Leibniz Universität Hannover
Wirtschaftswissenschaftliche Fakultät
Institut für Produktionswirtschaft
Prof. Dr. Stefan Helber

Hausarbeit im Rahmen der Veranstaltung
Entwicklung von Anwendungssystemen im WiSe 2014/2015
(Veranstaltungs-Nr. 173610)

Entwicklung einer Web-Applikation zur Lösung des ressourcenbeschränkten Projektplanungsproblems

Andreas Hipp	Robert Matern
Ungerstr. 24	Plathnerstr. 49
30451 Hannover	30175 Hannover
Matr.-Nr. 3027520	Matr.-Nr. 2798160

Abgabedatum: 24.03.2015

Inhaltsverzeichnis

Abbildungsverzeichnis **iii**

Abkürzungsverzeichnis **iii**

Symbolverzeichnis **vi**

1 Einleitung **1**

2 Grundlagen zum ressourcenbeschränkten Projektplanungsproblem und zum Framework Ruby on Rails **2**

2.1 Kapazitätsplanung 2

2.2 Kostenplanung 4

2.3 Ruby on Rails 6

3 Implementierung des ressourcenbeschränkten Projektplanungsproblems mittels des Frameworks Ruby on Rails **7**

3.1 Installation der Web-Applikation und der notwendigen Programme 7

3.2 Darstellung der Funktionsweise der Web-Applikation anhand eines Guides für User 8

3.3 Darstellung der Funktionsweise der Web-Applikation anhand eines Guides für Administratoren 14

3.4 Integration des Unterprogramms *rgl* und des Programms *GraphViz* in die Web-Applikation 21

4 Kritische Würdigung der Web-Applikation **25**

5 Fazit **27**

Literatur **28**

A Anhang **30**

A.1 Datenbankschema 30

A.2 GAMS-Implementierung des Beispiels 30

A.3 Ruby on Rails Programmcodes 37

Abbildungsverzeichnis

1 Terminalfenster unter Apple Mac OS X 6
2 Startseite der Web-Applikation Projektplanung 8
3 Anmeldebildschirm 9
4 Fehleranzeige bei der Anmeldung 10
5 Übersicht der Ressourcen für unangemeldete User 11
6 Profilseite eines Users 12
7 Übersicht der Ressourcen für User 13
8 Profilseite des Administrators 14
9 Übersicht der Ressourcen aus Sicht des Administrators 15
10 Projektplanung mit dem RCPSP - Übersicht 16
11 Vorgangsknoten-Netzplan 17
12 Fehler aufgrund eines Zyklus in der topologischen Reihenfolge 17
13 Einstellung des Starttermins anhand eines Kalendermenüs 18
14 Ergebnis der Kapazitätsplanung 19
15 Ergebnis der Kostenplanung - Ressourcenübersicht 21
16 Ergebnis der Kostenplanung - Vorgangsübersicht 22
17 Datenbankschema der Web-Applikation Projektplanung 30

Abkürzungsverzeichnis

Admin	Administrator
GAMS	General Algebraic Modeling System
RCPSP	Resource-Constrained Project Scheduling
RGL	Ruby Graph Library
RoR	Ruby on Rails

Quellcodeverzeichnis

1 Ausschnitt aus dem RoR-Controller für das RCPSP 20
2 Ausschnitt der Gemfile der Web-Applikation Projektplanung 22
3 Prüfung auf Zyklen mittels des Unterprogramms „rgl“ 23
4 Erstellung eines Graphen mittels des Unterprogramms „rgl“ 23
5 Ausschnitt aus dem RoR-Controller für die Vorgangsbeziehungen 24
6 Prüfung auf Zyklen mittels des Unterprogramms „rgl“ 24
7 Ausschnitt des RoR-Controller für die Vorgangsrelationen 25
8 GAMS-Code zur Kapazitätsplanung 30
9 GAMS-Code zur Kostenplanung 33
10 Gemfile der Web-Applikation Projektplanung 37
11 Routes-Datei der Web-Applikation Projektplanung 38
12 RoR-Controller für die Vorgangsrelationen 39
13 RoR-Controller für die Vorgangs-Ressourcen-Kombinationen 41
14 RoR-Controller für die Vorgänge 43
15 RoR-Controller für das Projekt 45
16 RoR-Controller für das RCPSP 47
17 RoR-Controller für die Ressourcen 54
18 RoR-Controller für die statischen Seiten 57
19 RoR-Controller für die User 57
20 RoR-Modell für die Vorgangsrelationen 58
21 RoR-Modell für die Vorgangs-Ressourcen-Kombinationen 59
22 RoR-Modell für die Vorgänge 59
23 RoR-Modell für das Projekt 59
24 RoR-Modell für die Ressourcen 60
25 RoR-Modell für die User 60
26 RoR-Seite für die Vorgangsrelationen - Formular 60
27 RoR-Seite für die Vorgangsrelationen - Übersicht 61
28 RoR-Seite für die Vorgangsrelationen - Erstellung 62
29 RoR-Seite für die Vorgangsrelationen - Grafische Darstellung 62
30 RoR-Seite für die Vorgangs-Ressourcen-Kombinationen - Formular 62
31 RoR-Seite für die Vorgangs-Ressourcen-Kombinationen - Bearbeitung 63
32 RoR-Seite für die Vorgangs-Ressourcen-Kombinationen - Übersicht 64
33 RoR-Seite für die Vorgangs-Ressourcen-Kombinationen - Erstellung 64
34 RoR-Seite für die Vorgangs-Ressourcen-Kombinationen - Anzeige 65
35 RoR-Seite für die Vorgänge - Formular 65
36 RoR-Seite für die Vorgänge - Bearbeitung 66
37 RoR-Seite für die Vorgänge - Übersicht 66
38 RoR-Seite für die Vorgänge - Erstellung 68

39 RoR-Seite für die Vorgänge - Anzeige 68
40 RoR-Seite für die Ressourcen - Formular 69
41 RoR-Seite für die Ressourcen - Tabelle als unangemeldeter User 70
42 RoR-Seite für die Ressourcen - Tabelle als angemeldeter User 70
43 RoR-Seite für die Ressourcen - Bearbeitung 72
44 RoR-Seite für die Ressourcen - Übersicht 72
45 RoR-Seite für die Ressourcen - Erstellung 73
46 RoR-Seite für die Ressourcen - Anzeige 73
47 RoR-Seite für die Optimierungsseite zur Projektplanung 73
48 RoR-Seite für die Startseite 76
49 Kopfzeile der Web-Apllikation 76
50 RoR-Seite für die User - Übersichtsseite 77
51 RoR-Seite für die User - Bearbeitung 77
52 RoR-Seite für die User - User-/Mitarbeiterübersicht 78
53 RoR-Seite für die User - Erstellung 78
54 RoR-Seite für die User - Anzeige 79
55 RoR-Datenbankschema 80
56 Beispieldaten für die Datenbank 82

Symbolverzeichnis

d_i	Dauer von Vorgang i
FE_i	frühestes Ende von Vorgang i
$i, h = 1, ..., I$	Vorgänge
k_{ir}	Kapazitätsbedarf von Vorgang i auf Ressource r
kp_r	verfügbare Kapazität von Ressource r je Periode
$\mathcal{N}_i$	Menge der direkten Nachfolger von Vorgang i
oc_r	Kosten einer Einheit Zusatzkapazität von Ressource r
O_{rt}	Zusatzkapazität von Ressource r in Periode t
$r = 1, ..., R$	Ressourcen
SE_i	spätestes Ende von Vorgang i
$t, \tau = 1, ..., T$	Perioden
$\mathcal{V}_i$	Menge der direkten Vorgänger von Vorgang i
$X_{it} \in \{0, 1\}$	gleich 1, falls Vorgang j in Periode t endet, sonst 0

1 Einleitung

Bei einem Projekt handelt es sich um eine zeitlich befristete, relativ innovative und risikobehaftete Aufgabe von erheblicher Komplexität, die meist einer gesonderten Planung bedarf.[1] Dementsprechend von großer Bedeutung ist die vorhergehende und genaue Planung von Projekten.[2] Projektplanung ist die Planung aller Arbeitsgänge eines Projekts durch Zuweisung eines Startzeitpunktes, so dass die Zeitbeziehung zwischen den Vorgängen eingehalten und knappe Ressourcenkapazitäten nicht überschritten werden.[2] Durch das Zerlegen des Projekts in einzelne Arbeitsgänge wird versucht, die Komplexität zu reduzieren und eine geordnete Abfolge der Arbeitsgänge zu erstellen, um das Projektziel zu erreichen.[3] Projektziele können dabei unterschiedlich kategorisiert werden, z. B. in Sach-, Termin- oder Kostenziele.[4]

Nach DIN 69900 hat ein Arbeitsgang oder ein einzelner Vorgang eines Projekts einen definierten Anfang sowie ein definiertes Ende und dient für das Projekt als Ablaufelement zur Beschreibung eines bestimmten Geschehens.[5] Trotz der Zerlegung besitzen die einzelnen Arbeitsgänge des Projekts eine Beziehung, mit der die Reihenfolge der Abläufe bestimmbar ist.[6] Oft wird zur Darstellung der Vorgangsrelationen ein Vorgangsknoten-Netzplan verwendet.[7] Ein Arbeitsgang ist i. d. R. verbunden mit dem Einsatz von Ressourcen, die wiederum mit Kosten verbunden sind. Eine Möglichkeit, das Projektziel unter minimaler Ressourcenverwendung zu erreichen, ist die effiziente Planung der Ablauffolge der Arbeitsgänge eines Projekts.[8] Damit ist es möglich, mehrere Projekte bei einer gegebenen Zeitvorgabe unter Einhaltung von Ressourcenrestriktionen fertigzustellen bzw. bei konstanter Ressourcenkapazität ein Projekt in kürzerer Zeit abzuschließen.

Zur Bestimmung der optimalen Ablauffolge der einzelnen Arbeitsgänge eines Projekts kann ein Optimierungsmodell verwendet werden, mit dem für eine festgelegten Ablauffolge eines Projekts und unter Berücksichtigung der Ressourcenbeschränkung die Fertigstellungszeit minimiert wird. In Kapitel 2 wird eine solche Modellformulierung für das ressourcenbeschränkte Projektplanungsproblem als sogenannte Kapazitätsplanung vorgestellt.[9] Alternativ wird in dem Kapitel das Optimierungsmodell um die Bedingung erweitert, dass Zusatzkapazitätseinheiten gebucht werden können. Mit dieser Modellerweiterung wird von der Kostenplanung in Projekten gesprochen. Bezeichnet wird im Allgemeinen das ressourcenbeschränkte Projektplanungsproblem mit dem englischen Begriff des *Resource-Constrained*

[1] Vgl. Voigt und Schewe (2014)
[2] Vgl. Zimmermann et al. (2006), S. VI
[3] Vgl. Zimmermann et al. (2006), S. 4
[4] Vgl. Felkai und Beiderwieden (2011), S. 52
[5] Vgl. DIN 69900 (2009), S. 15
[6] Vgl. Kellenbrink (2014), S. 6-7
[7] Vgl. Helber (2014), S. 205
[8] Vgl. Bartels (2009), S. 11-12
[9] Vgl. Helber (2014), S. 203

Project Scheduling Problem (RCPSP). Bei dem RCPSP handelt es sich um eine abstrakte mathematische Modellformulierung. Ziel der vorliegenden Arbeit ist es, das RCPSP in *Ruby on Rails (RoR)* zu implementieren. Bei *RoR* handelt es sich um ein Framework zur Entwicklung von Webdokumenten bzw. Internetseiten.[10] Es baut auf der Programmiersprache *Ruby* auf und ist ursprünglich von Yukihiro Matsumoto entwickelt.[11] Die Implementierung bedarf einer Verknüpfung von *RoR* und *GAMS*[12]. Unter *GAMS* wird eine algebraische Modellierungssprache für mathematische Optimierungsprobleme verstanden, mit der das RCPSP gelöst wird.[13] Im Kapitel 3 wird die Entwicklung des Anwendungssystems zum Lösen des RCPSP ausführlich beschrieben. Ergänzt wird diese Arbeit durch eine kritische Würdigung der Web-Applikation in Kapitel 4 sowie einem Fazit in Kapitel 5.

2 Grundlagen zum ressourcenbeschränkten Projektplanungsproblem und zum Framework Ruby on Rails

2.1 Kapazitätsplanung

Ein Großteil an Projekten besitzt die Eigenschaft eines beschränkten Ressourcenkontingents.[14] Soll demgemäß die vorgegebene Terminierung des Projekts als zuvor festgesetztes Ziel erreicht werden, muss neben der Reihenfolgerestriktion auch der Ressourcenbedarf der unterschiedlichen Arbeitsgänge sichergestellt sein. Mit der Einhaltung des Ressourcenbedarfs ist es möglich, alle zur Erfüllung des Projekts notwendigen Arbeitsgänge auszuführen und somit letztendlich das Projekt abzuschließen. Neben limitierten Ressourcen, die während des gesamten Projekts nur ein Mal zur Verfügung stehen, gibt es Ressourcen, die nach einer bestimmten Anzahl von Perioden erneuert werden können.[15] Erneuerbare Ressourcen sind bspw. die Produktionskapazität einer Maschine oder der Personaleinsatz für ein Projekt. In dieser Arbeit wird der Fokus auf erneuerbaren Ressourcen gelegt.

Zur Lösung des ressourcenbeschränkten Projektplanungsproblems wird das Modell RCPSP genutzt. Das RCPSP legt durch Fixierung der Aktivitätsstartzeitpunkte den Projektgrundablauf zur Zielerreichung der Minimierung der Projektdauer fest. Dies geschieht unter Einhaltung der Vorrangsbedingung der einzelnen Arbeitsgänge sowie der Kapazitätsbeschränkung der erneuerbaren Ressourcen.[16] Die im Folgenden aufgestellte Zielfunktion des RCPSP zur Minimierung der Projektdauer ist die gängigste Variante,[17] jedoch sind auch andere möglich.[18]

[10] http://www.rubyonrails.org
[11] http://www.ruby-lang.org/de/about
[12] General Algebraic Modeling System
[13] http://www.gams.com
[14] Vgl. Kellenbrink (2014), S. 11
[15] Vgl. Neumann-Braun et al. (2003), S. 21-22
[16] Vgl. Demeulemeester und Herroelen (2011), S. 23
[17] Vgl. Drexl et al. (1997), S. 98
[18] Vgl. Talbot (1982), S. 1200

Nachfolgend wird das deterministische RCPSP in diskreter Zeit formuliert.[19] Charakteristisch für eine mathematische Modellformulierung in diskreter Zeit sind die Zeiteinheiten, die den Perioden t, τ entsprechen.

Modell RCPSP

$$\min Z = \sum_{t=FE_I}^{SE_I} t \cdot X_{I,t} \tag{1}$$

unter Beachtung der Restriktionen

$$\sum_{t=FE_i}^{SE_i} X_{it} = 1 \qquad i = 1, ..., I \tag{2}$$

$$\sum_{t=FE_h}^{SE_h} t \cdot X_{ht} \leq \sum_{t=FE_i}^{SE_i} (t - d_i) \cdot X_{it} \qquad i = 1, ..., I;\ h \in \mathcal{V}_i \tag{3}$$

$$\sum_{i=1}^{I} \sum_{\tau=\max(t,FE_i)}^{\tau=\min(t+d_i-1,SE_i)} k_{ir} \cdot X_{i\tau} \leq kp_r \qquad r = 1, ..., R;\ t = 1, ..., T \tag{4}$$

$$X_{it} \in \{0, 1\} \qquad i \in \mathcal{I};\ t \in \{FE_i, ..., SE_i\} \tag{5}$$

Es wird ein Projekt betrachtet, das aus I unterschiedlichen Arbeitsgängen besteht. Jeder Arbeitsgang i hat eine definierte Menge von zu erledigenden Vorgängerarbeitsgängen $h \in \mathcal{V}_i$. Des Weiteren ist für die Fertigstellung des Projekts die Abarbeitung der Arbeitsgänge in topologischer Reihenfolge notwendig. D. h. der Vorgänger h hat stets eine kleinere Ordnungszahl als sein Nachfolger i ($h < i$) und muss zur Fortsetzung des Projektverlaufs beendet sein. Die Bearbeitungsdauer eines Arbeitsgangs i wird mit dem Parameter d_i festgelegt. Bei dem RCPSP in diskreter Zeit wird die Annahme getroffen, dass die Dauer durch einen ganzzahligen Parameter abgebildet wird. Der Startzeitpunkt des Projekts ist $t = 0$ und erstreckt sich über einen Gesamtzeitraum von T Perioden. Um die Reihenfolgebedingungen einzuhalten, werden einem Projekt zwei Dummy-Arbeitsgänge „Beginn“ ($i = 1$) und „Ende“ ($i = I$) hinzugefügt, welche mit einer Dauer von 0 Zeiteinheiten bewertet werden.[20] Dadurch findet eine exakte Terminierung des Projektbeginns und Projektendes statt. Der Parameter k_{ir}

[19] Vgl. Domschke und Drexl (2005), Abschnitt 5.5
[20] Vgl. Zimmermann et al. (2006), S. 66

stellt die benötigten Kapazitäten der erneuerbaren Ressource r bei der Durchführung von Arbeitsgang i dar. Die Ressourcen $r \in R$ sind in einer Periode innerhalb des Umfangs ihrer Kapazität kp_r nutzbar. Da es sich um erneuerbare Ressourcen handelt, stehen diese zu jeder neuen Periode in vollem Umfang erneut zur Verfügung. Ungenutzte Ressourcen sind jedoch nicht auf nachfolgende Arbeitsgänge und Perioden übertragbar.[21] Um den Fertigstellungszeitpunkt der einzelnen Arbeitsgänge i festlegen zu können, wird der Modellformulierung in diskreter Zeit die binäre Entscheidungsvariable X_{it} hinzugefügt.[22] Diese Binärvariable nimmt den Wert 1 an, falls der Arbeitsgang i zum Zeitpunkt t beendet wird.

Mittels der Zielfunktion (1) wird der Fertigstellungszeitpunkt des Projekts minimiert. Dafür wird der Zeitraum zwischen dem frühesten und spätesten Fertigstellungszeitpunkt (FE_I und SE_I) aller durchzuführenden Arbeitsgänge I betrachtet. Nebenbedingung (2) stellt sicher, dass ein Arbeitsgang i zwischen dem jeweiligen für diesen Arbeitsgang geltenden frühesten und spätesten Fertigstellungszeitpunkt nur exakt ein Mal durchgeführt wird. Die Reihenfolgerestriktion wird mit der Nebenbedingung (3) eingehalten. Sie stellt sicher, dass jeder Vorgänger $h \in \mathcal{V}_i$ beendet ist, bevor der Arbeitsgang i startet. Der Term $(t - d_i)$ garantiert für den Arbeitsgang i, dass dieser erst beginnt, sobald der Vorgänger h mit der Dauer d_i abgeschlossen ist. Der Parameter kp_r spiegelt die Kapazitätsgrenze für eine erneuerbare Ressource r je Periode t wider. In Nebenbedingung (4) findet zum einen eine formale Darstellung dieser Kapaiztätsbegrenzung statt. Zum anderen wird der Ressourcenverzehr während der gesamten Bearbeitungsdauer der Fertigstellung beachtet, in dem der Kapazitätsbedarf k_{ir} aller Arbeitsgänge I summiert wird. Eben diese Summe wird schließlich durch kp_r beschränkt. Mit der Nebenbedingung (5) wird die Binärvariable X_{it} für den Zeitraum $t = \{FE_i, ..., SE_i\}$ formal definiert. Aufgrund der Reihenfolgebeziehung (3) darf der jeweils betrachtete Arbeitsgang nur in diesem Zeitraum fertiggestellt werden. Die gemischtganzzahlige Modellformulierung lässt sich durch Standard-Lösungsverfahren exakt lösen.[23]

2.2 Kostenplanung

Aufbauend auf der Kapazitätsplanung kann das RCPSP um die Nutzung von Zusatzkapazitäten der Ressourcen erweitert werden, damit dem Projektplanungsmodell gestattet ist, den Vorgängen zusätzliche Kapazitätseinheiten der notwendigen Ressourcen bereitzustellen. Die Kapazitätsrestriktion wird dementsprechend um die Entscheidungsvariable $O_{rt} \geq 0$ erweitert. Die Variable O_{rt} beschreibt die Einheiten an Zusatzkapazitäten einer Ressource r in der Periode t. Im Vordergrund steht nicht die Einhaltung der verfügbaren Kapazitäten, sondern die aufgewendeten Zusatzkosten des Projekts unter Beachtung der Projektstruktur. Dem Optimierungsmodell ist es damit gestattet durch Erhöhung der Kapazitäten der Ressourcen

[21] Vgl. Kellenbrink (2014), S. 12
[22] Vgl. Pritsker et al. (1969), S. 94
[23] z. B. mittels eines Branch-and-Bound-Verfahrens, Vgl. Kellenbrink (2014), S. 14

die anfängliche Ressourcenbeschränkung zu umgehen. Zusätzlich wird bei der Modellerweiterung der Kostenplanung der Parameter oc_r eingeführt, der für eine betrachtete Ressource r die Kosten einer Einheit der Zusatzkapazitäten beschreibt. Ziel des Optimierungsmodells ist es die Kosten des Projekts zu minimieren. Dieses Modell zur Kostenplanung dient somit als Entscheidungsunterstützung, ob die Einbeziehung von Zusatzkapazitäten zu einer Verbesserung des Projektziels führt. Es handelt sich um den Trade-off des frühzeitigen Erreichens des Projektziels durch Nutzung von Zusatzkapazitäten und der gesamten Projektkosten, die für das Projekt aufgewendet werden sollen.

Nachfolgend wird das deterministische RCPSP+ in diskreter Zeit formuliert.

Modell RCPSP+

$$\min Z = \sum_{t=1}^{T} \sum_{t=1}^{R} oc_r \cdot O_{r,t} \tag{6}$$

unter Beachtung der Restriktionen (2), (3), (5) sowie

$$\sum_{i=1}^{I} \sum_{\tau=\max(t,FE_i)}^{\tau=\min(t+d_i-1,SE_i)} k_{ir} \cdot X_{i\tau} \leq kp_r + O_{rt} \qquad r = 1, ..., R;\ t = 1, ..., T \tag{7}$$

$$O_{rt} \geq 0 \qquad r = 1, ..., R;\ t = 1, ..., T \tag{8}$$

Bei dem RCPSP+ wird die Zielfunktion insoweit formuliert, dass über alle Perioden $t \in T$ und über alle Ressourcen $r \in R$ die Summe der Kosten oc_r für die Anzahl an notwendigen Einheiten an der Zusatzkapazität O_{rt} minimiert wird. Die Nebenbedingungen (2) und (3) bleiben bestehen, so dass jeder Vorgang exakt einmal zwischen dem frühesten und dem spätesten Ende (FE_i bzw. SE_i) fertiggestellt und die Topologie der Vorgänge eingehalten wird. Weiterhin gilt die Nebenbedingung (5), dass es sich bei der Entscheidungsvariable X_{jt} um eine binäre Variable handelt. Erweitert wird das RCPSP aus der Kapazitätsplanung mit einer modifizierten Nebenbedingung zur Einhaltung der Kapazitätsbeschränkung. Mit der Nebenbedingung (7) wird die Kapazitätsrestriktion für eine Ressource $r \in R$ in einer Periode $t \in T$ eingehalten, jedoch ist es dem Modell gestattet, die vorhandene Ressourcenkapazität kp_r um die Ausprägung der Entscheidungsvariable O_{rt} zu erweitern. Durch Lösen der LP-Relaxation wird der Ablaufplan des Projekts unter Beachtung der unterschiedlich zulässigen Gesamtdauern SE_I generiert. Weiter wird für jede Ressource $r \in R$ zur jeweiligen Periode $t \in T$ die notwendige Anzahl an benötigten Zusatzkapazitäten O_{rt} ermittelt. Die Nebenbedingung (8) beschreibt die Eigenschaft der Entscheidungsvariable O_{rt}. Es handelt sich um eine positive Variable bzw. einen Nullwert.

