# Dokumentation IPA Wochenplaner

IPA in Applikationsentwicklung

Auszubildender - Marius Küng

Auftraggeber - allink GmbH

Projektleiter - Silvan Spross

Experte - Antonio Di Luzio

Chefexperte - Hans Riesenmann

Durchführungsort - allink GmbH

Informatikmittelschule Basel

13.03. - 26.03.2012

# Inhaltsverzeichnis

# Teil I.

# Umfeld und Ablauf

# 1. Aufgabenstellung

## 1.1. Titel der Facharbeit

Webapplikation zur Ressourcenplanung von allink.creative

## 1.2. Thematik

Es soll eine Webapplikation mit Django erstellt werden, mit welcher die Geschäftsleitung die Ressourcenplanung der Mitarbeiter vornehmen kann. Damit soll eine ältere Webapplikation abgelöst werden.

## 1.3. Klassierung

- Applikationsentwicklung OO
- UNIX / Linux
- andere Programmiersprache

## 1.4. Durchführungsblock

Startblock 1: 12.03.2012 - 23.04.2012
IPA-Durchführung: 12.03.2012 - 26.03.2012
Einreichung bis: Montag, 30.01.2012

## 1.5. Ausgangslage

Bei allink besteht seit Mitte 2010 ein rudimentäres Ressourcenplanungstool. Zur Entwicklung wurden damals jedoch Technologien verwendet, die heute nicht mehr zur Kernkompetenz von allink zählen. Da dieses Tool jedoch jeden Freitag zur Planung der nächsten Woche verwendet wird, ist es seit längerem überfällig es in die bestehende Managementapplikation zu integrieren.

Dank der übermässig langen Testphase des Prototypen sind nun die Anforderungen an das definitive Tool gut bekannt. Daher soll eine Webapplikation mit Django erstellt werden, mit welcher die Geschäftsleitung von allink.creative die Ressourcenplanung der Mitarbeiter vornehmen kann. Damit soll eine auch die Webapplikation abgelöst werden.

## 1.6. Detaillierte Aufgabenstellung

Das bestehende Tool namens “Yatplaner” soll als Modul im bestehenden Management Tool namens “allink.planer” reimplementiert werden. Dabei soll die Bedienbarkeit verbessert werden. Das Ziel ist es das neue Tool so intuitiv bedienen zu können, dass für die Geschäftsleitung keine Schulung nötig ist. Der Praxistest wird voraussichtlich in der letzten IPA Woche an der Wochenplansitzung durchgeführt.

Nebst der begleitenden IPA Dokumentation, wo unter anderem der Funktionsumfang des bestehenden Tools analysiert wird, wird keine zusätzliche Dokumentation gefordert. Der Funktionsumfang des bestehenden Tools soll vom Lernenden in einer Analysephase aufgenommen werden. Dabei sollen die bestehenden Features als Muss- und mögliche neue Features als Kann-Ziele ausformuliert werden.

Der Quellcode des bestehenden Tools ist unter folgender Adresse einsehbar:
https://github.com/sspross/yatplaner/tree/rails

## 1.7. Mittel und Methoden

Folgende Technologien sind zwingend zu verwenden:

- Python 2.6
- Django 1.3
- Piston 2.3
- jQuery 1.8

- HTML5
- CSS3

Das Tool soll in folgenden Browsern fehlerfrei funktionieren:

- Firefox >= 8
- Chrome >= 10
- Safari >= 5

Der Internet Explorer muss explizit nicht unterstützt werden.

## 1.8. Vorkenntnisse

Dem Lernenden sind alle genannten Technologien bereits bekannt. Seit Beginn seines Praktikums im August 2011 setzt er sich damit auseinander. Gewisse Kombinationen wie z.B. mit jQuery einen AJAX Request zu erstellen, sind jedoch Neuland.

## 1.9. Vorarbeiten

Es findet keine explizite Vorarbeit statt.

## 1.10. Neue Lerninhalte

Wie bereits erwähnt sind dem Lernenden alle Technologien bereits bekannt. Jedoch sind gewisse Kombinationen noch nie vom Lernenden selbst angewandt worden. Das Know-how ist bei allink ausreichend vorhanden. Der Lernende kann zudem auf eine Vielzahl von bestehenden Projekten zurückgreifen, wo er unzählige Beispiele studieren kann.

## 1.11. Arbeiten in den letzten 6 Monaten

Der Lernende hat überwiegend Webseiten mit den oben genannten Technologien umgesetzt. Zu seinen umfassendsten Arbeiten zählen bis jetzt eine Webseite eines Immobilienunternehmens und einer Eventplattform eines Finanzkonzerns. Für ersteres arbeitete der Lernende rund 250 Stunden daran.

# 2. Einleitung

## 2.1. Projekorganisation

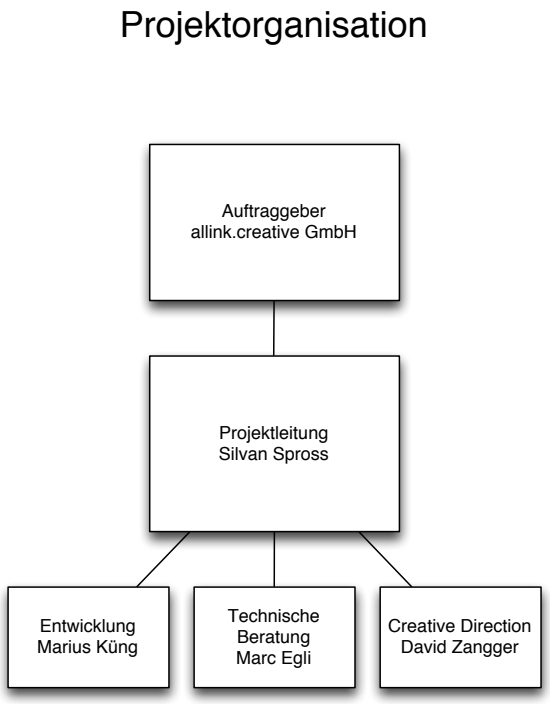


Abbildung 2.1.: Projekorganisation[1]

## 2.2. Vorkenntnisse

Technologien

- Python Grundkenntnisse
- Django fortgeschrittene Kenntnisse

[1] Eigene Darstellung

- Piston Grundkenntnisse
- jQuery fortgeschrittene Kenntnisse
- HTML5 gute Kenntnisse
- CSS3 gute Kenntnisse
- AJAX fortgeschrittene Kenntnisse

Anwendungen

- Mehrere komplette Webauftritte realisiert
- Applikationen in Django erstellt (News, Blog, Produkteübersicht)
- Schnittstellen programmiert (XML, JSON)
- Per AJAX dynamische Inhalte laden und einfügen
- Dynamische Formulare abschicken und Request entgegennehmen
- DOM-Elemente manipulieren, per Events steuern

## 2.3. Vorarbeiten

Ich habe, um einen ersten Eindruck über die Funktionalität zu erhalten, den bestehenden yatplaner ausprobiert. Ansonsten fand keine explizite Vorarbeit statt.

## 2.4. Firmenstandards

- Betriebsystem: Mac OS X
- Editor: Textmate
- Entwicklungsumgebung: Python, Django (Python-Framework für Webapplikationen)
- Lokale Testumgebung: Django Server
- Deployment: per fabric-script auf Apache-Server
- Versionierung: Git, auf Github veröffentlicht

## 2.5. Zeitplan

Ich plante in einzelnen Schritten oder fasste gewisse Aufgaben zusammen und schätzte wie lange ich an einem Feature brauchen würde. Dazwischen habe ich mir Meilensteine gesetzt mit einem Datum, dass ich einzuhalten plante. In der Planung habe ich bei kleinen Aufgaben meist zu viel geplant und dann in der Realität diese Überzeit aufgestockt. Damit wurde mehr Zeit für die Realisierung frei.

| Beschreibung | Planung | Realität |
|---|---|---|
| Teil 1 | | |
| Zeitplan erstellen | 2 | 2 |
| Dokumentation gliedern | 1 | 1 |
| Aufgabenstellung erfassen | 1 | 0.25 |
| Projektorganisation erfassen | 1 | 0.25 |
| Vorkenntnisse erfassen | 1 | 1 |
| Vorarbeiten erfassen | 1 | 0.25 |
| Deklaration des Firmenstandards erfassen | 1 | 0.5 |
| MS: Teil 1 abschliessen | 13.03. | 13.03. |

| Beschreibung | Planung | Realität |
|---|---|---|
| Teil 2 | | |
| Projektbeschreibung erfassen | 1 | 1 |
| Systembeschreibung (Django, planer) | 1 | 0.5 |
| Analyse IST-Zustand | 1 | 1.5 |
| Definition Muss-/Kann-Ziele | 2 | 2 |
| MS: Ziele definiert | 14.03. | 14.03. |
| MS: Planung abgeschlossen | 14.03. | 14.03. |
| ERM erstellen | 2 | 1 |
| rails yatplaner studieren | 1 | 2 |
| Projekt aufsetzen | 1 | 0.5 |
| Modellierung Django | 1.5 | 1 |
| MS: Modellierung | 14.03. | 14.03. |
| View erstellen | 4 | 3 |
| Tabelle Template (nur HTML) (Wochenansicht) | 3 | 2 |
| Daten in Tabelle einsetzen | 3 | 2 |
| Wochenansicht ausarbeiten | 7 | 7 |
| JS Funktionalität einbinden (Ajax) (MUSS) | 8 | 8 |
| MS: POST per AJAX schicken | 16.03. | 16.03. |
| Bugfixing | 5 | 6 |
| Realisierung beschreiben | 3 | 6 |
| MS: MUSS-Ziele erreicht | 21.03. | 20.03. |
| MS: Prototyp | 21.03. | 20.03. |

| Beschreibung | Planung | Realität |
|---|---|---|
| Kann-Ziele | 16 | 16 |
| MS: KANN-Ziele erreicht | 26.03. | 23.03. |
| Testfälle erstellen | 1 | 1 |
| Testing im allink.planer | 1 | 0.5 |
| Testfälle dokumentieren | 1 | 0.5 |
| Bugfixing | 3 | 2 |
| Codesäuberungen | 0.5 | 0.5 |
| Implementierung in allink.planer (master) | 0.5 | 0.25 |
| MS: Realisierung | 26.03. | 23.03. |
| Erreichte Ziele erfassen (Muss-Ziele, Kann-Ziele) | 1 | 1 |
| Websummary online erfassen | 1 | 1.5 |
| Quellenangabe | 1 | 1 |
| Glossar | 0.5 | 2 |
| Dokumentation abschliessen | 1 | 4 |
| Arbeit einreichen | 2 | 1 |
| | | |
| Total | 80 | 80 |

Tabelle 2.1.: Zeitplan

# Teil II.

# Projekt

# 3. Projektbeschreibung

## 3.1. Umfeld

Momentan existiert der Wochenplaner als eigenständige Applikation. Das Tool fungiert mehr als eine Tafel auf die Post-It's geklebt werden. D. h. die Aufgaben sind nicht direkt im Projektmanagementtool mit der Person verbunden. Somit bietet das Tool keinen Mehrwert bei der Auswertung des Projektablaufs.

Das Tool ist in Ruby on Rails programmiert, was nicht dem Firmenstandard entspricht, und kann deshalb nur mühsam gewartet und erweitert werden.

## 3.2. Präzisierung der Aufgabenstellung

Die Aufgabenstellung sieht vor, dass mindestens die bestehende Funktionalität vorhanden und die intuitive Bedienung verbessert wird. Zur weiteren Präzisierung werde ich eine detaillierte IST-Analyse vornehmen. Diese wird mit dem Auftraggeber besprochen, damit die MUSS- und KANN-Ziele erfasst werden können und ersichtlich wird, welche Funktionalität überhaupt noch Sinn machen und was weggelassen werden kann.

