# Domainanalyse

## Domainmodell

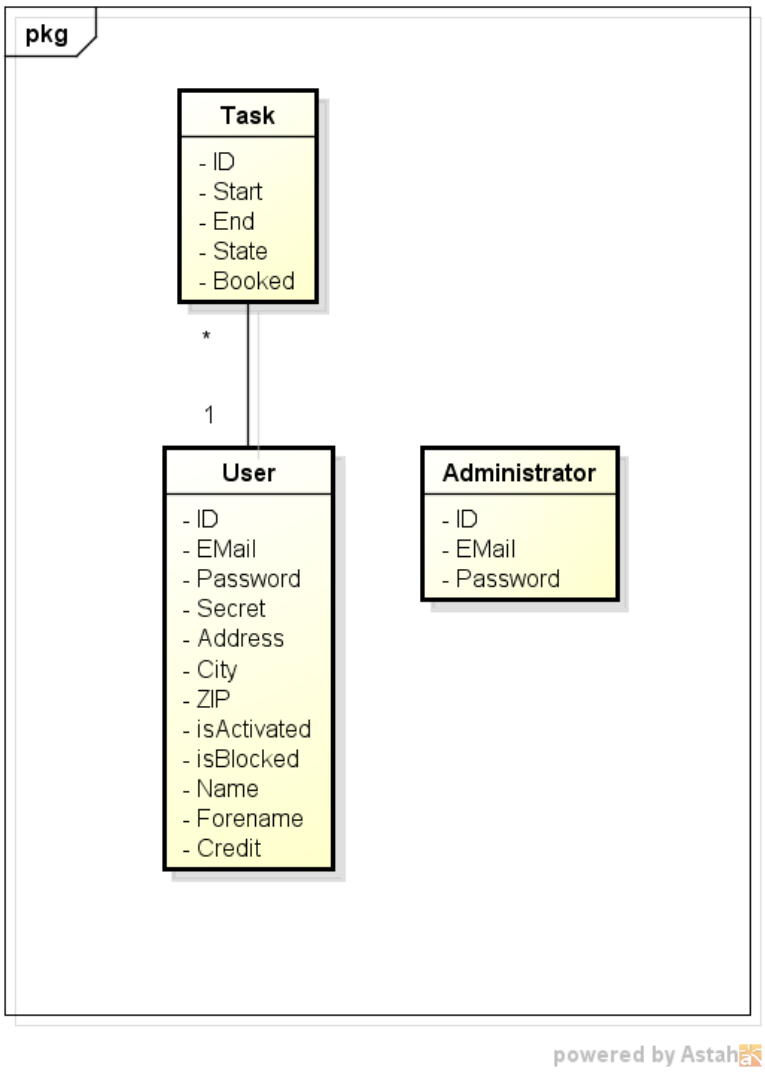



*Abbildung 2: Domainmodell*

## Beschreibung der Entitäten

### User

Der User stellt den Benutzer dar. Er wird beim Registrieren erstellt und nach dem Verifizieren aktiviert.

### Administrator

Er stellt den Administrator dar. Er ist ein Benutzer, welchem besondere Rechte zugewiesen werden. Er kann neben dem Verwalten der eigenen Tasks auch die Tasks von den anderen Usern verwalten.

### Task

Er stellt einen Job des Microsoft HPC Cluster auf der Ebene des „Parallelization as a Service Portal“ dar.

## Klassen- und Tabellendiagramm

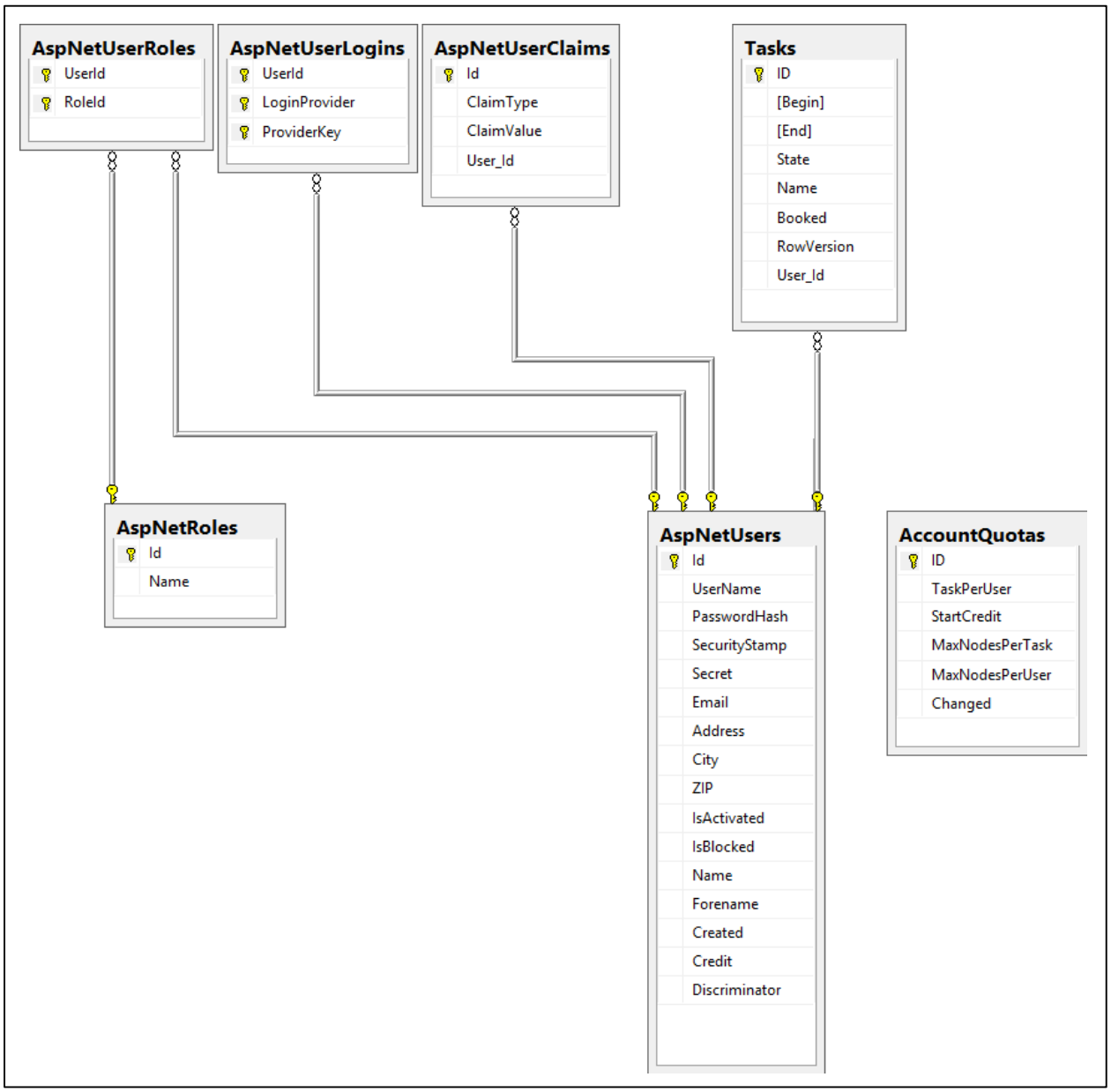



*Abbildung 3: Klassen- und Tabellendiagramm*

## Unterschiede zum Domainmodell

- AspNetUserRoles, AspNetUserLogins, AspNetUserClaims und AspNetRoles wurden durch das ASP.NET MVC IdentityUser-Framework generiert.
- AspNetUsers repräsentieren unsere User, welcher einer davon den vorgegebenen Namen „Administrator“ besitzt und somit den Administrator repräsentiert.
- Auf Tasks wird eine RowVersion geführt, um parallele Zugriffe auf die Zeile zu unterstützen.
- AccountQuotas wurden bei Beginn der Planung nicht als Domainklasse modelliert.