2.3 Ruby on Rails

Das Framework *Ruby on Rails (RoR)* zur Entwicklung von Web-Applikationen mit Datenbankbezug wurde von David Heinemeier Hansson im Jahre 2004 erstmals vorgestellt.[24] Mit dem Namen von RoR wird klar, dass das Framework die Programmiersprache *Ruby* nutzt. *Ruby* wird von den meisten gängigen Betriebssystemen unterstützt (Microsoft Windows, Apple Mac OS X, Linux, etc.) und ist bspw. in dem Betriebssystem Apple Mac OS X in der Version 1.8.7 standardmäßig integriert.[25] Bei *Ruby* handelt es sich um eine objekt-orientierte Programmiersprache mit dem Grundsatz *principle of least surprise* und folgt einigen Besonderheiten, wie z. B. einer einfachen Sprachsyntax, keiner typisierten Variablen und einer reinen Objektorientierung.[26] Abbildung 1 zeigt das Terminal von Apple Mac OS X mit typischen *Ruby*-Kommandobefehlen. *RoR* nutzt diese einfache Syntax zur Entwicklung von Web-Applikationen, wobei aufgrund der einfachen Bedienung auf integrierte Entwicklungsumgebungen zurückgegriffen wird, wie z. B. RadRails oder RubyMine.[27]

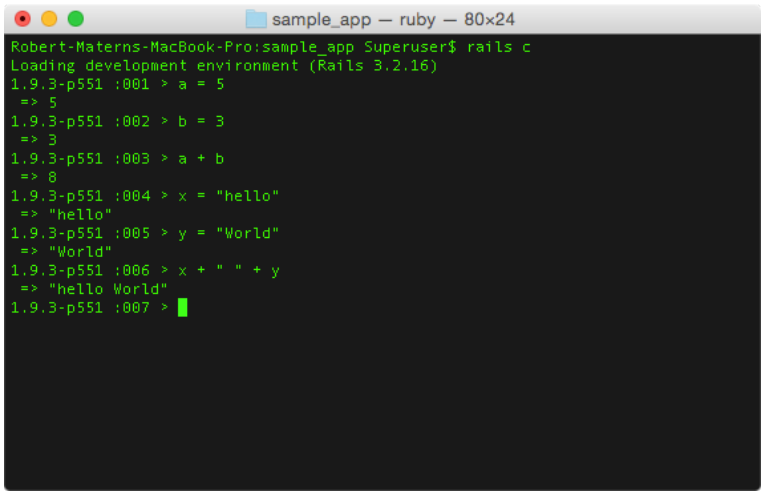


Abbildung 1 Terminalfenster unter Apple Mac OS X

Mit Hilfe des *RoR*-Frameworks lassen sich dadurch schnell Web-Applikationen mit Datenbankbezug entwickeln, wobei der wesentliche Vorteil in der Softwarearchitektur des Model-View-Controller-Paradigmas liegt.[28] Das Paradigma besagt, dass eine durch einen Browser angestoßene Anfrage an den Server durch den *RoR* `controller` verarbeitet wird. Der `controller` verarbeitet die Anfrage und leitet die nachfolgenden Schritte ein. Bei Web-Applikationen erfolgt eine solche Verarbeitung durch Anzeigen bzw. dem sogenannten *Rendern* von HTLM-Dokumenten der *RoR* `views` , die von Browsern angezeigt werden können. Der `controller` rendert die `views` und ermöglicht weitere *RoR*-Befehle im HTML-Dokument.

[24] Vgl. Grimmer (2006)
[25] Vgl. Wintermeyer (o. J.)
[26] Vgl. Walter (2008b), S. 297-298
[27] Vgl. Hartl (2012), S. 10
[28] Vgl. Walter (2008a), S. 463

Bei komplexen und dynamischen Seiten übernimmt der `controller` geforderte Daten aus den *RoR* `models`, die wiederum mit einer Datenbank verbunden sind. Durch diese Architektur lassen sind umfangreiche und an spezifische Anfragen angepasste Web-Applikationen entwickeln. Ein weiterer Vorteil von *RoR* ist die einfache Implementierung von Unterprogrammen. Ein Unterprogramm ist in *Ruby/RoR* ein `gem`, das durch den Bundler zur bestehenden Web-Applikation hinzugefügt wird und diese um weitere Konsolenbefehle erweitert.[29] Im nachfolgenden Kapitel wird die Entwicklung einer Web-Applikation mittels *RoR* beschrieben. Dabei liegt die Besonderheit der Ausarbeitung auf der Beschreibung der Schnittstelle zwischen *RoR* mit dem Programm *GAMS*, damit das in diesem Kapitel vorgestellte Projektplanungsprobem gelöst werden kann, und der Integration eines notwendigen Unterprogramms (`gem`) zur Analyse der Daten des Projektplanungsproblems. Zusätzlich ist in dem Kapitel beschrieben, inwieweit ein weiteres Programm implementiert wird, damit die Projektplanung als Vorgangsknoten-Netzplan dargestellt werden kann.

3 Implementierung des ressourcenbeschränkten Projektplanungsproblems mittels des Frameworks Ruby on Rails

3.1 Installation der Web-Applikation und der notwendigen Programme

Die Applikation kann über nachfolgenden Terminalbefehl auf ein lokales Computerverzeichnis geklont werden.

```
$ git clone https://github.com/rb4k/as-rcpsp.git
```

Anschließend ist eine Migration und Population der Datenbank erforderlich. Zusätzlich zur Web-Applikation (inkl. *RoR* in Version 1.9.3) müssen die Programme *GAMS* zur Lösung von mathematischen Optimierungsmodellen und *GraphViz* zur Visualisierung der Vorgangsbeziehungen auf dem lokalen Computerverzeichnis installiert sein. Die Programme können unter nachfolgenden Links bezogen werden:

- *GAMS*: `http://www.gams.com/download/`
- *GraphViz*: `http://www.graphviz.org/Download_windows.php`

[29] Vgl. Hartl (2012), S. 9-17

3.2 Darstellung der Funktionsweise der Web-Applikation anhand eines Guides für User

Die Funktionsweise der mit *RoR* programmierten Web-Applikation *„Projektplanung“* zur Lösung der Kapizitäts- und Kostenplanung des *RCPSP* lässt sich am anschaulichsten mit Hilfe eines Userguides darstellen. Neben der Besonderheiten, die durch das Problem der Projektplanung auftreten, können im selben Zuge auch die Spezifika der einzelnen Benutzerrollen aufgezeigt werden. Beachtet werden muss, dass die hier vorgestellte Web-Applikation auf der Arbeit von Hartl (2012) aufbaut.

Zunächst wird die Anwendung aus der Sicht eines Anwenders betrachtet, der sich nicht in die Applikation per Benutzererkennung eingeloggt hat. Konkret wird darunter ein potentiellen Mitarbeiter des entsprechenden Projektes verstanden, der sich über die Projektplanung informieren möchte, um sich gegebenenfalls als Mitarbeiter im Projekt (User) anzumelden. Im Testbetrieb wird der *Ruby*-Server gestartet und durch Eingabe der URL `http://localhost:3000/` in die Adresszeile eines beliebigen modernen Browers wird die Startseite der Projektplanung angezeigt (siehe Abbildung 2). Alternativ ist der Betrieb auf einem Webserver möglich, sofern die benötigte Software installiert und betriebsbereit ist. Auf der Startseite hat der User zum einen die Möglichkeit, sich anzumelden bzw. sich einzuloggen, für den Fall, dass er bereits User der Anwendung ist.

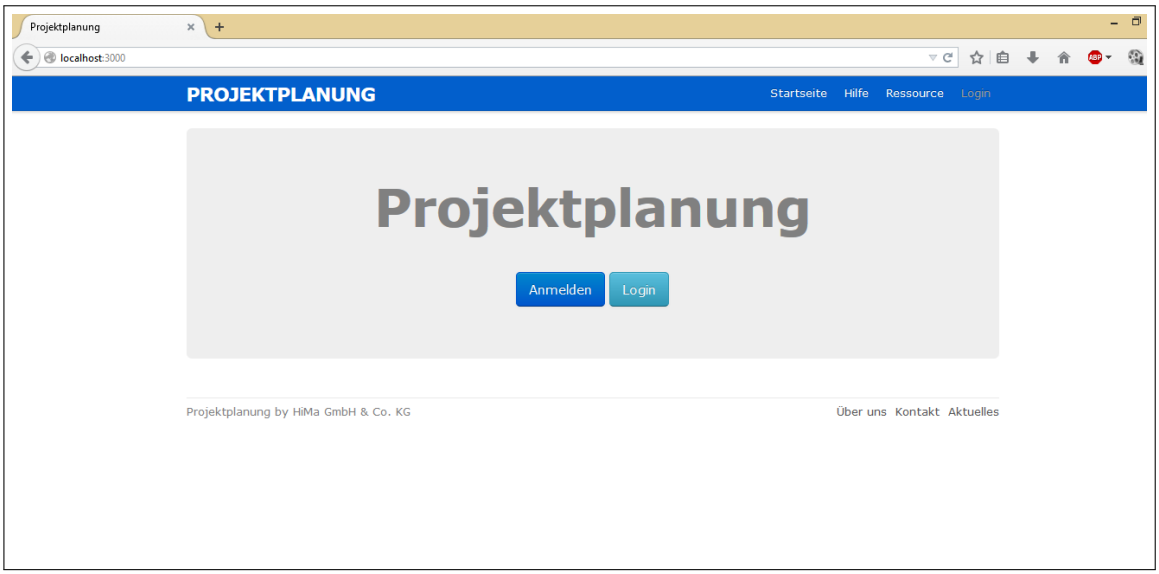


Abbildung 2 Startseite der Web-Applikation Projektplanung

Bei der Startseite (`home.html.rb`) der *RoR*-Applikation handelt es sich um eine statische Seite (`static_pages`) der `views`. Weiter gehören zu dieser Kategorie der HTML.RB-Dokumente die Seiten `about`, `contact`, `help` und `rcpsp`. Letztere wird zum späteren Zeitpunkt thematisiert. Ein Beispiel einer statischen Seite eines *RoR* `views` liefert Quellcode 48 im Anhang A.3.

Anhand der `static_pages` kann die Besonderheit von *RoR* deutlich gemacht werden. Durch das Model-View-Controller-Paradigma hilft der `static_pages_controller` bei der Verarbeitung von Anfragen. Es handelt sich hier um das typische Scaffolding (Bauprinzip) in *RoR*, bei dem `controller`, `models` und `views` erstellt werden.[30] Generiert werden können die Scaffolds durch einen *Ruby*-Befehl im Terminalfenster.

```
$ rails generate scaffold <name> <datenname:datentyp>
```

Wie der Name aber schon andeutet, bedarf es bei den statischen Seiten kaum der Verarbeitung von Datensätzen der *RoR* `models` zur Erstellung von dynamischen Seiten, wie der Quellcode 18 im Anhang A.3 zeigt.

Für die `static_pages` bedarf es einen speziellen *Match*, der in der `config/routes.rb` Datei hinterlegt wird (Vgl. Quellcode 11 im Anhang A.3). Die `config/routes.rb` ordnet den Scaffolds und HMTL-Dokumente spezifische Verzeichnisse in der Applikation zu. *RoR* erkennt die Unterseiten der angelegten Scaffolds und ermöglicht die Verlinkung der Seiten auch ohne spezifische Angaben (Vgl. Quellcode 11 im Anhang A.3).

Mit dem Link *Anmelden* erfolgt die Weiterleitung von der Startseite zur Anmeldeseite. Beschließt sich der Besucher der Seite, sich für das Projekt anzumelden, muss er alle Felder des Anmeldeformulars befüllen.

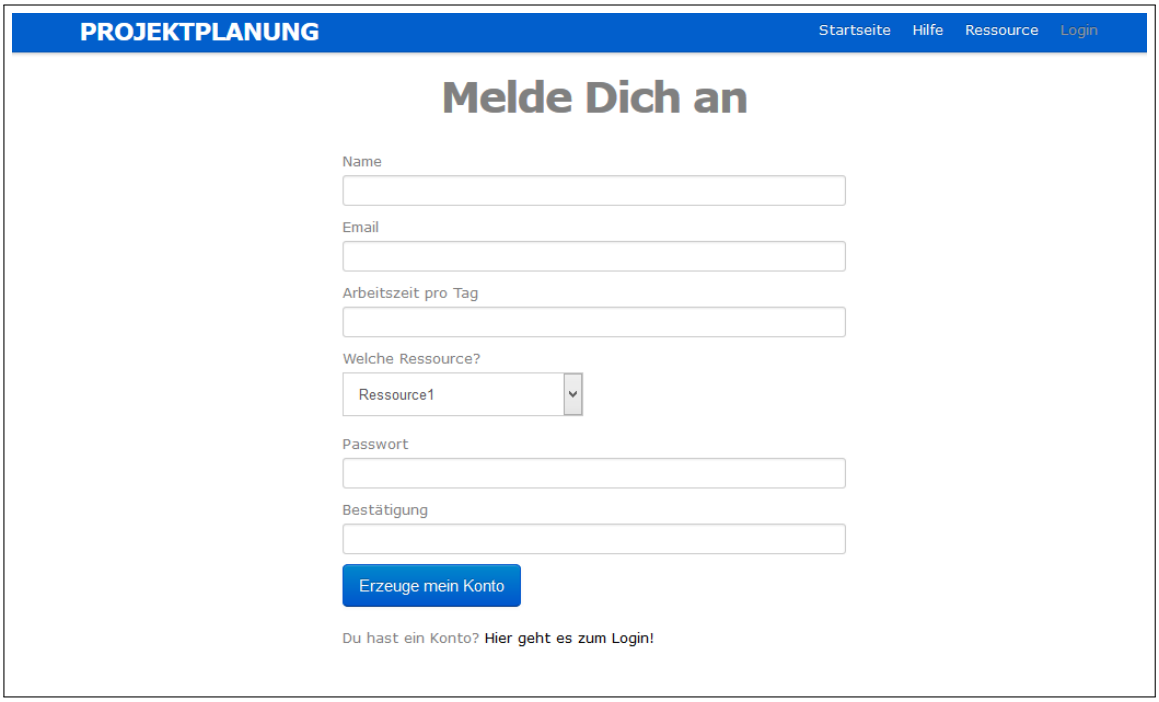


Abbildung 3 Anmeldebildschirm

[30] Vgl. Walter (2008a), S. 464

Neben dem Namen, einer Mailadresse und eines konformen Passwortes sind projektspezifische Informationen zur erfolgreichen Registrierung nötig. Im Feld *Arbeitszeit pro Tag* muss ein entsprechender Wert eingegeben werden, den der neue User bereit ist, pro Tag für das Projekt an Zeit zu investieren. Wird in diesem Feld keine ganze Zahl, sondern eine Dezimalzahl oder ein Wort eingegeben, kann die Anmeldung im System nicht stattfinden. Es wird ein Fehler angezeigt, der das Defizit aufzeigt und behoben werden muss (siehe Abbildung 4). Ausgelöst wird dieser Fehler durch einen Vermerk im zugehörigen *RoR* `models`, dass es sich um eine ganzahlige Zahl handelt (*Integer*). Der Quellcode 25 im Anhang A.3 zeigt dies anhand des hier betrachteten Beispiels.

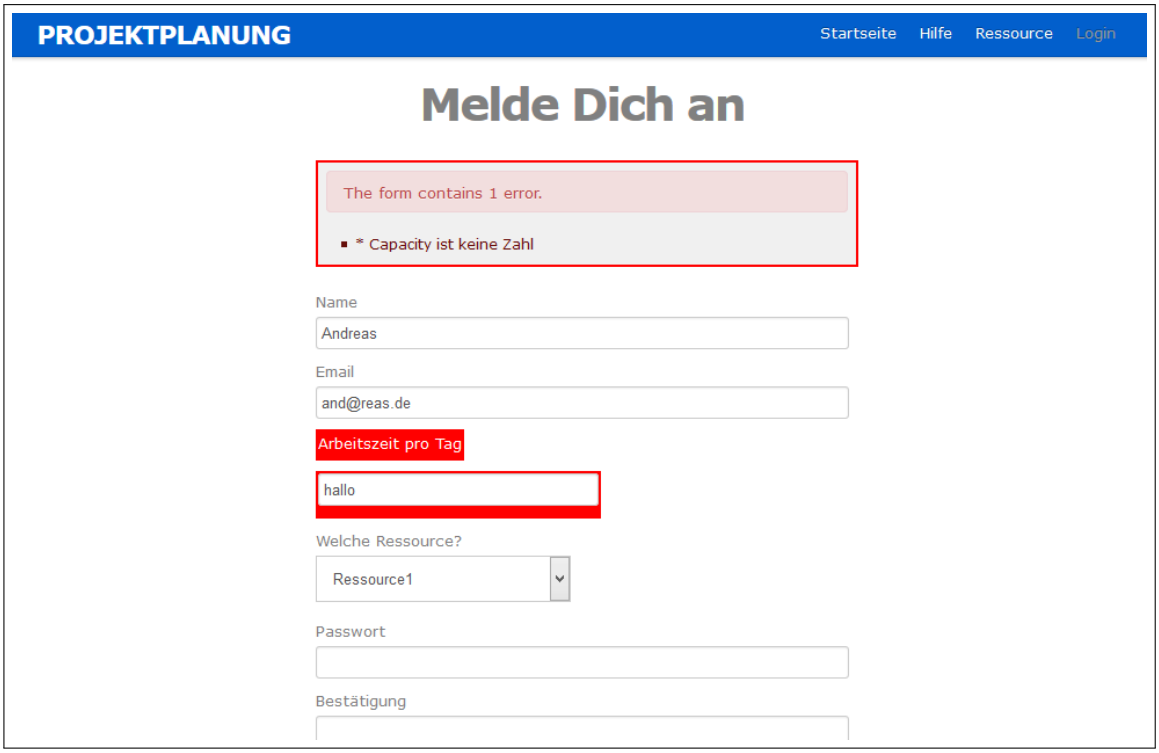


Abbildung 4 Fehleranzeige bei der Anmeldung

Auf der Startseite (sowie allen anderen Seiten) sind neben Links wie *Hilfe* und *Kontakt* auch der Link *Ressourcen* zu finden, der zur Ressourcen-Übersicht verweist. In dieser Übersicht sind alle Ressourcen des Projektes gelistet. Hier kommt der Grundsatz von RoR zum Tragen: *Don't repeat yourself*. Die Gestaltung und der Aufbau einer jeden Seite in der Web-Applikation orientiert sich anhand der CSS-Stylesheets bzw. der Layout-Dateien. Die Layout-Dateien sind unter `app/views/layouts` gelistet und definieren auf jeder Seite spezifische Bereiche. Die `application.html.erb` generiert für jede Seite dieses einheitliche Layout, unterstützt durch die Dateien `_footer.html.erb` und `_header.html.erb`. Im `_header.html.erb` ist der Link zur Ressourcen-Übersicht vermerkt (Vgl. Quellcode 49 im Anhang 49).

Der `_header.html.erb` zeigt einige *If*-Befehle, mit denen unterschiedliche Daten anhand der Eigenschaften unangemeldeten, angemeldeten und Admin-Usern angezeigt werden. Durch Folgen des Links *Ressource* wird die Index-Seite des *RoR* `views/ressources` aufgerufen (Siehe Abbildung 5). Der Quellcode 44 im Anhang A.3 zeigt die notwendige Programmierung für die Seite. *RoR* durchläuft aufgrund der Aktivierung des Links die Aktion *index* des dazugehörigen Controllers `resources_controller.rb` und generiert die zugehörige HTML-Seite (`views`). Die Indexseite prüft, ob der aktuelle User angemeldet ist. Abhängig dieser Entscheidung integriert *RoR* unterschiedliche Seiteninhalte. Sofern der aktuelle User nicht angemeldet ist, wird eine vereinfachte Ressourcen-Übersicht angezeigt. In dieser Ansicht sind alle jeweilig aktuellen Ressourcen mit zugehörigen Namen aufgelistet, sowie dem Link *Bewerben*, der wiederum mit der Anmeldeseite verlinkt ist.

Abbildung 5 Übersicht der Ressourcen für unangemeldete User

Findet keine Anmeldung in die Web-Applikation statt, sind keine weiterführenden Tätigkeiten möglich. Die Startseite liefert keine weiterführenden Informationen und bei der Eingabe von anderen Links in die Adresszeile des Browers wird der aktuelle User zur *Login*-Seite geführt, da alle Daten für nicht angemeldete Anwender gesperrt sind. Um die Applikation nutzen zu können, ist demzufolge die Anmeldung als User zwingend notwendig. Findet diese entweder nach erstmaliger Registrierung über den Link *Anmelden* oder über *Login* statt, wird die eigene Profilseite angezeigt (siehe Abbildung 6).

Die Profilseite gibt einen Überblick über all die Daten, die für den User in Hinblick auf das Projekt relevant sind. Es werden die Daten dargestellt, die bei der Anmeldung hinterlegt sind (Arbeitszeit, Rolle im Projekt) sowie die Vorgänge, die durch die Wahl der Ressource für diesen User relevant sind, in denen er also tätig sein muss. Der Aufbau der Seite ist im Quellcode 54 im Anhang A.3 dargestellt.

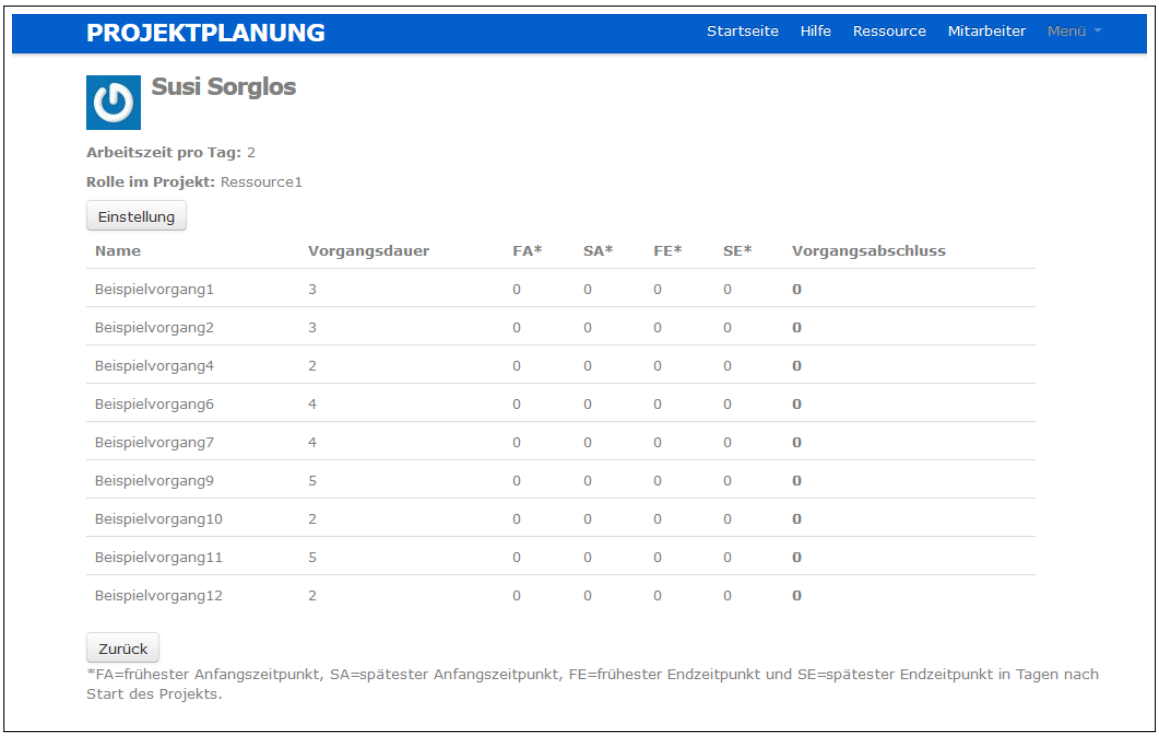


Abbildung 6 Profilseite eines Users

Zu jedem Vorgang wird die Dauer und gegebenenfalls die Zeitspanne angegeben, wann er jeweils stattfindet. Die Grenze liegt zwischen dem frühesten Startzeitpunkt FA_i und spätesten Endzeitpunkt SE_i des Vorgangs i. Ebenfalls wird der Vorgangsabschluss angezeigt. Dieser zeigt für die aufgeführten Vorgänge den Endzeitpunkt nach Start des Projekts unter Einhaltung der Ressourcenbeschränkung an, jeweils in Zeiteinheiten. Ob diese Tabelle mit Daten gefüllt ist, hängt davon ab, ob das Kapazitäts- bzw. Kostenplanungsproblem bereits gelöst ist. Möchte der User seine Daten, wie z.B. die Wahl der Ressource oder die Quantität der Arbeitszeit ändern, gelangt er über den Button *Einstellung* zu einer Seite, die äquivalent aufgebaut ist wie die Anmeldeseite, um dort seine Daten zu aktualisieren. Nach korrekter Eingabe können die Daten über den Button *Speichere Änderungen* gesichert werden. Auf der Profilseite erscheint daraufhin eine Anzeige *Profil updated* mit der Bestätigung, dass das Profil aktualisiert ist. Im Vergleich zum fremden Anwender gelangt der angemeldete User außerdem in der Kopfzeile über den Link *Mitarbeiter* auf eine Übersicht aller Mitarbeiter, die für das Projekt auf dieser Applikation angemeldet sind. Die Profilseite jedes Mitarbeiters kann betrachtet werden mit all den Informationen, die auch auf der eigenen Profilseite einzusehen sind. Es können jedoch keine Änderungen vorgenommen werden. Neben der Verlinkung zu der Übersicht der Mitarbeiter lässt sich in der Kopfzeile ein Feld *Menü* finden, das die Unterpunkte *Profil*, *Einstellungen* und *Logout* enthält. Die Verlinkung *Profil* stellt eine Verlinkung zur Profilseite dar, unter *Einstellungen* kann das eigene Profil aktualisiert werden.

