

# **Dokumentation IPA yatplaner**

IPA in Applikationsentwicklung

Auszubildender - Marius Küng

Auftraggeber - allink.creative GmbH

Projektleiter - Silvan Spross

Chefexperte - Hans Riesenmann

Experte - Antonio Di Luzio

Durchführungsort - allink.creative GmbH

Informatikmittelschule Basel

13.03. - 26.03.2012

# Inhaltsverzeichnis

<b>I. Umfeld und Ablauf</b>	<b>4</b>
<b>1. Aufgabenstellung</b>	<b>5</b>
1.1. Titel der Facharbeit . . . . .	5
1.2. Thematik . . . . .	5
1.3. Klassierung . . . . .	5
1.4. Durchführungsblock . . . . .	5
1.5. Ausgangslage . . . . .	6
1.6. Detaillierte Aufgabenstellung . . . . .	6
1.7. Mittel und Methoden . . . . .	6
1.8. Vorkenntnisse . . . . .	7
1.9. Vorarbeiten . . . . .	7
1.10. Neue Lerninhalte . . . . .	7
1.11. Arbeiten in den letzten 6 Monaten . . . . .	7
<b>2. Einleitung</b>	<b>8</b>
2.1. Projektorganisation . . . . .	8
2.2. Vorkenntnisse . . . . .	8
2.3. Vorarbeiten . . . . .	9
2.4. Firmenstandards . . . . .	9
<b>II. Projekt</b>	<b>11</b>
<b>3. Projektbeschreibung</b>	<b>12</b>
3.1. Umfeld . . . . .	12
3.2. Präzisierung der Aufgabenstellung . . . . .	12
3.3. Analyse Wochenplaner (IST-Zustand) . . . . .	12
3.3.1. Modellierung . . . . .	12
3.3.2. Darstellung . . . . .	13
3.3.3. Funktionsumfang . . . . .	13

3.3.4. Defizite . . . . .	13
3.4. Definition Ziele . . . . .	14
3.4.1. MUSS-Ziele . . . . .	14
3.4.2. KANN-Ziele . . . . .	14
<b>4. Realisierung</b>	<b>17</b>
4.1. ERM's . . . . .	17
4.1.1. allink.planer . . . . .	17
4.1.2. Wochenplaner . . . . .	18
4.2. Design . . . . .	19
4.2.1. Altes Design . . . . .	19
4.2.2. Neues Design . . . . .	20
4.3. Umsetzung . . . . .	20
4.4. Resultat . . . . .	21
4.5. Modellierung (Django, aufsetzen Projekt)/Backend . . . . .	22
4.6. Layout . . . . .	22
4.7. Entwicklungsumgebung . . . . .	22
4.8. Herausforderungen Entscheidungen . . . . .	22
4.8.1. JSON . . . . .	22
4.8.2. Queries für Wochenansicht . . . . .	22
4.8.3. JS Funktionalität (CRUD mit jQueryUI & AJAX) . . . . .	23
4.8.4. Sperrtage . . . . .	23
4.8.5. Task kopieren per jQueryUI . . . . .	23
4.9. Implementierung . . . . .	23
<b>5. Testing</b>	<b>24</b>
5.1. Testmethoden . . . . .	24
5.2. Testfälle erfassen . . . . .	24
5.3. Testbericht (Ziele) . . . . .	24
<b>6. Konklusion</b>	<b>25</b>
 <b>III. Arbeitsjournal</b>	 <b>26</b>
<b>7. Arbeitsjournal</b>	<b>27</b>
7.1. 13.03.2012 . . . . .	27
7.2. 14.03.2012 . . . . .	27
7.3. 15.03.2012 . . . . .	28
7.4. 16.03.2012 . . . . .	29

7.5. 19.03.2012 . . . . .	29
7.6. 20.03.2012 . . . . .	30
7.7. 21.03.2012 . . . . .	30
7.8. 22.03.2012 . . . . .	30
<b>8. Abbildungsverzeichnis</b>	<b>32</b>
<b>9. Tabellenverzeichnis</b>	<b>33</b>

**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 - 23.04.2012

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 ältere 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  $\geq 8$
- Chrome  $\geq 10$
- Safari  $\geq 5$