## 3.3. Analyse Wochenplaner (IST-Zustand)

### 3.3.1. Modellierung

Analysiert anhand der Rails-Modelle

Person

- Name
- Locked Days (gesperrte Tage, Auswahl von Montag bis Freitag)
- Beschreibungstext für jeden gesperrten Tag

Task

- Name
- Datum (Fälligkeitsdatum, DATE)
- Person (Fremdschlüssel)
- Dauer (Stundenanzahl)

### 3.3.2. Darstellung

Der Wochenplaner verfügt über mehrere Darstellungen:

- Kalenderansicht der aktuellen Woche (siehe Design-Screenshot)
- Übersicht aller Angestellten
- Monatsansicht pro Person
- Übersicht aller erfassten Tasks

### 3.3.3. Funktionsumfang

- Neue Task durch Doppelklick auf Personentag per Dialog erfassen
- Klick auf Task öffnet die Detailansicht (um bspw. Änderungen vorzunehmen oder die Task zu löschen)
- Verschieben eines Tasks per Drag & Drop
  - Task in anderen Tag verschieben
  - Task einer anderen Person zuweisen
- In der Kalenderwoche per Link vor- oder zurückspringen
- weitere 7 Tage in der Ansicht anzeigen (funktioniert auch umgekehrt)
- Wochentage können für wiederkehrende Events (Schule, Frei) markiert werden

### 3.3.4. Defizite

- Die Sperrtage haben kein Startdatum und sind für jeden Wochentag (Vergangenheit & Zukunft) erfasst
- Die Ajax-Funktionalität beschränkt sich auf Task erstellen und verschieben
- Die Bearbeitung der Tasks ist mühsam

## 3.4. Definition Ziele

In einer Sitzung mit der Projektleitung wurde der IST-Zustand besprochen und die Ziele definiert.

### 3.4.1. MUSS-Ziele

| Nr | Funktion | Beschreibung |
|---|---|---|
| 1 | Task erfassen | Der Benutzer kann eine neue Task erfassen. |
| 2 | Task Namen geben | Der Benutzer kann einem Task einen Namen geben. |
| 3 | Task Datum geben | Der Benutzer kann einem Task ein Datum geben. |
| 4 | Task Person zuweisen | Der Benutzer kann einem Task eine Person zuweisen. |
| 5 | Task Dauer geben | Der Benutzer kann einem Task eine Dauer (in Stunden) geben. |
| 6 | Task bearbeiten | Der Benutzer kann einen Task bearbeiten. |
| 7 | Task löschen | Der Benutzer kann einen Task löschen. |
| 8 | Person hinzufügen | Der Benutzer kann eine Person dem Wochenplaner hinzufügen. |
| 9 | Sperrtag Person zuweisen | Der Benutzer kann einer Person einen Sperrtag zuweisen. |
| 10 | Sperrtag Namen geben | Der Benutzer kann einem Sperrtag einen Namen geben. |
| 11 | Sperrtag bearbeiten | Der Benutzer kann einen Sperrtag bearbeiten. |
| 12 | Sperrtag löschen | Der Benutzer kann einen Sperrtag löschen. |
| 13 | Task einem Projekt zuweisen | Der Benutzer kann einen Task einem Projekt zuweisen. |
| 14 | Person als Partner | Eine Person als Partner kennzeichnen. |

Tabelle 3.1.: Zwingend umzusetzende Funktionen des Prototypen

### 3.4.2. KANN-Ziele

| Nr | Funktion | Beschreibung | Priorität |
|---|---|---|---|
| 15 | Task duplizieren | Der Benutzer kann einen Task duplizieren. | 1 |
| 16 | Sperrtag Startdatum geben | Der Benutzer kann einem Sperrtag ein Start- & Endatum geben. | 3 |
| 17 | Warnmeldung Max Stunden | Es erscheint eine Warnmeldung wenn die Summe aller Arbeitsstunden pro Tag 8h überschreiten. | 4 |
| 18 | Tasks sortieren | Tasks können innerhalb eines Tages sortiert werden. | 2 |

Tabelle 3.2.: Nicht zwingend umzusetzende Funktionen des Prototypen

# 4. Realisierung

## 4.1. ERMs

### 4.1.1. allink.planer

Als Ausgangslage dient mir das ERM des allink.planer. Die Entitäten Person und Project werden mit dem Wochenplaner verknüpft.

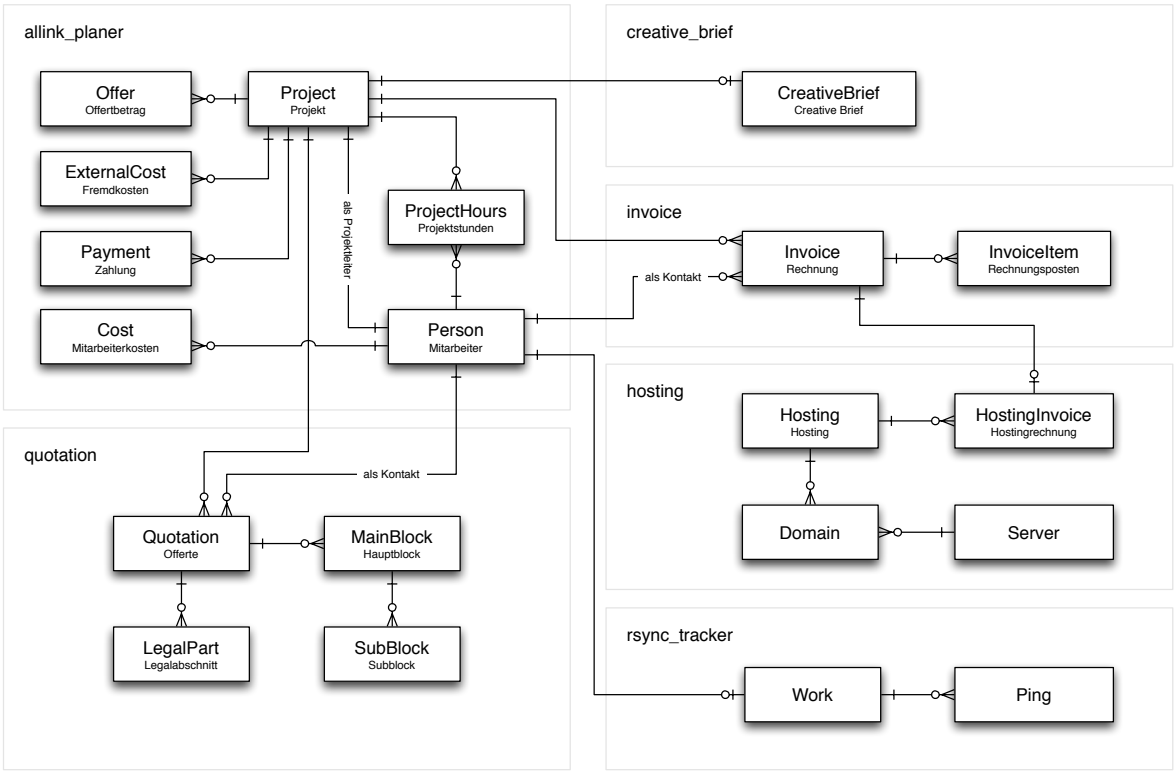


Abbildung 4.1.: ERM allink.planer[1]

[1]Entommen aus allink.planer

### 4.1.2. Wochenplaner

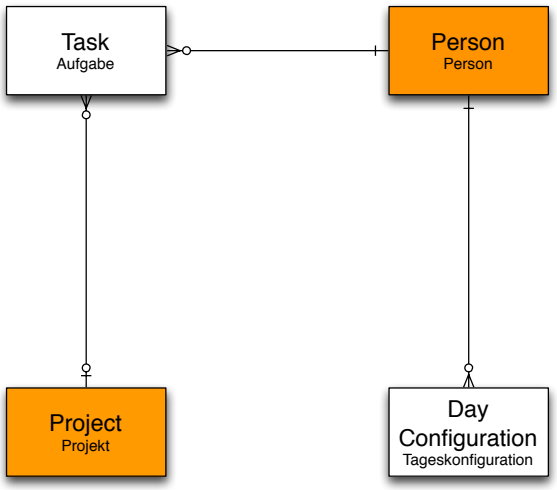


Abbildung 4.2.: ERM Wochenplaner (die Orange hinterlegten Entitäten stammen aus dem allink.planer)[2]

[2]Eigene Darstellung

## 4.2. Design

### 4.2.1. Altes Design

So sieht der momentane Wochenplaner aus:

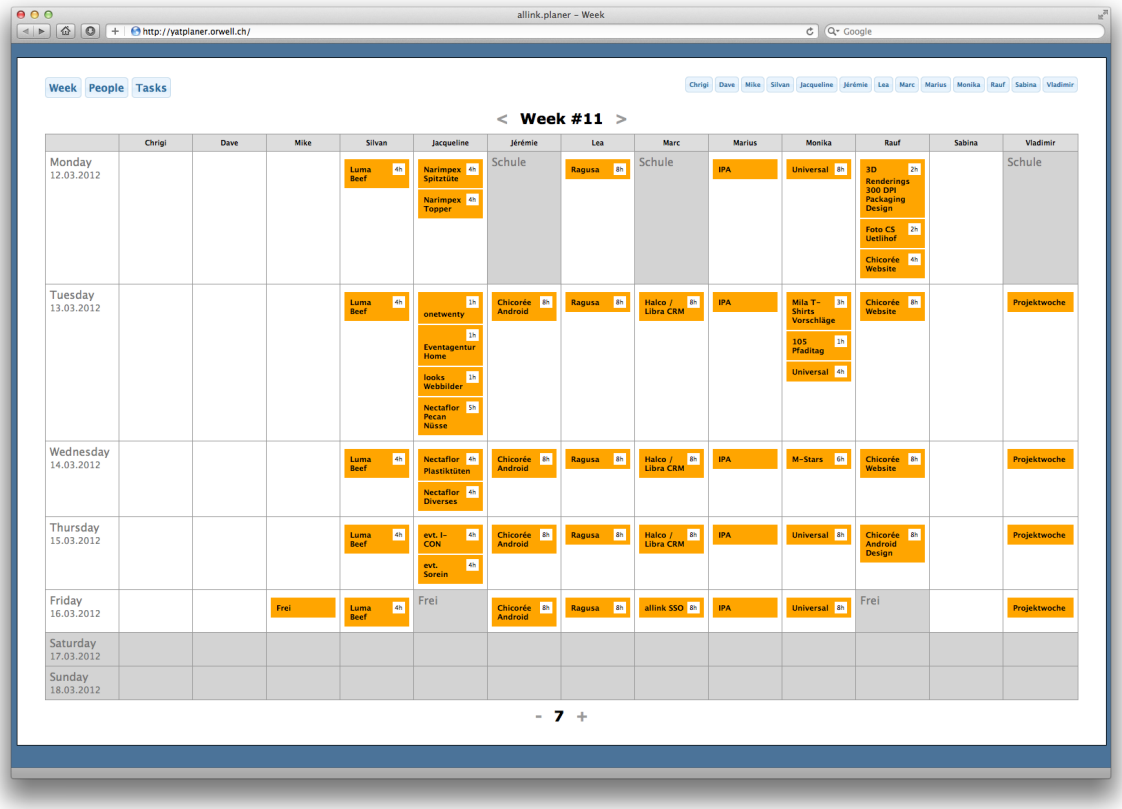


Abbildung 4.3.: Bisheriges Design des yatplaner

### 4.2.2. Neues Design

Mit folgendem Design soll der Wochenplaner umgesetzt und an das allink CI/CO angepasst werden:

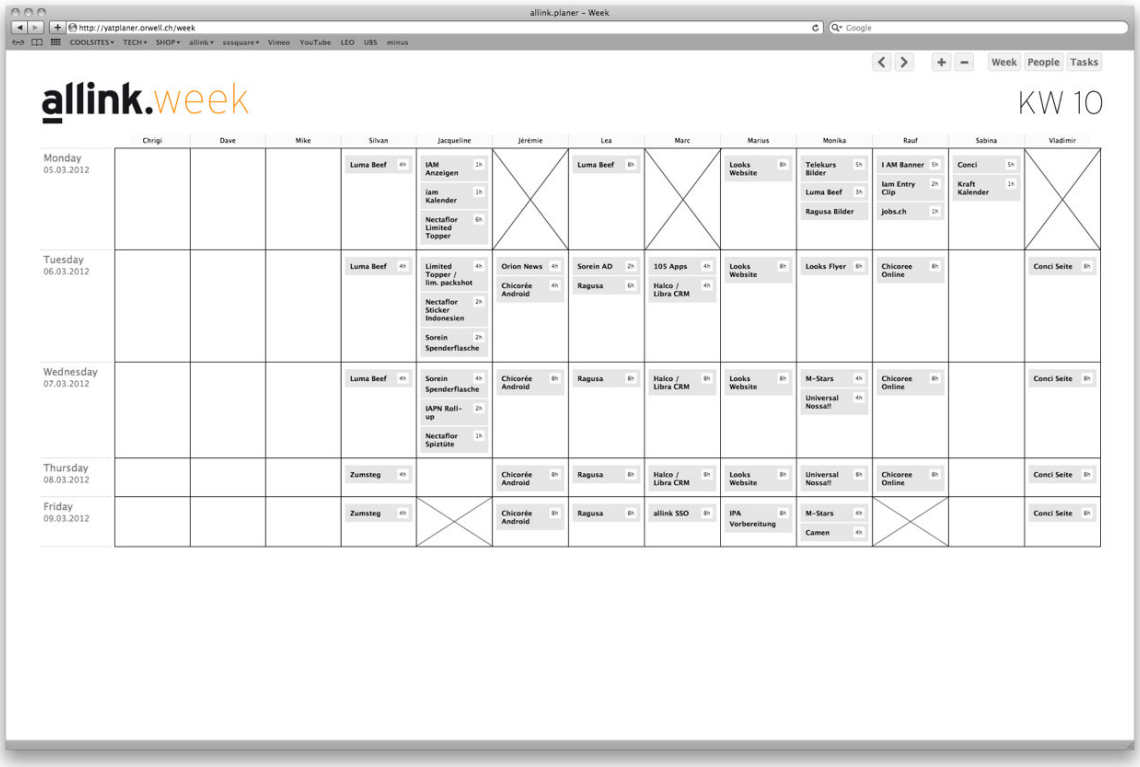


Abbildung 4.4.: Neues Design

## 4.3. Umsetzung

Da ich nicht auf die gesamte Entstehung des Quellcodes eingehen kann und der Verlauf im git-repository einsehbar ist, führe ich die Umsetzung der Ziele auf.

Der Wochenplaner besteht aus 2 Bereichen in denen Daten manipuliert werden können:

- Administrationsbereich (ist ein Standardfeature von Django um Applikationen zu verwalten)
- Wochenansicht (zusätzliche View im bestehenden Adminbereich)

Obwohl eine Intuitive Bedienung gefordert ist, müssen nicht alle Ziele in der Wochenansicht manipulierbar sein.
Die folgende Aufstellung zeigt auf welche Ziele wo umgesetzt wurden.

| Ziel-Nr | Admin | Wochenansicht |
|---|---|---|
| 1 | x | x |
| 2 | x | x |
| 3 | x | x |
| 4 | x | x |
| 5 | x | x |
| 6 | x | x |
| 7 | x | |
| 8 | x | |
| 9 | x | |
| 10 | x | |
| 11 | x | |
| 12 | x | x |
| 13 | x | x |
| 14 | x | |
| 15 | x | |
| 16 | | x |
| 17 | | x |
| 18 | | x |
| 19 | | x |

Tabelle 4.2.: Funktionsbereiche der Ziele

## 4.4. Resultat

Das Resultat meiner IPA ist eine Webapplikation, in der Tasks über eine HTML-Ansicht per Ajax intuitiv verwaltet werden können. Die Applikation ist in das Projektmanagementtool allink.planer integriert. Die Tasks sind mit Personen und Projekten verknüpft. Dadurch können in Zukunft diverse Analysen gemacht werden.

## 4.5. Modellierung

Über das Django-Framework kann man die Modelle der Klassen bzw. Entitäten erfassen. Alle Modelle können als Applikationen über einen Administratorbereich durch CRUD manipuliert werden.

### 4.5.1. Task

- Name (Char)
- Duedate (Date)
- Person (Fremdschlüssel)
- Duration (Stundenanzahl)
- Project (Fremdschlüssel)
- sortindex (int)

### 4.5.2. Tageskonfiguration

- Description (Char)
- Day (Int, ChoiceField)
- Person (Fremdschlüssel)
- startdate (Date)
- enddate (Date)

## 4.6. Layout

Das Layout bleibt, bis auf einige kleine Anpassungen und das neue Design, dasselbe.
Im neuen Layout hat die Woche nur noch 5 anstatt 7 Tage und die Links zum admin-Bereich wurden versetzt.

## 4.7. Entwicklungsumgebung

Die Entwicklungsumgebung besteht aus den in den Firmenstandards genannten Technologien.

## 4.8. Herausforderungen & Entscheidungen

### 4.8.1. JSON

Um dynamisch die Datenbank zu manipulieren, kann eine Methode in Django verwendet werden, über die Requests im JSON-Format an den Server geschickt werden. Über den django-piston Handler können diese Requests entgegengenommen und weitergeleitet werden.
Bis jetzt habe ich lediglich über einen JSON-Handler Daten aus der Datenbank gelesen oder neue Datensätze erstellt.
Einen bestehenden Datensatz zu ändern und diesen dann korrekt im Frontend wieder einzusetzen, ist ein bisschen knifflig, aber im Nachhinein doch sehr logisch zu verstehen. Durch eine Einführung von Silvan Spross konnte ich diesen Prozess verstehen und nachvollziehen.

### 4.8.2. Queries für Wochenansicht

Bisher habe ich alle Objekte die ich in der Wochenansicht benötigt der View übergeben. Im Template habe ich über die Objekte iteriert und die entsprechenden Daten ausgegeben. Dies ist jedoch nicht ‘schön’. Darum habe ich mit Unterstützung von Marc Egli, der mir die Verschachtelung von Tuples zeigte, die Objekte mit den korrekten Queries verschachtelt übergeben und dann im Template aufgelöst.

### 4.8.3. JS Funktionalität (CRUD mit jQueryUI & AJAX)

Die Wochenansicht der Tasks ist eine rein statische Ansicht. Alle Tasks und Sperrtage können über den Django admin per CRUD manipuliert werden. Dies ist aber nur ein Teil der bestehenden Funktionalität. Es wird eine CRUD-Funktionalität im Frontend benötigt. Diese werde ich mit jQueryUI realisieren da ich mit jQuery am meisten Erfahrung habe und Firmenstandard ist. Sich in Alternativen einzuarbeiten benötigt zu viel Zeit und ist nicht mit jQuery kompatibel. Per jQueryUI können alle Tasks frei zu jedem Tag und/oder Person verschoben werden. Per AJAX werden neue Tasks automatisch in die DB geschrieben und danach ins HTML eingefügt.

### 4.8.4. Sperrtage

Zuerst habe ich Sperrtage nur mit einem Tag versehen und später noch mit einem Startdatum. Wenn man aber nachvollziehen will wer wie lange wann gefehlt hat, darf man einen Sperrtag nicht einfach überschreiben. Also sind die Sperrtage auch mit einem Enddatum versehen. In der Abfrage für die Wochenansicht wird für jeden Tag geprüft, ob ein Sperrtag beginnt, endet oder kein Enddatum angegeben ist. Somit müssen keine Sperrtage gelöscht werden und es sind mehrere erfassbar, die sich automatisch ergänzen.

### 4.8.5. Task duplizieren per jQueryUI

Wie man einen Task per AJAX dupliziert ist auf verschiedenen Weisen möglich.
Zuerst habe ich es über ein sogenanntes Cloning versucht in dem eine Kopie des Tasks erstellt wird und dann als neuer Task eingesetzt werden kann. Da ich wusste, dass dieses Feature umfangreich wird, habe ich es in einen separaten “branch” ausgelagert. Jedoch stiess ich dabei auf grosse Probleme in der Umsetzung, da das cloning zusätzlich mit einer gedrückten Tastenkombination erfolgen sollte. Nach einer Besprechung mit der Projektleitung wurde beschlossen, dass den Task über einen Button dupliziert und unten eingesetzt wird.

## 4.9. Semantische Codebehebungen

Am Ende des Testings habe ich zur Erhöhung der Codequalität eine Codesäuberung mit pep8, pyflakes und jshint vorgenommen. Dabei wird falsche Syntax und nicht verwendete Imports hervorgehoben.

# 5. Testing

## 5.1. Testfälle erfassen

Die Testfälle beziehen sich auf die Ausführbarkeit der MUSS- und KANN-Ziele. Einige Ziele sind jeweils im Admin und in der Wochenansicht ausführbar. Ziele die auch in der Wochenansicht vorhanden sind, werde ich nicht im Admin testen, da es dieselben Prozesse sind.

## 5.2. Testmethoden

GUI Unit-Tests zu machen hätten den Rahmen meiner Arbeit gesprengt. Mit Selenium kann man Websiten automatisiert getestet werden. Alle Interaktion die ein User machen kann, könnten so getestet werden. Dies ist aber nur für grössere Webapplikationen sinnvoll.
Deshalb teste ich die User-Interaktionen manuell und prüfe über die Konsole was ich für eine Antwort erhalten habe. Dazu wird natürlich noch ein visuelles Feedback über HTML ausgegeben. Ich werde die definierten Testfälle durchführen und mit Screenshots dokumentieren ob wie sie ausgeführt werden. Auf den Bildern ist ausserdem (sofern vorhanden) der REST-Call aufgelistet, der die Kommunikation mit dem Server darstellt.

| Nr | Ziel-Funktion | Testfrage |
|---|---|---|
| 1 | 1 | Kann der Benutzer eine neue Task erfassen? |
| 2 | 2 | Kann der Benutzer einer Task einen Namen geben? |
| 3 | 3 | Kann der Benutzer einer Task ein Datum geben? |
| 4 | 4 | Kann der Benutzer einer Task eine Person zuweisen? |
| 5 | 5 | Kann der Benutzer einer Task eine Dauer geben? |
| 6 | 6 | Kann der Benutzer eine Task bearbeiten? |
| 7 | 7 | Kann der Benutzer eine Task löschen? |
| 8 | 8 | Kann der Benutzer einer Person einen Sperrtag zuweisen? |
| 9 | 9 | Kann der Benutzer einem Sperrtag einen Namen geben? |
| 10 | 10 | Kann der Benutzer einen Sperrtag bearbeiten? |
| 11 | 11 | Kann der Benutzer einen Sperrtag löschen? |
| 12 | 12 | Kann der Benutzer eine Task einem Projekt zuweisen? |
| 13 | 13 | Kann der Benutzer eine Person zum Wochenplaner hinzufügen? |
| 14 | 14 | Kann der Benutzer eine Person als Partner kennzeichnen? |
| 15 | 15 | Kann der Benutzer eine Task duplizieren? |
| 16 | 16 | Kann der Benutzer einem Sperrtag ein Start- & Enddatum geben? |
| 17 | 17 | Erscheint eine Warnmeldung wenn Taskdauer grösser als 8h |
| 18 | 18 | Kann der Benutzer Tasks in einem Tag sortieren? |