Unter *Ressourcen* kann der User, wie auch der nicht angemeldete Anwender, zur Übersicht der vorhandenen Ressourcen gelangen. Die Anzeige stellt sich für den angemeldeten User jedoch vielfältiger dar als für den einfachen Anwender (siehe Abbildung 7), da hier ein anderer Quellcode integriert ist (siehe Quellcode 44 und Quellcode 42 im Anhang A.3).

Abbildung 7 Übersicht der Ressourcen für User

Für den User sind alle Eigenschaften der verschiedenen Ressourcen einsehbar. Es werden die Gesamtkapazität, Kosten pro ME, Grundkosten und Zusatzkosten pro ME angezeigt. Ist bereits eine Lösung für das Problem der Kostenplanung ermittelt, werden die kalkulierten Werte für die Zusatzeinheiten, gesamten Zusatzkosten und die Gesamtkosten pro Ressource dargestellt. Zudem wird der Zielfunktionswert, bei der Kostenplanung die gesamten anfallenden Zusatzkosten, in Verbindung mit dem Zeitpunkt der Optimierung über der Tabelle dargestellt. Die Darstellung der Tabellen in dieser Web-Applikation orientiert sich prinzipiell an dem Bootstrap-Framework[31]. Alternativ bietet die App die Anzeige der Tabellen anhand einer JavaScript-Tabelle.[32] Über den Button *Anzeigen* in der hier betrachteten Tabelle sind die Eigenschaften einer Ressource separat einsehbar. Da der User bzw. Mitarbeiter in diesem Modell durch die Planung innerhalb des Projekts eingeteilt wird und seine Rechte nicht über die Organisation der eigenen Daten hinaus reicht, hat er keine weiteren Kompetenzen bei der Nutzung dieser Applikation.

[31] http://getbootstrap.com

[32] Auf Implementierung ist jedoch aufgrund der möglichen Inkompatibilität zu bestimmten Browser und aufgrund der Laufzeitverbesserung der Web-Applikation verzichtet.

3.3 Darstellung der Funktionsweise der Web-Applikation anhand eines Guides für Administratoren

Die Verwaltung der Mitarbeiter und die Organisation sowie Durchführung der Projektplanung kann ausschließlich nach der Anmeldung als Administrator (Admin) erfolgen. Der Admin gilt in dieser Anwendung als durchführende Gewalt. In dieser Testsituation ist *Example User* mit den dafür notwendigen Berechtigungen ausgestattet. Alternativ lässt sich durch Änderung der booleschen Variable `admin = true` der Datenbank zum RoR `models/users` die Eigenschaft auch auf andere Datensätze (User) übertragen. Nach der erfolgreichen Anmeldung als User mit administrativen Rechten erscheint zunächst erneut die Profilseite, sofern die Anmeldung über die Startseite erfolgt. Im Gegensatz zu normalen Usern bietet die Seite eines Admins jedoch zusätzliche Handlungsspielräume neben der einfachen Auflistung der Vorgänge (siehe Abbildung 8). Er hat die Möglichkeit, die Dauer der Vorgänge abzuändern oder Vorgänge aus dem Projekt zu löschen. Dies wird wieder über einen *If*-Befehl gesteuert, wie der Quellcode 54 aus dem Anhang A.3 zeigt.

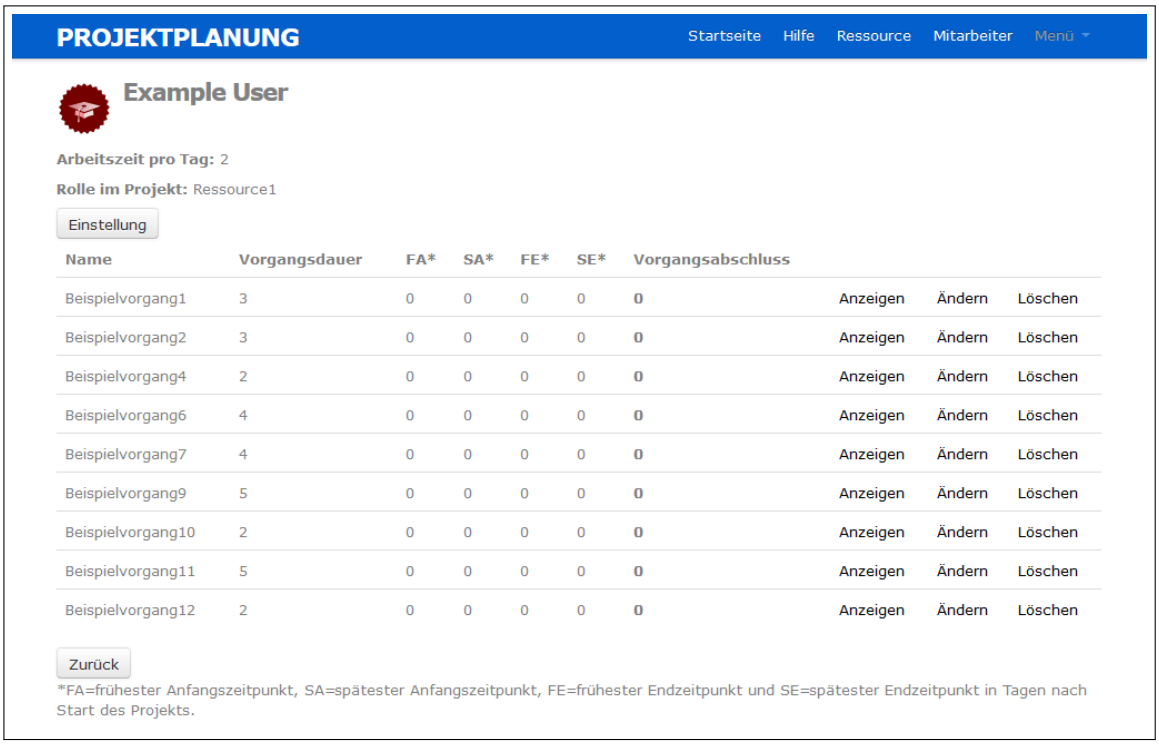


Abbildung 8 Profilseite des Administrators

In gleicher Weise stellt sich die Ausweitung der Kompetenzen bei den Ressourcen dar. Über die Auswahl des Links *Ressourcen* im *Header* existiert eine Verbindung zur Ressourcen-Übersicht. Dort können nun die Ressourcen ebenfalls gelöscht oder die Eigenschaften (Kosten und Zusatzkosten je ME) verändert werden. Des Weiteren kann über den Button *Neue Ressource anlegen* eben dies vollzogen werden (Vgl. Quellcode 44 im Anhang A.3).

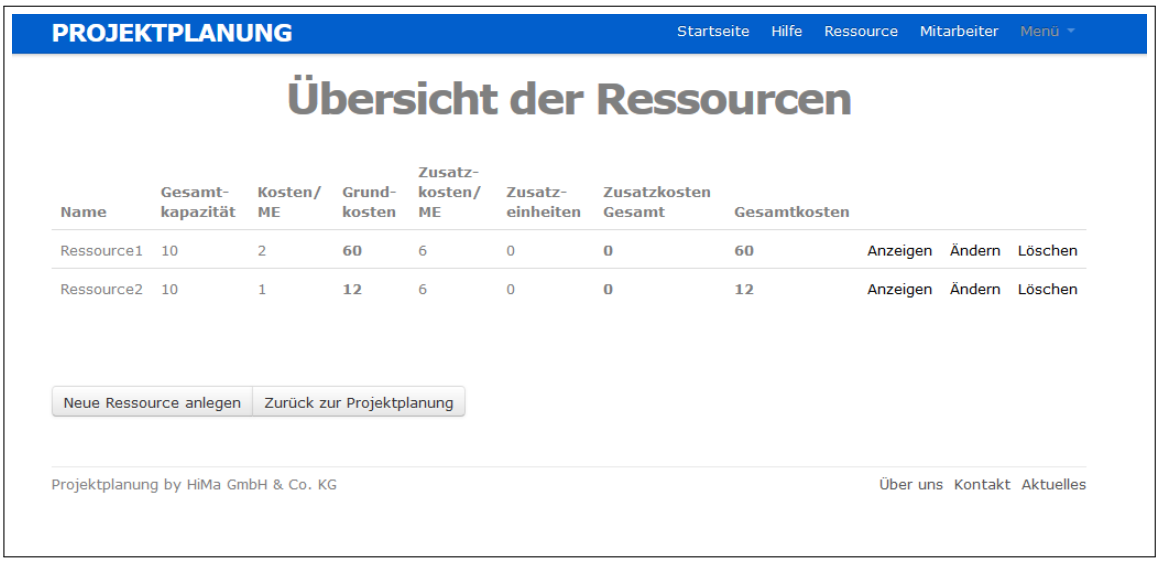


Abbildung 9 Übersicht der Ressourcen aus Sicht des Administrators

Um nun zur Kernaufgabe der Applikation, der Projektplanung zu gelangen (siehe Abbildung 10), kann entweder der Button *Zurück zur Projektplanung* auf der Seite zur Ressourcen-Übersicht getätigt oder ausgehend von jeder beliebigen Seite der Web-Applikation in der Kopfzeile (*Header*) unter *Menü* der Unterpunkt *Projektplanung* ausgewählt werden. Bei dieser statischen Seite fließen Daten aus dem RoR `models/project` ein. Bei diesem Modell handelt es sich um eine Hilfsdatenbank ohne weitere Beziehung zu anderen Modellen (Vgl. Abbildung 17 im Anhang A.1). Sie fungiert als Datenbank für unabhängige Parameter und hat damit nur einen Datensatz. Der Controller der statischen Seiten ruft über die Aktion `rcpsp` diesen Datensatz auf (Vgl. Quellcode 18 aus Anhang A.3). Dadurch kann das HTML.RB-Dokument `views/static_pages/` den Datensatz aufgreifen und Formularfelder zur Eingabe der unabhängigen Parameter bereitstellen. Zu den unabhängigen Parametern dieses Formulars zählen die Datenfelder `path`, `gvp`, `startdate` und `deadline`, auf die im Verlauf der weiteren Beschreibung der Web-Applikation näher eingegangen wird.

Im oberen Bereich der Seite sind die Verlinkungen zur Verwaltung der nötigen Inputs zur Lösung beider Planungsproblematiken angesiedelt. Neben den bereits behandelten Links zu den Vorgängen und Ressourcen finden sich Verlinkungen zu den *Vorgangsrelationen* und *Vorgang-Ressourcen-Kombinationen*.

Die Übersicht der Relationen zwischen den Vorgängen stellt eine Auflistung eines jeden Vorgängers und Nachfolgers dar. Durch anklicken auf den Button *Graph anzeigen* wird die aktuelle Projektstruktur mittels Vorgangsknoten-Netzplan dargestellt (siehe Abbildung 11). Die Implementierung dieses Graphen wird in Kapitel 3.4 beschrieben. Zur Anzeige des Graphen muss im Vorfeld die Web-Applikation mit dem Programm *GraphViz* verbunden sein.

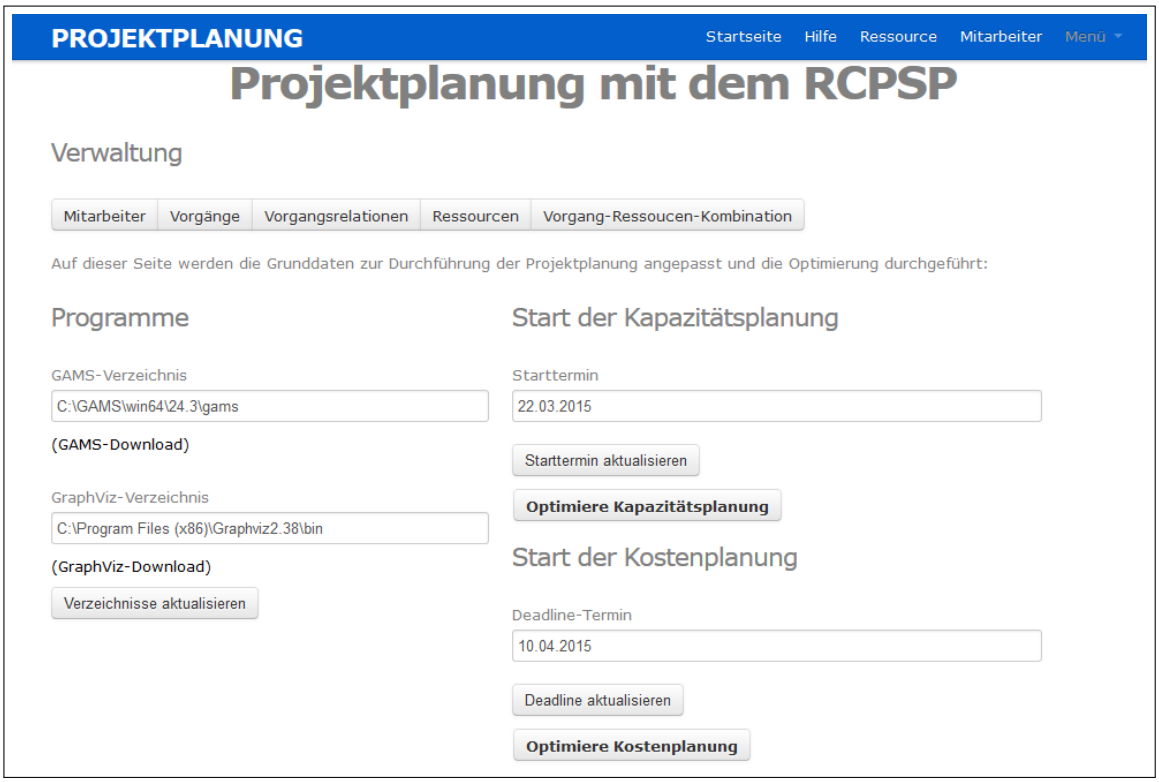


Abbildung 10 Projektplanung mit dem RCPSP - Übersicht

Ein Admin kann eine Relationen löschen oder neue anlegen. Wenn er sich dazu entschließt, eine neue anzulegen, ist zu beachten, dass ein Strukturplan eines Projektes keine Zyklen beinhalten darf. Damit Zyklen verhindert werden, findet beim Prozess des Anlegens einer neuen Vorgangsrelation eine Prüfung statt. Beinhaltet die neu angelegte Relation einen Zyklus, tritt ein Fehler auf und die Relationen muss überarbeitet werden (siehe Abbildung 12). So wird verhindert, dass der Strukturplan Zyklen enthält. Dieser Vorgang wird gesteuert durch die dafür zuständige Aktion `create` aus dem *RoR* `procedure_procedures_controller` (Vgl. Quellcode 12 im Anhang A.3). Inbegriffen in dieser Aktion ist ein frei-verfügbares Unterprogramm (`gem 'rgl'`[33]). In Kapitel 3.4 wird die Integration und Funktionsweise dieses Unterprogramms beschrieben.

Der Button *Vorgang-Ressourcen-Kombination* führt zu eben dieser Übersicht. Neben der Auflistung können die Kombinationen verändert, gelöscht oder neu erstellt werden. Bei der Veränderung oder Erstellung ist zu beachten, dass die Angabe des Kapazitätsbedarfs nur mit Hilfe einer ganzen Zahl erfolgen darf. Entsprechend der Kapazitätsangabe bei der Bearbeitung eines Profils erscheint bei jeder anderen Art von Eingabe ein Fehler, der die Datenspeicherung verhindert (Vgl. Quellcode 21 im Anhang A.3).

[33] https://github.com/monora/rgl

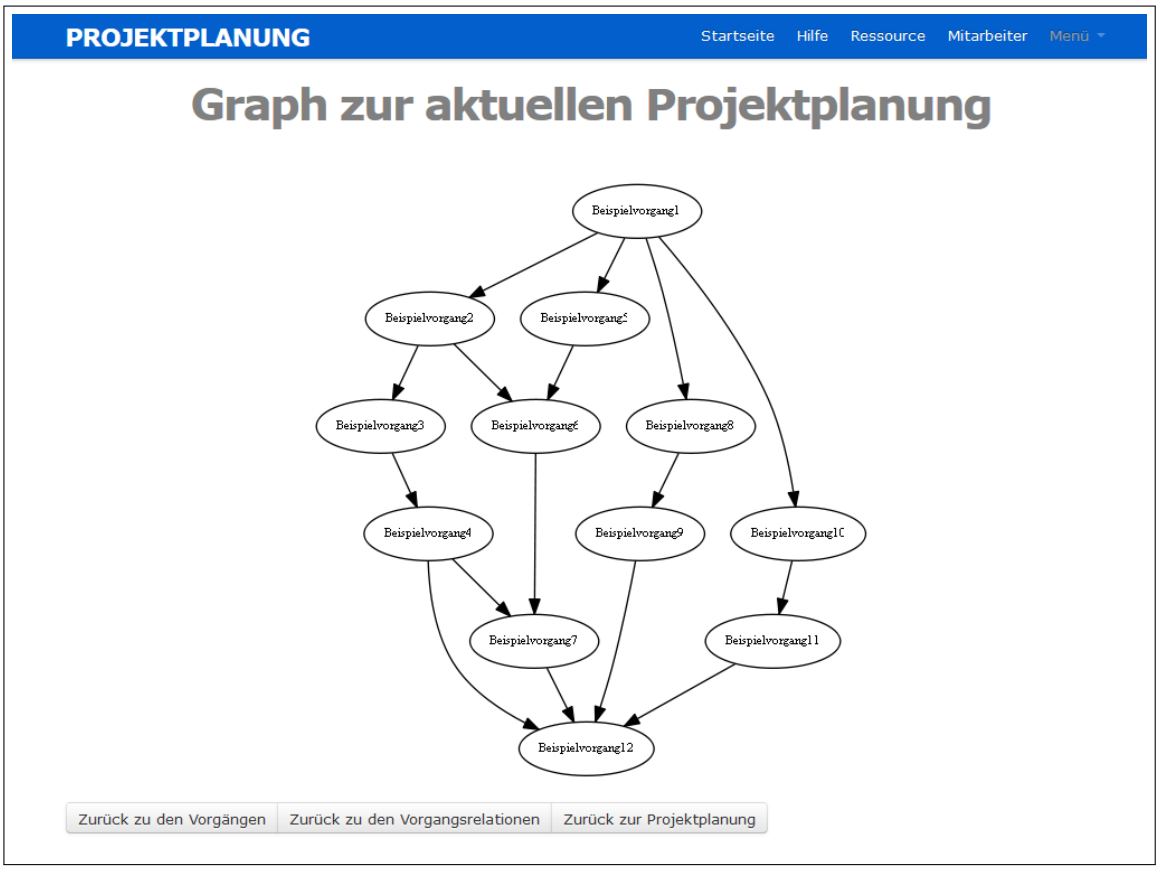


Abbildung 11 Vorgangsknoten-Netzplan

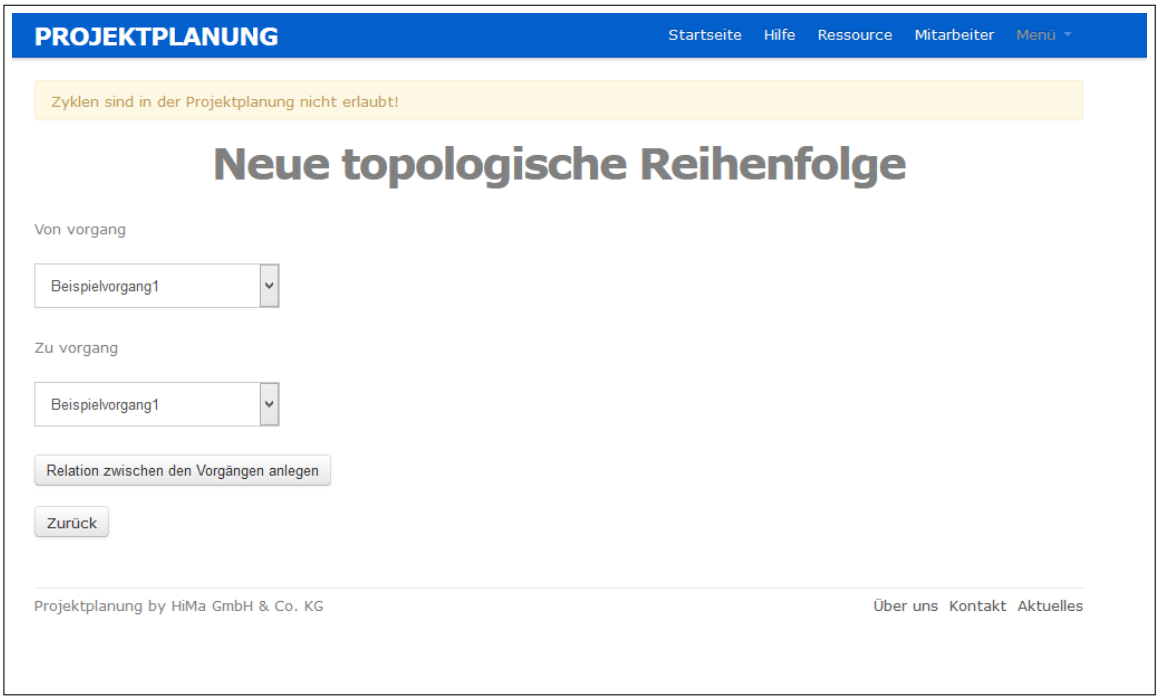


Abbildung 12 Fehler aufgrund eines Zyklus in der topologischen Reihenfolge

Nachdem all diese Daten geprüft und gegebenenfalls verändert sind, steht die Basis sowohl für die Optimierung der Kapazitäts- als auch der Kostenplanung. Bevor der Optimierungsprozess stattfinden kann, müssen noch einige Rahmenbedingungen geprüft werden. Da die Optimierung mit dem Programm *GAMS* stattfindet, muss die Applikation auf dieses Programm zurückgreifen können. Dafür muss *GAMS* auf dem hiesigen Computer installiert sein. Nach der Recherche des Installationsortes muss der korrekte Pfad in das dafür vorgesehene Feld der Übersichtsseite zur Projektplanung eingetragen werden, in dem der Beispielpfad zu sehen ist. Die Verbindung der Web-Applikation mit *GraphViz* erfolgt ebenfalls durch Eingabe des Verzeichnisses. Nach der Eingabe werden die Pfadzugriffe durch *Verzeichnisse aktualisieren* in der Datenbank des RoR `models/project` gesichert (siehe Abbildung 10). Neben den Programmpfaden muss ein Termin ausgewählt werden, zu dem das Projekt startet (siehe Abbildung 13). Anhand dieses Startdatums werden alle Daten bezüglich der Vorgänge berechnet, zudem stellt der Starttermin bei der Kostenplanung einen wichtigen Faktor dar. Durch die Betätigung des Feldes, das den aktuellen Datenwert anzeigt, öffnet sich ein Kalendermenü, in dem ein beliebiges Datum ausgewählt werden kann. Bei dem Datumsfeld handelt es sich ebenfalls um eine Applikationserweiterung (`gem`) namens „Bootstrap-Datepicker-Rails“[34] (Vgl. Quellcode 10 im Anhang A.3). Es handelt sich hier um ein Unterprogramm inkl. dazugehöriger JavaScrip-Datei.

