Dokumentation Projektverwaltung

Änderungsübersicht

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Autor | Bemerkung |
| 0.1 | 30.11.15 | Lars Anderegg | Inhaltsverzeichnis erstellt |
| 0.2 | 09.12.15 | Lars Anderegg | Projektplan & Erste zwei Iterationsziele |
| 0.3 | 12.12.15 | Lars Anderegg | Geschäftsprozessmodellierung angefangen |
| 0.4 | 15.12.15 | Lars Anderegg | Anforderungsanalyse angefangen |
| 0.5 | 19.12.15 | Lars Anderegg | Geschäftsprozessmodellierung abgeschlossen &  Neue Iterationsziele |
| 0.6 | 28.12.15 | Lars Anderegg | Anforderungsanalyse abgeschlossen |
| 0.7 | 03.01.16 | Lars Anderegg | Konfigurationsmanagement &  Neue Iterationsziele |
| 0.8 | 09.01.16 | Lars Anderegg | Geschäftsprozessmodellierung & Anforderungsanalyse aufgrund von Review überarbeitet |
| 0.9 | 10.01.16 | Lars Anderegg | Testkonzept & Risikoanalyse erstellt |
| 1.0 | 12.01.16 | Lars Anderegg | Testfälle erarbeitet |

Inhaltsverzeichnis

[1 Einleitung 3](#_Toc440210824)

[1.1 Allgemein 3](#_Toc440210825)

[2 Glossar 3](#_Toc440210826)

[3 Projektplan 3](#_Toc440210827)

[3.1 Iterationsplanung 4](#_Toc440210828)

[4 Risikoanalyse 5](#_Toc440210829)

[4.1 Allgemein 5](#_Toc440210830)

[4.2 Analyse möglicher Probleme 5](#_Toc440210831)

[5 Geschäftsprozessmodellierung 7](#_Toc440210832)

[5.1 Geschäftsprozess 7](#_Toc440210833)

[5.2 Geschäftsanwendungsfälle 8](#_Toc440210834)

[6 Anforderungsanalyse 9](#_Toc440210835)

[6.1 Systemanwendungsfälle 9](#_Toc440210836)

[7 Design 14](#_Toc440210837)

[7.1 Fachklassen 14](#_Toc440210838)

[7.2 Datenbank 14](#_Toc440210839)

[8 Qualitätsmanagement 14](#_Toc440210840)

[8.1 Testkonzept 14](#_Toc440210841)

[9 Konfigurationsmanagement 15](#_Toc440210842)

[9.1 Git-Repository 15](#_Toc440210843)

[10 Änderungsmanagement 16](#_Toc440210844)

[11 Anhang 16](#_Toc440210845)

# Einleitung

Im fünften Semester des Lehrgangs Techniker HF an der TEKO wird eine Semesterarbeit durchgeführt. Im Rahmen der Semesterarbeit bekommt jeder Schüler einen Projektauftrag den er selbstständig umsetzen muss.

## Allgemein

### Zweck des Dokuments

Blablabla.

# Glossar

# Projektplan

*Aus Excel einfügen*

## Iterationsplanung

### Einführung Iteration 1

|  |  |  |  |
| --- | --- | --- | --- |
| Ziel | Erreicht | Probleme | Massnahmen |
| Kickoff-Meeting | OK |  |  |
| Dokumenationsvorlage erstellt | OK |  |  |
| Pflichtenheftvorlage erstellt | OK |  |  |
| Projektplanvorlage erstellt | OK |  |  |

### Einführung Iteration 2

|  |  |  |  |
| --- | --- | --- | --- |
| Ziel | Erreicht | Probleme | Massnahmen |
| Pflichtenheft fertig | NOK | Zu wenig Zeit, ist aber zu 70% fertig | Noch kein kritischer Verzug, deshalb keine Massnahmen |
| Projektplan fertig | OK |  |  |
| Geschäftsprozessmodellierung bereit für Plenum | OK |  |  |
| Anforderungsanalyse bereit für Plenum | OK |  |  |