# System Architektur

## Systemdiagramm

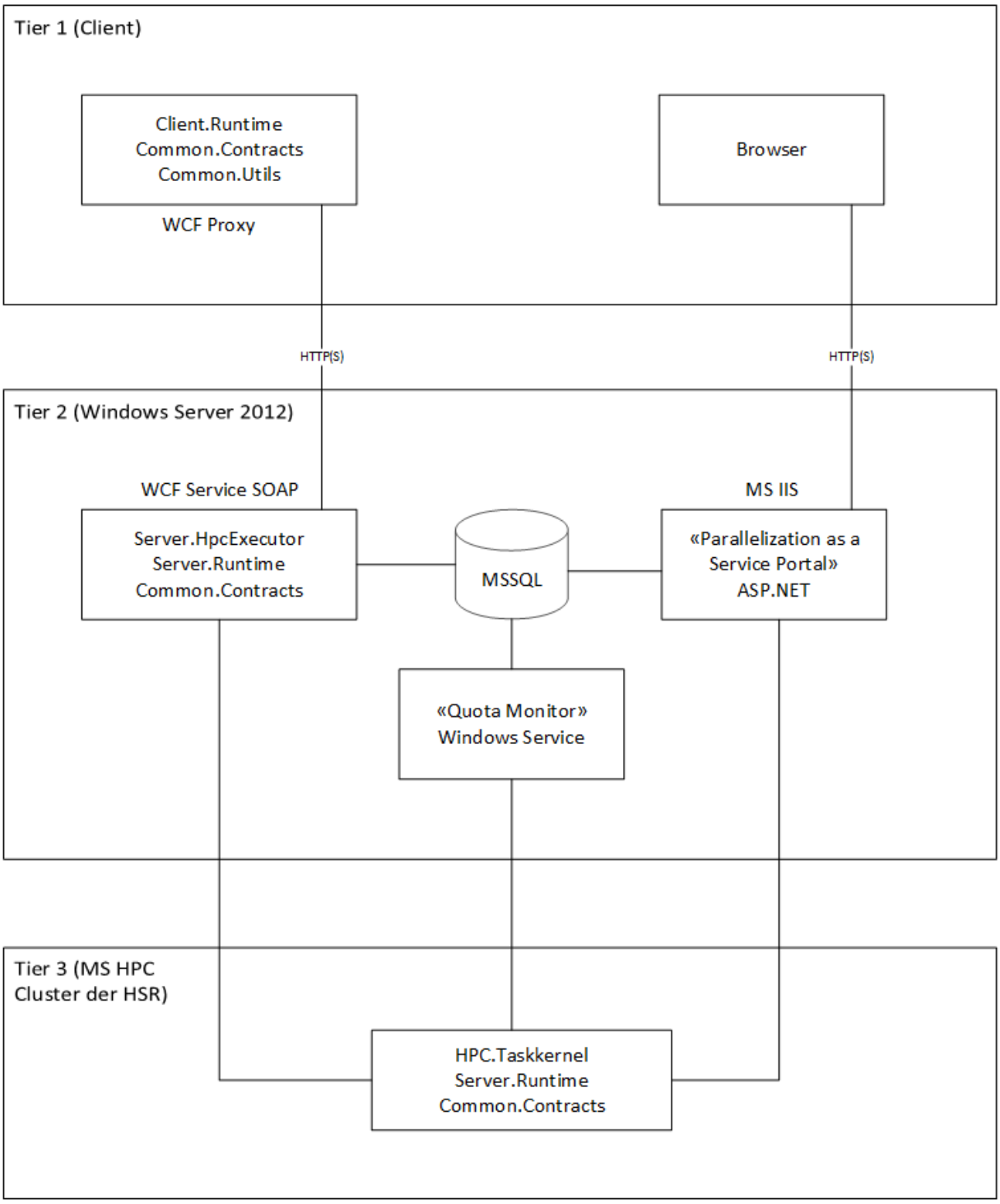



*Abbildung 4: Systemdiagramm*

## Beschreibung der Tiers

### Tier 1

Der Client ist das erste Tier. Auf diesen Geräten läuft via Browser die Kommunikation mit der Website und mit den Client.Runtime-Librarys die Kommunikation mit dem WCF Service in Tier 2.

### Tier 2

Tier 2 ist ein Microsoft Windows Server 2012 mit folgenden Komponenten:

#### *Webserver (Microsoft IIS)*

Der MS IIS wird verwendet, um einerseits unsere ASP.NET Website zu hosten und andererseits den WCF Service für das Parallel Task Dispatching von den Client Libraries anzubieten.

#### *Datenbankserver (Microsoft SQL Server)*

Um unsere Benutzer- und Taskinformationen persistent zu speichern, wird ein MSSQL Server eingesetzt. Zusätzlich werden noch die vordefinierten AccountQuotas darin gespeichert.

#### *Quota Monitor (Windows Service)*

Um die kontinuierliche asynchrone Überwachung der Tasks bzw. der Quotas sicherzustellen, läuft auf demselben System noch ein Windows Service der in einem festgelegten Zeitintervall die Werte überprüft.

### Tier 3

Das dritte Tier ist ein Microsoft HPC Cluster der HSR. Auf dem Cluster läuft ausserdem die Server.Runtime des WCF Service.

# Software Architektur

## Solution „SA_Parallelization“

Siehe Abbildung 5: Übersicht „SA_Parallelization“ im Solution Explorer.

### Project „ParallelizationPortal“

Dieses Project ist das eignetliche Webportal mit den Controller-, Model-, Utility-, Viewklassen und Ressourcen.

### Project “ParallelizationPortal_UnitTest”

In diesem Project sind die UnitTests für die Utilityklassen untergebracht.

### Project ParallelizationPortalMonitor

Dieses Project enthält den Code für den zu installierenden Quota Monitor als Windows Service.

### Project ParallelizationPortalUtility

Hier sind statische Utilityklassen für DB-, HPC-Zugriff und für das Hashing zu finden, welche in mehreren Projects und auch in der Solution „HSR.CloudTaskParallelism“ verwendet werden.

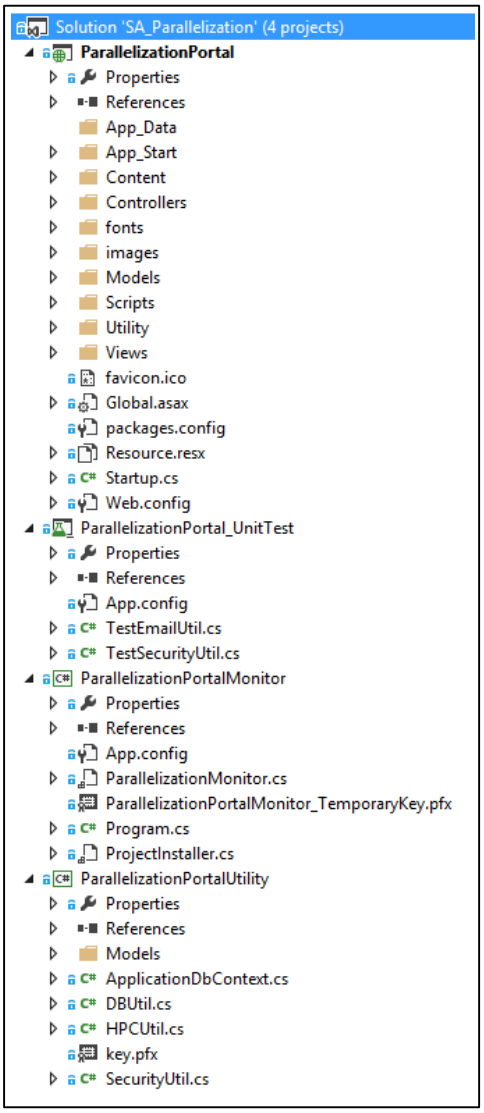



*Abbildung 5: Übersicht „SA_Parallelization“ im Solution Explorer*

## Solution „HSR.CloudTaskParallelism“

Siehe Abbildung 6: Übersicht „HSR.CloudTaskParallelism“ im Solution Explorer.

### Project „HSR.CloudTaskParallelism.Server.HpcExecutor“

Die einzige Anpassung unsererseits in dieser Solution ist im TaskExecutionService zu finden. Bei der Übermittlung eines neuen Tasks wird mit dem übermittelten Secret die Datenbank abgefragt um einerseits sicherzustellen, dass ein Benutzer mit diesem Secret existiert und andererseits genügend Credits vorhanden sind.

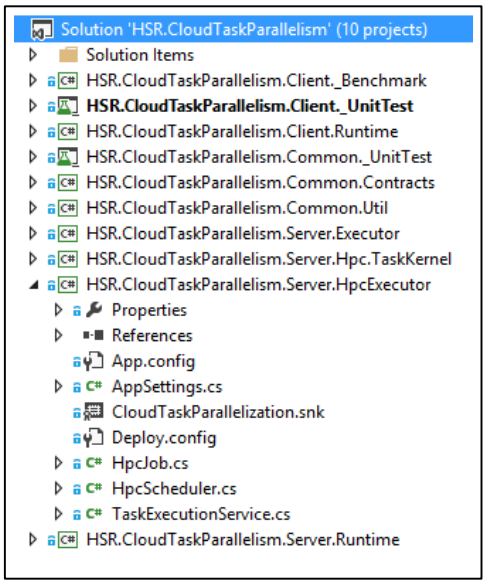



*Abbildung 6: Übersicht „HSR.CloudTaskParallelism“ im Solution Explorer*

# Designentscheide

## Frameworks

### *Entity Framework (Code First)*

Wir haben uns für die Verwendung des Entity Framework entschieden, da dies eine gute Möglichkeit für die Umsetzung von Code First bietet. Der Umgang mit Änderungen am Domainmodel und den damit verbunden Code wurde somit stark verbessert und automatisiert. Es ist einfach zu erlernen und bietet eine schnelle Entwicklung durch die Automatische Umsetzung der CRUD Operationen.

Es erlaubt die Verwendung von simplen Business Objects wie anhand unseres Beispiels von einem Task:

```
7 references
public class Tasks
{
    1 reference
    public int ID { get; set; }
    1 reference
    public DateTime? Begin { get; set; }
    1 reference
    public DateTime? End { get; set; }
    3 references
    public string State { get; set; }
    9 references
    public string Name { get; set; }
    4 references
    public int Booked { get; set; }
    5 references
    public virtual ApplicationUser User { get; set; }
```

*Abbildung 7: Task Class*

Durch diese simple Klasse wird automatisch eine Tabelle angelegt, inklusive den benötigten Einstellungen für ID und Foreigen-Keys. Ausserdem wird automatisch Lazy-Loading für den ApplicationUser verwendet.