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

#### Projekorganisation

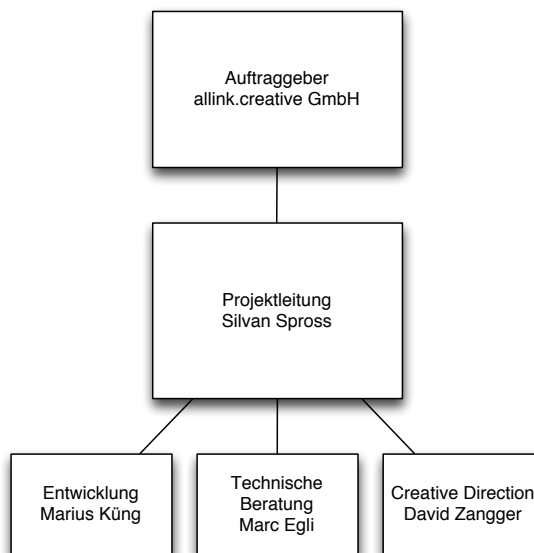


Abbildung 2.1.: Projekorganisation<sup>1</sup>

### 2.2. Vorkenntnisse

#### Technologien

- Python Grundkenntnisse
- Django Fortgeschrittene Kenntnisse

---

<sup>1</sup>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. Ausserdem den allink.planer studiert in den, der Wochenplaner implementiert wird. Ansonsten fand keine explizite Vorarbeit statt.

## 2.4. Firmenstandards

- Betriebssystem: Mac OS X
- Editor: Textmate
- Entwicklungsumgebung: Python, Django (Python-Framework für Webapplikationen)
- Virtuelle Testumgebung: Django Serversimulation (per Terminal steuerbar)
- Deployment: per fabric-script auf Apache-Server mit WSGI-Protokoll
- Versionierung: Git, auf Github gehostet