### Spezifikation Iteration 1

|  |  |  |  |
| --- | --- | --- | --- |
| Ziel | Erreicht | Probleme | Massnahmen |
| Geschäftsprozessmodellierung fertig | OK |  |  |
| Anforderungsanalyse fertig | OK |  |  |
| Git-Repository erstellen | OK |  |  |
| Erster Entwurf eines Datenmodells | OK |  |  |
| Auswahl des Frameworks | OK | Wurde unterschätzt |  |
| Aufsetzen eines Workspaces | OK |  |  |

### Spezifikation Iteration 2

|  |  |  |  |
| --- | --- | --- | --- |
| Ziel | Erreicht | Probleme | Massnahmen |
| Datenmodell fertig |  |  |  |
| Diagramme für Use Cases |  |  |  |
| Testplan für Use Cases |  |  |  |
| Risikoanalyse erstellen | OK |  |  |
| Konfigurationsmanagement ausformulieren |  |  |  |
| Mini Prototyp (Web<->Datenbank) fertig | OK |  |  |

### Konstruktion Iteration 1

|  |  |  |  |
| --- | --- | --- | --- |
| Ziel | Erreicht | Probleme | Massnahmen |
| UI Skizzen |  |  |  |
| Layout implementieren |  |  |  |
| Saubere Schichten Trennung |  |  |  |
| Änderungsmanagement erstellen |  |  |  |

### Konstruktion Iteration 2

|  |  |  |  |
| --- | --- | --- | --- |
| Ziel | Erreicht | Probleme | Massnahmen |
|  |  |  |  |

### Einführung Iteration 2

|  |  |  |  |
| --- | --- | --- | --- |
| Ziel | Erreicht | Probleme | Massnahmen |
|  |  |  |  |

# Risikoanalyse

## Allgemein

### Ziel der Risikoanalyse

Die Risikoanalyse dient zur differenzierten Betrachtung und Diskussion einzelner (oder aller) beobachteten Risiken für das Projekt „Liegenschaftsverwaltung ImmoGlobal“. Sie dient vor allem für alle Projektbeteiligten als Information, welche Situationen einRisiko für das Problem darstellen könnte.

### Projektmitarbeiter

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Name | Telefon | E-Mail |
| Projektleiter | Lars Anderegg | 079 765 12 34 | lan@software.ch |

## Analyse möglicher Probleme

### Krankheit / Unfall

|  |  |
| --- | --- |
| Id | P-01 |
| Verantwortlich | Lars Anderegg |
| Beschreibung | Der Projektleiter erkrankt oder verunfallt. Er ist nicht mehr in der Lage für das Projekt zu arbeiten. |
| Auswirkung | Da der Projektleiter alle Aufgaben übernimmt, verzögert sich das Projekt bis er wieder in Lage ist zu arbeiten. Es kann soweit hinhalten, dass das ganze Projekt gefährdet ist. |
| Wahrscheinlichkeit | Gering |
| Tragweite | Hoch |
| Massnahme | Der Projektleiter versucht Gefährlich Tätigkeiten zu vermeiden und zieht sich immer warm an wenn er nach Draussen geht. |

### Datenverlust

|  |  |
| --- | --- |
| Id | P-02 |
| Verantwortlich | Lars Anderegg |
| Beschreibung | Jegliche Daten von Dokumentation bis Sourcecode sind auf Festplatten abgelegt. Diese können durch ein viel Zahl von Gründen nicht mehr funktionieren und auf die Daten kann nicht mehr zugegriffen werden. |
| Auswirkung | Die Festplatten könnten plötzlich einen Defekt haben und die Daten wären nicht mehr zu retten. Das hätte tragische Auswirkungen auf das Projekt, denn die jeweiligen Fortschritte sind in den Dokumentationen festgehalten. |
| Wahrscheinlichkeit | Gering |
| Tragweite | Hoch |
| Massnahme | Die Daten werden auf verschiedenen Servern gespeichert. Dazu wurde auf github.com ein Repository eröffnet. |