### *ASP.NET MVC Framework*

Mit dem Entity Framework hatten wir auch einen leichten Zugang zu der eingebauten Userverwaltung von ASP.NET MVC und den damit schon abgedeckten Sicherheitsmassnahmen. ASP.NET MVC wurde bereits schon in der Aufgabenstellung gefordert und bietet uns eine gute Plattform um eine Webanwendung mit den drei Rollen von Model, View und Controller zu entwickeln.

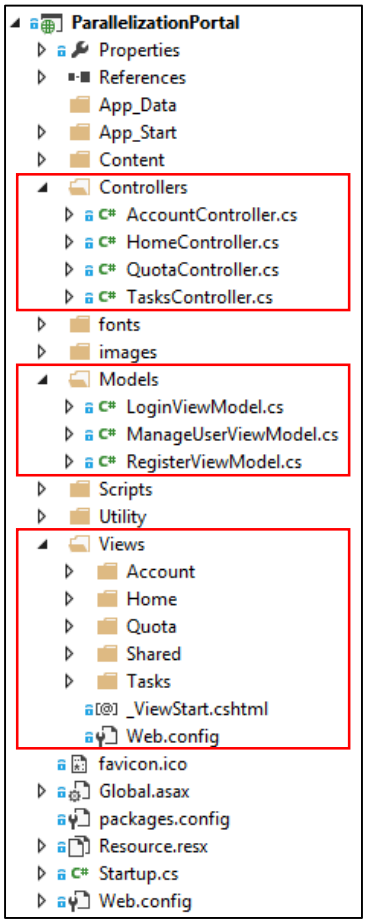



*Abbildung 8: ASP.NET MVC sichtbar in der Projektstruktur*

## Designentscheide auf Tier 1

### *Designentscheide in der Komponente „Client Runtime“*

#### Anmelde- und Identifikationsprozess

Um das Parallelization as a Service Portal verwenden zu können, muss ein Benutzerkonto erstellt werden. Um sicherzustellen dass eine gültige E-Mail-Adresse angegeben wurde, versendet das System nach der Registration eine E-Mail mit Bestätigungs-Link, welcher einem Benutzer zugewiesen werden kann. Danach ist der Benutzer freigeschaltet und kann den Service-Client herunterladen.

Zur Identifikation während der Verwendung des Service-Clients, wird ein Secret verwendet, welches nach der Anmeldung im Portal ausgelesen und im Configfile eingefügt werden kann. Standartmässig ist das Secret des Benutzers, welcher den Service-Client heruntergeladen hat, im Configfile gespeichert. So können wir verhindern, dass das Passwort des Benutzers unverschlüsselt im Configfile zu finden ist und trotzdem eine genügende Zugriffskontrolle vorhanden ist.

## Designentscheide auf Tier 2

### *Designentscheide in der Komponente „Parallelization as a Service Portal“*

#### ParallelizationPortalUtility

Da die verschiedenen Komponenten auf die Datenbank zugreifen müssen, haben wir uns dazu entschlossen, die Verbindung mit der Datenbank und dem HPC Cluster zu abstrahieren und in ein separates Library Project auszulagern.

Das Projekt verwendet das Entity Framework und besteht aus DBUtil, HPCUtil und den zugehörigen Model-Klassen.

Diese Auslagerung verhindert doppelten Code und hat den Vorteil, dass Änderungen nur an einem einzelnen Ort vorgenommen werden müssen. Ausserdem kann so die Verbindung mit dem HPC Cluster und der Datenbank auch in weiteren Projekten einfach implementiert werden.

#### Persistente Daten über die Tasks

Bei einem Update, Absturz oder Neustart des HPCs sind die Task- bzw. Jobinformationen eventuell nicht mehr persistent vorhanden. Auch soll das Parallelization as a Service Portal losgelöst vom HPC über bereits beendete Tasks bzw. Jobs funktionieren. Um einerseits den HPC nicht unnötig zu belasten und andererseits auf teure Remoteverbindung verzichten zu können, verwenden wir eine Tabelle für Tasks in der Datenbank.

Ein neu gestarteter Task vom Parallelization as a Service Portal wird in dieser Datenbank eingetragen, bevor der eigentliche Task an den HPC Cluster gesendet wird um zu verhindern, dass Tasks die nicht in die Datenbank eingetragen werden können auf dem HPC Cluster laufen und somit nicht verrechnet werden.

Die Tabelle der Tasks wird bei jeder Abfrage der TaskView im Parallelization as a Service Portal um die Daten des HPC aktualisiert. Ebenfalls wird der Quota Monitor-Task die Tabelle in einem festgelegtem Zeitintervall die Datenbank aktualisieren.

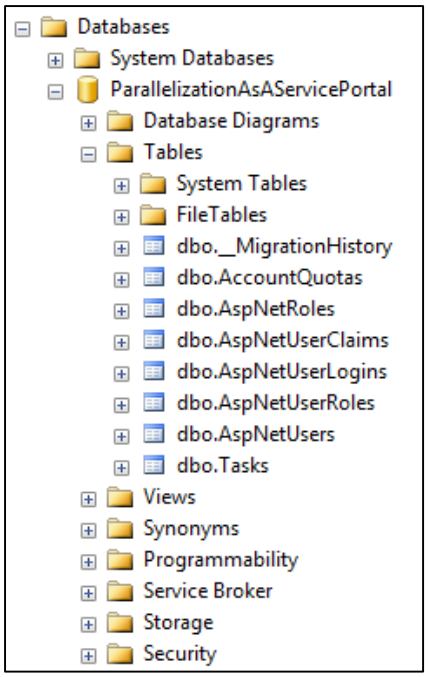



*Abbildung 9: Übersicht der Tabellen in der Datenbank „ParallelizationAsAServicePortal“*

## *Designentscheide in der Komponente „Quota Monitor“*

### Quota Monitor und Kostenmodell

Um eine Übernutzung des HPCs via dem Parallelization as a Service Portal zu verhindern, werden Quotas eingeführt. Diese sollen verhindern, dass Benutzer des Portals zu lange oder zu viele Tasks dem HPC zur Bearbeitung übermittelt werden können. Wie mit Herr Bläser abgesprochen verwenden wir **Laufzeit [s] * Anzahl MaxNodes** für die Einheit der Credits.

Auf der Entität des ApplicationUsers hat es dafür ein Field namens Credit:

| Credit |
|---|
| 1000 |
| 10000 |

*Abbildung 10: Field Credit*

Eine Tabelle namens AccountQuota mit folgenden Fields wurde für die Startwerte in die Datenbank eingefügt:

| | Column Name | Data Type | Allow Nulls |
|---|---|---|---|
| 🔑 | ID | int | ☐ |
| | TaskPerUser | int | ☐ |
| | StartCredit | int | ☐ |
| | MaxNodesPerTask | int | ☐ |
| | MaxNodesPerUser | int | ☐ |
| | Changed | datetime | ☐ |
| | | | ☐ |

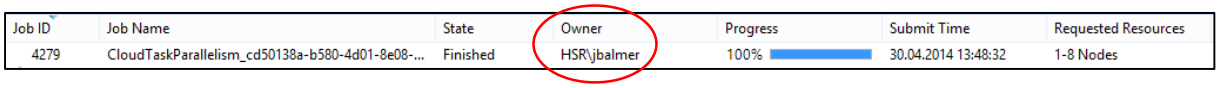
*Abbildung 11: Table AccountQuota*

Wegen der zusätzlichen Verwendung durch den Quota Monitor-Task wird auf eine Lösung mit Configfile verzichtet. Die obige Tabelle wird verwendet, um die vorkonfigurierten Werte zu speichern. Sie ist als SingleRow-Tabelle vorgesehen. Die einzelnen Werte können vom Administrator des Parallelization as a Service Portal festlegen werden.

Der Quota Monitor-Task wird als Windows Service implementiert und überprüft in regelmässigen Zeitabstand die festgelegten Werte der Tasks. Falls eine grobe Überschreitung dieser Werte erkannt wurde, kann der Quota-Monitor-Task auch laufende Task auf dem HPC beenden um eine übermässige Benutzung von einer Person zu verhindern.