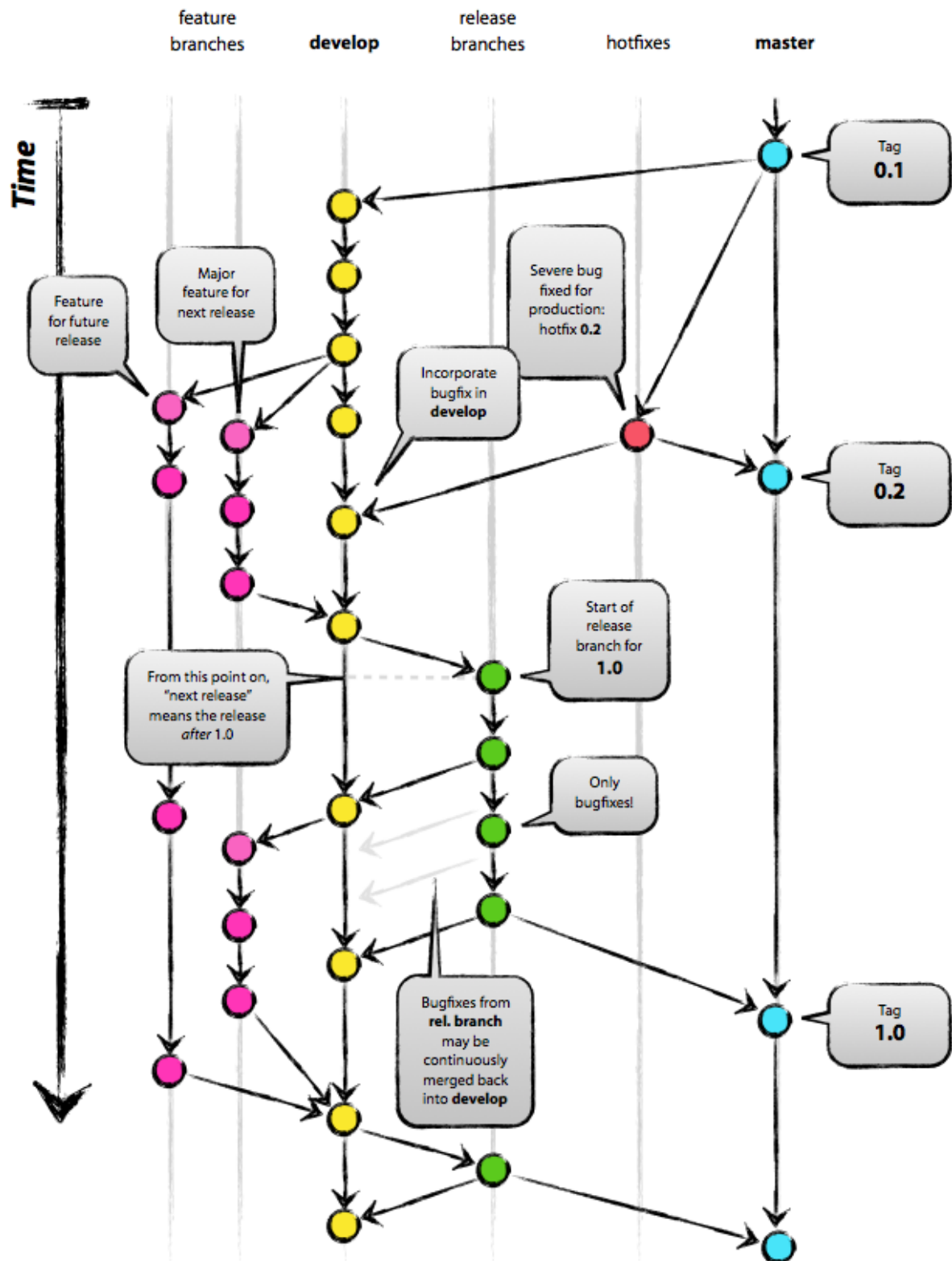


Abbildung 2.2.: A successful Git branching model von Vincent Driessen

# **Teil II.**

# **Projekt**

## 3. Projektbeschreibung

### 3.1. Umfeld

Momentan läuft der Wochenplaner vom allink.planer getrennt und kann nicht in den allink Projektablauf einbezogen werden. 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 (mehr) dem Firmenstandard entspricht, und kann deshalb nur mühsam gewartet und erweitert werden.

### 3.2. Präzisierung der Aufgabenstellung

Da die Aufgabenstellung detailliert erfasst wurde und ein Prototyp existiert, ist relativ klar was die eigentliche Aufgabenstellung umfasst. 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 Funktionalitäten 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 (Screenshot)
- Übersicht aller Angestellten
- Monatsansicht einer Person
- Übersicht aller erfassten Tasks

#### 3.3.3. Funktionsumfang

- Neue Task durch Doppelklick auf Personentag per Dialog erfassen (Screenshot)
- Klick auf Task öffnet die Detailansicht (um bspw. Änderungen vorzunehmen)
- Verschiebung einer Task 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 hinzufügen (kann auch rückgängig gemacht werden)
- Wochentage können für wiederkehrende Events (Schule, Frei) markiert werden

#### 3.3.4. Defizite

- Die Sperrtage haben kein Startdatum und sind für jeden Wochentag 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**

### **3.4.2. KANN-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	Sperntag Person zuweisen	Der Benutzer kann einer Person einen Sperntag zuweisen.
10	Sperntag Namen geben	Der Benutzer kann einem Sperntag einen Namen geben.
11	Sperntag bearbeiten	Der Benutzer kann einen Sperntag bearbeiten.
12	Sperntag löschen	Der Benutzer kann einen Sperntag löschen.
13	Task einem Projekt zuweisen	Der Benutzer kann einen Task einem Projekt zuweisen.
14	Person zu Wochenplaner	Eine Person zum Wochenplaner hinzufügen.
15	Person als Partner	Eine Person als Partner kennzeichnen.

Tabelle 3.1.: Zwingend umzusetzende Funktionen des Prototypen



Nr	Funktion	Beschreibung	Priorität
16	Task duplizieren	Der Benutzer kann einen Task duplizieren.	1
17	Sperntag Startdatum geben	Der Benutzer kann einem Sperntag ein Startdatum geben.	3
18	Warnmeldung Max	Es erscheint eine Warnmeldung wenn die Summe aller Arbeitsstunden pro Tag 8h überschreiten.	4
19	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.