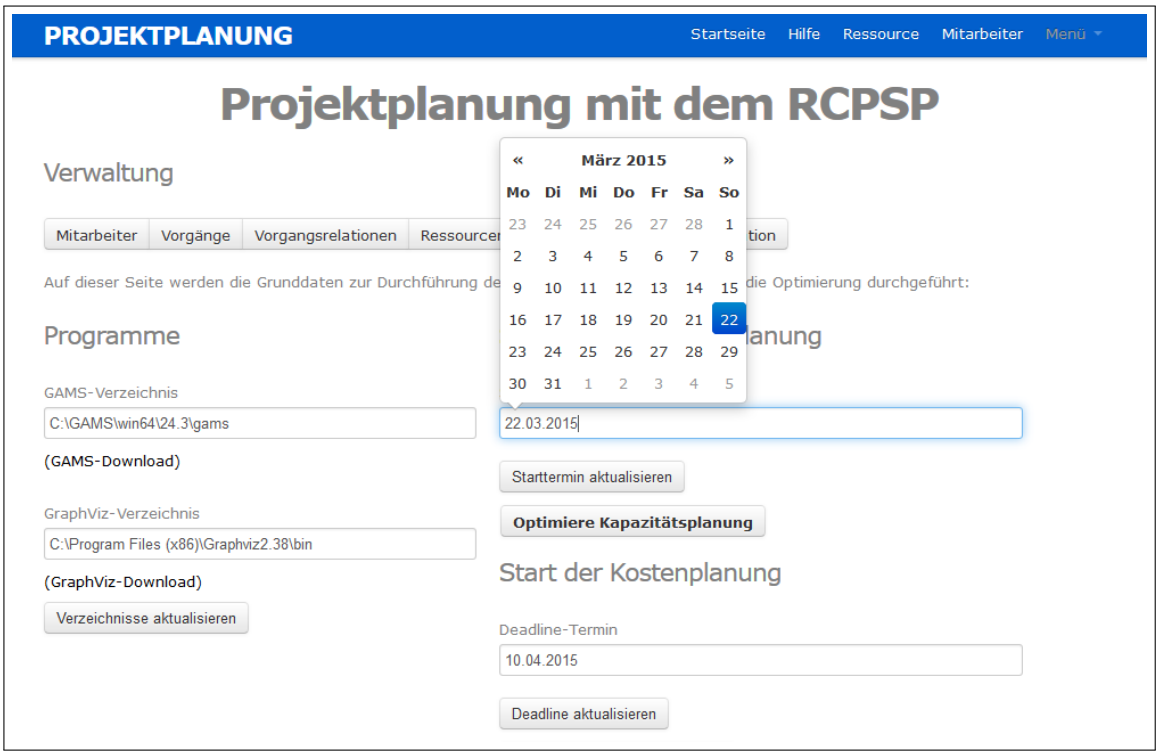


Abbildung 13 Einstellung des Starttermins anhand eines Kalendermenüs

[34] https://github.com/Nerian/bootstrap-datepicker-rails

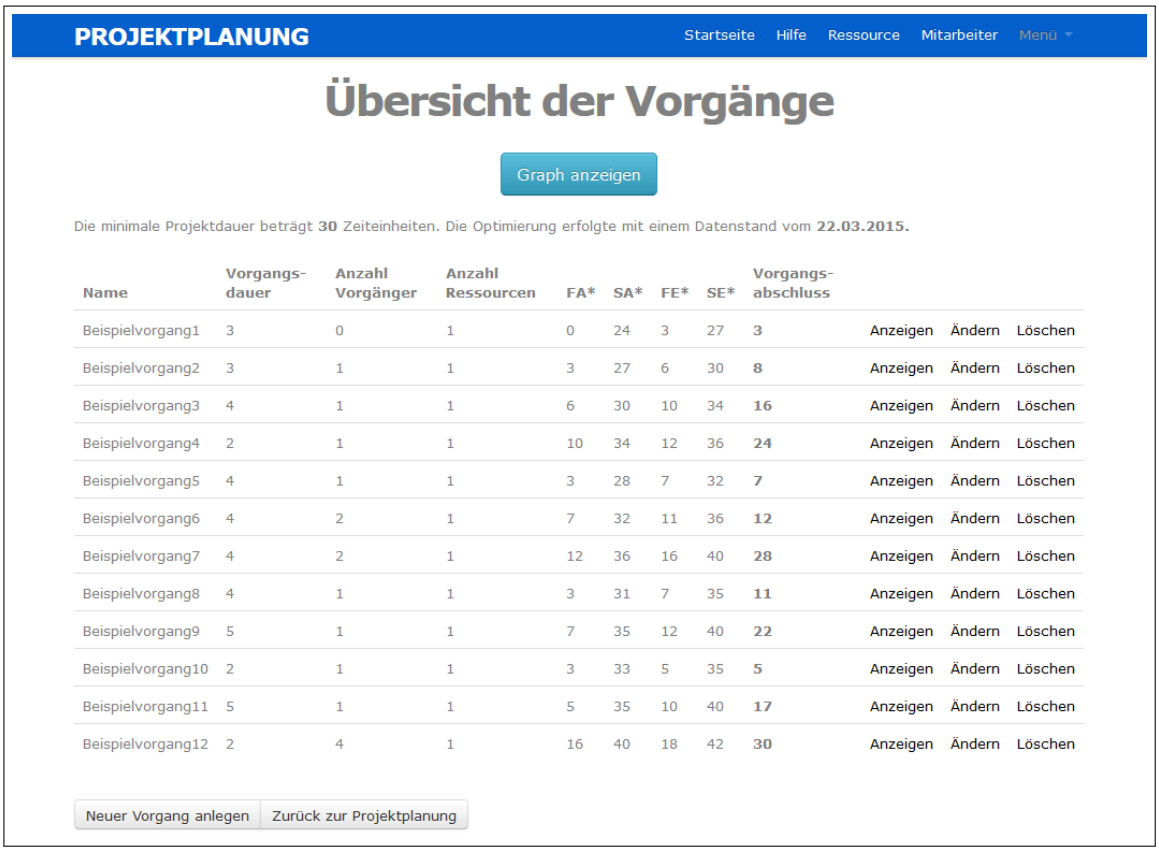
Nachdem das *GAMS*-Verzeichnis und der Starttermin eingestellt sind, kann die Kapazitätsplanung durch die Betätigung des Buttons *Optimiere Kapazitätsplanung* durchgeführt werden. Es handelt sich hier um die Aktion `optimize` des *RoR* `rcpsps_controller` (Vgl. Quellcode 16 im Anhang A.3). Die Aktion dient dazu, die Include-Dateien für die *GAMS*-Optimierung zu schreiben und eben diese durch Aufrufen der *GAMS*-Software zu starten. Bei der *GAMS*-Optimierung handelt es sich um die Datei mit dem Quellcode 8 aus Anhang A.2. Nach einer Rechenzeit, währenddessen der Button, mit dem die Optimierung gestartet wurde, auf den Rechenprozess hinweist, leitet die Applikation den Admin direkt zu der Übersicht der Vorgänge. Dieser Schritt wird durch die Hilfsaktion `solution` des *RoR* `rcpsps_controller` unterstützt, die parallel aufgerufen wird. Nachdem die *GAMS*-Optimierung vollzogen ist, liest der *RoR* `rcpsps_controller` die von der *GAMS*-Optimierung erstellten Text-Dateien ein und schreibt diese direkt in die dafür vorgesehene Datenbank. Bei der Kapazitätsplanung wird hauptsächlich die Datenbank des *RoR* `models/procedure` angesprochen. Wie bereits in Abbildung 8 dargestellt, sind in der Übersicht der Vorgänge alle möglichen Zeitpunkte dargestellt, an denen die einzelnen Vorgänge stattfinden können. Zusätzlich wird die Projektdauer über der Tabelle und der Vorgangsabschluss, durch den die Projektdauer erzielt wird, in die Tabelle geschrieben (siehe Abbildung 14).

Die Besonderheit der Aktion `optimize` ist, dass *RoR* die durch die *GAMS*-Optimierung (Quellcode 9 aus Anhang A.2) erstellte Textdatei zum Parameter X_{it} trotz der zwei Indizes auslesen kann. Dies erfolgt durch einen *If*-Befehl, siehe Quellcode 1.

Quellcode 1 Ausschnitt aus dem RoR-Controller für das RCPSP

```
fi=File.open("RCPSP1_solution_x.txt", "r")
fi.each { |line|
sa=line.split(";")
if sa[0].to_i == 1
sa0=sa[0]
sa1=sa[1]
sa2=sa[2].delete "t" + " \n"
procedure=Procedure.find_by_name(sa1)
procedure.optp = sa2
procedure.save
end
}
fi.close
```

Über den Button *Zurück zur Projektplanung* gelangt ein Admin zurück zur Verwaltungsseite. Sofern die Kostenplanung gewünscht ist, kann diese über die Verwaltungsseite gestartet werden. Bevor der optimale Kostenplan für das vorhandene Projekt berechnet werden kann, muss zunächst äquivalent zur Einstellung des Starttermins eine Deadline eingerichtet werden. Dies funktioniert erneut über ein Kalendermenü des „Bootstrap-Datepicker-Rails“. Es



Name	Vorgangs-dauer	Anzahl Vorgänger	Anzahl Ressourcen	FA*	SA*	FE*	SE*	Vorgangs-abschluss			
Beispielvorgang1	3	0	1	0	24	3	27	**3**	Anzeigen	Ändern	Löschen
Beispielvorgang2	3	1	1	3	27	6	30	**8**	Anzeigen	Ändern	Löschen
Beispielvorgang3	4	1	1	6	30	10	34	**16**	Anzeigen	Ändern	Löschen
Beispielvorgang4	2	1	1	10	34	12	36	**24**	Anzeigen	Ändern	Löschen
Beispielvorgang5	4	1	1	3	28	7	32	**7**	Anzeigen	Ändern	Löschen
Beispielvorgang6	4	2	1	7	32	11	36	**12**	Anzeigen	Ändern	Löschen
Beispielvorgang7	4	2	1	12	36	16	40	**28**	Anzeigen	Ändern	Löschen
Beispielvorgang8	4	1	1	3	31	7	35	**11**	Anzeigen	Ändern	Löschen
Beispielvorgang9	5	1	1	7	35	12	40	**22**	Anzeigen	Ändern	Löschen
Beispielvorgang10	2	1	1	3	33	5	35	**5**	Anzeigen	Ändern	Löschen
Beispielvorgang11	5	1	1	5	35	10	40	**17**	Anzeigen	Ändern	Löschen
Beispielvorgang12	2	4	1	16	40	18	42	**30**	Anzeigen	Ändern	Löschen

Abbildung 14 Ergebnis der Kapazitätsplanung

sollte bei der Bestimmung der Deadline darauf geachtet werden, dass die Deadline in einem sinnvollen Verhältnis zum Starttermin steht. Eine zu kurze oder lange Zeitspanne zwischen den beiden Terminen kann zu unbrauchbaren Ergebnissen führen. Ist eine geeignete Deadline ausgewählt und die Optimierung des Kostenplans gestartet, öffnet sich nach einer kurzen Rechenzeit die Übersicht der Ressourcen (siehe Abbildung 15). Dieses erfolgt mit der Aktion `optimize2` und `solution2` des `rcpsps_controller`.

Auf der Seite mit der Ressourcen-Übersicht sind die Projektkosten und die Zusatzkosten, die jede Ressource durch Einhaltung der Deadline verursacht, ausgelesen (siehe Abbildung 9). Bei der Kostenplanung ist jedoch nicht nur relevant, wie hoch die Kosten zur Durchführung des Projektes sind, sondern auch die Zeitpunkte, zu denen die Vorgänge stattfinden. Um dies zu untersuchen, bietet sich dem Admin die Möglichkeit, ein weiteres Mal die Seite mit der Übersicht der Vorgänge aufzurufen. Auf dieser Seite sind die frühesten und spätesten Zeitpunkte sowie der Abschluss jedes Vorgangs unter Einhaltung der Ressourcenbeschränkung in die Tabelle eingelesen. Die alten Ergebnisse der Kapazitätsplanung sind gelöscht, damit kein veralteter Wert angezeigt wird (siehe Abbildung 16).

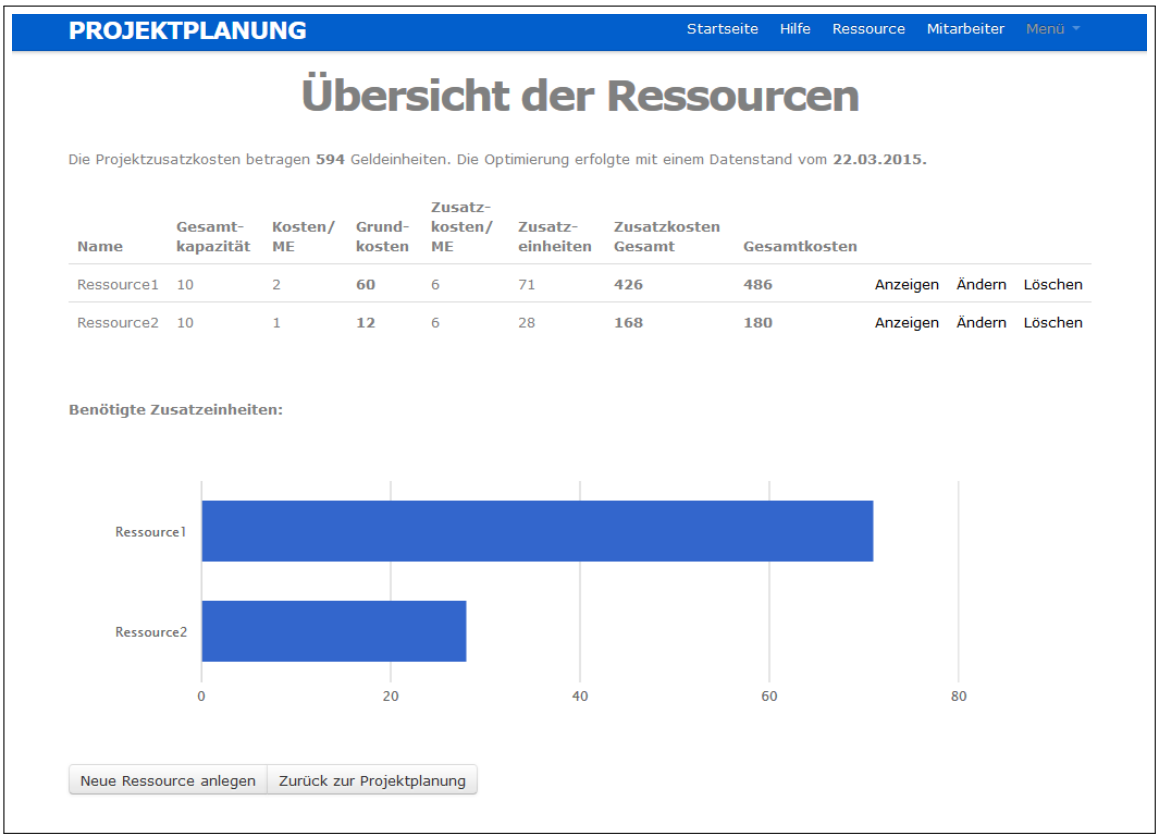


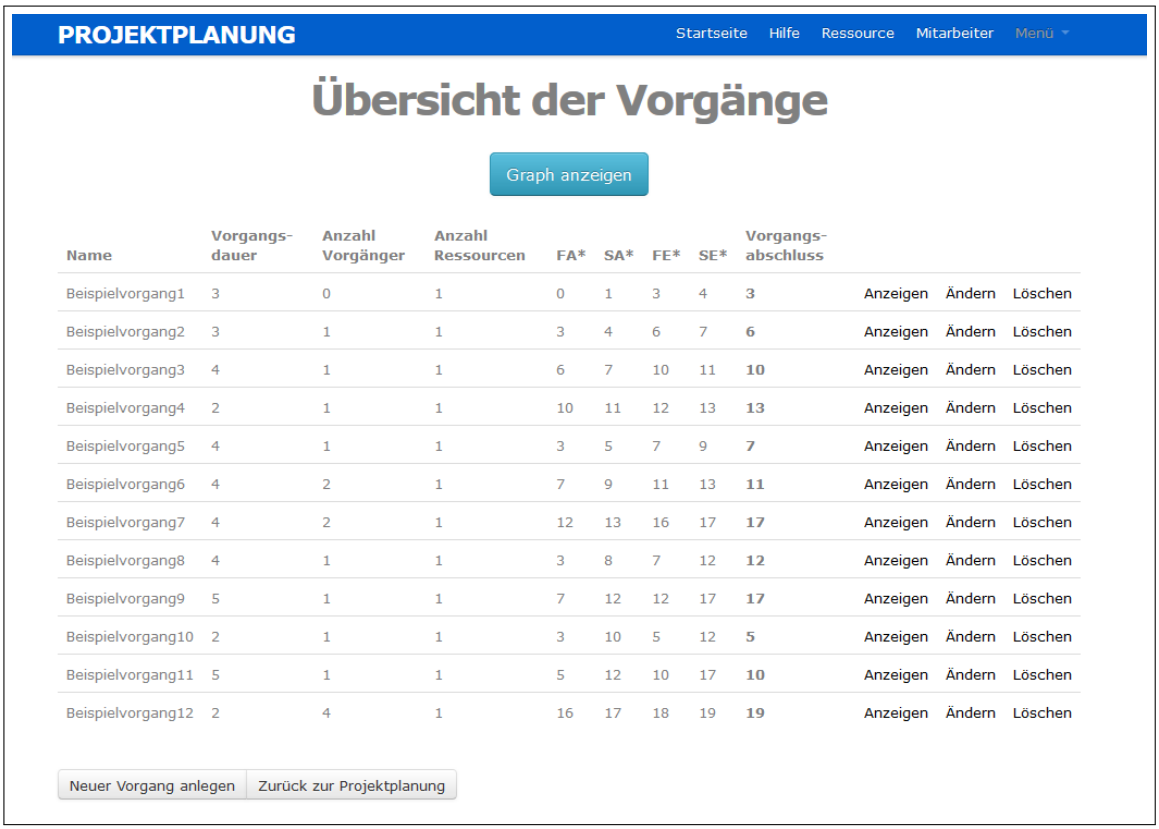
Abbildung 15 Ergebnis der Kostenplanung - Ressourcenübersicht

3.4 Integration des Unterprogramms *rgl* und des Programms *GraphViz* in die Web-Applikation

Für die Lösung des Projektplanungsproblems bedarf es der topologischen Reihenfolgebeziehung der einzelnen Vorgänge untereinander. Weiter dürfen keine Zyklen in der Projektstruktur inbegriffen sein.[35] Zur Prüfung der Projektstruktur der aktuell gespeicherten Datensätze zu dem Projekt (Vorgänge und Vorgangsrelationen) muss in *RoR* ein Algorithmus hinterlegt werden, der die Projektstruktur nach Zyklen untersucht. Eine Sammlung von hierfür geeigneten Algorithmen bietet die Ruby Graph Library (RGL) in Form des Unterprogramms `gem 'rgl'`[36]. Mit dieser `gem` wird die Web-Applikation um die dafür notwendigen Kommandobefehle erweitert. Bei der `gem 'rgl'` handelt es sich um eine Sammlung von Kommandobefehlen für die Graphentheorie, deren Datenstruktur und die zur Analyse notwendigen Algorithmen. Alternativ gibt es andere Sammlungen bzw. Erweiterungen der `gem 'rgl'`, wie z. B. die `gem 'plexus'`.

In der hier betrachteten Web-Applikation zur Lösung des Projektplanungsproblems ist die `gem 'rgl'` implementiert. Eine bestehende Web-Applikation kann durch eine `gem` mittels

[35] Vgl. Helber (2014), S. 208
[36] https://github.com/monora/rgl



Name	Vorgangs-dauer	Anzahl Vorgänger	Anzahl Ressourcen	FA*	SA*	FE*	SE*	Vorgangs-abschluss			
Beispielvorgang1	3	0	1	0	1	3	4	**3**	Anzeigen	Ändern	Löschen
Beispielvorgang2	3	1	1	3	4	6	7	**6**	Anzeigen	Ändern	Löschen
Beispielvorgang3	4	1	1	6	7	10	11	**10**	Anzeigen	Ändern	Löschen
Beispielvorgang4	2	1	1	10	11	12	13	**13**	Anzeigen	Ändern	Löschen
Beispielvorgang5	4	1	1	3	5	7	9	**7**	Anzeigen	Ändern	Löschen
Beispielvorgang6	4	2	1	7	9	11	13	**11**	Anzeigen	Ändern	Löschen
Beispielvorgang7	4	2	1	12	13	16	17	**17**	Anzeigen	Ändern	Löschen
Beispielvorgang8	4	1	1	3	8	7	12	**12**	Anzeigen	Ändern	Löschen
Beispielvorgang9	5	1	1	7	12	12	17	**17**	Anzeigen	Ändern	Löschen
Beispielvorgang10	2	1	1	3	10	5	12	**5**	Anzeigen	Ändern	Löschen
Beispielvorgang11	5	1	1	5	12	10	17	**10**	Anzeigen	Ändern	Löschen
Beispielvorgang12	2	4	1	16	17	18	19	**19**	Anzeigen	Ändern	Löschen

Abbildung 16 Ergebnis der Kostenplanung - Vorgangsübersicht

GitHub und dazugehörigem Terminalbefehl erweitert werden. Jedoch muss beachtet werden, dass in der `Gemfile`, in der `Rakefile` sowie in ggf. anderen Dateien Fehler durch den `pull`-Befehl auftreten könnten. Daher empfiehlt es sich einen neuen `git branch` zu erstellen. Nachfolgende Kommandobefehle im Terminal zeigen die vorher beschriebenen Schritte:

```
$ cd <Verzeichnis der Web-Applikation>
$ git branch -b <Branch-Name>
$ git pull https://github.com/monora/rgl.git
```

Nach dem das Verzeichnis der Web-Applikation um die Dateien der `gem 'rgl'` erweitert ist, muss in der `Gemfile` das nötige `gem` aktiviert werden (siehe Ausschnitt aus Quellcode 10).