### Zeitplanung

|  |  |
| --- | --- |
| Id | P-03 |
| Verantwortlich | Lars Anderegg |
| Beschreibung | Das Projekt hat einen fixen Abgabetermin, damit dieser erreicht werden kann wurde zu Beginn eine Planung gemacht. |
| Auswirkung | Wird die Planung nicht eingehalten, verschieben sich alle Tätigkeiten nach hinten. So können wichtige Funktionen oder Dokumentationsbestandteile nicht abgeschlossen werden. |
| Wahrscheinlichkeit | Mittel |
| Tragweite | Hoch |
| Massnahme | Durch vorausschauendes Projektmanagement und sorgfältige iterative Planung soll ein Desaster vermieden werden. |

# Geschäftsprozessmodellierung

## Geschäftsprozess

|  |  |
| --- | --- |
| Name | Projekt verwalten |
| Beschreibung | Alle Aktivitäten welche notwendig sind um ein Projekt verwalten zu können. Hierzu ist es notwendig das Projekt anzulegen, Vorgehensmodelle auszuwählen, und anschliessend das Projekt zu aktualisieren und Reportings zu generieren. |
| Enthaltene Geschäftsanwendungsfälle | -Projekt anlegen  -Vorgehensmodell auswählen  -Projekt aktualisieren  -Reporting generieren  -Projekt abschliessen |
| Verantwortliche | Projektleiter |
| Beteiligte | Projektleiter, Mitarbeiter |

Vorgehensmodell auswählen

Projekt anlegen

Projekt aktualisieren

Reporting generieren

Projekt abschliessen

Mitarbeiter auswählen

## Geschäftsanwendungsfälle

### Projekt anlegen

|  |  |
| --- | --- |
| Name | Projekt anlegen |
| Beschreibung | Projekt mit allen initialen Daten aufnehmen |
| Akteure | Projektleiter |
| Auslöser | Neues Projekt startet |
| Ergebnis | Projekt erstellt |
| Eingehende Daten | -Projektreferenz  -Projektname  -Projektbeschreibung  -Priorität des Projekts |
| Vorbedingungen |  |
| Ablauf | -Eingehende Daten erfassen |
| Priorität | Hoch, Projekt ist Grundlage für die folgenden Anwendungsfälle |

### Vorgehensmodell auswählen

|  |  |
| --- | --- |
| Name | Vorgehensmodell auswählen |
| Beschreibung | Vorgehensmodell anlegen oder bearbeiten und schliesslich eines auswählen |
| Akteure | Projektleiter |
| Auslöser | Projekt angelegt |
| Ergebnis | Vorgehensmodell ausgewählt |
| Eingehende Daten | Daten für Vorgehensmodell |
| Vorbedingungen |  |
| Ablauf | -Prüfen ob Vorgehensmodell vorhanden  -Vorgehensmodell anlegen oder bearbeiten  -Vorgehensmodell auswählen  -Projektphasen generieren |
| Priorität | Hoch, Vorgehensmodell muss ausgewählt werden |

### Mitarbeiter auswählen

|  |  |
| --- | --- |
| Name | Mitarbeiter auswählen |
| Beschreibung | Mitarbeiter in das Projekt aufnehmen. |
| Akteure | Projektleiter, Mitarbeiter |
| Auslöser | Projekt angelegt |
| Ergebnis | Mitarbeiter dem Projekt zugeordnet |
| Eingehende Daten | Funktion und Pensum von Mitarbeiter |
| Vorbedingungen | Mitarbeiter vorhanden |
| Ablauf | -Mitarbeiter aussuchen  -Verfügbarkeit der Mitarbeiter prüfen  -Mitarbeiter zuordnen |
| Priorität | Hoch, ein Projekt braucht Mitarbeiter |