Tabelle 5.1.: Zu erfüllende Funktionen des Prototypen

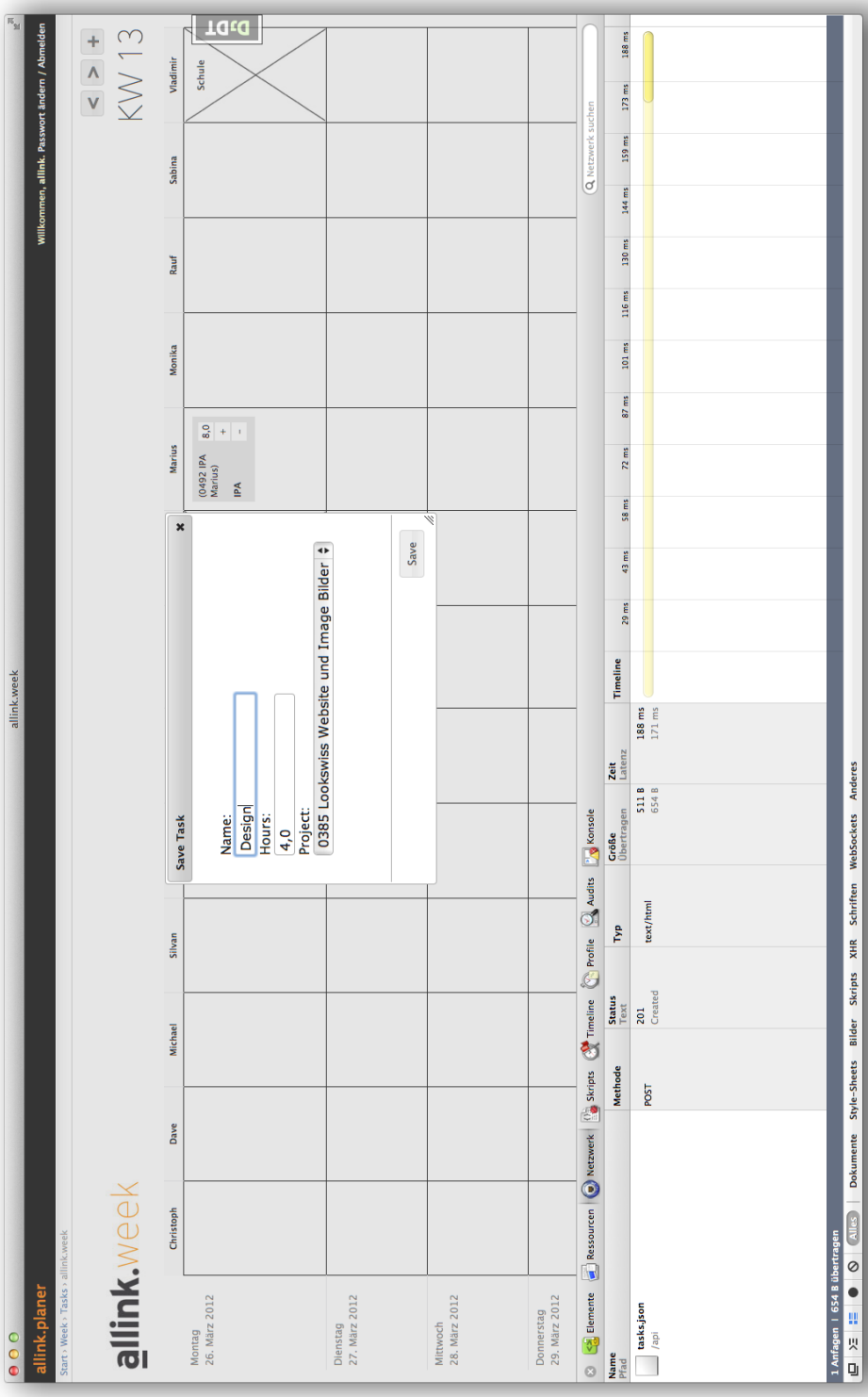


Abbildung 5.1.: Abgedeckte Ziele: Nr. 1, 2, 3, 4, 5, 12

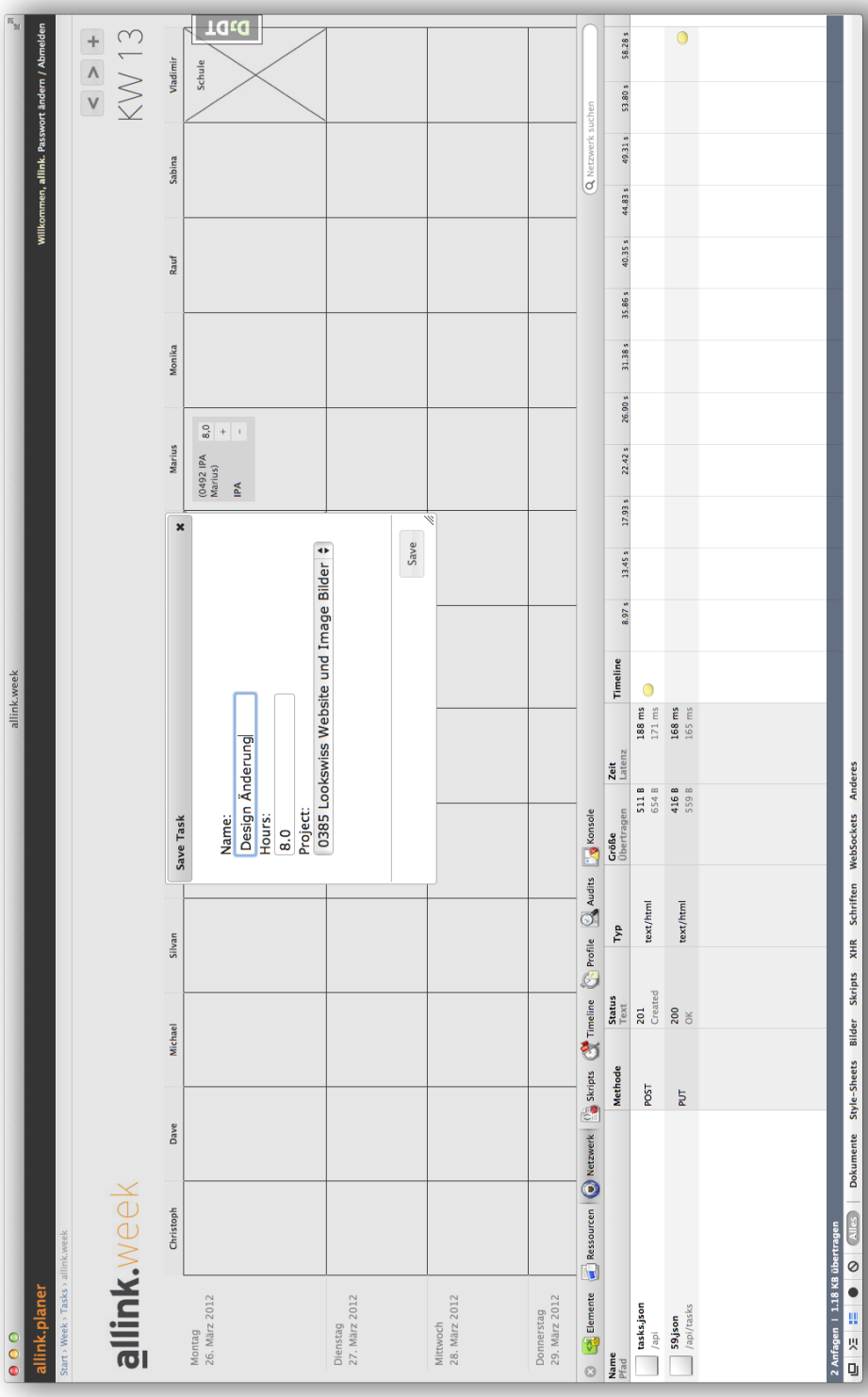


Abbildung 5.2.: Abgedeckte Ziele: Nr. 6

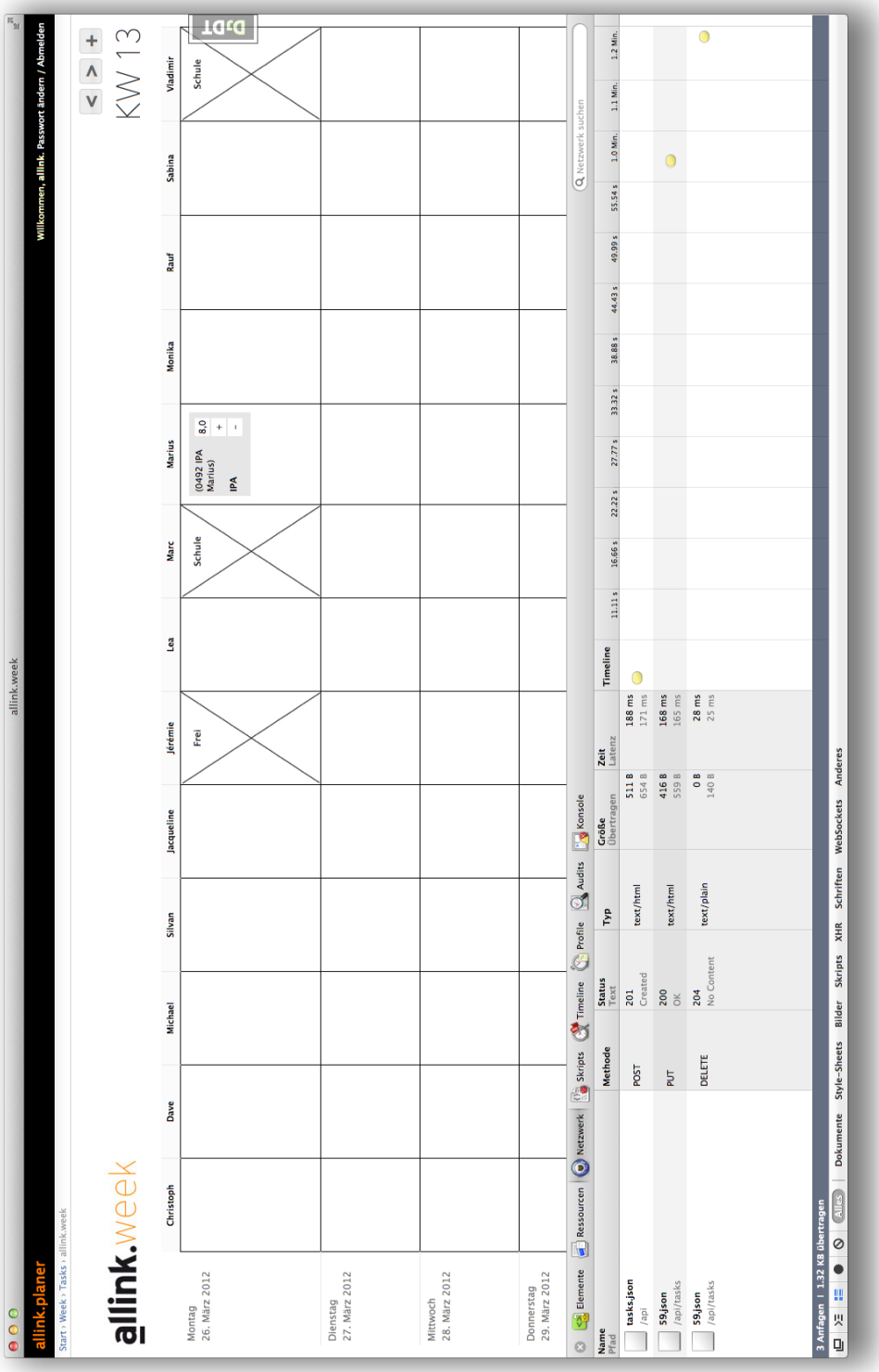


Abbildung 5.3.: Abgedeckte Ziele: Nr. 7

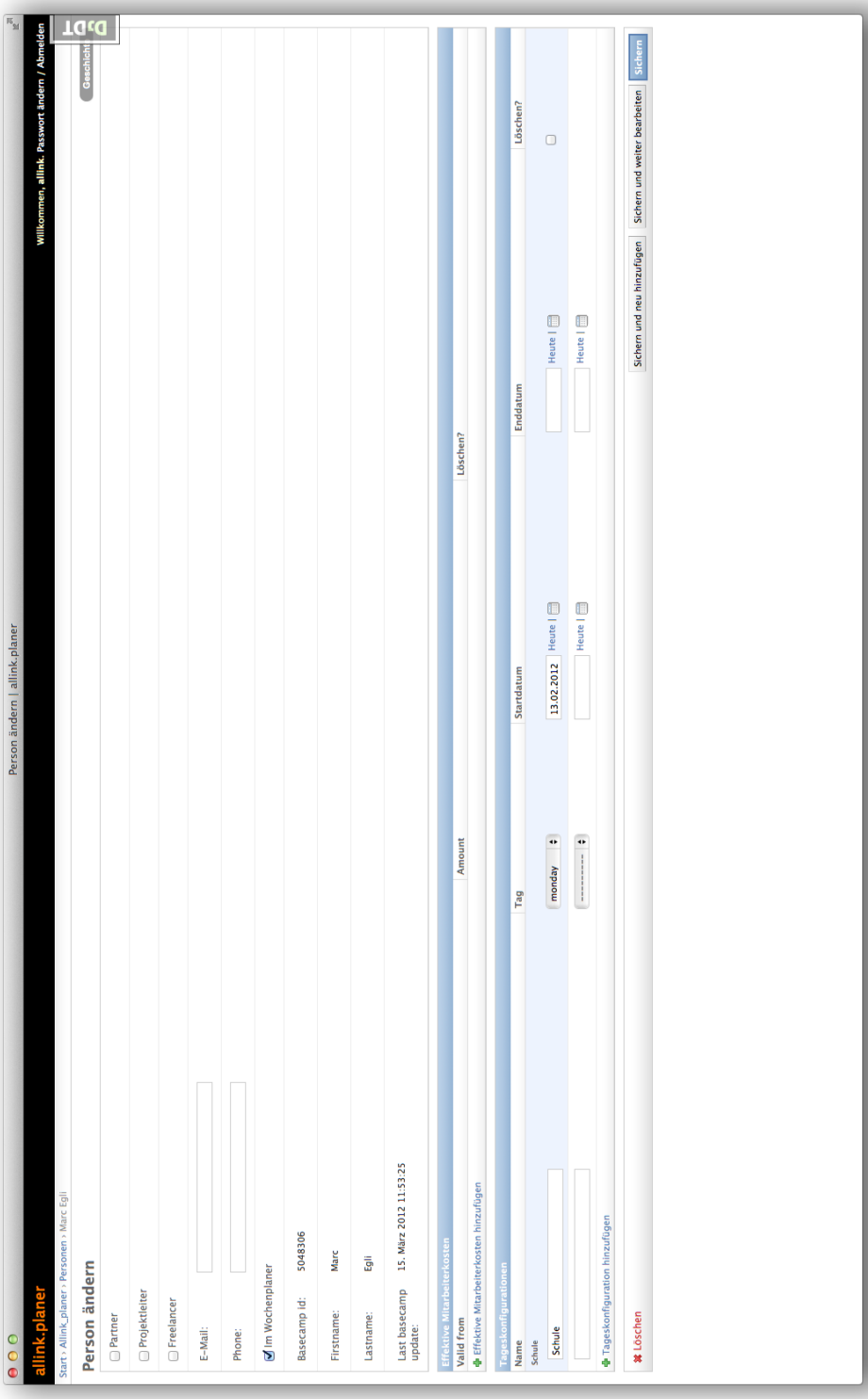


Abbildung 5.4.: Abgedeckte Ziele: Nr. 8, 9

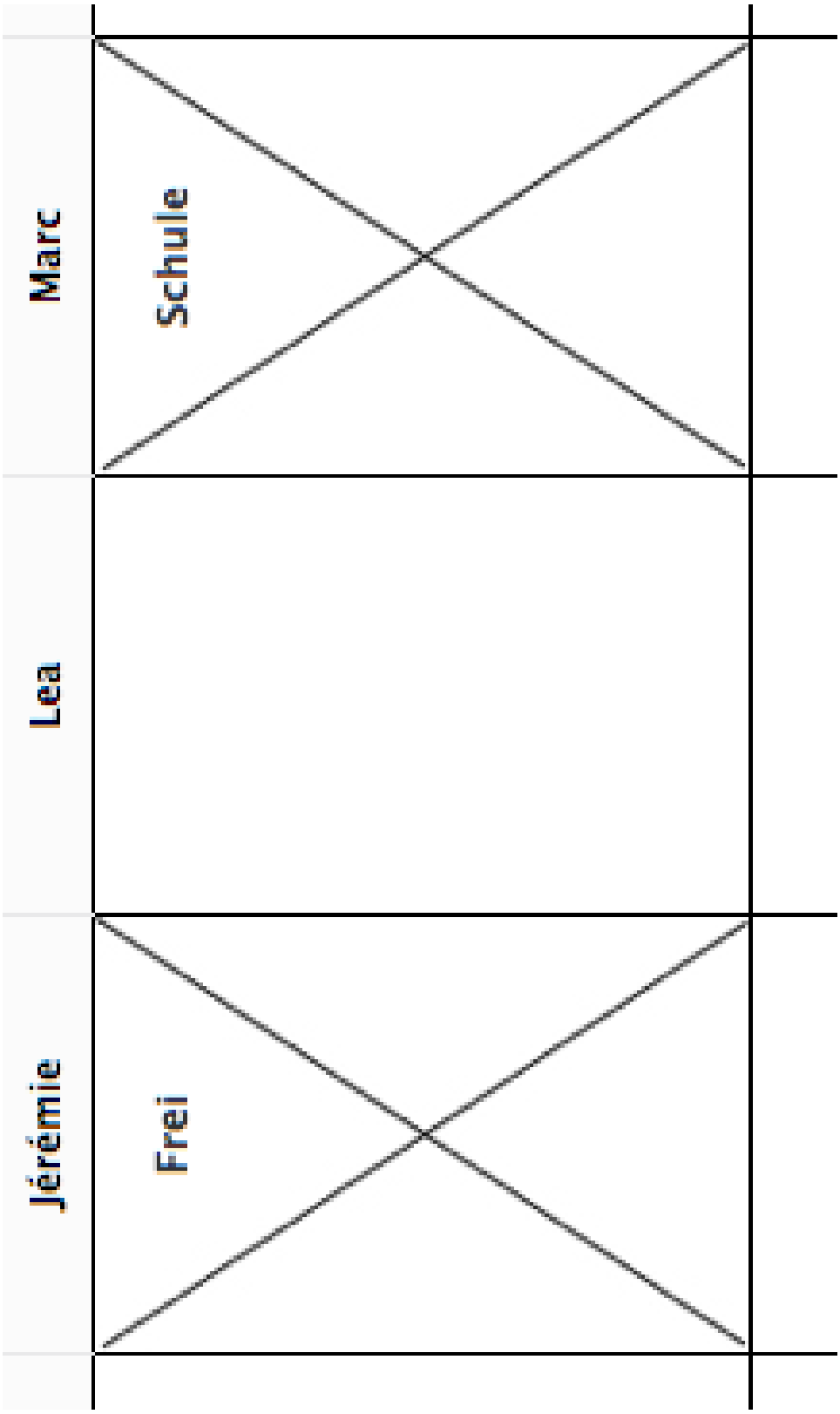