### Benutzerkonten von HPC und Parallelization as a Service Portal

Um Jobs auf dem HPC ausführen zu können, braucht jeder Benutzer ein Benutzerkonto auf dem HPC. Das Erstellen eines neuen HPC Benutzerkontos für jeden Benutzer des Parallelization as a Service Portal erfolgt einerseits ausserhalb des IFS und andererseits nicht automatisiert, was viel Zeitaufwand bedeutet. Deshalb haben wir uns entschlossen, eine weitere Schicht von Benutzer einzuführen. Diese sind auf dem Server des Parallelization as a Service Portal in einer Datenbank persistent gespeichert. Jedoch sind dann alle Benutzer des Portals mit demselben HPC Benutzerkonto auf dem HPC aktiv. Mit Hilfe des HPC Job Managers kann nicht mehr unterschieden werden, welcher Task bzw. Job von welchem Benutzer ursprünglich in Auftrag gegeben wurde. Auf der Stufe HPC treten alle Portal-Benutzer als dedizierter Benutzer auf.



| Job ID | Job Name | State | Owner | Progress | Submit Time | Requested Resources |
|---|---|---|---|---|---|---|
| 4279 | CloudTaskParallelism_cd50138a-b580-4d01-8e08-... | Finished | HSR\jbalmer | 100% | 30.04.2014 13:48:32 | 1-8 Nodes |

*Abbildung 12:HPC Job Manager: Owner*

### Verifikations-E-Mail und E-Mail Versand

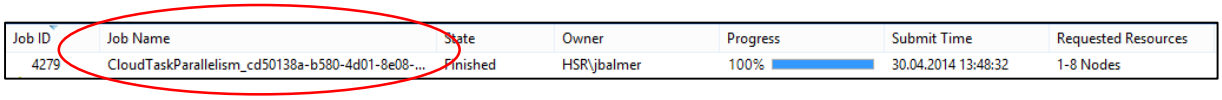
Um das Parallelization as a Service Portal verwenden zu können, muss ein Benutzerkonto auf der Webseite erstellt werden. Um sicherzustellen, dass eine gültige E-Mail-Adresse angegeben wurde, versendet das System nach der Registration eine E-Mail mit Bestätigungs-Link. Dieser Link enthält einen Hashwert, um ihn so einem Benutzer zuweisen zu können. Danach ist der Benutzer freigeschaltet und hat Zugriff auf weitere Funktionen wie z.B. das Herunterladen des Service-Client.

Um die Verifikations-E-Mails versenden zu können, haben wir während der Entwicklungszeit Gmail als E-Mail-Anbieter verwendet. Dort kann ein Konto mit wenigen Klicks erstellt werden. Wegen dem administrativen Aufwand von Externen, wurde kein anonymes E-Mail-Konto bei der HSR beantrag. Ein solches kann nachträglich noch einfach eingebaut werden und würde die Seriosität des Parallelization as a Service Portals erhöhen.

### Task.Name als Identifier

Da erst der HPC Cluster die Task.ID festlegt, haben wir keine Möglichkeit diese vorher auf dem Parallelization as a Service Portal zu setzen. Deswegen verwenden wir den Task.Name als Identifier, da wir diesen selbst bestimmen können. Da dieser nicht unbedingt eindeutig ist, haben wir uns entschlossen, den Task.Name generisch so zu wählen, dass es praktisch gesehen eindeutig ist.

Hier die Sicht aus dem HPC Job Manager; sichtbar ist der generierte Job Name:



| Job ID | Job Name | State | Owner | Progress | Submit Time | Requested Resources |
|---|---|---|---|---|---|---|
| 4279 | CloudTaskParallelism_cd50138a-b580-4d01-8e08-... | Finished | HSR\jbalmer | 100% | 30.04.2014 13:48:32 | 1-8 Nodes |

*Abbildung 13: HPC Job Manager: Job Name*

## *Designentscheide in der Komponente „MSSQL“*

### RowVersion wird nur auf Task mitgeführt

Da nur der Administrator Zugriff auf die Tabelle „Quota“ hat und es nur einen einzigen Administrator gibt, ist die RowVersion auf dieser Tabelle unnötig. Bei der Tabelle „ApplicationUser“ ist es nicht möglich, eine RowVersion zu führen, da das die vordefinierte Klasse des IdentityUsers blockiert.

# Teststrategie

## Unittests

Die Applikation besteht vorwiegend aus ASP.NET-Seiten, den dazugehörigen Controllern und Hilfsklassen (zum Versand von E-Mail oder zum Abfragen des HPCs). Desswegen setzen wir Unittesting nur sehr punktuell ein, da Unittesting nicht bei allen von den obengenannten Klassen sinnvoll ist.

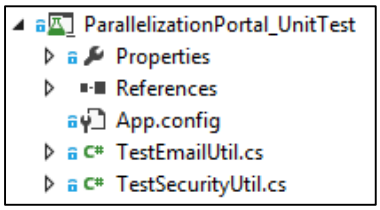



*Abbildung 14: Übersicht des Project UnitTest in der Solution „ParallelizationPortal“*

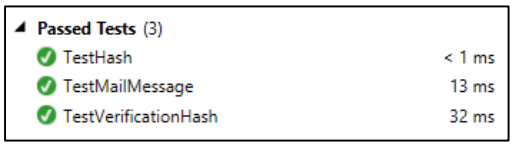



*Abbildung 15: Ansicht Test Explorer*

## Systemtests

Dafür setzen wir auf Systemtests, die das korrekte Zusammenspiel der Klassen gut testen können. Der komplette Testreport ist im Anhang zu finden.

# Resultat

## Beschreibung des Prototyps

### ASP.Net MVC Seite

#### *Konfiguration*

Alle benötigten Connectionstrings und die Einstellungen für den HPC Cluster sind über die Configfiles zugänglich. Die Werte können auch direkt im IIS eingestellt werden.

Beim Start der Seite sollte man zuerst einen Administrator registrieren. Der Administrator muss den Username „Administrator“ verwenden. Nach erstellen des Administrators können die Quotas definiert werden und damit sind die nötigen Konfigurationen abgeschlossen.

#### *Publicansichten:*

Registration und Anmeldung:
Nachdem sich ein neuer User registriert hat, erhält er eine E-Mail, welche an die von ihm angegebene Adresse gesendet wird. In der Email ist ein Link enthalten der zum vollständigen Freischalten des Accounts benötigt wird.

Nach dem bestätigen der Emailadresse kann sich der User anmelden.

Ansicht Info:
Es gibt eine Gebrauchsanweisung mit einer Schritt-für-Schritt-Erklärung für den Gebrauch des Portals und des heruntergeladenen Clients. Diese ist unter dem Punkt Info zu finden.

Ansicht Kontakt:
Seite mit Kontaktinformationen für den User.

#### *Useransichten*

Service Client:
Der User kann den Service-Client herunterladen und erhält eine vorkonfigurierte Version mit seinen Credentials. So kann er die DLLs direkt in seinem Projekt einbinden und verwenden.
Es wird zuerst ein File mit dem Secret des Users erstellt und danach mit den DLL’s des Service-Clients gezipt und zum Download angeboten. Dach dem Download wird das erstellte Zipfile wieder gelöscht.

Tasks:
Der User hat die Übersicht über seine Tasks und deren Status sowie eine Übersicht über seine Credits die er noch zur Verfügung hat.
Die laufenden Tasks können über die Detailansicht gestoppt werden.

#### *Administratoransichten*

Quotas:
Hier erhält der Administrator Einsicht in die gesetzten Quotas. Die Werte können geändert werden und werden mit einem Klick auf „speichern“ in die Datenbank geschrieben. Sie sind ab diesem Zeitpunkt auch gültig.

Benutzerverwaltung:
Hier kann der Administrator die Benutzerdaten verwalten.
Folgende Punkte sind änderbar:

- UserName
- Name
- Vorname
- IsActivated
- IsBlocked
- Credit

Tasks:
Der Administrator Sieht alle Tasks des Parallelizationportals und kann diese in der Detailansicht des Tasks stoppen.

## Monitor

Der Monitor ist ein Windows Service der die Einhaltung der Quotas überwacht und durchsetzt. Der Monitor sorgt auch dafür, dass die Tasks in der Datenbank regelmässig mit denen vom HPC Cluster abgeglichen werden damit sie aktuell sind.

Der Monitor sorgt ausserdem dafür, dass die Zeiten der Tasks verbucht werden und diese dem Nutzer abgezogen werden.

Der Monitor protokolliert zudem seine Iterationen und schreibt sie in Ein EventLog. Zusätzlich werden Fehler und das Starten sowie Stoppen des Monitors im Log erfasst.

Der Monitor wird mittels Developer Command Prompt von Visual Studio installiert.

```
Installieren: installutil.exe Monitor.exe
Deinstallieren: installutil.exe /u Monitor.exe
```