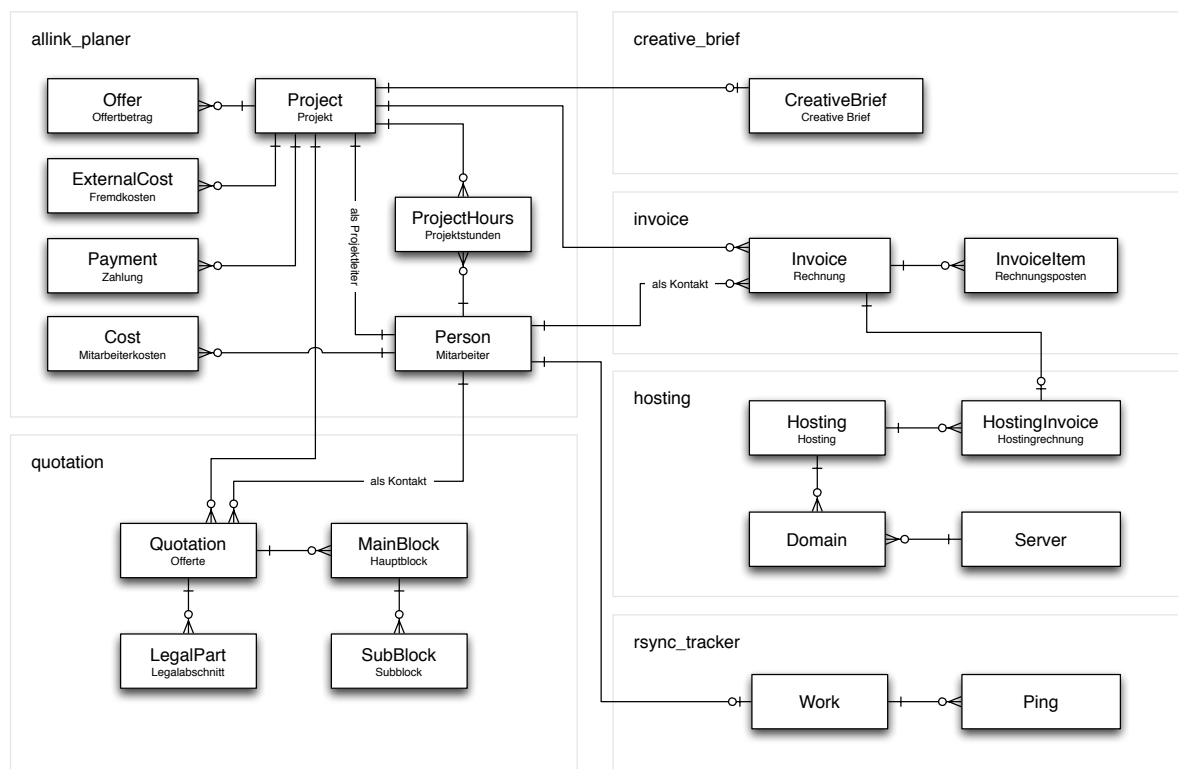


Abbildung 4.1.: ERM allink.planer<sup>1</sup>

<sup>1</sup>Entommen aus allink.planer

#### 4.1.2. Wochenplaner