Abbildung 5.5.: Abgedeckte Ziele: Nr. 8, 9

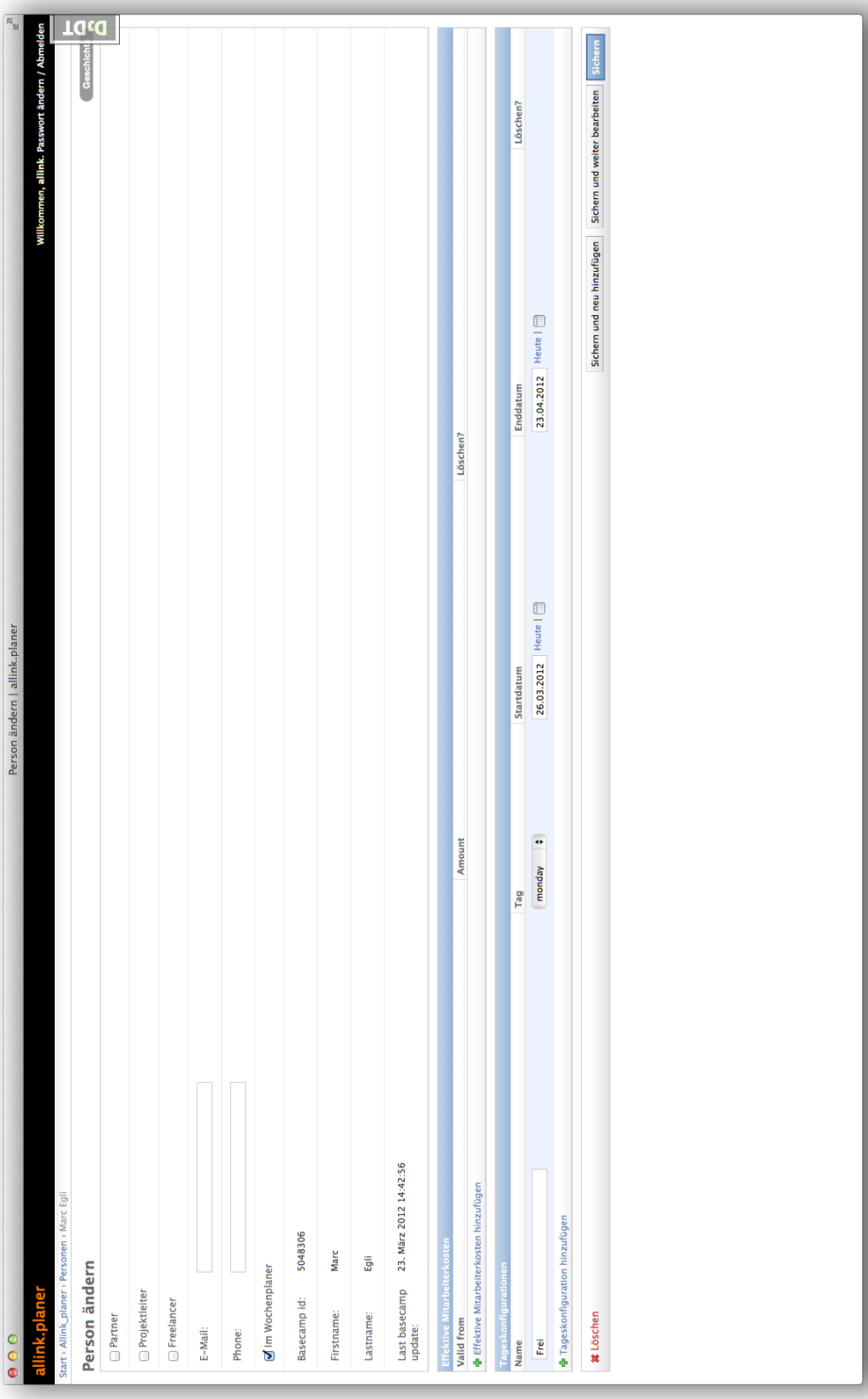


Abbildung 5.6.: Abgedeckte Ziele: Nr. 10

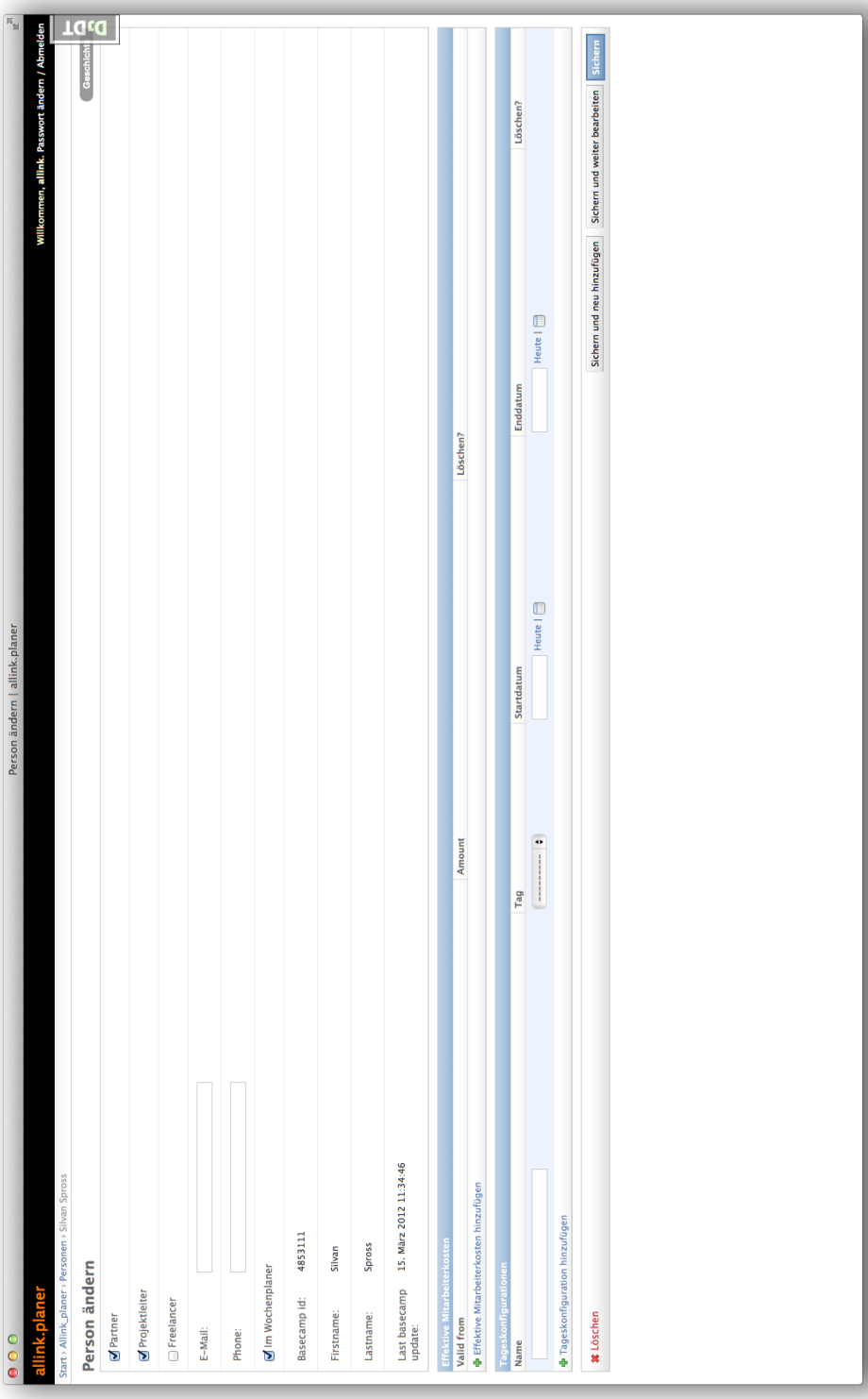


Abbildung 5.7.: Abgedeckte Ziele: Nr. 11

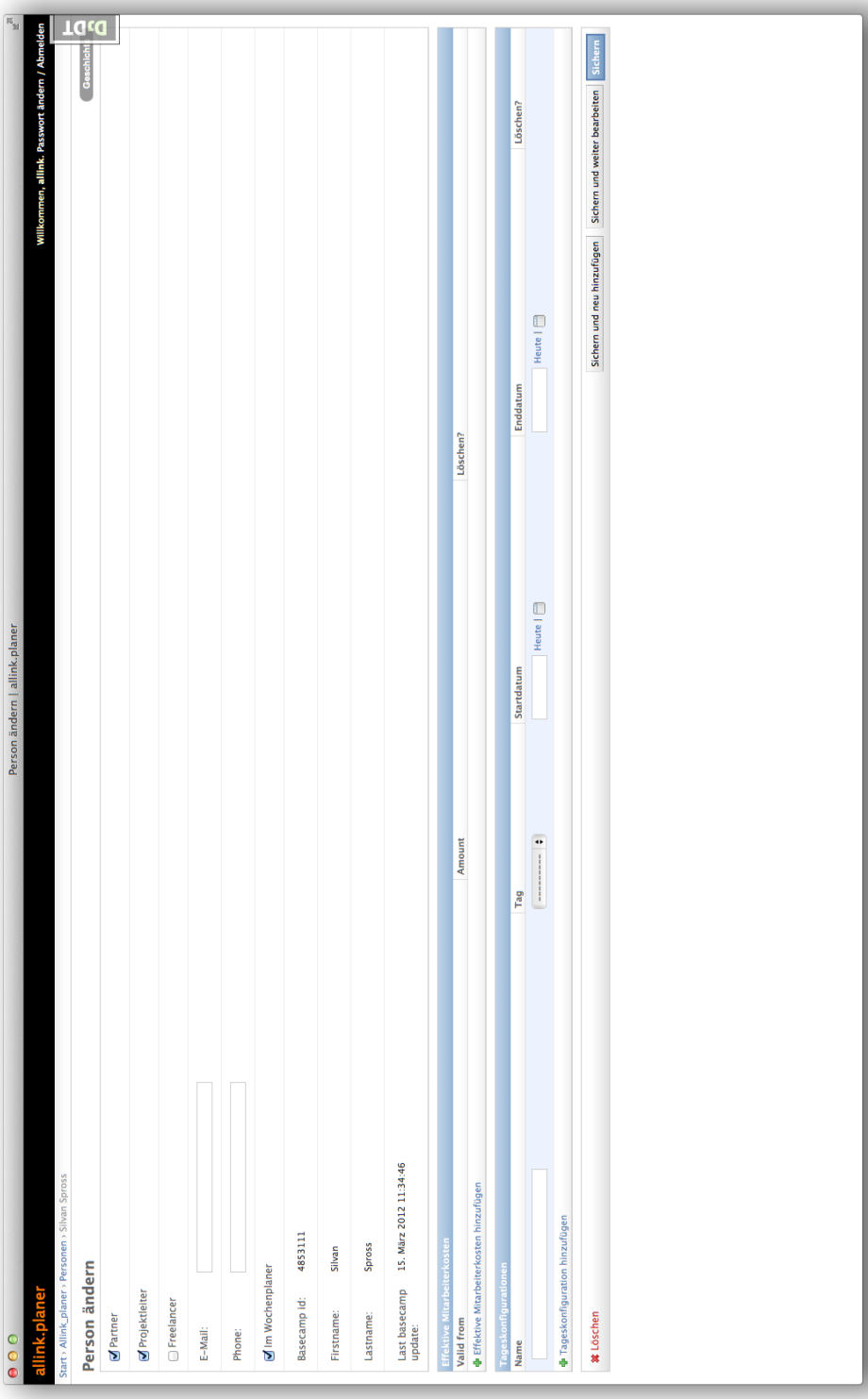


Abbildung 5.8.: Abgedeckte Ziele: Nr. 13, 14

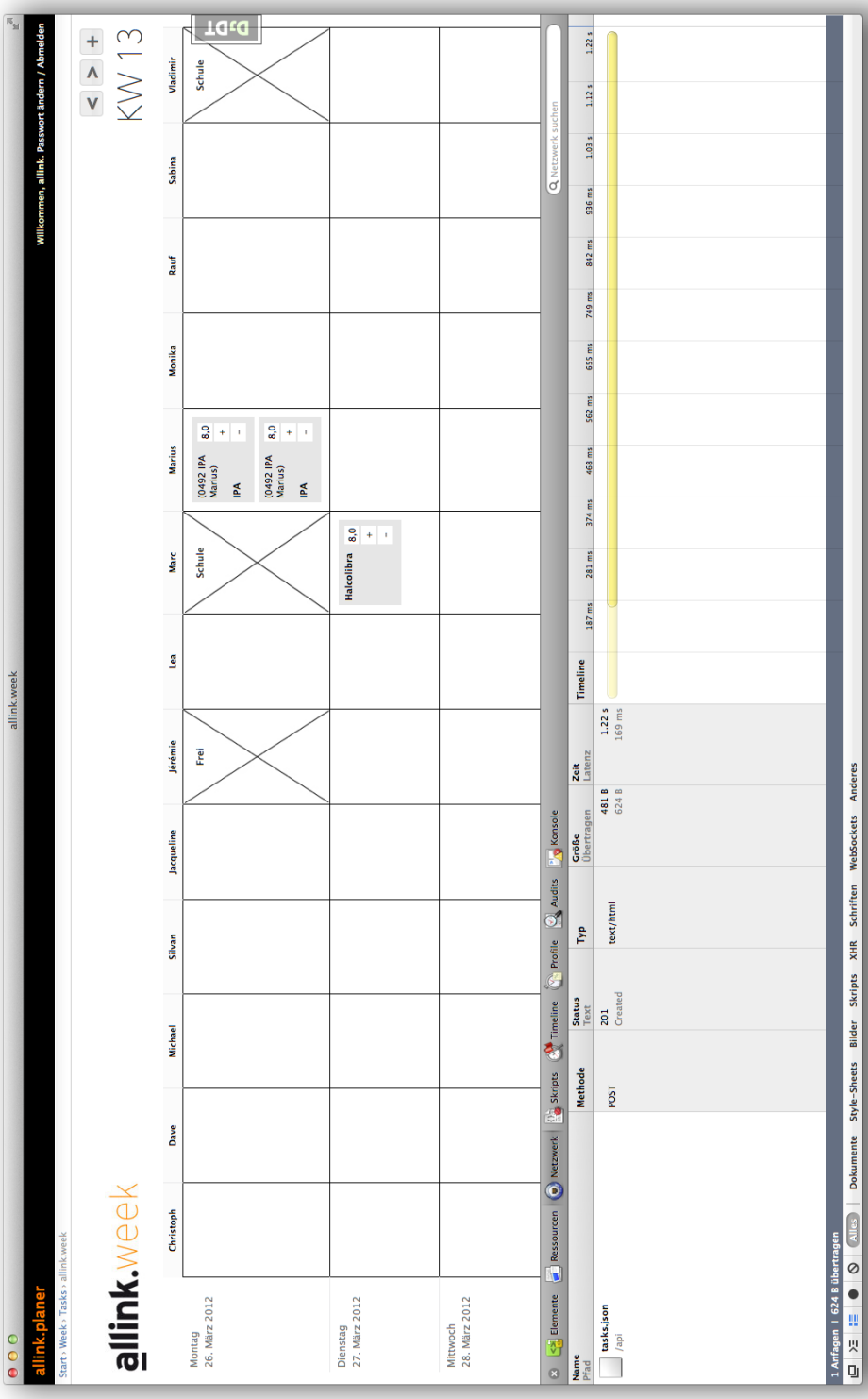


Abbildung 5.9.: Abgedeckte Ziele: Nr. 15

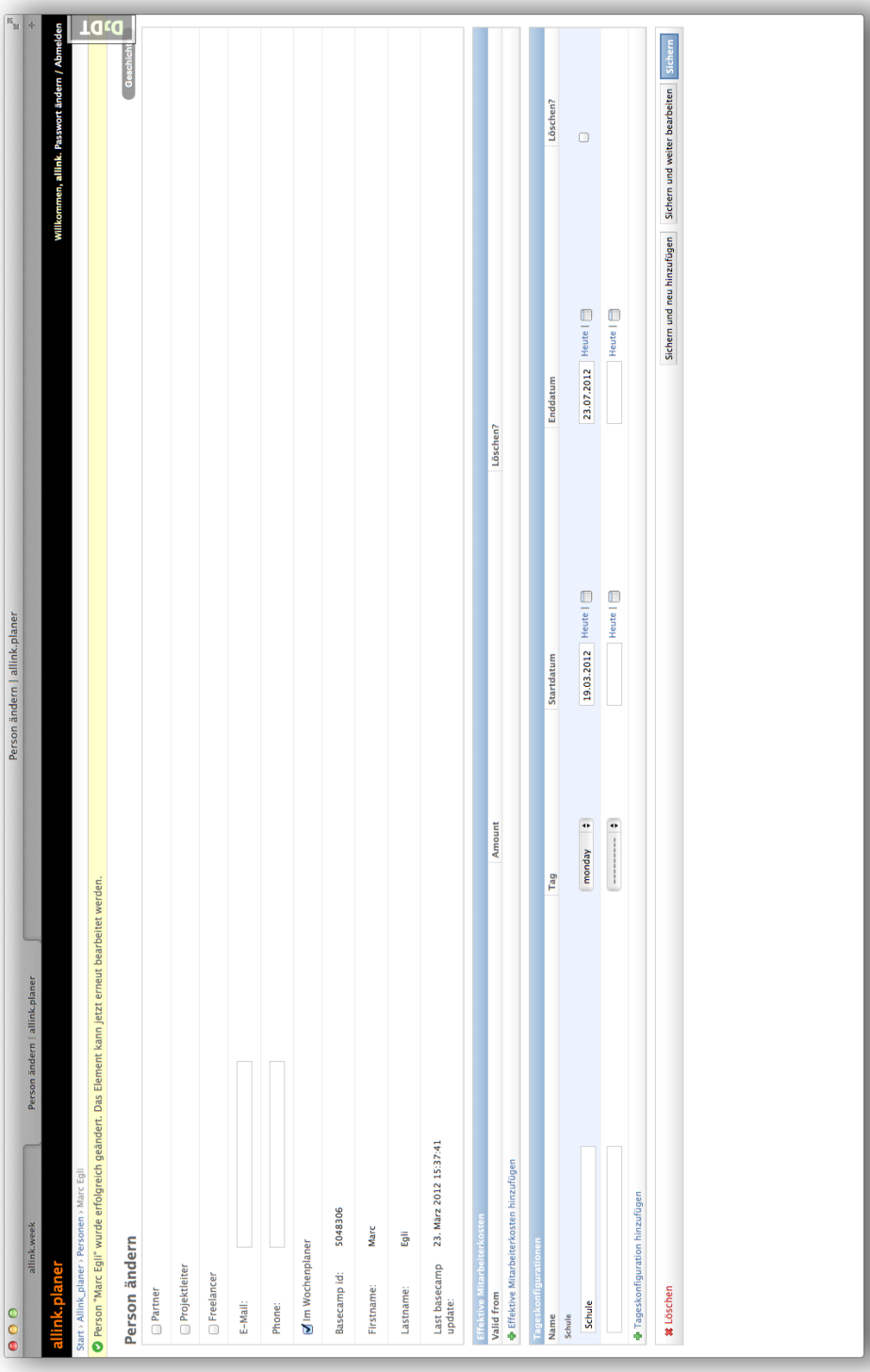


Abbildung 5.10.: Abgedeckte Ziele: Nr. 16