Quellcode 2 Ausschnitt der Gemfile der Web-Applikation Projektplanung

```
gem "chartkick"
gem 'bootstrap-datepicker-rails', '~> 1.3.1.1'
gem 'ruby-graphviz', '~> 1.2.1'
gem 'graphviz', '~> 0.1.0'
gem 'rgl'
```

Durch den Befehl `$ bundle install` wird die Web-Applikation um die ergänzten `gem`-Dateien erweitert. Damit sind neue Kommandobefehle innerhalb der Programmierung der Web-Applikation möglich. Zur Prüfung der Projektstruktur auf Zyklen wird der Algorith-

mus bzw. die *Klasse* `RGL::TopsortIterator` der `gem 'rgl'` verwendet.[37] Die *Klasse* ordnet die Datensätze der Vorgangsbeziehung in eine lineare Ordnung ein. Anschließend wird der dadurch generierte Graph nach Zyklen durchsucht. Abbildung 6 zeigt dies anhand eines Beispiels.

Quellcode 3 Prüfung auf Zyklen mittels des Unterprogramms „rgl“

```
>> require 'rgl/adjacency'
>> dg=RGL::DirectedAdjacencyGraph[1,2 ,2,3 ,2,4, 4,5, 6,4, 1,6]
(1-2)(1-6)(2-3)(2-4)(4-5)(6-4)
>> require 'rgl/topsort'
>> dg.acyclic?
true
```

Für die hier betrachtete Web-Applikation bedarf es eines Zugriffs auf das Datenbankmodell des *RoR* `models/procedure_procedure.rb` zur Prüfung der Projektstruktur. Das Modell dient dazu, die einzelnen Vorgänge in eine Beziehung zueinander zu bringen, damit in der *GAMS*-Optimierung die topologische Reihenfolge eingehalten werden kann. Das Modell hilft aber auch zur Generierung eines Graphen für das Unterprogramm `rgl`. Dafür wird eine Schleife erstellt, in der die Vorgänger (`prepro_id`) und Nachfolger (`sucpro_id`) eines jeden Vorgangs (`procedure`) in einen direkten Nachbarschaftsgraphen ergänzt werden (siehe Quellcode 4).

Quellcode 4 Erstellung eines Graphen mittels des Unterprogramms „rgl“

```
require 'rgl/adjacency'
result = RGL::DirectedAdjacencyGraph.new ProcedureProcedure.all.each { | x |
result.add_edge x.prepro_id , x.sucpro_id }
```

Nachdem dieser Graph generiert ist, wird er durch die Methode `#acyclic?`[38] auf bestehende Zyklen untersucht. Diese Untersuchung ist in dem *RoR* `controller` unter der Aktion `create` ergänzt. Quellcode 5 zeigt den Ausschnitt mit der hier beschriebenen Aktion.

Weiter bietet das Unterprogramm `rgl` die Möglichkeit, eine `dot`-Datei zu erstellen, damit der Graph mittels des Programms *GraphViz* visualisiert wird.

[37] http://www.rubydoc.info/github/monora/rgl/RGL/TopsortIterator
[38] http://www.rubydoc.info/github/monora/rgl/RGL/Graph#acyclic%3F-instance_method

Quellcode 5 Ausschnitt aus dem RoR-Controller für die Vorgangsbeziehungen

```
  def create
  require 'rgl/adjacency'
  require 'rgl/topsort'
  @procedure_procedure = ProcedureProcedure.new(params[:procedure_procedure↩
      ])
    respond_to do |format|
      if @procedure_procedure.save
        result = RGL::DirectedAdjacencyGraph.new
        ProcedureProcedure.all.each { |x|
          result.add_edge x.prepro_id, x.sucpro_id }
        if result.acyclic? == true
        format.html { redirect_to procedure_procedures_path, notice: '↩
            Relation wurde erfolgreich angelegt!' }
        format.json { render json: procedure_procedures_path, status: :↩
            created, location: @procedure_procedure }
        else
          @procedure_procedure.destroy
          sleep(5)
          format.html { redirect_to :back, notice: 'Zyklen sind in der ↩
              Projektplanung nicht erlaubt!' }
          format.json { render json: @procedure_procedure.errors, status: :↩
              unprocessable_entity }
        end
      end
    end

  end
```

Quellcode 6 Prüfung auf Zyklen mittels des Unterprogramms „rgl“

```
>> require 'rgl/adjacency'
>> dg=RGL::DirectedAdjacencyGraph[1,2 ,2,3 ,2,4, 4,5, 6,4, 1,6]
>> require 'rgl/dot'
>> dg.write_to_graphic_file
"graph.dot"
```

Zur Visualisierung der `dot`-Datei bedarf es jedoch der Integration des Open-Source-Programms *GraphViz* in die Web-Applikation. Mit dem Programm ist es möglich Strukturinformationen als abstrakte Graphen und Netzwerke darzustellen.[39] Damit *RoR* mit dem Programm kommunizieren kann, wird seitens der Anbieter empfohlen das Unterprogramm `gem 'ruby-graphviz'` zu verwenden. Nachdem die Installation des Unterprogramms durchgeführt und das Programm *GraphViz* in einem lokalen Computerverzeichnis gespeichert ist, kann die `dot`-Datei mittels Kommandobefehl ausgelesen und visualisiert werden. Dies erfolgt im *RoR* `controllers/procedure_procedures_controller.rb` sowie auf der Untersei-

[39] http://www.graphviz.org/About.php

te `graph.html.rb` der *RoR* `views/procedure_procedures` (Vgl. Quellcode 12 bzw. 29 im Anhang A.3). Abbildung 7 zeigt die Aktion zur Erstellung des Graphen.

Quellcode 7 Ausschnitt des RoR-Controller für die Vorgangsrelationen

```
def graph
  @procedure_procedures = ProcedureProcedure.all
  @project = Project.find(1)
  require 'rgl/adjacency'
  require 'rgl/dot'
  result = RGL::DirectedAdjacencyGraph.new
  @procedure_procedures.each { |x|
    result.add_edge  x.prepro.name, x.sucpro.name }
  result.write_to_graphic_file('png')
  require 'graphviz'
  GraphViz.parse( "graph.dot", :path => @project.gvp.to_s ).output(:png =>↩
      "app/assets/images/graph.png", :path => @project.gvp.to_s)
end
```

4 Kritische Würdigung der Web-Applikation

Die hier implementierte Applikation behandelt ausschließlich das *RCPSP(+)* bei exakt einem vorhandenen Projekt. Die Multiprojektplanung ist außer Acht gelassen. Mit der Einführung der Multiprojektplanung eröffnen sich zwar neue Problemstellungen bei der Implementierung, es bietet jedoch auch neue Anwendungsmöglichkeiten der Applikation. Anstatt einzelne Aspekte eines Projektes zu optimieren, kann nun ein Portfolio an mehreren Projekten erstellt werden. Ein Mitarbeiter hat somit die Wahl, an welchen Projekten er partizipieren möchte und die Durchführung des Projektportfolios ist Ziel der Optimierung. Es findet dementsprechend eine Projektion der Kapazitäts- und Kostenplanung auf die Multiprojektplanung statt.

Die tatsächlich durchgeführte Programmierung bietet ebenfalls Verbesserungspotential. Auf der Übersichtsseite der Projektplanung, auf der die Optimierungsvorgänge gestartet werden können, finden sich auch die Felder zur Aktualisierung des Starttermins und der Deadline. Die Felder und Funktionen, die in diesem Zusammenhang ausgelöst werden, funktionieren zwar fehlerfrei, die zeitliche Abhängigkeit ist aber nicht in der Form gegeben, wie es die Bezeichnungen voraussetzen. Wird die zeitliche Reihenfolge vertauscht, führt das Modell die Optimierung fehlerhaft durch. Der Termin der Deadline muss laut der implementierten Regularien zeitlich nachfolgend zum Starttermin stattfinden.

Ein weiteres Problem tritt bei der Aktualisierung der Pfade für die Programme *GAMS* und *GraphViz* auf (siehe Abbildung 10). Die Lösung aller Planungsprobleme erfolgt erst nach der Angabe des korrekten Pfads von *GAMS*. Nach diesem Schema funktioniert die Erstellung des

Graphen der Vorgangsrelationen nur, wenn erfolgreich auf *Graphviz* zurückgegriffen werden kann. Diese Kausalitäten führen zu der Frage, was bei der Angabe von falschen Pfaden für das jeweilige Programm passiert. Faktisch funktionieren sie nicht. Beim Start der Optimierung mit einem falschen *GAMS*-Pfad sucht der Controller vergeblich nach der Anwendung *GAMS* im aktuell gespeicherten Verzeichnis. Statt den Vorgang abzubrechen, bleibt der *Ruby*-Server an dieser Stelle hängen und der Nutzer kann keine weiteren Aktionen ausführen. Um dieses „Abstürzen“ des Servers zu verhindern, falls der Programmpfad inkorrekt ist, könnte ein `Timeout`-Befehl die Aktion nach einer vorgegebenen Zeitspanne abbrechen. So wäre es möglich, den Pfad zu ändern und den Vorgang erneut zu starten. Die Aktion `Timeout` ist dem Framework *RoR* inbegriffen. Eine andere Möglichkeit als die Aktion `Timeout` ist das Unterprogramm `Terminator`. Das Unterprogramm ist keine interne *RoR*-Methode, daher muss der passende `gem` installiert werden, damit *RoR* diese Methode verwenden kann.[40]

Auch bei der Verwaltung der Vorgänge und Vorgangsrelationen offenbaren sich Lücken, die in späteren Ausarbeitungen überarbeitet werden sollten. Das Anlegen, Ändern und Löschen von Vorgängen funktioniert reibungslos. Zur Darstellung der Vorgangsrelationen wird das Programm *GraphViz* verwendet, unter Zuhilfenahme des Unterprogramms `gem 'ruby-graphviz'`. Der Programmcode dieses Unterprogramms ist übernommen, nur eine Anwendung des Unterprogramms an die Gegebenheiten der Problemstellung findet statt. Diese Datenübernahme hat zur Folge, dass der Name jedes Vorgangs aus **mindestens zwölf Zeichen** bestehen muss. Bei einem kürzeren Vorgangsnamen zeichnet das Programm keinen Graphen. Die Gründe dafür konnten in dieser Arbeit nicht abschließend geklärt werden, da der gesamte Programmcode des Unterprogramms und des Programms *GraphViz* in der kurzen Zeit der Erstellung dieser Arbeit nicht vollständig untersucht werden konnte. Daher ist in dieser Arbeit eine Mindestlänge für den Namen eines Vorgangs vorgegeben. Das ursprüngliche Problem von *GraphViz* besteht dadurch weiterhin. Auf andere Funktionen der Vorgänge oder Vorgangsrelationen hat dieser Umstand keinen Einfluss. Die Vorgangsrelationen stellen einen wichtigen Part der Projektplanung dar. Schließlich ist die Bestimmung der Reihenfolge der abzuarbeitenden Vorgänge die Basis für viele andere Operationen auf dem Weg zur Projektoptimierung. Beim Anlegen einer neuen Vorgangsrelation dürfen, wie bereits beschrieben, keine Zyklen entstehen. Es findet aber keine Prüfung statt, ob die neu angelegte Vorgangsrelation bereits existiert. Folglich wird nicht verhindert, dass z.B. die Relation von *Vorgang 1* zu *Vorgang 2* zwei Mal angelegt wird. Für alle weiteren Prozesse resultieren keine Schäden, da doppelt vorhandene Relationen nicht beachtet werden, trotzdem handelt es sich hier um eine Prüfungslücke im Modell der Vorgangsrelationen. Diese Problematik der Möglichkeit von inhaltlich identischen Datensätzen betrifft jedes Datenmodell.

[40] https://github.com/ahoward/terminator

5 Fazit

In dieser Ausarbeitung sind zunächst die Problematiken der Projektplanung dargelegt sowie das *RCPSP(+)* für die Kapazitäts- und Kostenplanung formuliert. Dabei steht neben der Lösung des Modells mit einem Programm für mathematische Modellformulierungen auch die Integration sowie Verknüpfung mit einer Web-Applikation im Vordergrund. Ziel ist es, eine Web-Applikation zu erstellen, die im Hintergrund der Anwendung auf das Lösungstool, das die Rechnungen durchführt, zurückgreift, die Ergebnisse abruft und darstellt.

Das *RCPSP(+)* wird in dem Programm *GAMS* für mathematische Optimierungsprobleme formuliert und mit dessen Hilfe wird die LP-Relaxation gelöst. Da der Schwerpunkt dieser Arbeit auf dem Framework *RoR* liegt, sind in der *GAMS*-Modellformulierung nur die essentiellen Bestandteile zur Lösung des *RCPSP(+)* implementiert. Dank einer Schnittstelle zwischen dem Framework *RoR* und dem Programm *GAMS* ist es möglich, einen Großteil der Daten in *RoR*-Programmiersprache zu formulieren, um diese dann in *GAMS* einlesen zu lassen. Der Schwerpunkt liegt dementsprechend auf der Implementierung in *RoR* und der Bildung der Schnittstelle.

Bei der Erstellung der Applikation ist die herausragende Rolle der Controller zu erwähnen. In den jeweiligen Controllern werden alle dynamischen Aktionen definiert, auf die in anderen Verzeichnissen oder Datenmodellen zurückgegriffen werden. Nur Aktionen, die im Controller hinterlegt sind, können in der HTML-Darstellung der Seite aufgerufen bzw. genutzt werden. Alternativ wird der Controller für die Schnittstelle zu *GAMS* oder *GraphViz* benötigt. Falls der Controller fehlerhaft programmiert ist, führt dies zwangsläufig an einem gewissen Punkt der Applikation zu Komplikationen. Am Beispiel des *RCPSP*-Controller (Quellcode 16 im Anhang A.3) lässt sich die Signifikanz des Controllers in *RoR* demonstrieren.

Im *RCPSP*-Controller sind alle Aktionen bezüglich der Optimierung der Kapazitäts- und Kostenplanung hinterlegt. Die Aktion `optimize` beinhaltet alle Schritte, die *RoR* durchführen muss, damit die *GAMS*-Datei zur Lösung der Kapazitätsplanung den vollständigen Input erhält, das System gestartet sowie der Output korrekt ausgelesen wird. Die durch das Betätigen des Buttons *Optimiere Kapazitätsplanung* ausgelöste Aktion `optimize` (siehe Abbildung 10), funktioniert ausschließlich durch die korrekte Programmierung des Controllers.

Schließlich liefert die Arbeit eine umfangreiche Darstellung der Web-Applikation für die Projektplanung mittels *RoR*, *GAMS* und *GraphViz*. Dabei wird auf die Besonderheiten der Programmierung eingegangen und explizit auf die Integration weiterer Unterprogramme in dem Framework *RoR*.

Literatur

Bartels, J.H. (2009): Projektplanung–Grundlagen und Anwendungsbeispiele. In: Anwendung von Methoden der ressourcenbeschränkten Projektplanung mit multiplen Ausführungsmodi in der betriebswirtschaftlichen Praxis. Springer, S. 7–42.

Demeulemeester, E. und Herroelen, W. (2011): Robust project scheduling. Bd. 3. Now Publishers Inc.

DIN 69900 (2009): Projektmanagement - Netzplantechnik; Beschreibung und Begriffe. In: Berlin: Beuth.

Domschke, W. und Drexl, A. (2005): Einführung in operations research. Bd. 5. Springer.

Drexl, A.; Kolisch, R. und Sprecher, A. (1997): Neuere Entwicklungen in der Projektplanung. In: Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung, S. 95–120.

Felkai, R. und Beiderwieden, A. (2011): Analysieren und Formulieren von Projektzielen. In: Projektmanagement für technische Projekte. Springer, S. 45–64.

Grimmer, L. (2006): Interview with David Heinemeier Hansson from Ruby on Rails.

Hartl, M. (2012): Ruby on Rails Tutorial: Learn Web Development with Rails. Pearson Education.

Helber, S. (2014): Operations Management Tutorial.

Kellenbrink, C. (2014): Einführung in die ressourcenbeschränkte Projektplanung. In: Ressourcenbeschränkte Projektplanung für flexible Projekte. Springer, S. 5–18.

Neumann-Braun, K.; Schwindt, C. und Zimmermann, J. (2003): Project scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions. Springer.

Pritsker, A.A.B.; Waiters, L.J. und Wolfe, P.M. (1969): Multiproject scheduling with limited resources: A zero-one programming approach. In: Management science. Bd. 16, Nr. 1, S. 93–108.

Talbot, F.B. (1982): Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. In: Management Science. Bd. 28, Nr. 10, S. 1197–1210.

Voigt, K.I. und Schewe, G. (2014): Definition Projekt Version 7 - Gabler Wirtschaftslexikon.

Walter, T. (2008a): Das Ruby-Framework Ruby on Rails. In: Kompendium der Web-Programmierung. Springer, S. 463–491.

Walter, T. (2008b): X.media.press. Springer Berlin Heidelberg. S. 297–336.

Wintermeyer, S. (o. J.): Installation von Ruby on Rails 3.2 mit RVM. http://ruby-auf-schienen.de.

Zimmermann, J.; Stark, C.; Rieck, J. et al. (2006): Projektmanagement. In: Projektplanung. Springer Berlin Heidelberg, S. 1–113.

A Anhang

A.1 Datenbankschema

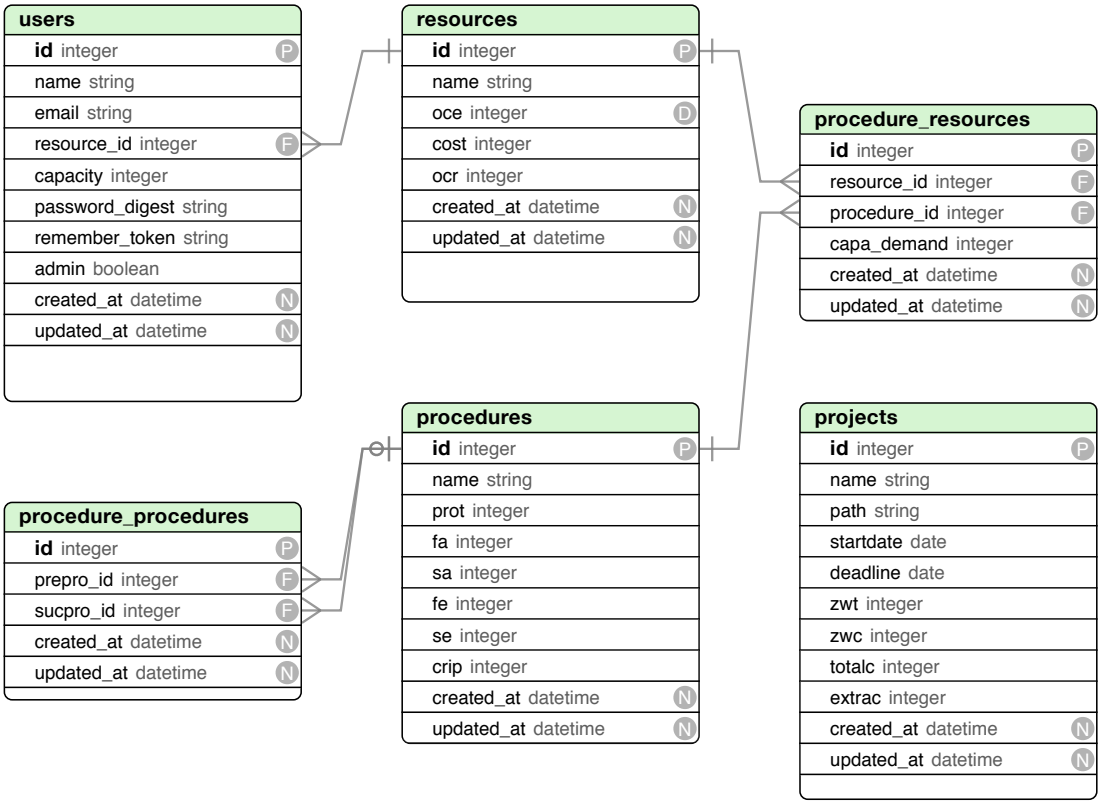


Abbildung 17 Datenbankschema der Web-Applikation Projektplanung

P = Primary Key, N = Not Null, D = Default Value, F = Foreign Key

A.2 GAMS-Implementierung des Beispiels

Quellcode 8 GAMS-Code zur Kapazitätsplanung

```
* Ressourcenbeschraenkte Projektplanung in diskreter Zeit
* Zwei Modellvarianten:
* Variante 1: Minimierung der Projektdauer bei gegebenen Kapazitaeten

set
    i Vorgang
    t Periode
    r Ressource;

alias(t,tau);
alias(h,i);
```

```
set
    VN(h,i) Vorgaenger-Nachfolger-Relation zwischen h und i;

parameter
    d(i)    Dauer
    FE(i)   Fruehester Endzeitpunkt
    SE(i)   Spaetester Endzeitpunkt
    FA(i)   Fruehester Anfangszeitpunkt
    SA(i)   Spaetester Anfangszeitpunkt
    k(i,r)  Kapazitaetsbedarf von Vorgang i auf Ressource r
    KP(r)   Kapazitaet je Periode von Ressource r
    ihilf
    Deadline
    MinimaleDauer   ;

binary variables
    x(i,t)  gleich 1 wenn Vorgang i in Periode t beendet wird;

free variables
    z       Zielfunktionswert;

$include "RCPSP1_Input.inc";

* Zeitrechnung
* Achtung: Topologische Sortierung wird unterstellt

MinimaleDauer=0;
FA(i)=0;
FE(i)=d(i);

loop(i,
     loop(h$VN(h,i),
         if(FE(h)>FA(i),
             FA(i)=FE(h);
             FE(i)=FA(i)+d(i);
             if( FE(i)>MinimaleDauer,
                 MinimaleDauer = FE(i)
             );
         );
     );

);

SE(i)=max(MinimaleDauer, Deadline);
SA(i)=SE(i)-d(i);

for(ihilf=card(i) downto 1,
```

```
     loop(i$(ord(i)=round(ihilf)),
         loop(h$VN(i,h),
             if(SA(h)<SE(i),
                 SE(i)=SA(h);
                 SA(i)=SE(i)-d(i);
             );
         );
     );
);

display d, FA, FE, SA, SE, MinimaleDauer;

Equations
    ZielfunktionZeit,
    JederVorgangEinmal(i)
    Projektstruktur(h,i)
    Kapazitaetsrestriktion(r,t);

ZielfunktionZeit..
    z=e=sum(i$(ord(i)=card(I)),
        sum(t$(FE(i)<=ord(t)-1 and ord(t)-1 <= SE(i)),
             (ord(t)-1)*x(i,t)));

JederVorgangEinmal(i)..
    sum(t$(FE(i)<=ord(t)-1 and ord(t)-1 <= SE(i)), x(i,t)) =e= 1;

Projektstruktur(h,i)$VN(h,i)..
    sum(t$(FE(h)<=ord(t)-1 and ord(t)-1 <= SE(h)),
          (ord(t)-1)*x(h,t))  =l=
    sum(t$(FE(i)<=ord(t)-1 and ord(t)-1 <= SE(i)),
          (ord(t)-1-d(i))*x(i,t));

Kapazitaetsrestriktion(r,t)..
    sum(i,
    sum(tau$((ord(tau)-1 >= max(ord(t)-1, FE(i)))  and
              (ord(tau)-1 <= min(ord(t)-1+d(i)-1, SE(i)))),
          k(i,r)*x(i,tau)))=l=KP(r);

model RCPSP1 /
    ZielfunktionZeit
    JederVorgangEinmal
    Projektstruktur
    Kapazitaetsrestriktion/;

RCPSP1.optcr=0.0;
RCPSP1.limrow=500;
```

```
solve RCPSP1 minimizing z using mip;

display z.l, x.l;

file outputfile1 / 'RCPSP1_solution_zeit.txt'/;
put outputfile1;

loop(i,
     put i.tl:0, '; ' FA(i), ' ; ' SA(i), ' ; ' FE(i), ' ; ' SE(i) /
);
putclose outputfile1;

file outputfile2 / 'RCPSP1_solution_x.txt'/;
put outputfile2;

loop(t,
loop(i,
     put x.l(i,t), ';' i.tl:0, ';' t.tl:0  /
);
);
putclose outputfile2;

file outputfile3 /'RCPSP1_solution_zw.txt'/;
put outputfile3;

put 'Zielfunktionswert: ',z.l /
put '**********************'

putclose outputfile3;
```