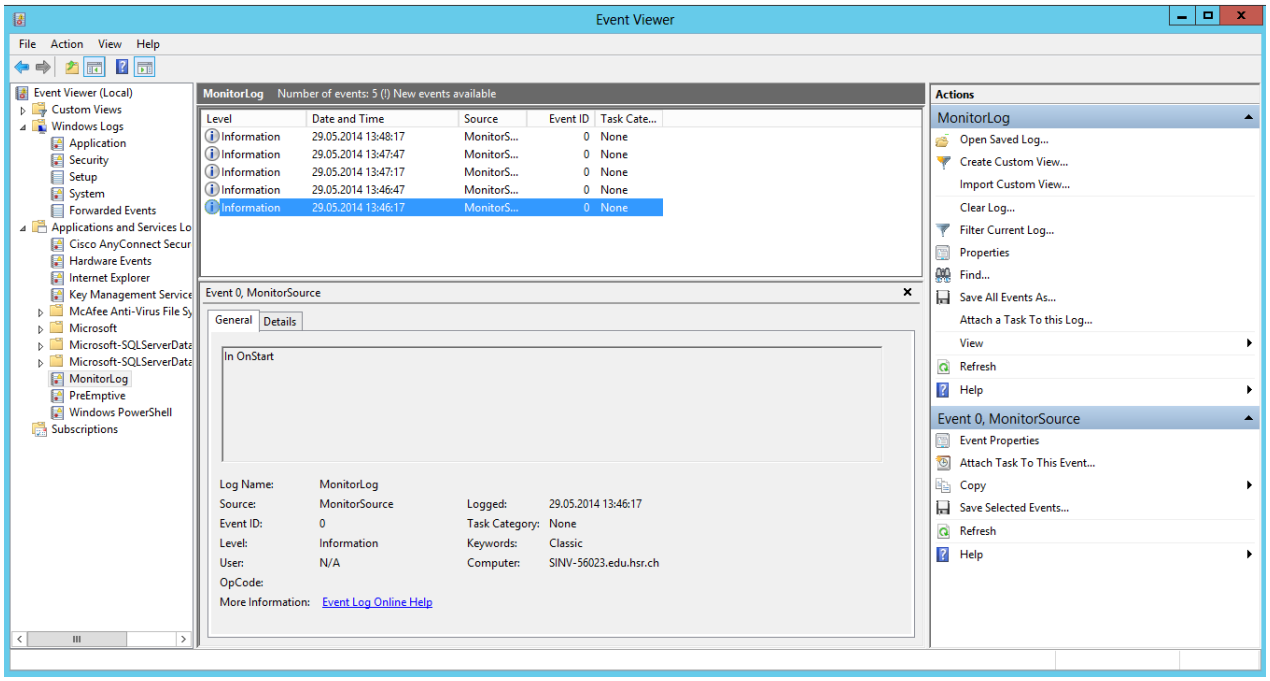



*Abbildung 16: Screenshot Event Viewer*

## Screenshots

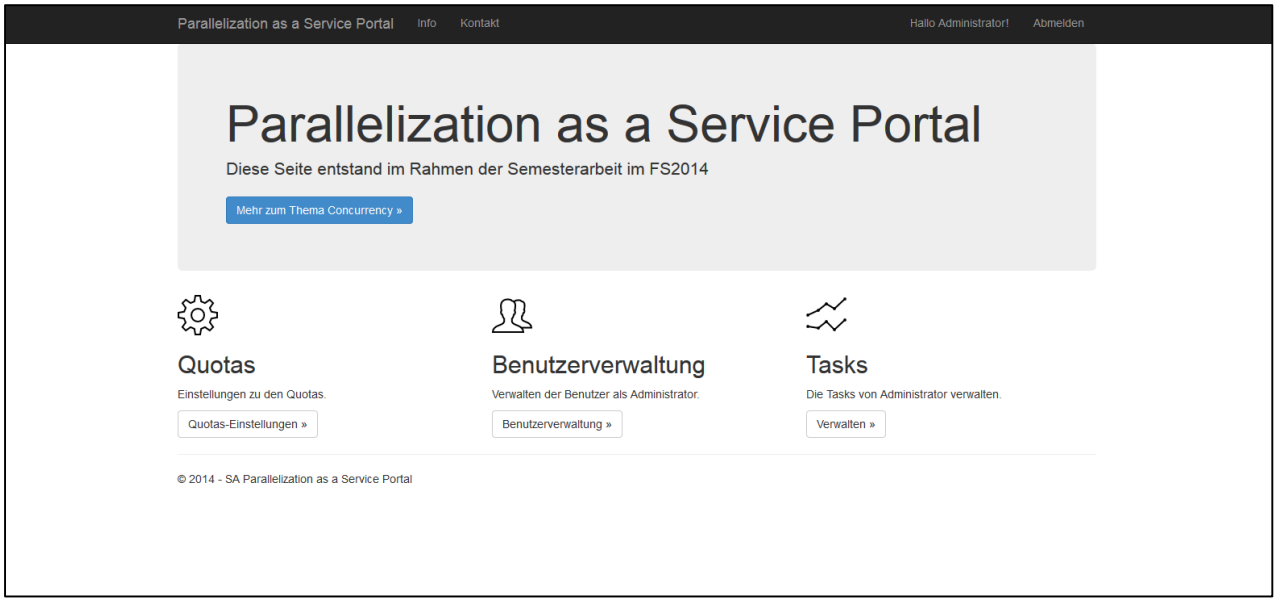



*Abbildung 17: Ansicht des Administrators*

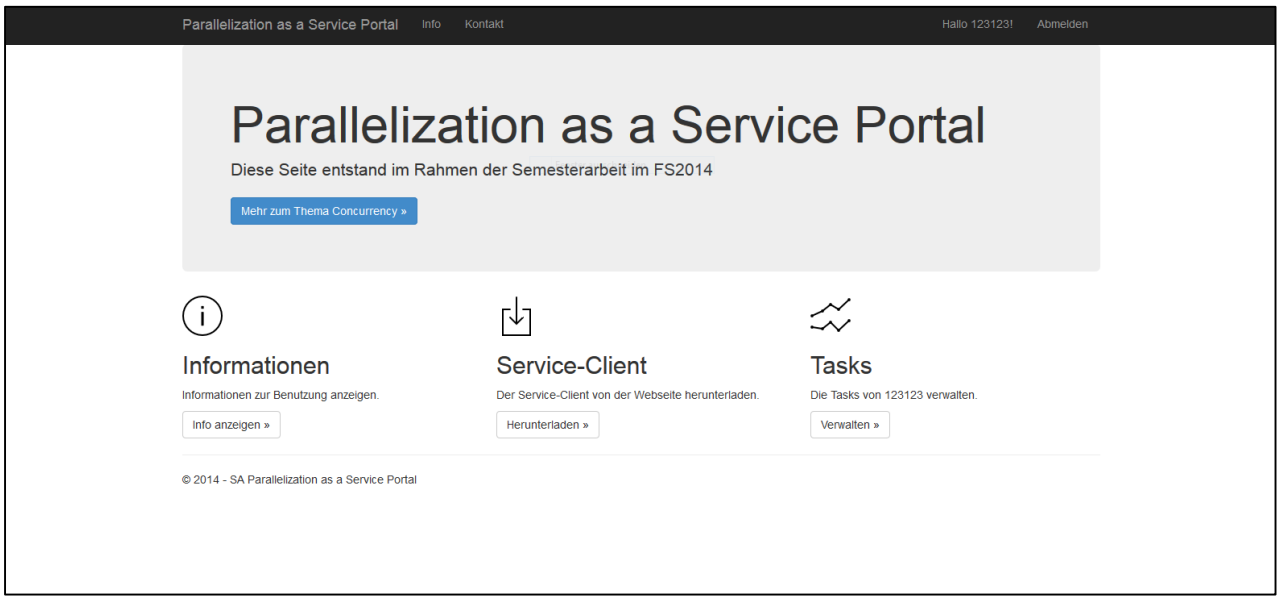



*Abbildung 18: Ansicht eines Benutzers*


*Abbildung 19: Ansicht ohne Login: Registrieren und Anmelden*

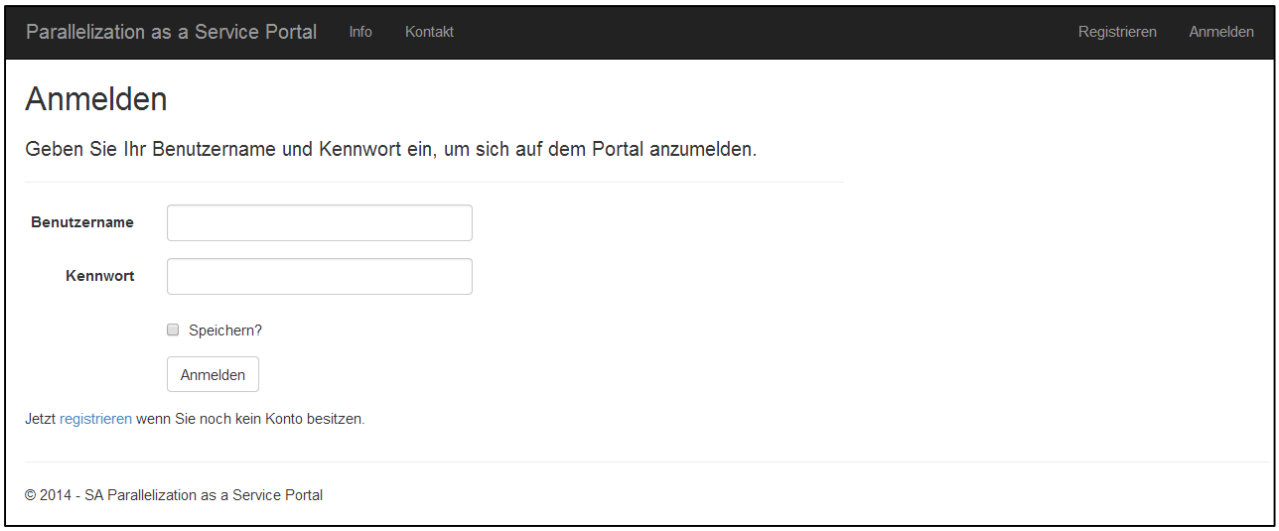



*Abbildung 20: Ansicht Anmelden*

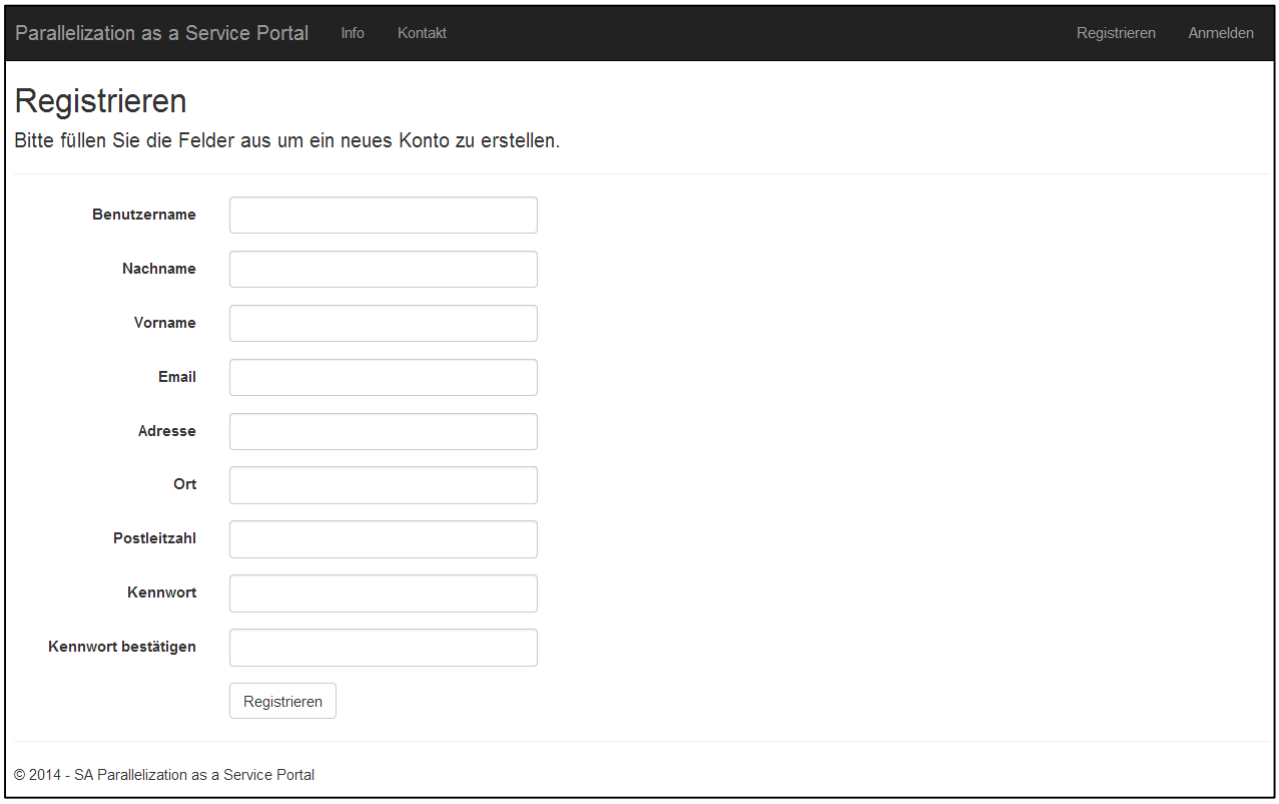



*Abbildung 21: Ansicht Registrieren*

## Ablaufbeschreibung für Benutzer

Schritt 1: Registrieren auf dem Parallelization as a Service Portal:

- Zuerst brauchen Sie ein Benutzerkonto für das Parallelization as a Service Portal. Klicken Sie dafür oben rechts auf der Webseite auf "Registrieren".
- Füllen Sie dann die Felder korrekt aus und klicken Sie auf "Registrieren"
- Sie werden vom System eine E-Mail erhalten. Klicken Sie zur Verifizierung auf den Link im E-Mail.
- Sie können Sich nun anmelden.

Schritt 2: Herunterladen und einbinden des Serviceclients

- Nach dem Anmelden klicken Sie unten rechts bei Service-Client auf Herunterladen.
- Binden Sie die heruntergeladenen Assemblys in Ihr Projekt ein.

Schritt 3: Verwenden des Serviceclients und Monitoring auf dem Parallelization as a Service-Portal:

- Fügen Sie die Benötigte `using HSR.CloudTaskParallelism.Client.Runtime;` ein.
- Verwenden Sie die neue Klasse wie ein klassisches Threadpool:

```
Parallel.For(0, verticalPixels, (row) =>
  doSomething();
});
```

- Sie können nun im Parallelization as a Service Portal den Task in der Taskübersicht unten mittig überwachen und allenfalls auch stoppen.
- Vergessen Sie nicht, dass wenn Ihr Credit aufgebraucht ist, Ihre Tasks möglicherweise auch abgebrochen werden können.

## Ausblick (Ideen)

### Payment

Um das Projekt in der Zukunft kommerziell nutzbar zu machen, braucht es ein zusätzliches Modul das für das Payment zuständig ist.

Es wäre leicht in die bisherige Umgebung einzusetzen. Es bräuchte nur eine Funktion die der User nutzen kann um seine Credits wieder aufzuladen.

Denkbar sind verschiedenste Zahlungsmöglichkeiten wie PayPal oder Kreditkarte. Die Creditpunkte könnten ähnlich einer Onlinewährung in Paketen gekauft werden.

### UserGruppen

Eine Unterteilung der User in verschiedene Gruppen wäre nützlich um sie mit verschiedenen Quotas kontrollieren zu können. Beispielsweise Studenten der HSR könnten einen höheren Startcredit erhalten als andere User. So könnten auch spezielle Projektgruppen umgesetzt werden, denen mehr Tasks pro Nutzer zur Verfügung stehen.

### HSR Konto

Es könnte für das Portal ein eigenes Benutzerkonto angelegt werden. So könnten die Tasks am HPC Cluster sowie der Mailverkehr einfacher geregelt und zugeordnet werden.