### Projekt aktualisieren

|  |  |
| --- | --- |
| Name | Projekt aktualisieren |
| Beschreibung | Projekt mit den neusten Daten ergänzen. Hierzu gehört das verwalten der Aktivitäten, Meilensteine, Ressourcen, Dokumentreferenzen und Projektmanagement spezifische Informationen (Phasenstatus usw.). |
| Akteure | Projektleiter, Mitarbeiter |
| Auslöser | Neue Daten zum ergänzen vorhanden |
| Ergebnis | Projekt mit neuen Daten ergänzt |
| Eingehende Daten | -Aktivitäten  -Meilensteine  -Ressourcen  -Dokumentreferenzen  -Projektmanagement spezifische Informationen (Phasenstatus usw.) |
| Vorbedingungen | Projekt ist gestartet |
| Ablauf | -Aktivitäten verwalten  -Meilensteine verwalten  -Ressourcen verwalten  -Dokumentreferenzen verwalten  -Projektmanagement spezifische Informationen verwalten |
| Priorität | Hoch, da Kernprozess |

### Reporting generieren

|  |  |
| --- | --- |
| Name | Reporting generieren |
| Beschreibung | Aus den vorhandenen Projektdaten wird ein Reporting PDF generiert. Darin ist eine Übersicht mit dem aktuellen Projektstand in Zeit und Kosten Dimension ersichtlich. |
| Akteure | Projektleiter |
| Auslöser | Projektleiter muss/will einen Report erstellen |
| Ergebnis | Projektübersicht mit aktuellem Stand |
| Eingehende Daten | Alle Daten die dem Projekt zugeordnet sind. |
| Vorbedingungen | Projekt ist gestartet |
| Ablauf | -Reporting generieren  -Reporting versenden |
| Priorität | Niedrig, nice to have |

### Projekt abschliessen

|  |  |
| --- | --- |
| Name | Projekt abschliessen |
| Beschreibung | Projekt abschliessen, so dass keine Änderungen mehr möglich sind. |
| Akteure | Projektleiter |
| Auslöser | Projekt fertig |
| Ergebnis | Projekt kann nicht mehr geändert werden. |
| Eingehende Daten |  |
| Vorbedingungen | Alle Daten für das Projekt sind erfasst |
| Ablauf |  |
| Priorität | Mittel |

# Anforderungsanalyse

## Systemanwendungsfälle

### Projekt anlegen

|  |  |
| --- | --- |
| Anwendungsfall | Projekt anlegen |
| Ziel | Erfassen aller initialen Daten |
| Vorbedingungen | Projektdaten vorhanden |
| Nachbedingungen Erfolg | Projekt ist mit erforderlichen Daten angelegt |
| Nachbedingungen Fehlschlag | Der Benutzer bricht das Anlegen ab |
| Akteur | Projektleiter |
| Auslöser | Neues Projekt startet |
| Beschreibung | Erfassen der initialen Daten |
| Alternativen | - |
| Erweiterungen | - |
| Priorität | hoch |
| Häufigkeit | Niedrig, einmal pro Projekt |
| Offene Punkte | - |
| Sonstiges | - |



### Vorgehensmodell auswählen

|  |  |
| --- | --- |
| Anwendungsfall | Vorgehensmodell auswählen |
| Ziel | Ein Vorgehensmodell ist ausgewählt. |
| Vorbedingungen | Projekt vorhanden |
| Nachbedingungen Erfolg | Dem Projekt ist ein Vorgehenmodell zugewiesen |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter |
| Auslöser | Projekt angelegt |
| Beschreibung | Das gewünschte Vorgehensmodell wird ausgewählt. Ist das Modell noch nicht vorhanden wird ein neues angelegt oder ein bestehendes bearbeitet. |
| Alternativen | -Bestehendes Vorgehensmodell bearbeiten  -Neues Vorgehensmodell anlegen |
| Erweiterungen | - |
| Priorität | hoch |
| Häufigkeit | Niedrig, einmal pro Projekt |
| Offene Punkte | - |
| Sonstiges | - |



### Bestehendes Vorgehensmodell bearbeiten

|  |  |
| --- | --- |
| Anwendungsfall | Bestehendes Vorgehensmodell bearbeiten |
| Ziel | Das Vorgehensmodell ist angepasst |
| Vorbedingungen | Alle Projekte welche dieses Vorgehensmodell verwenden sind abgeschlossen. |
| Nachbedingungen Erfolg | Das Vorgehensmodell ist mit neuen Daten abgespeichert |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter |
| Auslöser | Kein Vorgehensmodell passt |
| Beschreibung | Das gewünschte Vorgehensmodell ist nicht vollständig vorhanden, deshalb wird ein bestehendes verändert. |
| Alternativen | -Neues Vorgehensmodell anlegen |
| Erweiterungen | - |
| Priorität | Niedrig, Workaround mit neuem Modell anlegen |
| Häufigkeit | Niedrig |
| Offene Punkte | - |
| Sonstiges | - |