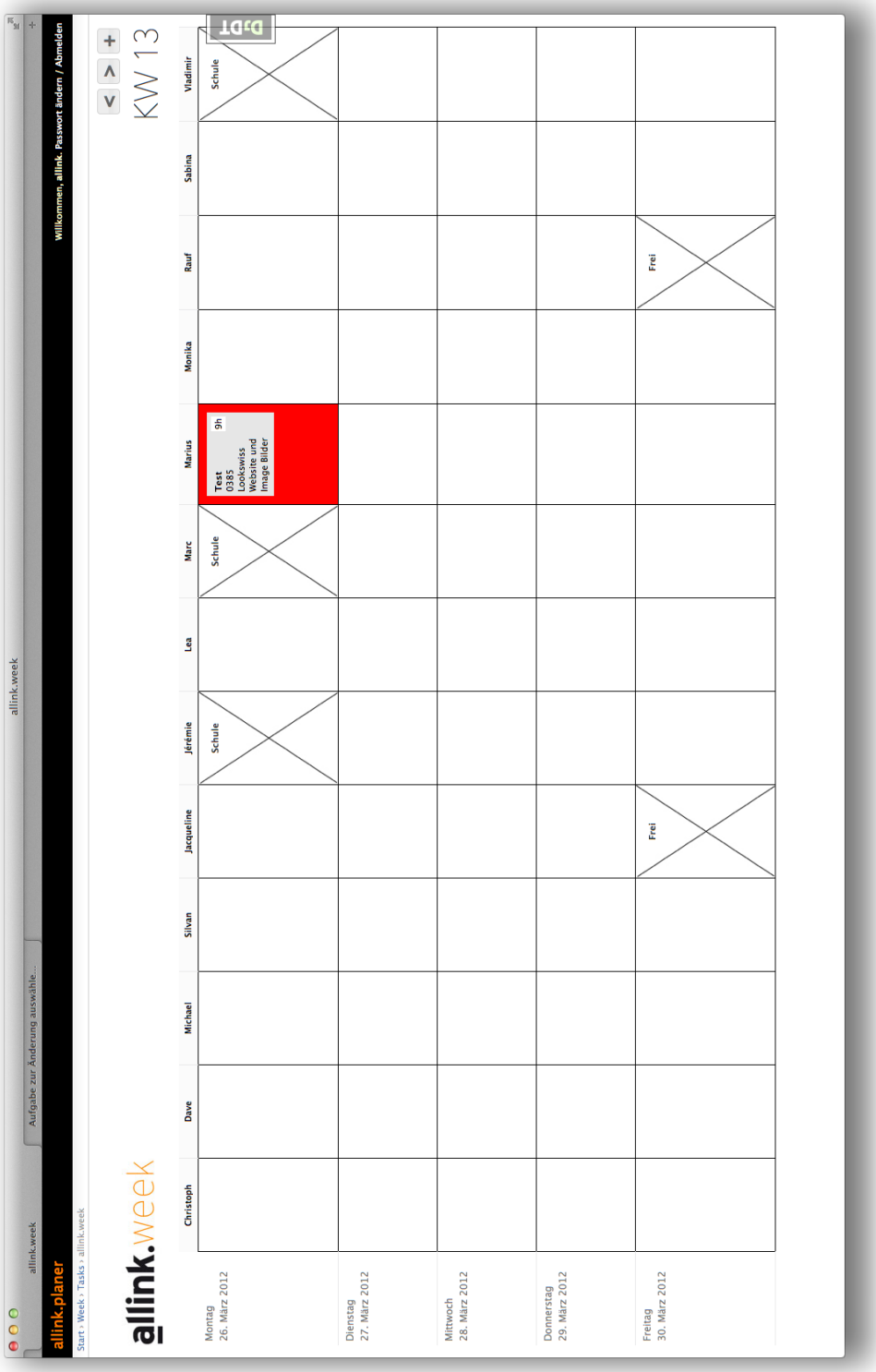


Abbildung 5.11.: Abgedeckte Ziele: Nr. 17

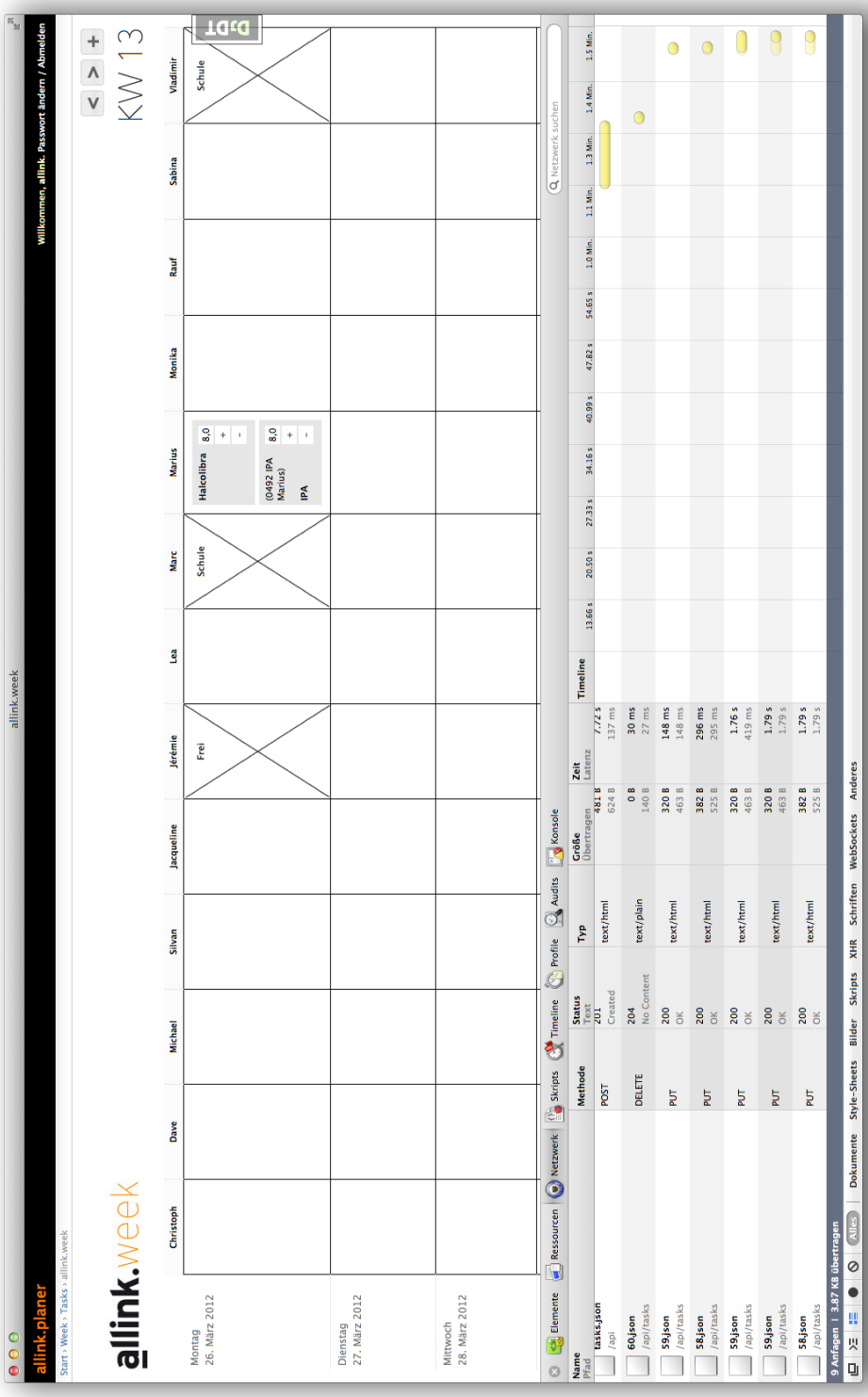


Abbildung 5.12.: Abgedeckte Ziele: Nr. 18

### 5.2.1. Resultate Testing

Hier sind die Resulate der Tests aufgelistet:

| Nr | MUSS-Funktion | erfüllt |
|---|---|---|
| 1 | 1 | ja |
| 2 | 2 | ja |
| 3 | 3 | ja |
| 4 | 4 | ja |
| 5 | 5 | ja |
| 6 | 6 | ja |
| 7 | 7 | ja |
| 8 | 8 | ja |
| 9 | 9 | ja |
| 10 | 10 | ja |
| 11 | 11 | ja |
| 12 | 12 | ja |
| 13 | 13 | ja |
| 14 | 14 | ja |
| 15 | 15 | ja |
| 16 | 16 | ja |
| 17 | 17 | ja |
| 18 | 18 | ja |

Tabelle 5.2.: Resultate Testfälle

[0]Eigene Darstellung

### 5.2.2. Field-Test

Am 24.03. hat die Geschäftsführung an ihrer wöchentlichen Planungssitzung das Tool das erste Mal verwendet. Dabei wurde das Tool auf seine Intuitivität geprüft. Am 26.03. habe ich Feedback erhalten um anfallende Anpassungen vornehmen zu können.