# Fazit

Das Primärziel eines funktionierenden Prototyps wurde erreicht und die anfänglich definierten Use-Cases konnten umgesetzt werden. Der Prototyp bestand die System- und Unittests und erreichte zufriedenstellende Resultate im Bereich Performance gerade durch das verwendete Entity Framework. Durch die Wahl des Code-First-Approaches im Entity Framework waren Anpassungen in der Datenbank leicht umsetzbar. So werden zukünftige Erweiterungen auch leichter umsetzbar.

Es zeigte sich, dass das verwendete API für den HPC Cluster gut Dokumentiert ist, wodurch eine Einarbeitung leicht gefallen ist. Die verwendete HPC Infrastruktur lief bis auf wenige Einbrüche sehr Stabil und beeinträchtigt die Performance der Seite nicht. Hier muss aber noch weiter abgeklärt werden, wo sich die Grenzen für die Anzahl der User und Tasks befinden.

Das Ziel eines funktionierenden Konzepts wurde erreicht, jedoch nahm die Analyse und Einarbeitung in die Frameworks mehr Zeit in Anspruch als ursprünglich geplant, dadurch verzögerte sich der Start der Implementierung. Die Verzögerung führte dazu, dass der Prototyp am Ende leider nur Lokal getestet werden konnte und es blieb keine Zeit mehr die Lauffähigkeit auf dem VServer der HSR ausführlich zu testen.

# Persönliches Statement

## Jan Balmer

Ich habe viel gelernt über die Verwendung von .NET in Webapplikationen. Es war sehr spannend die für mich neuen Frameworks zu verwenden und mir Wissen über diese neuen Technologien anzueignen.

Gerade die Aufteilung in die verschiedenen Komponenten in unserem Projekt war anfangs sehr schwierig, da man so gleichzeitig verschiedene „Baustellen“ hatte. Sobald wir aber die Datenbankzugriffe in ein separates Projekt abstrahiert hatten, wurde die Programmierarbeit stark erleichtert.

Gerade diese Aufteilung fand ich am Schluss des Projektes eine sehr praktische Eigenschaft, da man ohne die eine Komponente zu verändern, an einer anderen arbeiten konnte.

## Adrian Rieser

Auch wenn es mein erstes .NET Projekt war, konnte ich mich rasch mit der eingesetzten ASP.NET MVC Architektur anfreunden. Bei Fragen bezüglich C# war MSDN sehr hilfreich, da dort viele Themen sehr gut dokumentiert sind. Die Verwendung des HPC Clusters, dem Entity Framework und dem Webconfig- und Ressourcenfiles waren mit nützlichen Codebeispielen unterstützt.

Trotzdem dauerte alles etwas länger, da ein Teil der Zeit doch für die Einarbeitung in die neuen Technologien verwendet wurde. Es gab Gebiete, in denen ich mich auskannte und welche, die neu für mich waren und ich viel lernen konnte. Als Webapplikation mit ASP.NET-Seiten, Datenbankanbindung, WCF Schnittstellen und gab es viele Aspekte.

Die endgültige Struktur die wir nach dem letzten Codereview und Refactoring nun haben, gibt dem Projekt meiner Meinung nach ein guter Aufbau, den ich für weitere Projekte verwenden kann.