Quellcode 9 GAMS-Code zur Kostenplanung

```
* Ressourcenbeschraenkte Projektplanung in diskreter Zeit
* Zwei Modellvarianten:
* Variante 2: Minimierung der Kosten fuer Zusatzkapazitaet bei
*             gegebener Deadline

set
    i Vorgang
    t Periode
    r Ressource;

alias(t,tau);
alias(h,i);

set
    VN(h,i) Vorgaenger-Nachfolger-Relation zwischen h und i;
```

```
parameter
    d(i)     Dauer
    FE(i)    Fruehester Endzeitpunkt
    SE(i)    Spaetester Endzeitpunkt
    FA(i)    Fruehester Anfangszeitpunkt
    SA(i)    Spaetester Anfangszeitpunkt
    k(i,r)   Kapazitaetsbedarf von Vorgang i auf Ressource r
    KP(r)    Kapazitaet je Periode von Ressource r
    oc(r)    Kosten einer Einheit Zusatzkapazitaet
    ihilf
    Deadline
    MinimaleDauer   ;

binary variables
    x(i,t)  gleich 1 wenn Vorgang i in Periode t beendet wird;

free variables
    z       Zielfunktionswert;

positive variables
    O(r,t)  Zusatzkapazitaet von Ressource r in Periode t;

$include "RCPSP2_Input.inc";

* Zeitrechnung
* Achtung: Topologische Sortierung wird unterstellt

MinimaleDauer=0;
FA(i)=0;
FE(i)=d(i);

loop(i,
     loop(h$VN(h,i),
         if(FE(h)>FA(i),
             FA(i)=FE(h);
             FE(i)=FA(i)+d(i);
             if( FE(i)>MinimaleDauer,
                 MinimaleDauer = FE(i)
             );
         );
     );
);

SE(i)=max(MinimaleDauer, Deadline);
SA(i)=SE(i)-d(i);

for(ihilf=card(i) downto 1,
```

```
     loop(i$(ord(i)=round(ihilf)),
         loop(h$VN(i,h),
             if(SA(h)<SE(i),
                 SE(i)=SA(h);
                 SA(i)=SE(i)-d(i);
             );
         );
     );
);

display d, FA, FE, SA, SE, Deadline, MinimaleDauer;

Equations
    ZielfunktionKosten,
    JederVorgangEinmal(i)
    Projektstruktur(h,i)
    Kapazitaetsrestriktion(r,t)
    KapazitaetsrestriktionFlex(r,t);

ZielfunktionKosten..
    z=e=sum((r,t),oc(r)*O(r,t));

JederVorgangEinmal(i)..
    sum(t$(FE(i)<=ord(t)-1 and ord(t)-1 <= SE(i)), x(i,t)) =e= 1;

Projektstruktur(h,i)$VN(h,i)..
    sum(t$(FE(h)<=ord(t)-1 and ord(t)-1 <= SE(h)),
          (ord(t)-1)*x(h,t))  =l=
    sum(t$(FE(i)<=ord(t)-1 and ord(t)-1 <= SE(i)),
          (ord(t)-1-d(i))*x(i,t));

KapazitaetsrestriktionFlex(r,t)..
    sum(i,
    sum(tau$((ord(tau)-1 >= max(ord(t)-1, FE(i)))  and
              (ord(tau)-1 <= min(ord(t)-1+d(i)-1, SE(i)))),
          k(i,r)*x(i,tau)))=l=KP(r)+O(r,t);

model RCPSP2 /
    ZielfunktionKosten
    JederVorgangEinmal
    Projektstruktur
    KapazitaetsrestriktionFlex/;

RCPSP2.optcr=0.0;
RCPSP2.limrow=500;

solve RCPSP2 minimizing z using mip;
```

```
parameter
    zkr(r)  Berechnung Zusatzkosten;

zkr(r) = sum(t,O.l(r,t));

display x.l, O.l, zkr;

file outputfile1 / 'RCPSP2_solution_kosten.txt'/;
put outputfile1;

loop(r,
    put r.tl:0, '; ' zkr(r) /
);

putclose outputfile1;

file outputfile2 / 'RCPSP2_solution_x.txt'/;
put outputfile2;

loop(t,
loop(i,
     put x.l(i,t), ';' i.tl:0, ';' t.tl:0  /
);
);
putclose outputfile2;

file outputfile3 / 'RCPSP2_solution_zeit.txt'/;
put outputfile3;

loop(i,
     put i.tl:0, '; ' FA(i), ' ; ' SA(i), ' ; ' FE(i), ' ; ' SE(i) /
);
putclose outputfile3;

file outputfile4 /'RCPSP2_solution_zw.txt'/;
put outputfile4;

put 'Zielfunktionswert: ',z.l /
put '**********************'

putclose outputfile4;
```

A.3 Ruby on Rails Programmcodes

Quellcode 10 Gemfile der Web-Applikation Projektplanung

```
source 'http://rubygems.org'

gem 'rails', '3.2.8'
gem 'bootstrap-sass', '2.0.4'
gem 'bcrypt-ruby', '3.0.1'
gem 'faker', '1.0.1'
gem 'will_paginate', '3.0.3'
gem 'bootstrap-will_paginate', '0.0.6'
gem 'jquery-rails', '2.0.2'
gem 'best_in_place'
gem "chartkick"
gem 'bootstrap-datepicker-rails', '~> 1.3.1.1'
gem 'ruby-graphviz', '~> 1.2.1'
gem 'graphviz', '~> 0.1.0'
gem 'rgl'

group :development, :test do
  gem 'sqlite3', '1.3.5'
  gem 'rspec-rails', '2.11.0'
  gem 'guard-rspec', '1.2.1'
  gem 'annotate', '2.5.0'
  gem 'wdm', '~> 0.0.3'
  gem 'guard-spork', '1.2.0'
  gem 'spork', '0.9.2'
end

group :development do
  gem 'better_errors'
  gem 'binding_of_caller'
end

# Gems used only for assets and not required
# in production environments by default.
group :assets do
  gem 'sass-rails',   '3.2.5'
  gem 'coffee-rails', '3.2.2'
  gem 'uglifier', '1.2.3'
  gem 'jquery-datatables-rails'
  gem 'jquery-ui-rails'
end

group :test do
  gem 'capybara', '1.1.2'
  gem 'factory_girl_rails', '4.1.0'
```

```
  gem 'cucumber-rails', '1.2.1', :require => false
  gem 'database_cleaner', '0.7.0'
  gem 'growl', '1.0.3'
  gem 'rb-fchange', '0.0.5'
  gem 'rb-notifu', '0.0.4'
  gem 'win32console', '1.3.0'
end

group :production do
  gem 'pg', '0.12.2'
end

if RUBY_VERSION =~ /1.9/ # assuming you're running Ruby ~1.9
  Encoding.default_external = Encoding::UTF_8
  Encoding.default_internal = Encoding::UTF_8
end
```

Quellcode 11 Routes-Datei der Web-Applikation Projektplanung

```
SampleApp::Application.routes.draw do

  resources :projects
  resources :procedure_resources
  resources :procedures
  resources :resources
  resources :procedure_procedures
  resources :users
  resources :sessions, only: [:new, :create, :destroy]

  root to: 'static_pages#home'

  match '/signup', to: 'users#new'
  match '/signin', to: 'sessions#new'
  match '/signout', to: 'sessions#destroy', via: :delete

  match '/help', to: 'static_pages#help'
  match '/about', to: 'static_pages#about'
  match '/contact', to: 'static_pages#contact'
  match '/rcpsp', to: 'static_pages#rcpsp'

  match 'procedure_procedure/graph', :to => 'procedure_procedures#graph'

  match 'rcpsp/optimize', :to => 'rcpsps#optimize'
  match 'rcpsp/solution', to: 'rcpsps#solution'

  match 'rcpsp/optimize2', :to => 'rcpsps#optimize2'
  match 'rcpsp/solution2', to: 'rcpsps#solution2'
end
```

Quellcode 12 RoR-Controller für die Vorgangsrelationen

```
class ProcedureProceduresController < ApplicationController
    respond_to :html, :json
    before_filter :signed_in_user
    before_filter :admin_user

  def graph
    @procedure_procedures = ProcedureProcedure.all
    @project = Project.find(1)
    require 'rgl/adjacency'
    require 'rgl/dot'
    result = RGL::DirectedAdjacencyGraph.new
    @procedure_procedures.each { |x|
      result.add_edge  x.prepro.name, x.sucpro.name }
    result.write_to_graphic_file('png')
    require 'graphviz'
    GraphViz.parse( "graph.dot", :path => @project.gvp.to_s ).output(:png =>↩
        "app/assets/images/graph.png", :path => @project.gvp.to_s)
  end

  def index
    @procedure_procedures = ProcedureProcedure.all
  end

  def show
  end

  def new
    @procedure_procedure = ProcedureProcedure.new
    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @procedure_procedure }
    end
  end

  def edit
  end

  def create
  require 'rgl/adjacency'
  require 'rgl/topsort'
  @procedure_procedure = ProcedureProcedure.new(params[:procedure_procedure↩
      ])
    respond_to do |format|
      if @procedure_procedure.save
```

```
        result = RGL::DirectedAdjacencyGraph.new
        ProcedureProcedure.all.each { |x|
          result.add_edge x.prepro_id, x.sucpro_id }
        if result.acyclic? == true
        format.html { redirect_to procedure_procedures_path, notice: '↩
            Relation wurde erfolgreich angelegt!' }
        format.json { render json: procedure_procedures_path, status: :↩
            created, location: @procedure_procedure }
        else
          @procedure_procedure.destroy
          sleep(5)
          format.html { redirect_to :back, notice: 'Zyklen sind in der ↩
              Projektplanung nicht erlaubt!' }
          format.json { render json: @procedure_procedure.errors, status: :↩
              unprocessable_entity }
        end
      end
    end

  end

  def update
  end

  def destroy
    @procedure_procedure = ProcedureProcedure.find(params[:id])
    @procedure_procedure.destroy
    respond_to do |format|
      format.html { redirect_to procedure_procedures_url }
      format.json { head :no_content }
    end
  end

  private

  def signed_in_user
    unless signed_in?
      store_location
      redirect_to signin_path, notice: "Bitte anmelden."
    end
  end

  def admin_user
    redirect_to(root_path) unless current_user.admin?
  end

end
```

Quellcode 13 RoR-Controller für die Vorgangs-Ressourcen-Kombinationen

```
class ProcedureResourcesController < ApplicationController
  respond_to :html, :json
  before_filter :signed_in_user
  before_filter :admin_user
  # GET /procedure_resources
  # GET /procedure_resources.json
  def index
    @procedure_resources = ProcedureResource.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @procedure_resources }
    end
  end
  # GET /procedure_resources/1
  # GET /procedure_resources/1.json
  def show
    @procedure_resource = ProcedureResource.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @procedure_resource }
    end
  end
  # GET /procedure_resources/new
  # GET /procedure_resources/new.json
    def new
      @procedure_resource = ProcedureResource.new

      respond_to do |format|
        format.html # new.html.erb
        format.json { render json: @procedure_resource }
      end
    end
  # GET /procedure_resources/1/edit
  def edit
    @procedure_resource = ProcedureResource.find(params[:id])
  end

  # POST /procedure_resources
  # POST /procedure_resources.json
    def create
      @procedure_resource = ProcedureResource.new(params[:procedure_resource↩
          ])

      respond_to do |format|
```

```
      if @procedure_resource.save
        format.html { redirect_to @procedure_resource, notice: 'Vorgangs-↩
            Ressourcen-Relation wurde erfolgreich angelegt!' }
        format.json { render json: @procedure_resource, status: :created, ↩
            location: @procedure_resource }
      else
        format.html { render action: "new" }
        format.json { render json: @procedure_resource.errors, status: :↩
            unprocessable_entity }
      end
    end
  end
# PUT /procedure_resources/1
# PUT /procedure_resources/1.json
def update
  @procedure_resource = ProcedureResource.find(params[:id])

  respond_to do |format|
    if @procedure_resource.update_attributes(params[:procedure_resource])
      format.html { redirect_to @procedure_resource, notice: 'Vorgang-↩
          Ressourcen-Relation wurde erfolgreich aktualisiert.' }
      format.json { head :no_content }
    else
      format.html { render action: "edit" }
      format.json { render json: @procedure_resource.errors, status: :↩
          unprocessable_entity }
    end
  end
end
# DELETE /procedure_resources/1
# DELETE /procedure_resources/1.json
  def destroy
    @procedure_resource = ProcedureResource.find(params[:id])
    @procedure_resource.destroy

   respond_to do |format|
     format.html { redirect_to procedure_resources_url }
     format.json { head :no_content }
   end
 end

private

def signed_in_user
  unless signed_in?
    store_location
    redirect_to signin_path, notice: "Bitte anmelden."
```

```
    end
  end

  def admin_user
    redirect_to(root_path) unless current_user.admin?
  end

end
```

Quellcode 14 RoR-Controller für die Vorgänge

```
class ProceduresController < ApplicationController
  respond_to :html, :json
  before_filter :signed_in_user
  before_filter :admin_user
# GET /procedures
# GET /procedures.json
  def index
    @procedures = Procedure.all
    @project = Project.find(1)
    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @procedures }
    end
  end
# GET /procedures/1
# GET /procedures/1.json
  def show
    @procedure = Procedure.find(params[:id])
    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @procedure }
    end
  end
# GET /procedures/new
# GET /procedures/new.json
  def new
    @procedure = Procedure.new
    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @procedure }
    end
  end
# GET /procedures/1/edit
  def edit
    @procedure = Procedure.find(params[:id])
  end
# POST /procedures
```

```
# POST /procedures.json
  def create
    @procedure = Procedure.new(params[:procedure])
    respond_to do |format|
      if @procedure.save
        format.html { redirect_to @procedure, notice: 'Vorgang wurde ↩
            angelegt' }
        format.json { render json: @procedure, status: :created, location: ↩
            @procedure }
      else
        format.html { render action: "new" }
        format.json { render json: @procedure.errors, status: :↩
            unprocessable_entity }
      end
    end
  end
# PUT /procedures/1
# PUT /procedures/1.json
  def update
    @procedure = Procedure.find(params[:id])
    respond_to do |format|
      if @procedure.update_attributes(params[:procedure])
        format.html { redirect_to @procedure, notice: 'Vorgang wurde ↩
            aktualisiert' }
        format.json { head :no_content }
      else
        format.html { render action: "edit" }
        format.json { render json: @procedure.errors, status: :↩
            unprocessable_entity }
      end
    end
  end
# DELETE /procedures/1
# DELETE /procedures/1.json
  def destroy
    @procedure = Procedure.find(params[:id])
    @procedure.destroy
    respond_to do |format|
      format.html { redirect_to procedures_url }
      format.json { head :no_content }
    end
  end

  private

  def signed_in_user
    unless signed_in?
```

```
      store_location
      redirect_to signin_path, notice: "Bitte anmelden."
    end
  end

  def admin_user
    redirect_to(root_path) unless current_user.admin?
  end

end
```

Quellcode 15 RoR-Controller für das Projekt

```
class ProjectsController < ApplicationController
  respond_to :html, :json
  #before_filter :admin_user
# GET /projects
# GET /projects.json
  def index
    @projects = Project.all
    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @projects }
    end
  end
# GET /projects/1
# GET /projects/1.json
  def show
    @project = Project.find(params[:id])
    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @project }
    end
  end
# GET /projects/new
# GET /projects/new.json
  def new
    @project = Project.new
    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @project }
    end
  end
# GET /projects/1/edit
  def edit
    @project = Project.find(params[:id])
  end
# POST /projects
```

```
# POST /projects.json
  def create
    @project = Project.new(params[:project])
    respond_to do |format|
      if @project.save
        format.html { redirect_to @project, notice: 'Projekt wurde angelegt'↩
            }
        format.json { render json: @project, status: :created, location: ↩
            @project }
      else
        format.html { render action: "new" }
        format.json { render json: @project.errors, status: :↩
            unprocessable_entity }
      end
    end
  end
# PUT /projects/1
# PUT /projects/1.json
  def update
    @project = Project.find(params[:id])
    respond_to do |format|
      if @project.update_attributes(params[:project])
        format.html { redirect_to rcpsp_path, notice: 'Daten wurden ↩
            aktualisiert' }
        format.json { head :no_content }
      else
        format.html { render action: "edit" }
        format.json { render json: @project.errors, status: :↩
            unprocessable_entity }
      end
    end
  end
# DELETE /projects/1
# DELETE /projects/1.json
  def destroy
    @project = Project.find(params[:id])
    @project.destroy
    respond_to do |format|
      format.html { redirect_to projects_url }
      format.json { head :no_content }
    end
  end

  private

  def signed_in_user
    unless signed_in?
```

```
      store_location
      redirect_to signin_path, notice: "Bitte anmelden."
    end
  end

  def admin_user
    redirect_to(root_path) unless current_user.admin?
  end

end
```

Quellcode 16 RoR-Controller für das RCPSP

```
#encoding: UTF-8

class RcpspsController < ApplicationController
  respond_to :html, :json
  before_filter :signed_in_user
  before_filter :admin_user

   def optimize
         if File.exist?("RCPSP1_solution_x.txt")
      File.delete("RCPSP1_solution_x.txt")
    end
    if File.exist?("RCPSP1_solution_zeit.txt")
      File.delete("RCPSP1_solution_zeit.txt")
    end
    if File.exist?("RCPSP1_solution_zw.txt")
      File.delete("RCPSP1_solution_zw.txt")
    end

    @resources = Resource.all
    @procedures = Procedure.all
    @procedure_procedures = ProcedureProcedure.all
    @procedure_resources = ProcedureResource.all
    @projects = Project.all
    @project = Project.find(1)

    @procedures.each { |proc|
      proc.fa=nil
      proc.sa=nil
      proc.fe=nil
      proc.se=nil
      proc.save
    }

    @projects.each { |projekt|
      projekt.zwc=nil
```

```
    projekt.save
  }

  @resources.each { |resources|
    resources.oce=0
    resources.save
  }

  if File.exist?("RCSPSP1_Input.inc")
    File.delete("RCPSP1_Input.inc")
  end
  f=File.new("RCPSP1_Input.inc", "w")

  printf(f, "set r / \n")
  @resources.each { |res| printf(f, res.name + "\n") }
  printf(f, "/;" + "\n\n")

  printf(f, "set i / \n")
  @procedures.each { |proc| printf(f, proc.name + "\n") }
  printf(f, "/;" + "\n\n")

  printf(f, "set t / t0*t")
  printf(f, Procedure.sum(:prot).to_s)
  printf(f, " /;"+ "\n\n")

  printf(f, "VN(h,i)=no;\n\n")

  @procedure_procedures.each { |proc_proc|
    printf(f, "VN('" + proc_proc.prepro.name+"','" + proc_proc.sucpro.name↩
        +"')=yes;\n")
  }

  printf(f, "\n")

  @procedures.each { |time|
    printf(f, "d('" + time.name + "')= "+ time.prot.to_s + ";\n")
  }

  printf(f, "\n")
  printf(f, "k(i,r)=0;\n\n")

  @procedure_resources.each { |k|
    printf(f, "k('" + k.procedure.name + "','" + k.resource.name + "')= "+↩
        k.capa_demand.to_s + ";\n")
  }

  printf(f, "\n")
```

```
@resources.each { |grenze|
  printf(f, "KP('" + grenze.name + "')= "+ User.sum(:capacity, :↩
      conditions => {:resource_id => grenze}).to_s + ";\n")
}

printf(f, "\n")

printf(f, "Deadline= "+ Procedure.sum(:prot).to_s + ";\n")

f.close

if File.exist?("RCPSP1_solution.txt")
  File.delete("RCPSP1_solution.txt")
end

system @project.path.to_s +  " RCPSP1"

redirect_to url_for(:controller => :rcpsps, :action => :solution)

if (File.exist?("RCPSP1_solution_zeit.txt") and File.exists?("↩
    RCPSP1_solution_zw.txt"))

  fi=File.open("RCPSP1_solution_x.txt", "r")
  fi.each { |line|
  sa=line.split(";")
  if sa[0].to_i == 1
  sa0=sa[0]
  sa1=sa[1]
  sa2=sa[2].delete "t" + " \n"
  procedure=Procedure.find_by_name(sa1)
  procedure.optp = sa2
  procedure.save
  end
  }
  fi.close

  fi=File.open("RCPSP1_solution_zeit.txt", "r")
  fi.each { |line|
    sa=line.split(";")
    sa0=sa[0]
    sa1=sa[1]
    sa2=sa[2]
    sa3=sa[3]
    sa4=sa[4].delete " \n"
    procedure=Procedure.find_by_name(sa0)
    procedure.fa = sa1
```

```
      procedure.sa = sa2
      procedure.fe = sa3
      procedure.se = sa4
      procedure.save
    }
    fi.close

    fi=File.open("RCPSP1_solution_zw.txt", "r")
    line=fi.readline
    fi.close
    sa=line.split(" ")
    sa0=sa[0]
    sa1=sa[1].delete " \n"
    project=Project.find_by_id(1)
    project.zwt = sa1
    project.save
  #else
   # flash.now[:not_available] = "Die Lösung wurde noch nicht berechnet!"
  end

end

def solution
  @procedures = Procedure.all
  @project = Project.find(1)
  until File.exist?("RCPSP1_solution_zeit.txt") and File.exists?("↩
     RCPSP1_solution_zw.txt")
    sleep( 5 )
  end

  render 'procedures/index'
end

def optimize2
  if File.exist?("RCPSP2_solution_x.txt")
    File.delete("RCPSP2_solution_x.txt")
  end
  if File.exist?("RCPSP2_solution_kosten.txt")
    File.delete("RCPSP2_solution_kosten.txt")
  end
  if File.exist?("RCPSP2_solution_zeit.txt")
    File.delete("RCPSP2_solution_zeit.txt")
  end
  if File.exist?("RCPSP2_solution_zw.txt")
    File.delete("RCPSP2_solution_zw.txt")
  end
```

```
@resources = Resource.all
@procedures = Procedure.all
@procedure_procedures = ProcedureProcedure.all
@procedure_resources = ProcedureResource.all
@projects = Project.all
@project = Project.find(1)

@procedures.each { |proc|
  proc.fa=nil
  proc.sa=nil
  proc.fe=nil
  proc.se=nil
  proc.save
}

@projects.each { |projekt|
  projekt.zwt=nil
  projekt.save
}

@resources.each { |res|
  res.oce=nil
  res.save
}

if File.exist?("RCSPSP2_Input.inc")
  File.delete("RCPSP2_Input.inc")
end
f=File.new("RCPSP2_Input.inc", "w")

printf(f, "set r / \n")
@resources.each { |res| printf(f, res.name + "\n") }
printf(f, "/;" + "\n\n")

printf(f, "set i / \n")
@procedures.each { |proc| printf(f, proc.name + "\n") }
printf(f, "/;" + "\n\n")

printf(f, "set t / t0*t")
printf(f, Procedure.sum(:prot).to_s)
printf(f, " /;"+ "\n\n")

printf(f, "VN(h,i)=no;\n\n")

@procedure_procedures.each { |proc_proc|
  printf(f, "VN('" + proc_proc.prepro.name+"','" + proc_proc.sucpro.name↩
     +"')=yes;\n")
```

```
}

printf(f, "\n")

@procedures.each { |time|
  printf(f, "d('" + time.name + "')= "+ time.prot.to_s + ";\n")
}

printf(f, "\n")
printf(f, "k(i,r)=0;\n\n")

@procedure_resources.each { |k|
  printf(f, "k('" + k.procedure.name + "','" + k.resource.name + "')= "+↩
      k.capa_demand.to_s + ";\n")
}

printf(f, "\n")

@resources.each { |grenze|
  printf(f, "KP('" + grenze.name + "')= "+ User.sum(:capacity, :↩
     conditions => {:resource_id => grenze}).to_s + ";\n")
}

printf(f, "\n")

@resources.each { |zusatz|
  printf(f, "oc('" + zusatz.name + "')= "+ zusatz.ocr.to_s + ";\n")
}

printf(f, "\n")

deadline = (@project.deadline - @project.startdate).to_i

printf(f, "Deadline=" + deadline.to_s + ";\n")

f.close

if File.exist?("RCPSP2_solution.txt")
  File.delete("RCPSP2_solution.txt")
end

system @project.path.to_s +  " RCPSP2"

redirect_to url_for(:controller => :rcpsps, :action => :solution2)

if (File.exist?("RCPSP2_solution_kosten.txt") and File.exists?("↩
   RCPSP2_solution_zw.txt"))
```

```
  fi=File.open("RCPSP2_solution_x.txt", "r")
  fi.each { |line|
    sa=line.split(";")
    if sa[0].to_i == 1
      sa0=sa[0]
      sa1=sa[1]
      sa2=sa[2].delete "t" + " \n"
      procedure=Procedure.find_by_name(sa1)
      procedure.optp = sa2
      procedure.save
    end
  }
  fi.close

  fi=File.open("RCPSP2_solution_kosten.txt", "r")
  fi.each { |line|
    sa=line.split(";")
    sa0=sa[0]
    sa1=sa[1].delete " \n"
    resource=Resource.find_by_name(sa0)
    resource.oce = sa1
    resource.save
  }
  fi.close
end

if (File.exist?("RCPSP2_solution_zeit.txt"))

  fi=File.open("RCPSP2_solution_zeit.txt", "r")
  fi.each { |line|
    sa=line.split(";")
    sa0=sa[0]
    sa1=sa[1]
    sa2=sa[2]
    sa3=sa[3]
    sa4=sa[4].delete " \n"
    procedure=Procedure.find_by_name(sa0)
    procedure.fa = sa1
    procedure.sa = sa2
    procedure.fe = sa3
    procedure.se = sa4
    procedure.save
  }
  fi.close
end
```

```
    if File.exist?("RCPSP2_solution_zw.txt")
      fi=File.open("RCPSP2_solution_zw.txt", "r")
      line=fi.readline
      fi.close
      sa=line.split(" ")
      sa0=sa[0]
      sa1=sa[1].delete " \n"
      project=Project.find_by_id(1)
      project.zwc = sa1
      project.save
    end

  end

  def solution2
    @resources = Resource.all
    @project = Project.find(1)
    until File.exist?("RCPSP2_solution_zeit.txt") and File.exists?("↩
        RCPSP2_solution_zw.txt") and File.exist?("RCPSP2_solution_kosten.txt"↩
        )
      sleep( 5 )
    end

    render 'resources/index'
  end

  private

  def signed_in_user
    unless signed_in?
      store_location
      redirect_to signin_path, notice: "Bitte anmelden."
    end
  end

  def admin_user
    redirect_to(root_path) unless current_user.admin?
  end

end
```