## Wochenplaner

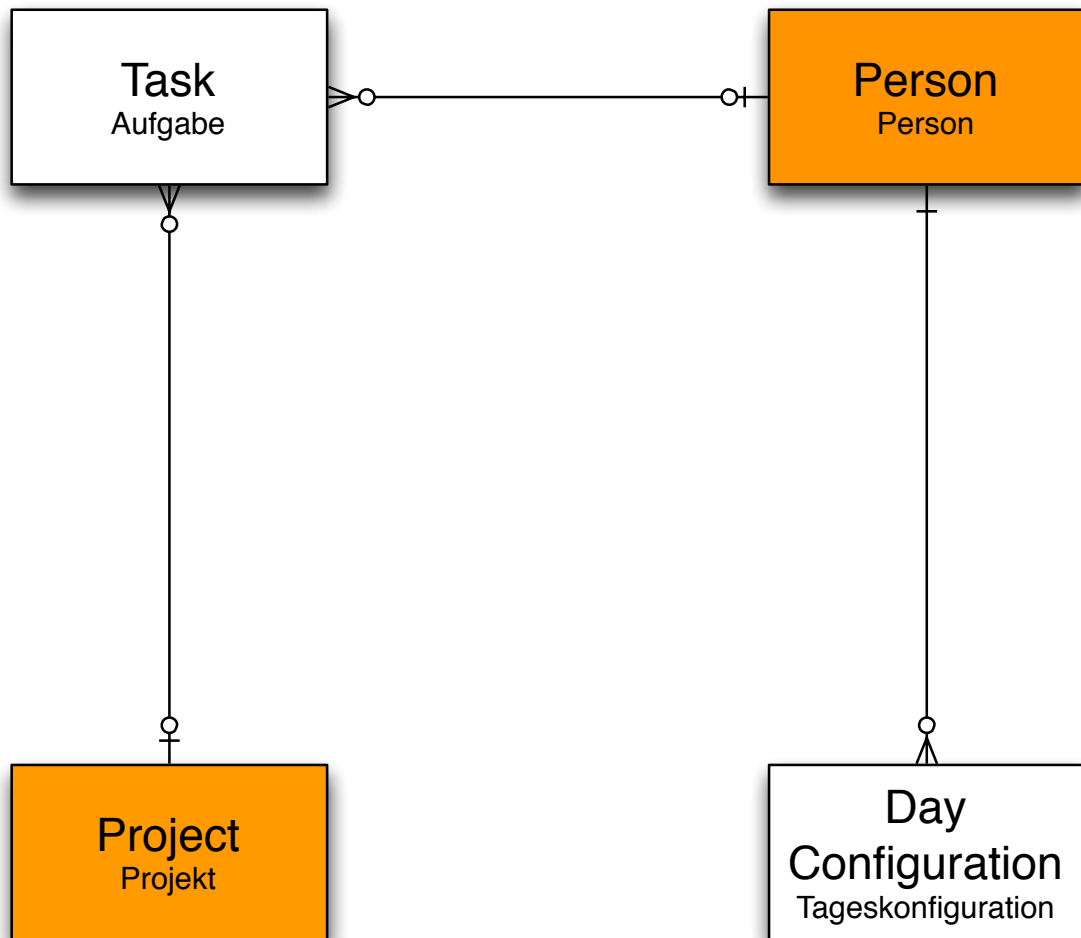


Abbildung 4.2.: ERM Wochenplaner (die Orange hinterlegten Entitäten stammen aus dem allink.planer)<sup>2</sup>