# Anhang

## Literaturverzeichnis

[1] Using HPC, http://msdn.microsoft.com/en-us/library/cc907080%28v=vs.85%29.aspx, letzter Zugriff am 29.5.2014

[2] HPC Reference, http://msdn.microsoft.com/en-us/library/cc853441%28v=vs.85%29.aspx, letzter Zugriff am 29.5.2014

[3] Appendix 1: HPC Cluster Networking, http://technet.microsoft.com/en-us/library/ff919486%28v=ws.10%29.aspx, letzter Zugriff am 29.5.2014

[4] Jess Chadwick, Todd Snyder und Hrusikesh Panda, “Programming ASP.NET MVC 4”, First Edition, Oktober, 2012

[5] Getting Started with ASP.NET MVC 5, http://www.asp.net/mvc/tutorials/mvc-5/introduction/getting-started, letzter Zugriff am 29.5.2014

[6] Code First für eine neue Datenbank, http://msdn.microsoft.com/de-ch/data/jj193542.aspx, letzter Zugriff am 29.5.2014

[7] Working with .resx Files Programmatically, http://msdn.microsoft.com/en-us/library/gg418542%28v=vs.110%29.aspx, letzter Zugriff am 29.5.2014

[8] Web.config File - ASP.NET, http://www.codeproject.com/Articles/301726/Web-config-File-ASP-NET, letzter Zugriff am 29.5.2014

[9] Walkthrough: Creating a Windows Service Application in the Component Designer, http://msdn.microsoft.com/en-us/library/zt39148a(v=vs.110).aspx, letzter Zugriff am 29.5.2014

## Projektplanung

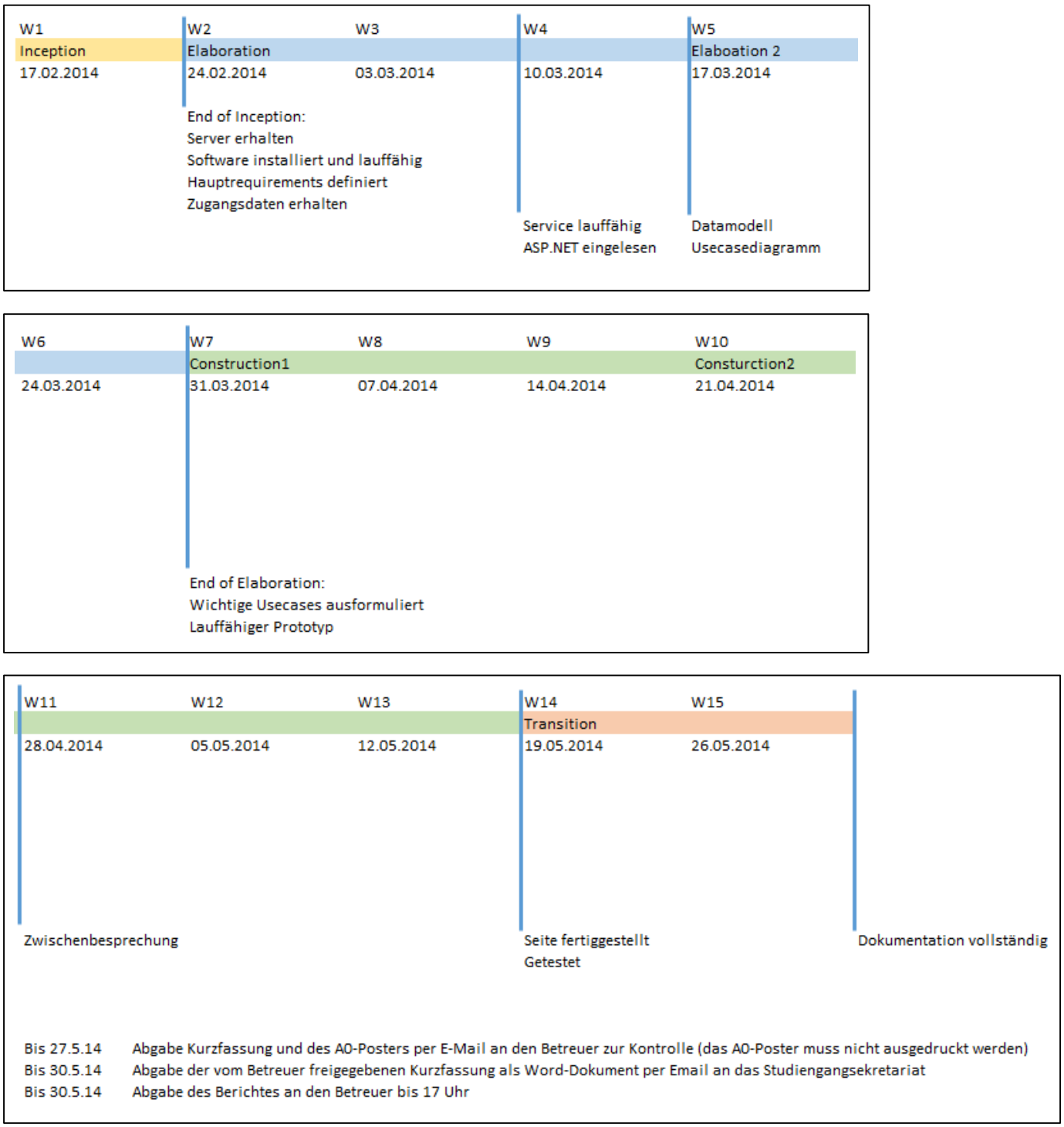



*Abbildung 22: Übersicht Zeitplan und Meilensteine*

## Soll-Ist

| | | Adrian R. | Jan B. | IST | SOLL |
|---|---|---|---|---|---|
| Woche 1 | Einlesen in Code | 6 | 7 | 13 | 10 |
| | Milestones erfassen | 1 | 2 | 3 | 2 |
| | Requirements | 2 | 3 | 5 | 4 |
| | Risikos erfassen | 2 | 1 | 3 | 2 |
| | Installieren von Software | 2 | 2 | 4 | 4 |
| | | 13 | 15 | 28 | 22 |
| Woche 2 | HPC SDK einrichten | 2 | 2 | 4 | 4 |
| | Service einrichten | 3 | 4 | 7 | 4 |
| | Workpackage für nächste Iteration | 1 | 3 | 4 | 4 |
| | Iterationen/Zeitplan überarbeiten | 1 | 1 | 2 | 1 |
| | ASP.NET einlesen | 4 | 5 | 9 | 6 |
| | Framework besprechen | 2 | 2 | 4 | 4 |
| | | 13 | 17 | 30 | 23 |
| Woche 3 | Datamodel | 3 | 1 | 4 | 4 |
| | Datenbank installieren und einrichten | 2 | 2 | 4 | 4 |
| | Usecasediagram erstellen | 4 | 1 | 5 | 4 |
| | Usecases ausformuliert | 2 | 1 | 3 | 4 |
| | HTML Template | 0 | 3 | 3 | 4 |
| | Frameworks installieren | 1 | 2 | 3 | 2 |
| | Anmeldegui für Prototyp | 1 | 2 | 3 | 3 |
| | Registrierungsgui für Prototyp | 1 | 3 | 4 | 4 |
| | Testen | 2 | 1 | 3 | 3 |
| | | 16 | 16 | 32 | 32 |
| Woche 4 | Anforderungsspezifikationsdokument | 4 | 1 | 5 | 2 |
| | ASP.NET einlesen | 3 | 0 | 3 | 4 |
| | HPC Reference eingelesen | 3 | 2 | 5 | 4 |
| | Domainmodel | 3 | 0 | 3 | 4 |

| | | | | | |
|---|---|---|---|---|---|
| | Einrichten von IIS | 0 | 2 | 2 | 4 |
| | Einrichten MSSQL | 0 | 3 | 3 | 2 |
| | Registrieren/Anmelden von Usern | 2 | 6 | 8 | 6 |
| | Testing | 1 | 3 | 4 | 4 |
| | Einrichten von Service auf Server | 1 | 1 | 2 | 2 |
| | | 17 | 18 | 35 | 32 |
| Woche 5 | Auslesen von Jobliste | 3 | 1 | 4 | 3 |
| | Abbrechen eines laufenden Jobs | 3 | 1 | 4 | 3 |
| | Taskverwaltung ASP.NET | 1 | 5 | 6 | 4 |
| | Testen | 2 | 2 | 4 | 4 |
| | Lösen von Serverproblem | 4 | 1 | 5 | 4 |
| | HPC Tutorial durcharbeiten | 1 | 1 | 2 | 2 |
| | Umstellung auf entity Framework | 2 | 4 | 6 | 8 |
| | | 16 | 15 | 31 | 28 |
| Woche 6 | Datenbankverbindung von Service | 1 | 3 | 4 | 4 |
| | Erstellen von neuem Datenbankeintrag bei neuem Job | 1 | 5 | 6 | 4 |
| | Useransicht der Jobs | 1 | 3 | 4 | 4 |
| | Detailansicht von einzelnem Job | 2 | 1 | 3 | 4 |
| | Util Klasse für HPC funktionen | 2 | 1 | 3 | 4 |
| | Testen | 3 | 2 | 5 | 4 |
| | Fehlerkorrektur | 3 | 3 | 6 | 6 |
| | | 13 | 18 | 31 | 30 |
| Woche 7 | Canceln von laufendem Job | 2 | 5 | 7 | 8 |
| | Anpassung Domainmodel | 3 | 1 | 4 | 4 |
| | Einlesen der Credentials via XML | 1 | 3 | 4 | 4 |