Quellcode 17 RoR-Controller für die Ressourcen

```
class ResourcesController < ApplicationController
  respond_to :html, :json
  before_filter :signed_in_user, only: [:edit, :update, :new, :create, :↩
      destroy]
  before_filter :admin_user, only: [:edit, :update, :new, :create, :destroy]
```

```
# GET /resources
# GET /resources.json
  def index
    @resources = Resource.all
    @procedures = Procedure.all
    @project = Project.find(1)
    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @resources }
    end
  end
# GET /resources/1
# GET /resources/1.json
  def show
    @resource = Resource.find(params[:id])
    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @resource }
    end
  end
# GET /resources/new
# GET /resources/new.json
  def new
    @resource = Resource.new
    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @resource }
    end
  end
# GET /resources/1/edit
  def edit
    @resource = Resource.find(params[:id])
  end
# POST /resources
# POST /resources.json
  def create
    @resource = Resource.new(params[:resource])
    respond_to do |format|
      if @resource.save
        format.html { redirect_to @resource, notice: 'Ressource wurde ↩
            angelegt' }
        format.json { render json: @resource, status: :created, location: ↩
            @resource }
      else
        format.html { render action: "new" }
        format.json { render json: @resource.errors, status: :↩
            unprocessable_entity }
```

```
      end
    end
  end
# PUT /resources/1
# PUT /resources/1.json
  def update
    @resource = Resource.find(params[:id])
    respond_to do |format|
      if @resource.update_attributes(params[:resource])
        format.html { redirect_to @resource, notice: 'Ressource wurde ↩
            aktualisiert' }
        format.json { head :no_content }
      else
        format.html { render action: "edit" }
        format.json { render json: @resource.errors, status: :↩
            unprocessable_entity }
      end
    end
  end
# DELETE /resources/1
# DELETE /resources/1.json
  def destroy
    @resource = Resource.find(params[:id])
    @resource.destroy
    respond_to do |format|
      format.html { redirect_to resources_url }
      format.json { head :no_content }
    end
  end

  private

  def signed_in_user
    unless signed_in?
      store_location
      redirect_to(root_path)
    end
  end

  def correct_user
    @user = User.find(params[:id])
    redirect_to(root_path) unless current_user?(@user)
  end

  def admin_user
    redirect_to(root_path) unless current_user.admin?
  end
```

```
end
```

Quellcode 18 RoR-Controller für die statischen Seiten

```
class StaticPagesController < ApplicationController
  def home
  end

  def help
  end

  def about
  end

  def contact
  end

  def rcpsp
    @project = Project.find(1)
  end

end
```

Quellcode 19 RoR-Controller für die User

```
class UsersController < ApplicationController
  before_filter :signed_in_user, only: [:index, :show, :edit, :update, :↩
      destroy]
  before_filter :correct_user, only: [:edit, :update]
  before_filter :admin_user, only: [:destroy]

  def show
    @user = User.find(params[:id])
    @resource = Resource.find(@user.resource_id)
  end

  def new
    @user = User.new
  end

  def create
    @user = User.new(params[:user])
    if @user.save
      sign_in @user
      flash[:success] = "Willkommen zur Projektplanung!"
      redirect_to @user
    else
```

```
      render 'new'
    end
  end

  def edit
  end

  def update
    if @user.update_attributes(params[:user])
      flash[:success] = "Profile updated"
      sign_in @user
      redirect_to @user
    else
      render 'edit'
    end
  end

  def index
    @users = User.paginate(page: params[:page])
  end

  def destroy
    User.find(params[:id]).destroy
    flash[:success] = "User destroyed."
    redirect_to users_url
  end

  private

  def signed_in_user
    unless signed_in?
      store_location
      redirect_to signin_path, notice: "Bitte anmelden."
    end
  end

  def correct_user
    @user = User.find(params[:id])
    redirect_to(root_path) unless current_user?(@user)
  end

  def admin_user
    redirect_to(root_path) unless current_user.admin?
  end

end
```

Quellcode 20 RoR-Modell für die Vorgangsrelationen

```
class ProcedureProcedure < ActiveRecord::Base
  attr_accessible :prepro_id, :sucpro_id

  belongs_to :prepro, class_name: "Procedure"
  belongs_to :sucpro, class_name: "Procedure"

  end
```

Quellcode 21 RoR-Modell für die Vorgangs-Ressourcen-Kombinationen

```
class ProcedureResource < ActiveRecord::Base
  attr_accessible :procedure_id, :resource_id, :capa_demand

  belongs_to :procedure, class_name: "Procedure"
  belongs_to :resource, class_name: "Resource"

  validates :capa_demand, :numericality => {:only_integer => true}

end
```

Quellcode 22 RoR-Modell für die Vorgänge

```
class Procedure < ActiveRecord::Base
  attr_accessible :created_at, :fa, :fe, :name, :prot, :sa, :se, :updated_at↩
      , :optp

  has_many :procedure_resources, :dependent => :destroy
  has_many :resources, through: :procedure_resources
  has_many :procedure_procedures, foreign_key: "prepro_id", :dependent => :↩
      destroy
  has_many :reverse_procedure_procedures, foreign_key: "sucpro_id", ↩
      class_name: "ProcedureProcedure", :dependent => :destroy

  validates :prot, :numericality => {:only_integer => true}
  validates :name, length: { minimum: 12}

end
```

Quellcode 23 RoR-Modell für das Projekt

```
class Project < ActiveRecord::Base
  attr_accessible :created_at, :startdate, :deadline, :gvp, :updated_at, :↩
      path, :zwt, :zwc, :totalc, :extrac
end
```

Quellcode 24 RoR-Modell für die Ressourcen

```
class Resource < ActiveRecord::Base
  attr_accessible :created_at, :name, :ocr, :cost, :oce

  has_many :procedure_resources, :dependent => :destroy
  has_many :procedures, through: :procedure_resources
  has_many :users, :dependent => :destroy

  validates :ocr, :numericality => {:only_integer => true}
  validates :cost, :numericality => {:only_integer => true}

end
```

Quellcode 25 RoR-Modell für die User

```
class User < ActiveRecord::Base
  attr_accessible :email, :name, :password, :password_confirmation, :↩
     capacity, :resource_id
  has_secure_password

  belongs_to :resource

  before_save { |user| user.email = email.downcase }
  before_save :create_remember_token

  validates :name, presence: true, length: { maximum: 50 }

  VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i
  validates :email, presence: true, format: { with: VALID_EMAIL_REGEX },
            uniqueness: { case_sensitive: false}
  validates :password, presence: true, length: { minimum: 6}
  validates :password_confirmation, presence: true
  validates :capacity, :numericality => {:only_integer => true}

  private
    def create_remember_token
      self.remember_token = SecureRandom.urlsafe_base64
    end
end
```

Quellcode 26 RoR-Seite für die Vorgangsrelationen - Formular

```
<%= form_for(@procedure_procedure) do |f| %>
    <% if @procedure_procedure.errors.any? %>
        <div id="error_explanation">
          <h2><%= pluralize(@procedure_procedure.errors.count, "error") %> ↩
              prohibited this procedure_procedure from being
            saved:</h2>
```

```
          <ul>
            <% @procedure_procedure.errors.full_messages.each do |msg| %>
                <li><%= msg %></li>
            <% end %>
          </ul>
        </div>
    <% end %>

    <div id="prepro_id" class="field">
      <%= f.label 'Von Vorgang' %><br />
      <%= f.collection_select :prepro_id, Procedure.all, :id, :name %>
    </div>

    <div class="field">
      <%= f.label 'Zu Vorgang' %><br />
      <%= f.collection_select :sucpro_id, Procedure.all, :id, :name %>
    </div>

    <div class="actions">
      <%= f.submit 'Relation zwischen den Vorgängen anlegen', :class => "btn↩
        " %>
    </div>
<% end %>
```

Quellcode 27 RoR-Seite für die Vorgangsrelationen - Übersicht

```
<% provide(:title, 'Übersicht der Vorgangsrelationen') %>
<h1>Übersicht der Vorgangsrelationen</h1>

<div class="center">
<th><%= link_to "Graph anzeigen", procedure_procedure_graph_path, class: "↩
    btn btn-info btn-large" %></th>
</div>

<br/>

<table id="procedure_periods" class="table table-hover">
  <thead>
  <tr>
    <th>Von Vorgang</th>
    <th>Zu Vorgang</th>
    <th></th>
  </tr>
  </thead>
  <tbody>
  <% @procedure_procedures.each do |procedure_procedure| %>
      <tr>
        <td><%= procedure_procedure.prepro.name %></td>
```

```
        <td><%= procedure_procedure.sucpro.name %></td>
        <td><%= link_to 'Löschen', procedure_procedure, method: :delete, ↩
            data: {confirm: 'Sind Sie sicher?'} %></td>
      </tr>
  <% end %>
  </tbody>
</table>
<br/>

<div class="btn-group btn-lg">
<%= link_to 'Neue Relation anlegen', new_procedure_procedure_path, :class =>↩
    "btn"%>
<%= link_to 'Zurück zur Projektplanung', rcpsp_path, :class => "btn"%>
</div>
```

Quellcode 28 RoR-Seite für die Vorgangsrelationen - Erstellung

```
<% provide(:title, 'Neue topologische Reihenfolge') %>
<h1>Neue topologische Reihenfolge</h1>

<%= render 'form' %>

<%= link_to 'Zurück', :back, :class => "btn" %>
```

Quellcode 29 RoR-Seite für die Vorgangsrelationen - Grafische Darstellung

```
<% provide(:title, 'Graph zur aktuellen Projektplanung') %>
<h1>Graph zur aktuellen Projektplanung</h1>

<br/>

<div class="center">
<img src="/assets/graph.png" />
</div>

<br/>

<div class="btn-group btn-lg">
  <%= link_to 'Zurück zu den Vorgängen', procedures_path, :class => "btn"%>
  <%= link_to 'Zurück zu den Vorgangsrelationen', procedure_procedures_path,↩
      :class => "btn"%>
  <%= link_to 'Zurück zur Projektplanung', rcpsp_path, :class => "btn"%>
</div>
```

Quellcode 30 RoR-Seite für die Vorgangs-Ressourcen-Kombinationen - Formular

```
<%= form_for(@procedure_resource) do |f| %>
    <% if @procedure_resource.errors.any? %>
```

```
        <div id="error_explanation">
          <h2><%= pluralize(@procedure_resource.errors.count, "error") %> ↩
              prohibited this procedure_resource from being
            saved:</h2>

          <ul>
            <% @procedure_resource.errors.full_messages.each do |msg| %>
                <li><%= msg %></li>
            <% end %>
          </ul>
        </div>
    <% end %>

    <div class="field">
      <%= f.label 'Vorgang' %><br />
      <%= f.collection_select :procedure_id, Procedure.all, :id, :name %>
    </div>

    <div class="field">
      <%= f.label 'Ressource' %><br />
      <%= f.collection_select :resource_id, Resource.all, :id, :name %>
    </div>

    <div class="field">
      <%= f.label 'Kapazitätsbedarf' %><br />
      <%= f.text_field :capa_demand %>
    </div>

    <div class="actions">
      <%= f.submit 'Vorgang zur Ressource zuordnen oder aktualisieren', :↩
          class => "btn" %>
    </div>
<% end %>
```

Quellcode 31 RoR-Seite für die Vorgangs-Ressourcen-Kombinationen - Bearbeitung

```
<% provide(:title, 'Zuordnung des Vorgangs zur Ressource ändern') %>
<h1>Zuordnung des Vorgangs zur Ressource ändern</h1>

<%= render 'form' %>

<div class="btn-group btn-lg">
<%= link_to 'Anzeigen', @procedure_resource, :class => "btn" %>
<%= link_to 'Zurück', procedure_resources_path, :class => "btn" %>
</div>
```

Quellcode 32 RoR-Seite für die Vorgangs-Ressourcen-Kombinationen - Übersicht

```
<% provide(:title, 'Übersicht der Zuordnung der Vorgänge zu den Ressourcen')↩
    %>
<h1>Übersicht der Zuordnung der Vorgänge zu den Ressourcen</h1>
<table id="procedure_resources" class="table table-hover">
  <thead>
  <tr>
    <th>Vorgang</th>
    <th>Ressource</th>
    <th>Kapazitätsbedarf</th>
    <th></th>
  </tr>
  </thead>
  <tbody>

  <% @procedure_resources.each do |procedure_resource| %>
      <tr>
        <td><%= procedure = Procedure.find(procedure_resource.procedure_id)
                procedure.name %></td>
        <td><%= resource = Resource.find(procedure_resource.resource_id)
                resource.name %></td>
        <td><%= procedure_resource.capa_demand  %></td>
        <td><%= link_to 'Anzeigen', procedure_resource %></td>
        <td><%= link_to 'Ändern', edit_procedure_resource_path(↩
            procedure_resource) %></td>
        <td><%= link_to 'Löschen', procedure_resource, method: :delete, data↩
            : {confirm: 'Sind Sie sicher?'} %></td>
      </tr>
  <% end %>
  </tbody>
</table>

<br/>

<div class="btn-group btn-lg">
<%= link_to 'Vorgang einer Ressource neu zuordnen', ↩
    new_procedure_resource_path, :class => "btn" %>
<%= link_to 'Zurück zur Projektplanung', rcpsp_path, :class => "btn" %>
</div>
```

Quellcode 33 RoR-Seite für die Vorgangs-Ressourcen-Kombinationen - Erstellung

```
<% provide(:title, 'Neue Vorgangs-Ressourcen-Kombination') %>
<h1>Neue Vorgangs-Ressourcen-Kombination</h1>

<%= render 'form' %>

<%= link_to 'Zurück', :back, :class => "btn" %>
```

Quellcode 34 RoR-Seite für die Vorgangs-Ressourcen-Kombinationen - Anzeige

```
<% provide(:title, 'Vorgangs-Ressourcen-Kombination') %>
<h1>Vorgangs-Ressourcen-Kombination</h1>

<p>
  <b>Vorgang:</b>
<td><%= procedure = Procedure.find(@procedure_resource.procedure_id)
        procedure.name %></td>
</p>

<p>
  <b>Ressource:</b>
<td><%= resource = Resource.find(@procedure_resource.resource_id)
        resource.name %></td>
</p>

<p>
  <b>Kapazitätsbedarf:</b>
<td><%= @procedure_resource.capa_demand %></td>
</p>

<div class="btn-group btn-lg">
<%= link_to 'Ändern', edit_procedure_resource_path(@procedure_resource), :↩
    class => "btn" %>
<%= link_to 'Zurück', procedure_resources_path, :class => "btn" %>
</div>
```

Quellcode 35 RoR-Seite für die Vorgänge - Formular

```
<%= form_for(@procedure) do |f| %>
  <% if @procedure.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@procedure.errors.count, "error") %> prohibited this↩
          procedure from being saved:</h2>

      <ul>
      <% @procedure.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :name %><br />
```

```
    <%= f.text_field :name %>
  </div>
  <div class="field">
    <%= f.label 'Vorgangsdauer' %><br />
    <%= f.text_field :prot %>
  </div>
  <div class="actions">
    <%= f.submit 'Vorgang anlegen oder aktualisieren', :class => "btn" %>
  </div>
<% end %>
```

Quellcode 36 RoR-Seite für die Vorgänge - Bearbeitung

```
<% provide(:title, 'Vorgangsbearbeitung') %>
<h1>Vorgangsbearbeitung</h1>

<%= render 'form' %>

<div class="btn-group btn-lg">
<%= link_to 'Anzeigen', @procedure, :class => "btn" %>
<%= link_to 'Zurück', procedures_path, :class => "btn" %>
</div>
```

Quellcode 37 RoR-Seite für die Vorgänge - Übersicht

```
<% provide(:title, 'Übersicht der Vorgänge') %>
<h1>Übersicht der Vorgänge</h1>

<div class="center">
  <th><%= link_to "Graph anzeigen", procedure_procedure_graph_path, class: "↩
      btn btn-info btn-large" %></th>
</div>

<br/>

<% if @project.zwt!=nil %>
    Die minimale Projektdauer beträgt <strong><%= Procedure.maximum("optp")↩
        %></strong> Zeiteinheiten. Die Optimierung erfolgte mit einem ↩
       Datenstand vom <strong><%= @project.updated_at.strftime("%d.%m.%Y") ↩
       %>. </strong><br> <br>
<% end %>

<table id="procedures" class="table table-hover">
  <thead>
  <tr>
    <th>Name</th>
    <th>Vorgangs- dauer</th>
    <th>Anzahl Vorgänger</th>
```

```
    <th>Anzahl Ressourcen</th>
    <th>FA*</th>
    <th>SA*</th>
    <th>FE*</th>
    <th>SE*</th>
    <th><b>Vorgangs- abschluss</b></th>
    <th></th>
    <th></th>
    <th></th>
  </tr>
  </thead>
  <tbody>

<% @procedures.each do |procedure| %>
  <tr>
    <td><%= procedure.name %></td>
    <td><%= procedure.prot %></td>
    <td><%= procedure.reverse_procedure_procedures.count %></td>
    <td><%= procedure.resources.count %></td>
    <td><%= procedure.fa %></td>
    <td><%= procedure.sa %></td>
    <td><%= procedure.fe %></td>
    <td><%= procedure.se %></td>
    <td><b><%= procedure.optp %></b></td>
    <td><%= link_to 'Anzeigen', procedure %></td>
    <td><%= link_to 'Ändern', edit_procedure_path(procedure) %></td>
    <td><%= link_to 'Löschen', procedure, method: :delete, data: { confirm: ↩
        'Sind sie sicher?' } %></td>
  </tr>
<% end %>
</tbody>
</table>

<br />

<div class="btn-group btn-lg">
<%= link_to 'Neuer Vorgang anlegen', new_procedure_path, :class => "btn" %>
<%= link_to 'Zurück zur Projektplanung', rcpsp_path, :class => "btn" %>
</div>

<br />

<thead>
*FA=frühester Anfangszeitpunkt, SA=spätester Anfangszeitpunkt, FE=frühester ↩
   Endzeitpunkt und SE=spätester Endzeitpunkt in Tagen nach Start des ↩
   Projekts.
</thead>
```

```
<br/>
```

Quellcode 38 RoR-Seite für die Vorgänge - Erstellung

```
<% provide(:title, 'Neuer Vorgang') %>
<h1>Neuer Vorgang</h1>

<%= render 'form' %>

<%= link_to 'Zurück', procedures_path, :class => "btn" %>
```

Quellcode 39 RoR-Seite für die Vorgänge - Anzeige

```
<% provide(:title, 'Vorgang') %>
<h1>Vorgang</h1>

<p>
  <b>Name:</b>
  <%= @procedure.name %>
</p>

<p>
  <b>Vorgangsdauer:</b>
  <%= @procedure.prot %>
</p>

<div class="btn-group btn-lg">
<%= link_to 'Zurück', procedures_path, :class => "btn" %>
<%= link_to 'Ändern', edit_procedure_path(@procedure), :class => "btn" %>
<%= link_to 'Vorgang einer Ressource zuordnen', new_procedure_resource_path,↩
    :class => "btn" %>
<%= link_to 'Neue Vorgangsrelation anlegen', new_procedure_procedure_path, :↩
   class => "btn"%>
</div>

<br/><br/>
<div class="row">
  <div class="span6">
<table id="procedure_resources" class="table table-hover">
  <thead>
  <tr>
    <th>Ressource</th>
    <th>Kapazitätsbedarf</th>
    <th></th>
  </tr>
  </thead>
  <tbody>
```

```
  <% @procedure.procedure_resources.each do |procedure_resource| %>
      <tr>
        <td><%= Resource.find(procedure_resource.resource_id).name %></td>
        <td><%= procedure_resource.capa_demand  %></td>
        <td><%= link_to 'Ändern', edit_procedure_resource_path(↩
            procedure_resource) %></td>
        <td><%= link_to 'Löschen', procedure_resource, method: :delete, data↩
            : {confirm: 'Sind Sie sicher?'} %></td>
      </tr>
  <% end %>
  </tbody>
</table>
</div>

  <div class="span6">
<table id="procedure_periods" class="table table-hover">
  <thead>
  <tr>
    <th>Vorgänger</th>
    <th></th>
  </tr>
  </thead>
  <tbody>
  <% @procedure.reverse_procedure_procedures.each do |procedure_procedure| ↩
      %>
      <tr>
        <td><%= procedure_procedure.prepro.name %></td>
        <td><%= link_to 'Löschen', procedure_procedure, method: :delete, ↩
            data: {confirm: 'Sind Sie sicher?'} %></td>
      </tr>
  <% end %>
  </tbody>
</table>
  </div>
</div>
```