---

<sup>2</sup>Eigene Darstellung

## 4.2. Design

## 4.2. Design

### 4.2.1. Altes Design

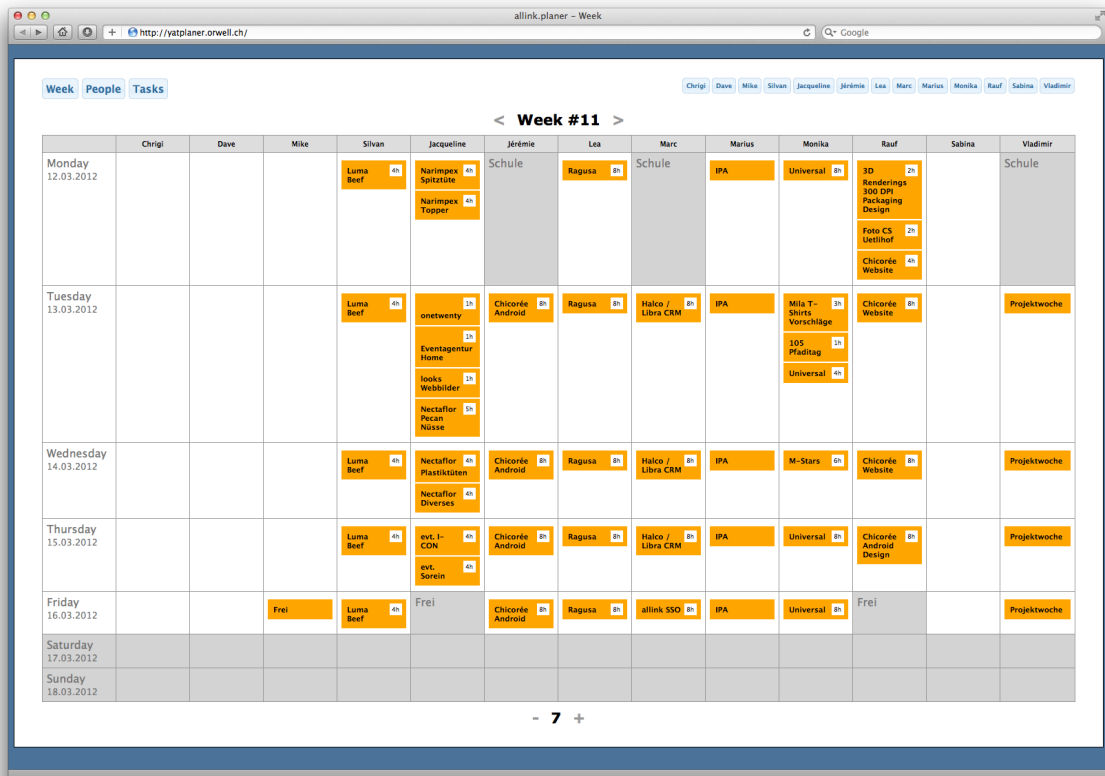


Abbildung 4.3.: Bisheriges Design des yatplaner

## 4.3. Umsetzung

### 4.2.2. Neues Design

	Chrigi	Dove	Mike	Silvan	Jacqueline	Jérémie	Lea	Marc	Marius	Monika	Rauf	Subina	Vladimir
Monday 05.03.2012				Luma Beef 4h	IAM Anzeigen 1h IAM Kalender 1h NectraFlor Limited Topper 6h		Luma Beef 8h		Looks Website 8h	Telekurs Bilder 1h Luma Beef 1h Ragusa Bilder 1h	I AM Banner 1h IAM Entry Clip 1h Jobsch 1h	Conci 1h Kraft Kalender 1h	
Tuesday 06.03.2012				Luma Beef 4h	Limited Topper / film parkuhot 4h NectraFlor Sticker Indonesien 2h Sorein Spenderflasche 2h	Orion News 4h Chicoree Android 4h	Sorein AD 2h Ragusa 8h	105 Apps 4h Halco / Libra CRM 4h	Looks Website 8h	Looks Flyer 8h	Chicoree Online 8h		Conci Seite 8h
Wednesday 07.03.2012				Luma Beef 4h	Sorein Spenderflasche 4h IAPN Roll-up 2h NectraFlor Spottstein 1h	Chicoree Android 8h	Ragusa 8h	Halco / Libra CRM 8h	Looks Website 8h	M-Stars 4h Universal Nossal! 4h	Chicoree Online 8h		Conci Seite 8h
Thursday 08.03.2012				Zumstag 4h		Chicoree Android 8h	Ragusa 8h	Halco / Libra CRM 8h	Looks Website 8h	Universal Nossal! 8h	Chicoree Online 8h		Conci Seite 8h
Friday 09.03.2012				Zumstag 4h		Chicoree Android 8h	Ragusa 8h	allink SSO 8h	IPA Vorbereitung 8h	M-Stars 4h Camen 4h			Conci Seite 8h

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:

- Administratorbereich
- Wochenansicht

Obwohl eine intuitive Bedienung gefordert ist, müssen nicht alle Ziele in der Wochenansicht manipulierbar sein.

Die folgende Aufstellung zeigt auf welche Ziele wie 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	o
8	x	o
9	x	o
10	x	o
11	x	o
12	x	o
13	x	x
14	x	o
15	x	o

Tabelle 4.2.: Funktionsbereiche der Ziele

## 4.4. Resultat

Das Resultat meiner IPA ist einen Wochenplaner in das Projektmanagementtool allink.planer zu integrieren. Dieser Wochenplaner ist über eine HTML-Ansicht dynamisch bearbeitbar. Die Tasks sind mit Personen und Projekten verknüpft. Dadurch können Analysen gemacht werden.

### 4.5. Modellierung (Django, aufsetzen Projekt)/Backend

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

### 4.6. Layout

Das Layout bleibt, bis auf einige kleine Anpassungen, 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 verwendet allink eine Methode die über das JSON-Format Requests an den Server schickt. Über den django-piston Handler können diese Requests entgegengenommen und weitergeleitet werden.

Bis jetzt habe ich lediglich über einen JSON-Handler (django-piston) Daten aus der Datenbank geholt oder neue Datensätze erstellt.