| | | | | | |
|---|---|---|---|---|---|
| | Ergänzungen von Personendaten | 2 | 2 | 4 | 4 |
| | Weiteres Workpackage erstellen | 1 | 1 | 2 | 2 |
| | Anpassung HPC Utility | 1 | 0 | 1 | 2 |
| | Testen | 0 | 1 | 1 | 2 |
| | Fehlerkorrektur | 1 | 1 | 2 | 2 |
| | Featureliste | 2 | 0 | 2 | 2 |
| | | 13 | 14 | 27 | 30 |
| Woche 8 | Abklären von Beschränkung für Jobs | 2 | 0 | 2 | 2 |
| | Emailverifikation | 1 | 6 | 7 | 8 |
| | Downloadseite für Client | 1 | 3 | 4 | 4 |
| | Testen | 2 | 2 | 4 | 4 |
| | beginn mit Adminansicht | 0 | 2 | 2 | 4 |
| | | 6 | 13 | 19 | 22 |
| Woche 9 | Dokumentation | 2 | 1 | 3 | 4 |
| | Einrichtung auf ASP seite | 0 | 2 | 2 | 2 |
| | Admin Userverwaltung | 0 | 4 | 4 | 4 |
| | Admin Jobverwaltung | 0 | 4 | 4 | 4 |
| | Testen | 3 | 2 | 5 | 4 |
| | Anpassen Detailansicht von Task | 0 | 1 | 1 | 2 |
| | Einfache Constraints | 0 | 1 | 1 | 2 |
| | | 5 | 15 | 20 | 22 |
| Woche 10 | Software Architektur Dokument | 4 | 1 | 5 | 4 |
| | Seite aktualisieren und mit Inhalt füllen | 6 | 0 | 6 | 4 |
| | Quotas einfügen | 0 | 5 | 5 | 4 |
| | Quota-Monitor einfügen | 0 | 7 | 7 | 6 |
| | Refactoring | 2 | 0 | 2 | 2 |
| | Testing | 3 | 2 | 5 | 4 |
| | | 15 | 15 | 30 | 24 |

| | | | | | |
|---|---|---|---|---|---|
| Woche 11 | Seite auf Server einrichten | 0 | 5 | 5 | 4 |
| | Monitor erstellen | 0 | 4 | 4 | 4 |
| | Monitor testen | 0 | 2 | 2 | 2 |
| | Dokumentation | 3 | 1 | 4 | 4 |
| | Umsetzung Constraints | 0 | 2 | 2 | 2 |
| | Seite aktualisieren und mit Inhalt füllen | 5 | 0 | 5 | 4 |
| | Icons eingefügt | 4 | 0 | 4 | 4 |
| | | 12 | 14 | 26 | 24 |
| Woche 12 | Monitor fertiggestellt | 0 | 6 | 6 | 8 |
| | Monitor auf Server einrichten | 0 | 2 | 2 | 2 |
| | Defaultwert bei Constraint einrichten | 0 | 1 | 1 | 2 |
| | Überprüfung Constraint bei Tasksubmit | 0 | 2 | 2 | 2 |
| | Dokumentation Nachführen | 4 | 2 | 6 | 4 |
| | Beschreib Resultat | 0 | 3 | 3 | 4 |
| | XML generierung | 2 | 0 | 2 | 2 |
| | ZIP generierung | 5 | 0 | 5 | 4 |
| | Systemtest | 4 | 0 | 4 | 4 |
| | | 15 | 16 | 31 | 32 |
| Woche 13 | Refactoring | 2 | 3 | 5 | 4 |
| | Abarbeiten der CodeReviewliste | 7 | 6 | 13 | 10 |
| | Dokumentation | 2 | 4 | 6 | 6 |
| | Testing | 3 | 0 | 3 | 4 |
| | | 14 | 13 | 27 | 24 |
| Woche 14 | Refactoring gemäss Code-Reviewliste | 14 | 16 | 30 | 20 |
| | Draft Kurzfassung/Abstract | 1.5 | 0 | 1.5 | 2 |
| | Draft Plakat | 1 | 0 | 1 | 2 |
| | Testing | 2 | 1 | 3 | 4 |
| | Bericht | 12 | 10 | 2 | 2 |
| | | 30.5 | 17 | 57.5 | 30 |

## Eingesetzte Werkzeuge

- Visual Studio 2013 mit Visual Studio Online
- MS HPC Cluster mit HPC Job Manager
- Prototyp von CloudTaskParallelization
- MSSQL Server mit SQL Server Management Studio
- Astah Community und Visio 2013 für Diagramme
- Word und Excel 2013 für Dokumente

## Testing

### Systemtest

#### *Voraussetzungen*

#### An den Server:

- VPN Verbindung steht um die Verbindung mit dem HPC zu ermöglichen.
- MS SQL vorhanden.
- Windows Service des Quota-Monitors muss laufen.
- Ein spezieller Benutzer namens "Administrator" muss vorhanden sein.

#### An den Client:

- Konnektivität zum Parallelization as a Service Portals.
- IDE zur Verwendung des Service-Clients vorhanden.

*Spezifikation*

| Nr. | Beschreibung des Tests | Erwartetes Resultat |
|---|---|---|
| 1a | Registrieren<br>mit kompletten und gültigen Angaben | Erfolgreiches registrieren |
| 1b | Registrieren<br>mit bereits verwendetem Benutzernamen | System meldet dass der Benutzername bereits verwendet wird |
| 1c | Registrieren<br>mit nicht vollständig ausgefüllten Folmularfelder | System meldet welche Felder noch ausgefüllt werden müssen |
| 1d | Registrieren<br>mit ungültiger E-Mail-Adresse | System meldet den Fehler |
| 1e | Registrieren<br>mit ungültiger PLZ | System meldet den Fehler |
| 2 | Verifizieren<br>via Link im Verifizierungs-E-Mail | Erfolgreiches verifizieren<br>(Mutationen auch in der DB) |
| 3a | Anmelden<br>mit korrekt Verifiziertem Benutzer | Erfolgreiches anmelden<br>(Es werden folgende Kacheln angezeigt: Informationen, Service-Client und Tasks) |
| 3b | Anmelden<br>mit falschem Passwort | System meldet dass der Benutzername oder das Passwort ungültig ist |
| 3c | Anmelden<br>mit nichtvorhandenem Benutzer | System meldet dass der Benutzername oder das Passwort ungültig ist |
| 3d | Anmelden<br>mit einem nicht verifizierten Benutzer | System meldet dass der Benutzername oder das Passwort ungültig ist |
| 3e | Anmelden<br>als Administrator | Erfolgreiches Anmelden<br>(Es werden folgende Kacheln angezeigt: Quotas, Benutzerverwaltung und Tasks) |
| 4a | Herunterladen des Service-Clients<br>ohne Timeout | Erfolgreiches herunterladen des personalisierten Zip-Archivs: Zip-Dateinamen enthält Namen des entsprechenden Benutzer, XML enthält Secret von entsprechendem Benutzer |
| 4b | Herunterladen des Service-Clients<br>nach 10 Minuten | Abbruch des herunterladens |
| 5 | Task abbrechen<br>von einem zu editierenden Tasks | Task wird abgebrochen: Sichtbar via HPC Job Manager |
| 6a | Benutzerverwaltung:<br>Sperren eines Benutzers | CRUD-Operationen sind möglich und werden gesperrt: Sichtbar in der Datenbank |
| 6b | Benutzerverwaltung:<br>Ein nicht verifizierter Benutzer versucht einen Task zu submitten | Abbruch |
| 6c | Benutzerverwaltung:<br>Ein gesperrter Benutzer versucht einen Task zu submitten | Abbruch |
| 7 | Abmelden | Der Benutzer wird abgemeldet und wird auf die Hauptseite geleitet |
| 8 | Quotas:<br>Mutieren und speichern | Quotas werden gemäss Eingaben mutiert: Sichtbar auf der Datenbank |
| 9 | Task submitten | Task wird via Portal auf den HPC übertragen und ausgeführt: Sichtbar via HPC Job Manager |

*Protokoll vom 29.5.2014*

Systemtest auf lokalem Rechner

| Nr. | Fehler | Status |
|---|---|---|
| 1a | Keiner | OK |
| 1b | Keiner | OK |
| 1c | Keiner | OK |
| 1d | Keiner | OK |
| 1e | Keiner | OK |
| 2 | Keiner | OK |
| 3a | Keiner | OK |
| 3b | Keiner | OK |
| 3c | Keiner | OK |
| 3d | Keiner | OK |
| 3e | Keiner | OK |
| 4a | Keiner | OK |
| 4b | Keiner | OK |
| 5 | Keiner | OK |
| 6a | Keiner | OK |
| 6b | Keiner | OK |
| 6c | Keiner | OK |
| 7 | Keiner | OK |
| 8 | Keiner | OK |
| 9 | Keiner | OK |

## Verteilung der Arbeitspakete

| **Grobaufteilung der Arbeitspakete** | **Bearbeitet druch** |
|---|---|
| Anpassung WCF Service | Jan Balmer |
| Monitor Service | Jan Balmer |
| Grundstruktur ASP.Net MVC Seite | Jan Balmer |
| Testing | Adrian Rieser |
| Abstraktion Datenbank und HPC Cluster Verbindung | Jan Balmer |
| Auslagerung in Config- und Ressourcenfiles | Adrian Rieser |
| Webseitenaufbau und -Inhalt | Adrian Rieser |
| Implementation Datenbankverbindung (Entity Framework) | Jan Balmer |
| ZIP- und XML-Generierung | Adrian Rieser |