### 5.2.3. Resultat Field-Test

Das Tool wurde als gut befunden, die gewünschte Intuitivität ist umgesetzt worden und der Auftrag gilt seitens der Projektleitung als erfüllt.
Jedoch wurde die Bedienung, um ein Projekt auszuwählen, als zu umständlich empfunden. Es wird gewünscht, dass nach meiner IPA ein Formular mit “autocompletion” implementiert wird.

# 6. Konklusion

## 6.1. Was ich gelernt habe

In meiner 10-tägigen IPA habe ich folgendes gelernt:

- wie man eine bestehende Applikation analysiert und Muss-/Kann-Ziele erfasst.
- planen wie viel Zeit man für eine Aufgabe benötigt
- Eine komplexe und datenlastige HTML-Ansicht umzusetzen
- Ziele zu definieren und für den zeitlichen Rahmen abzugrenzen
- verschachtelte Queries erstellt um eine kompakte Datenabfrage zu generieren
- eine dynamische Kommunikation mit dem Server herzustellen
- dynamische CRUD-Request erstellen
- die jQuery Library jQueryUI anzuwenden
- Testfälle erstellen und durchführen

## 6.2. Positives

Ich hatte sehr viel Spass an meiner Arbeit, da es sich um eine Webapplikation handelte die über AJAX CRUD-Funktionalität verfügt und ich etwas derartiges noch nie gemacht habe. Was mich besonders überraschte war:

- in Django Schnittstellen zu programmieren und diese zu erweitern ist relativ simpel.
- dass jQuery ohne grössere Probleme einen AJAX-Request bauen und per callback wieder Daten entgegen nehmen kann.

Was mich natürlich auch freute war, dass ich alle Ziele erreicht habe, wenn auch nicht unbedingt in der zu Beginn geplanten Umsetzungszeit.

## 6.3. Negatives

### 6.3.1. Zeitplanung

Dass ich die Zeitplanung nicht mit einer Timeline umgesetzt habe, machte mich nicht ganz zufrieden. Jedoch empfand ich es als schwer ein so kleines Projekt so genau planen zu können. Es traten in der Realität immer wieder kleine Probleme auf, die alle zusammen mein Zeitmanagement durcheinander gebracht hätten. In Zukunft werde ich versuchen Aufgaben besser zu planen damit sie besser mit dem zeitlichen Rahmen übereinstimmen.

### 6.3.2. Testing

Im Nachhinein wurde mir klar, dass ich meine Testmethode besser von Anfang an in den Programmierprozess hätte einbeziehen sollen oder einen besseren Weg finden, um für diese Funktionalität ein Frontend-Testing durchzuführen. Ich war der Ansicht, dass die Testmethode für die Testfälle ausreichend ist, da die Realisierung stark auf die Intuitiviät im Frontend ausgerichtet war.

# A. Abbildungsverzeichnis

# B. Tabellenverzeichnis

# C. Abkürzungsverzeichnis

| Abkürzung | Bedeutung |
|---|---|
| AJAX | Asynchronous JavaScript and XML |
| API | Application programming interface |
| branch | Kette von commits in Repository |
| CSS | Cascading Style Sheets |
| CRUD | Create, Read, Update, Delete Datenbankfunktionlität |
| Django | Python Webdevelopment Framework |
| django-piston | REST Framework for Django |
| ERM | Entity Relation Model |
| Git | version control system |
| Github | Git hosting platform |
| HTML | Hyper Text Markup Language |
| jQuery | Javascript Framework |
| jQueryUI | Javascript Framework for User Interfaces |
| JSON | Javascript Object Notation |
| Repository | Gesamtes versioniertes Projekt |
| REST | Representational State Transfer |

Tabelle C.1.: Abkürzungen

# D. Literaturverzeichnis

Python Dokumentation:
http://docs.python.org/
Django Dokumentation:
https://docs.djangoproject.com/en/1.3/
Javascript Dokumentation:
http://www.w3schools.com/jsref/default.asp
jQuery Dokumentation:
http://docs.jquery.com/Main_Page

# E. Arbeitsjournal

## E.1. 13.03.2012

Beginn IPA yatplaner

- Dokumentation auf github hosten und versionieren
- Kapitelstrukur aufbauen
- Zeitplan erstellen (Planung)
- Aufgabenstellung erfassen
- Projektorganisation grafisch erfassen
- Designbesprechung mit Dave
- Vorkenntnisse erfassen
- Vorarbeiten erfassen
- Firmenstandards erfassen
- Ziel erreicht: Teil 1 Der Doku erfasst
- Umfeld erfassen
- IST-Analyse vorerfassen (Ansichten, Funktionalität)

## E.2. 14.03.2012

- IST-Analyse vorerfassen (Modellierung der Rails App)
- Sitzungen mit Silvan Spross und Marc Egli zur Definition der MUSS- und KANN-Ziele
- MUSS-Ziele erfasst
- KANN-Ziele erfasst
- Planung abgeschlossen

- ERM für Wochenplaner erstellt
- allink.planer github repository gecloned
- Entwicklungsumgebung aufgesetzt
- Ziel erreicht: Anfang Teil 2 der Dokumentation erfasst und mit Realisierung begonnen
- mit Django-Modellierung begonnen (Task, DayConfiguration)

## E.3. 15.03.2012

- Kapitel Realisierung weiterführen
- Django admin einrichten für week
- Testdaten (Personen) in die Entwicklungsumgebung importieren
- Problem: Die Wochentage zusammenzufassen war relativ knifflig damit man durch jeden Tag iterieren kann
- View erstellen für alle Tasks, Personen und Tage
- Template erstellt und Tabelle so aufgesetzt damit Daten korrekt eingefügt werden
- Template angefangen zu stylen (CSS)
- Ziel erreicht: Daten korrekt aus Modell laden und in Template einsetzen, Template funktioniert
- (Bin im Zeitplan mit Realisierung)

## E.4. 16.03.2012

- Ansicht funktioniert für erste Versuche mit jQueryUI
- Task sind verschiebbar und sortierbar in andere Tage (ohne callback/Änderungen werden nicht gespeichert)
- alle Tasks werden zu javascript Objekten der Klasse Task
- jQuery Methoden hinzugefügt (Click-Events)
- Task-Dialog hinzugefügt um neue Task zu erfassen
- Task Objekt erweitert mit save Methode wenn eine Task gespeichert wird

- Django Task Formular
- Django Task Piston Handler (json)
- AJAX POST schicken
- piston nimmt json-POST entgegen und führt das Formular, aus welches die Daten in die Datenbank speichert
- wenn eine neue Task erstellt wurde, wird die neue task.id zurückgeschickt um das neue Javascript Task Objekt abzuspeichern
- Ziel erreicht: jQueryUI einrichten, mind. POST per Ajax möglich und dynamisches einfügen in Tabelle

## E.5. 19.03.2012

- Taskobjekte der aktuellen Woche in javascript dictionary speichern
- Hilfestellung Silvan: PUT Funktionalität in Django abhandeln bei existierendem Objekt
- Verschieben von Task (PUT Funktionalität)
- Klick auf eine Task öffnet Dialog und man kann die Task bearbeiten und abspeichern
- Bugfix: wenn man eine Task verschoben wird, öffnet es danach den Bearbeiten-Dialog. Per deaktivieren des click events kann dies behoben werden.
- Bugfix: man konnte eine neu erstellte Task nicht verschieben
- Projekte werden dynamisch in den Dialog eingefügt und können ausgewählt werden
- Task einem Projekt zuweisen
- Ziel: PUT Funktionalität erstellen für Task verschieben und ändern
- Dokumentation weiter führen
- Besprechung mit Silvan Stand IPA

## E.6. 20.03.2012

- Umsetzung MUSS-Ziele erreicht
- Dokumentation weiter geschrieben
- feature task duplizieren ausprobiert (Kann-Ziel)
- Bugfix: Taskzelle ist zu wenig hoch um problemlos neue Task in Zelle zu verschieben
- Bugfix: Bei Bearbeitung einer Task wird aktuelles Projekt nicht mitgeschickt

## E.7. 21.03.2012

- Sperrtag bild ausgewechselt
- Bugfix: wenn man im Template die Woche wechselt kann das Datumsformat nicht vearbeitet werden
- Queries optimieren mit Marc, verschachtelte Tuples für queries
- Dokumentation weiter geschrieben
- Kleinere Optimierungen
- Cloning weiter versuchen

## E.8. 22.03.2012

- Fehlerbehebungen im Template: Kalenderwoche wird nicht aktualisiert
- KANN-Ziel Tasks sortieren
- KANN-Ziel Warnung wenn Tasks in Tag mehr als 8h Stunden betragen
- KANN-Ziel Einer Tageskonfiguration ein Startdatum setzen
- Bugfix wenn neue Task erstellt wurde konnte sie nicht direkt bearbeitet werden.
- Besprechung Sperrtage habe ein Start- und Enddatum und können von weiteren Sperrtagen ersetzt ohne aus der DB gelöscht zu werden
- Mit Unterstützung von Marc die beste Lösung finden die Sperrtage auszulesen (Bei jedem Tag abfragen)
- Bugfix man kann nicht weniger als 1 Woche anzeigen

- Bugfix wenn man die Tagesanzahl erhöht wird aktuelles Datum nicht berücksichtigt
- Besprechung wie das Task-cloning am besten implementiert wird

## E.9. 23.03.2012

- Task kann per Klick dupliziert werden (es wird eine neue mit gleichem Inhalt erzeugt und unten eingefügt)
- Task kann per Klick gelöscht werden
- Doku schreiben
- Mit Silvan Wochenplaner angeschaut und Darstellungsanpassungen vorgenommen
- Abnahme Wochenplaner Silvan

## E.10. 26.03.2012

- Bugfix bei verschieben von Task auf max. Stunden Anzahl prüfen.
- Bugfix wenn bei neuem Task keine Stunden angegeben sind.
- Dokumentation überarbeiten
- Dokumentation fertigstellen
- IPA Dokumentation einreichen