Quellcode 40 RoR-Seite für die Ressourcen - Formular

```
<%= form_for(@resource) do |f| %>
  <% if @resource.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@resource.errors.count, "error") %> prohibited this ↩
          resource from being saved:</h2>
      <ul>
      <% @resource.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
```

```
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :name %><br />
    <%= f.text_field :name %>
  </div>
    <div class="field">
      <%= f.label 'Kosten je Kapazität' %><br />
      <%= f.text_field :cost %>
    </div>
    <div class="field">
      <%= f.label 'Zusatzkostensatz je Kapazität' %><br />
      <%= f.text_field :ocr %>
    </div>
  <div class="actions">
    <%= f.submit 'Ressource anlegen oder aktualisieren', :class => "btn" %>
  </div>
<% end %>
```

Quellcode 41 RoR-Seite für die Ressourcen - Tabelle als unangemeldeter User

```
<table id="resources" class="table table-hover">
  <thead>
  <tr>
    <th>Name</th>
    <th></th>
  </tr>
  </thead>
  <tbody>
  <% elements = [] %>
  <% @resources.each do |resource| %>
      <tr>
        <td><%= resource.name %></td>
            <td><%= link_to 'Bewerben', signup_path %></td>
      </tr>
  <% end %>
  </tbody>
</table>

<%= link_to 'Hier geht es zur Anmeldung', signup_path, :class => "btn" %>
```

Quellcode 42 RoR-Seite für die Ressourcen - Tabelle als angemeldeter User

```
<% if @project.zwc!=nil %>
    Die Projektzusatzkosten betragen <strong><%=  @project.zwc %></strong> ↩
       Geldeinheiten. Die Optimierung erfolgte mit einem Datenstand vom <↩
```

```
        strong><%=  @project.updated_at.strftime("%d.%m.%Y") %>.</strong><br>↩
         <br>
<% end %>

<table id="resources" class="table table-hover">
  <thead>
  <tr>
    <th>Name</th>
    <th>Gesamt- kapazität</th>
    <th>Kosten/ ME</th>
    <th>Grund- kosten</th>
    <th>Zusatz- kosten/ ME</th>
    <th>Zusatz- einheiten</th>
    <th>Zusatzkosten Gesamt</th>
    <th>Gesamtkosten</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>
  </thead>
  <tbody>

  <% @resources.each do |resource| %>
      <tr>
        <td><%= resource.name %></td>
        <td><%= User.sum(:capacity, :conditions => {:resource_id => resource↩
            }) %></td>
        <td><%= resource.cost %></td>
        <th><%= resource.cost * resource.procedures.sum(&:prot) %></th>
        <td><%= resource.ocr %></td>
        <td><%= resource.oce %></td>
        <th><%= resource.ocr * resource.oce %></th>
        <th><%= resource.ocr * resource.oce + (resource.cost *  resource.↩
            procedures.sum(&:prot) ) %></th>
        <td><%= link_to 'Anzeigen', resource %></td>
        <% if signed_in? && current_user.admin? %>
            <td><%= link_to 'Ändern', edit_resource_path(resource) %></td>
            <td><%= link_to 'Löschen', resource, method: :delete, data: { ↩
                confirm: 'Sind sie sicher?' } %></td>
        <% end %>
      </tr>
  <% end %>
  </tbody>
</table>

<br/>
<br/>
```

```
<% begin %>
<% if @project.zwc!=nil %>
    <%= javascript_include_tag "//www.google.com/jsapi" %>
    <%= javascript_include_tag "chartkick" %>
    <% require 'chartkick' %>
    <b>Benötigte Zusatzeinheiten:</b>
    <%= bar_chart Resource.group(:name).sum(:oce) %>
<% end %>
<% end %>

<br />

<% if current_user.admin? %>
    <div class="btn-group btn-lg">
      <%= link_to 'Neue Ressource anlegen', new_resource_path, :class => "↩
          btn" %>
      <%= link_to 'Zurück zur Projektplanung', rcpsp_path, :class => "btn" ↩
          %>
    </div>
<% end %>
```

Quellcode 43 RoR-Seite für die Ressourcen - Bearbeitung

```
<% provide(:title, 'Bearbeitung der Ressource') %>
<h1>Bearbeitung der Ressource</h1>

<%= render 'form' %>

<div class="btn-group btn-lg">
<%= link_to 'Anzeigen', @resource, :class => "btn" %>
<%= link_to 'Zurück', resources_path, :class => "btn" %>
</div>
```

Quellcode 44 RoR-Seite für die Ressourcen - Übersicht

```
<% provide(:title, 'Übersicht der Ressourcen') %>
<h1>Übersicht der Ressourcen</h1>

<% if signed_in? %>
<%= render 'resources/signed_in' %>
<% else %>
<%= render 'resources/free' %>
<% end %>
```

Quellcode 45 RoR-Seite für die Ressourcen - Erstellung

```
<% provide(:title, 'Neue Ressource') %>
<h1>Neue Ressource</h1>

<%= render 'form' %>

<%= link_to 'Zurück', resources_path, :class => "btn" %>
```

Quellcode 46 RoR-Seite für die Ressourcen - Anzeige

```
<% provide(:title, 'Ressource') %>
<h1>Ressource</h1>

<p>
  <b>Name:</b>
  <%= @resource.name %>
</p>

<p>
  <b>Kosten je Kapazität:</b>
  <%= @resource.cost %>
</p>

<p>
  <b>Zusatzkosten je Kapazität:</b>
  <%= @resource.ocr %>
</p>

<p>
  <b>Aktuelle Gesamkapazität:</b>
  <%= User.sum(:capacity, :conditions => {:resource_id => @resource}) %>
</p>

<% if current_user.admin? %>
    <div class="btn-group btn-lg">
<%= link_to 'Ändern', edit_resource_path(@resource), :class => "btn" %>
    <% end %>
<%= link_to 'Zurück', resources_path, :class => "btn" %>
</div>
```

Quellcode 47 RoR-Seite für die Optimierungsseite zur Projektplanung

```
<% provide(:title, 'RCPSP') %>
<h1>Projektplanung mit dem RCPSP</h1>

<h2>Verwaltung</h2>

<div class="btn-group btn-lg">
```

```
  <%= link_to "Mitarbeiter", users_path, :class => "btn" %>
  <%= link_to "Vorgänge", procedures_path, :class => "btn" %>
  <%= link_to "Vorgangsrelationen", procedure_procedures_path, :class => "↩
      btn" %>
  <%= link_to "Ressourcen", resources_path, :class => "btn" %>
  <%= link_to "Vorgang-Ressoucen-Kombination", procedure_resources_path, :↩
      class => "btn" %>
</div>

<br>

<p>
  Auf dieser Seite werden die Grunddaten zur Durchführung der Projektplanung↩
       angepasst und die Optimierung durchgeführt: <br> <br>
</p>

<% if signed_in? %>

<div class="row">
 <div class="span5">
    <h2>Programme</h2>
    <%= form_for(@project) do |f| %>
        <div class="field">
          <%= f.label :path, "GAMS-Verzeichnis" %>
          <%= f.text_field :path %>
          <a href="http://www.gams.com/download/">(GAMS-Download)</a>
        </div>

        <br/>

        <div class="field">
          <%= f.label :gvp, "GraphViz-Verzeichnis" %>
          <%= f.text_field :gvp %>
          <a href="http://www.graphviz.org/Download..php">(GraphViz-Download↩
              )</a>
        </div>

        <div class="actions">
          <%= f.submit 'Verzeichnisse aktualisieren', :class => "btn"  %>
        </div>
     </div>

     <div class="span6">
      <h2>Start der Kapazitätsplanung</h2>

      <div class="field">
           <%= f.label :startdate, "Starttermin" %>
```

```
          <%= f.text_field :startdate, :class => "datepicker", :value => ↩
              @project.startdate.strftime("%d.%m.%Y")    %>
        </div>
        <div class="actions">
          <%= f.submit 'Starttermin aktualisieren', :class => "btn"  %>
        </div>
       <table>
        <tr>
          <th><%= link_to "Optimiere Kapazitätsplanung", ↩
              rcpsp_optimize_path, :class => "btn", data: {disable_with: "<i↩
               class='fa fa-spinner fa-spin'></i> Berechnung gestartet..."} ↩
              %></th>
        </tr>
       </table>
     <% end %>

       <br/>

    <h2>Start der Kostenplanung</h2>
    <%= form_for(@project) do |f| %>
       <div class="field">
          <%= f.label :deadline, "Deadline-Termin" %>
          <%= f.text_field :deadline, :class => "datepicker", :value => ↩
              @project.deadline.strftime("%d.%m.%Y")    %>
       </div>
       <div class="actions">
           <%= f.submit 'Deadline aktualisieren', :class => "btn"  %>
       </div>
        <table>
          <tr>
            <th><%= link_to "Optimiere Kostenplanung", rcpsp_optimize2_path,↩
                :class => "btn", data: {disable_with: "<i class='fa fa-↩
               spinner fa-spin'></i> Berechnung gestartet..."} %>
          </tr>
        </table>
    <% end %>
<% end %>

 </div>
</div>

<script>
    $('.datepicker').datepicker({
        src:'js/bootstrap-datepicker.de.js',
        language:'de',
        format: 'dd.mm.yyyy'
    });
```

```
</script>
```

Quellcode 48 RoR-Seite für die Startseite

```
<div class="center hero-unit">
  <h1>Projektplanung</h1>
  <br/>
  <br/>
  <% if signed_in? %>
      <th><%= link_to "Zur Optimierung", rcpsp_path, class: "btn btn-info ↩
          btn-large" %></th>
<%else %>
      <th><%= link_to "Anmelden", signup_path, class: "btn btn-primary btn-↩
          large" %></th>
      <th><%= link_to "Login", signin_path, class: "btn btn-info btn-large" ↩
          %></th>
  <% end %>
</div>
```

Quellcode 49 Kopfzeile der Web-Apllikation

```
<header class="navbar navbar-fixed-top">
  <div class="navbar-inner">
    <div class="container">
      <%= link_to "Projektplanung", root_path, id: "logo" %>
      <nav>
        <ul class="nav pull-right">
          <li><%= link_to "Startseite", root_path, id: "main_links" %></li>
          <li><%= link_to "Hilfe", help_path, id: "main_links" %></li>
          <li><%= link_to "Ressource", resources_path, id: "main_links" %></↩
              li>

          <% if signed_in? %>
              <li><%= link_to "Mitarbeiter", users_path, id: "main_links" ↩
                  %></li>
              <li id="fat-menu" class="dropdown">
                <a href="#" class="dropdown-toggle" data-toggle="dropdown"> ↩
                    Menü <b class="caret"></b>
                </a>
                <ul class="dropdown-menu">
                  <li><%= link_to "Profil", current_user %></li>
                  <li><%= link_to "Einstellungen", edit_user_path(↩
                      current_user) %></li>
                  <% if current_user.admin? %>
                      <li><%= link_to "Projektplanung", rcpsp_path %></li>
                  <% end %>
                  <li class="divider"></li>
                  <li>
```

```
                    <%= link_to "Logout", signout_path, method: "delete" %>
                  </li>
                </ul>
              </li>
          <% else %>
              <li><%= link_to "Login", signin_path %></li>
          <% end %>
        </ul>
      </nav>
    </div>
  </div>
</header>
```

Quellcode 50 RoR-Seite für die User - Übersichtsseite

```
<li>
  <%= gravatar_for user, size: 52 %>
  <%= link_to user.name, user %>

  <% if current_user.admin? && !current_user?(user) %>
    | <%= link_to "Löschen", user,   method: :delete,
                                     data: { confirm: "Bist du dir sicher?" ↩
                                        } %>
  <% end %>
</li>
```

Quellcode 51 RoR-Seite für die User - Bearbeitung

```
<% provide(:title, 'Profil') %>
<h1>Aktualisiere Dein Profil</h1>

<div class="row">
  <div class="span6 offset3">
    <%= form_for(@user) do |f| %>
        <%= render 'shared/error_messages' %>

        <%= f.label :name %>
        <%= f.text_field :name %>

        <%= f.label :email %>
        <%= f.text_field :email %>

        <%= f.label :capacity, "Arbeitszeit pro Tag" %>
        <%= f.text_field :capacity %>

        <%= f.label :resource_id, "Welche Ressource?" %>
        <%= f.collection_select :resource_id, Resource.all, :id, :name %>
```

```
        <%= f.label :password, "Passwort" %>
        <%= f.password_field :password %>

        <%= f.label :password_confirmation, "Bestätigung" %>
        <%= f.password_field :password_confirmation %>

        <%= f.submit "Speichere Änderungen", class: "btn btn-large btn-↩
            primary" %>
    <% end %>
    <%= gravatar_for @user %>
    <a href="http://gravatar.com/emails">Profilbild bearbeiten</a>
  </div>
</div>
```

Quellcode 52 RoR-Seite für die User - User-/Mitarbeiterübersicht

```
<% provide(:title, 'Alle Mitarbeiter') %>
<h1>Alle Mitarbeiter</h1>

<%= will_paginate %>

<ul class="users">
      <%= render @users %>
</ul>

<%= will_paginate %>
```

Quellcode 53 RoR-Seite für die User - Erstellung

```
<% provide(:title, 'Melde Dich an') %>
<h1>Melde Dich an</h1>

<div class="row">
  <div class="span6 offset3">
    <%= form_for(@user) do |f| %>
      <%=  render 'shared/error_messages' %>

      <%= f.label :name %>
      <%= f.text_field :name %>

      <%= f.label :email %>
      <%= f.text_field :email %>

      <%= f.label :capacity, "Arbeitszeit pro Tag" %>
      <%= f.text_field :capacity %>

      <%= f.label :resource_id, "Welche Ressource?" %>
      <%= f.collection_select :resource_id, Resource.all, :id, :name %>
```

```
      <%= f.label :password, "Passwort" %>
      <%= f.password_field :password %>

      <%= f.label :password_confirmation, "Bestätigung" %>
      <%= f.password_field :password_confirmation %>

      <%= f.submit "Erzeuge mein Konto", class: "btn btn-large btn-primary" ↩
          %>
    <% end %>
    <p>Du hast ein Konto? <%= link_to "Hier geht es zum Login!", signin_path↩
        %></p>
  </div>
</div>
```

Quellcode 54 RoR-Seite für die User - Anzeige

```
<% provide(:title, @user.name) %>
<div class="row">
  <aside class="span4">
    <section>
      <h1>
        <%= gravatar_for @user %>
        <%= @user.name %>
      </h1>
    </section>
  </aside>
</div>

<p>
  <b>Arbeitszeit pro Tag:</b>
  <%= @user.capacity %>
</p>

<p>
  <b>Rolle im Projekt:</b>
  <%= @resource.name%>
</p>

<% if current_user?(@user) %>
    <%= link_to 'Einstellung', edit_user_path(current_user), :class => "btn"↩
        %>
<% end %>
<table id="procedures" class="table table-hover">
  <thead>
  <tr>
    <th>Name</th>
    <th>Vorgangsdauer</th>
```

```
    <th>FA*</th>
    <th>SA*</th>
    <th>FE*</th>
    <th>SE*</th>
    <th><b>Vorgangsabschluss</b></th>
    <th></th>
    <th></th>
    <th></th>
  </tr>
  </thead>
  <tbody>

  <% @resource.procedures.each do |procedure| %>
      <tr>
        <td><%= procedure.name %></td>
        <td><%= procedure.prot %></td>
        <td><%= procedure.fa %></td>
        <td><%= procedure.sa %></td>
        <td><%= procedure.fe %></td>
        <td><%= procedure.se %></td>
        <td><b><%= procedure.optp %></b></td>
        <% if current_user.admin? %>
            <td><%= link_to 'Anzeigen', procedure %></td>
            <td><%= link_to 'Ändern', edit_procedure_path(procedure) %></td>
            <td><%= link_to 'Löschen', procedure, method: :delete, data: { ↩
                confirm: 'Sind sie sicher?' } %></td>
        <% end %>
      </tr>
  <% end %>
  </tbody>
</table>
  <%= link_to 'Zurück', users_path, :class => "btn" %>

<br>

<thead>
*FA=frühester Anfangszeitpunkt, SA=spätester Anfangszeitpunkt, FE=frühester ↩
    Endzeitpunkt und SE=spätester Endzeitpunkt in Tagen nach Start des ↩
    Projekts.
</thead>
```

Quellcode 55 RoR-Datenbankschema

```
# encoding: UTF-8
# This file is auto-generated from the current state of the database. ↩
    Instead
# of editing this file, please use the migrations feature of Active Record ↩
    to
```

```
# incrementally modify your database, and then regenerate this schema ↩
    definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more ↩
    migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended to check this file into your version control ↩
    system.

ActiveRecord::Schema.define(:version => 20150310092848) do

  create_table "procedure_procedures", :force => true do |t|
    t.integer  "prepro_id"
    t.integer  "sucpro_id"
    t.datetime "created_at", :null => false
    t.datetime "updated_at", :null => false
  end

  create_table "procedure_resources", :force => true do |t|
    t.integer  "resource_id"
    t.integer  "procedure_id"
    t.integer  "capa_demand"
    t.datetime "created_at",   :null => false
    t.datetime "updated_at",   :null => false
  end

  create_table "procedures", :force => true do |t|
    t.string   "name"
    t.datetime "created_at",                :null => false
    t.datetime "updated_at",                :null => false
    t.integer  "prot"
    t.integer  "fa",          :default => 0
    t.integer  "sa",          :default => 0
    t.integer  "fe",          :default => 0
    t.integer  "se",          :default => 0
    t.integer  "optp",        :default => 0
  end

  create_table "projects", :force => true do |t|
    t.string   "gvp"
    t.string   "path"
    t.date     "startdate"
    t.date     "deadline"
```

```
    t.datetime "created_at", :null => false
    t.datetime "updated_at", :null => false
    t.integer  "zwt"
    t.integer  "zwc"
    t.integer  "totalc"
    t.integer  "extrac"
  end

  create_table "resources", :force => true do |t|
    t.string   "name"
    t.datetime "created_at",                :null => false
    t.integer  "oce",        :default => 0
    t.integer  "cost"
    t.integer  "ocr"
    t.datetime "updated_at",                :null => false
  end

  create_table "users", :force => true do |t|
    t.string   "name"
    t.string   "email"
    t.integer  "resource_id"
    t.integer  "capacity"
    t.datetime "created_at",                        :null => false
    t.datetime "updated_at",                        :null => false
    t.string   "password_digest"
    t.string   "remember_token"
    t.boolean  "admin",           :default => false
  end

  add_index "users", ["email"], :name => "index_users_on_email", :unique => ↩
      true
  add_index "users", ["remember_token"], :name => "↩
      index_users_on_remember_token"

end
```

Quellcode 56 Beispieldaten für die Datenbank

```
#encoding: UTF-8

namespace :db do
  desc "Fill database with sample data"
  task populate: :environment do
    admin = User.create!(name: "Example User",
                         email: "example@railstutorial.org",
                         password: "foobar",
                         password_confirmation: "foobar",
                         capacity: 2,
```

```
                        resource_id: 1)
    admin.toggle!(:admin)

    User.create!(name: "Susi Sorglos",
                 email: "susi@sorglos.de",
                 password: "foobar",
                 password_confirmation: "foobar",
                 capacity: 2,
                 resource_id: 1)

    (3..5).each do |n|
#       name = Faker::Name.name
      name = "Nutzer-#{n}"
      email = "example-#{n}@railstutorial.org"
      password = "password"
      capacity = 2
      resource_id = 1
      User.create!(name: name,
                   email: email,
                   password: password,
                   password_confirmation: password,
                   capacity: capacity,
                   resource_id: resource_id)
    end

    (6..10).each do |n|
#       name = Faker::Name.name
      name = "Nutzer-#{n}"
      email = "example-#{n}@railstutorial.org"
      password = "password"
      capacity = 2
      resource_id = 2
      User.create!(name: name,
                   email: email,
                   password: password,
                   password_confirmation: password,
                   capacity: capacity,
                   resource_id: resource_id)
    end

    (1..12).each do |n|
      name = "Beispielvorgang#{n}"
      prot = rand(2..5)
      Procedure.create!(name: name,
                   prot: prot)
    end
```

```
(1..2).each do |n|
  name = "Ressource#{n}"
  ocr = rand(5..10)
  cost = rand(1..3)
  Resource.create!(name: name,
                   ocr: ocr,
                   cost: cost)
end

ProPro1 = ProcedureProcedure.create!(prepro_id: 1, sucpro_id: 2)
ProPro2 = ProcedureProcedure.create!(prepro_id: 1, sucpro_id: 5)
ProPro3 = ProcedureProcedure.create!(prepro_id: 1, sucpro_id: 8)
ProPro4 = ProcedureProcedure.create!(prepro_id:  1, sucpro_id: 10)
ProPro5 = ProcedureProcedure.create!(prepro_id: 2, sucpro_id: 3)
ProPro6 = ProcedureProcedure.create!(prepro_id: 2, sucpro_id: 6)
ProPro7 = ProcedureProcedure.create!(prepro_id: 3, sucpro_id: 4)
ProPro8 = ProcedureProcedure.create!(prepro_id: 4, sucpro_id: 7)
ProPro9 = ProcedureProcedure.create!(prepro_id: 4, sucpro_id: 12)
ProPro10 = ProcedureProcedure.create!(prepro_id: 5, sucpro_id: 6)
ProPro11 = ProcedureProcedure.create!(prepro_id: 6, sucpro_id: 7)
ProPro12 = ProcedureProcedure.create!(prepro_id: 7, sucpro_id: 12)
ProPro13 = ProcedureProcedure.create!(prepro_id: 8, sucpro_id: 9)
ProPro14 = ProcedureProcedure.create!(prepro_id: 9, sucpro_id: 12)
ProPro15 = ProcedureProcedure.create!(prepro_id: 10, sucpro_id: 11)
ProPro16 = ProcedureProcedure.create!(prepro_id: 11, sucpro_id: 12)

ProRes1 = ProcedureResource.create!(procedure_id: 1, resource_id: 1, ↩
    capa_demand: 8)
ProRes2  = ProcedureResource.create!(procedure_id: 2, resource_id: 1, ↩
    capa_demand: 8)
ProRes3 = ProcedureResource.create!(procedure_id: 3, resource_id: 2, ↩
    capa_demand: 10)
ProRes4 = ProcedureResource.create!(procedure_id: 4, resource_id: 1, ↩
    capa_demand: 8)
ProRes5 = ProcedureResource.create!(procedure_id: 5, resource_id: 2, ↩
    capa_demand: 8)
ProRes6 = ProcedureResource.create!(procedure_id: 6, resource_id: 1, ↩
    capa_demand: 9)
ProRes7 = ProcedureResource.create!(procedure_id: 7, resource_id: 1, ↩
    capa_demand: 8)
ProRes8 = ProcedureResource.create!(procedure_id: 8, resource_id: 2, ↩
    capa_demand: 10)
ProRes9 = ProcedureResource.create!(procedure_id: 9, resource_id: 1, ↩
    capa_demand: 8)
ProRes10 = ProcedureResource.create!(procedure_id: 10, resource_id: 1, ↩
    capa_demand: 9)
ProRes11 = ProcedureResource.create!(procedure_id: 11, resource_id: 1, ↩
```

```
        capa_demand: 8)
    ProRes12 = ProcedureResource.create!(procedure_id: 12, resource_id: 1, ↩
        capa_demand: 8)

    Proj1 = Project.create!(gvp: "C:\\Program Files (x86)\\Graphviz2.38\\bin↩
        ", path: "C:\\GAMS\\win64\\24.3\\gams", startdate: Date.today, ↩
        deadline: Date.today + 19.days)

    #(1..10).each do |n|
      #procedure_id = n
      #resource_id = rand(1..5)
      #ProcedureResource.create!(resource_id: resource_id, procedure_id: ↩
          procedure_id)
    #end

  end
end
```