Einen bestehenden Datensatz zu ändern und diesen dann korrekt im Frontend wieder einzusetzen ist das 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 habe über die 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 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 gesteuert werden. Dies ist aber nur ein Teil der bestehenden Funktionalität. Es wird eine CRUD-Funktionalität im Frontend benötigt. Diese werde ich mit jQueryUI möglich machen da ich mit jQuery am meisten Erfahrung habe. Sich in Alternativen einzuarbeiten benötigt zu viel Zeit und ist nicht mit jQuery kompatibel. Per jQueryUI können alle Task frei verschoben werden zu jedem Tag und/oder Person. 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, 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 Abfragen für die Wochenansicht wird für jeden Tag geprüft, ob in dem 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 kopieren per jQueryUI

## 4.9. Implementierung

Der Wochenplaner kann als App mit Abhängigkeiten (Person, Projekt, piston, jQueryUI) in den allink.planer direkt integriert werden.



## **5. Testing**

### **5.1. Testmethoden**

### **5.2. Testfälle erfassen**

Die Testfälle beziehen die Ausführbarkeit der MUSS- und KANN-Ziele. Einige Ziele sind jeweils im Admin und in der Wochenansicht ausführbar. Daher werde ich beide auf ihren Erfolg testen.

### **5.3. Testbericht (Ziele)**

## **6. Konklusion**

**Teil III.**

# **Arbeitsjournal**

## 7. Arbeitsjournal

### 7.1. 13.03.2012

Beginn IPA yatplaner

- Dokumentation auf github hosten und versionieren
- Kapitelstruktur aufbauen Teil 1
- Zeitplan erstellen (Planung)
- Aufgabenstellung erfassen
- Projektorganisation grafisch erfassen
- Designbesprechung mit Dave
- Vorkenntnisse erfassen
- Vorarbeiten erfassen
- Firmenstandards erfassen
- Ziel erreicht: Teil 1 Der Doku erfasst Teil 2
- Umfeld erfassen
- IST-Analyse vorerfassen (Ansichten, Funktionalität)

### 7.2. 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: Teil 2 zum Grossteil der Dokumentation erfasst und mit Realisierung begonnen
- mit Django-Modellierung begonnen (Task, DayConfiguration)
- Habe bis jetzt für Doku mehr Zeit gebraucht als im Zeitplan vorgesehen

### **7.3. 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 begonnen zu stylen
- Ziel erreicht: Daten korrekt aus Modell laden und in Template einsetzen, Template funktioniert
- (Bin im Zeitplan mit Realisierung)

## 7.4. 16.03.2012

- Ansicht funktioniert für erste Versuche mit jQueryUI
- Task sind verschiebbar und sortierbar in andere Tage (ohne callback)
- alle Tasks werden zu javascript Objekte der Klasse Task
- jquery methoden hinzugefügt
- dialog hinzugefügt
- Task Objekt erweitert mit save methode wenn eine Task gespeichert wird
- Django Task Formular
- Django Task Piston Handler (json)
- AJAX POST schicken
- json piston nimmt POST entgegen und führt das formular aus welches die daten in die db speicher
- wenn neuer eintrag gemacht wurde wird die neue task.id zurückgeschickt um es lokal im js task objekt zu speichern
- Ziel: jQueryUI einrichten, mind. POST per Ajax möglich und dynamisches einfügen in Tabelle (beide erreicht)

## 7.5. 19.03.2012

- Todo: Taskobjekte holen der aktuellen woche
- 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 draggt 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
- Ziel: PUT Funktionalität erstellen für Task verschieben und ändern
- Dokumentation weiter führen

- Besprechung mit Silvan Stand IPA

## **7.6. 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

## **7.7. 21.03.2012**

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

## **7.8. 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



## 8. Abbildungsverzeichnis

2.1. Projekorganisation . . . . .	8
2.2. A successful Git branching model von Vincent Driessen . . . . .	10
4.1. . . . .	17
4.2. . . . .	18
4.3. Bisheriges Design des yatplaner . . . . .	19
4.4. Neues Design . . . . .	20

## 9. Tabellenverzeichnis

3.1. Zwingend umzusetzende Funktionen des Prototypen . . . . .	15
3.2. Nicht zwingend umzusetzende Funktionen des Prototypen . . . . .	16
4.2. Funktionsbereiche der Ziele . . . . .	21