### Neues Vorgehensmodell anlegen

|  |  |
| --- | --- |
| Anwendungsfall | Neues Vorgehensmodell anlegen |
| Ziel | Neues Vorgehensmodell ist erfasst |
| Vorbedingungen | - |
| Nachbedingungen Erfolg | Das neue Vorgehensmodell ist abgespeichert |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter |
| Auslöser | Kein Vorgehensmodell passt |
| Beschreibung | Das gewünschte Vorgehensmodell ist nicht vorhanden, deshalb wird ein neues erfasst. |
| Alternativen | - |
| Erweiterungen | - |
| Priorität | Hoch, muss zwingend einmal gemacht werden |
| Häufigkeit | Niedrig |
| Offene Punkte | - |
| Sonstiges | - |



### Projektphasen generieren

|  |  |
| --- | --- |
| Anwendungsfall | Vorgehensmodell auswählen |
| Ziel | Die Projektphasen wurden erstellt und dem Projekt zugeordnet |
| Vorbedingungen | Projekt vorhanden und Vorgehensmodell ausgewählt |
| Nachbedingungen Erfolg | Die Projektphasen sind abgespeichert |
| Nachbedingungen Fehlschlag | Ein Systemfehler tritt auf |
| Akteur | System |
| Auslöser | Vorgehensmodell ausgewählt |
| Beschreibung | Aufgrund des ausgewählten Vorgehensmodells werden die Projektphasen generiert und dem Projekt zugeordnet. Für jede Phase wird ein Meilenstein erstellt welcher mit dem Phasenende verknüpft wird. |
| Alternativen | - |
| Erweiterungen | - |
| Priorität | Hoch, muss zwingend einmal gemacht werden |
| Häufigkeit | Niedrig, einmal pro Projekt |
| Offene Punkte | - |
| Sonstiges | - |



### Mitarbeiter auswählen

|  |  |
| --- | --- |
| Anwendungsfall | Mitarbeiter auswählen |
| Ziel | Mitarbeiter ist ausgewählt |
| Vorbedingungen | Projekt vorhanden |
| Nachbedingungen Erfolg | Mitarbeiter ist dem Projekt zugewiesen |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter |
| Auslöser | Projekt angelegt |
| Beschreibung | Der gewünschte Mitarbeiter wird ausgewählt. Ist dieser noch nicht vorhanden wird ein neuer angelegt oder ein bestehender bearbeitet. |
| Alternativen | - |
| Erweiterungen | - |
| Priorität | Hoch, muss zwingend einmal gemacht werden |
| Häufigkeit | Niedrig |
| Offene Punkte | - |
| Sonstiges | - |



### Mitarbeiter anlegen

|  |  |
| --- | --- |
| Anwendungsfall | Mitarbeiter anlegen |
| Ziel | Neuer Mitarbeiter ist erfasst |
| Vorbedingungen | - |
| Nachbedingungen Erfolg | Der neue Mitarbeiter ist abgespeichert |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter |
| Auslöser | Mitarbeiter ist nicht vorhanden |
| Beschreibung | Der gewünschte Mitarbeiter ist nicht vorhanden, deshalb wird er erfasst. |
| Alternativen | - |
| Erweiterungen | - |
| Priorität | Hoch, muss zwingend einmal gemacht werden |
| Häufigkeit | Niedrig |
| Offene Punkte | - |
| Sonstiges | - |



### Mitarbeiter bearbeiten

|  |  |
| --- | --- |
| Anwendungsfall | Mitarbeiter bearbeiten |
| Ziel | Mitarbeiter ist angepasst |
| Vorbedingungen | Mitarbeiter vorhanden |
| Nachbedingungen Erfolg | Der Mitarbeiter ist mit den neuen Daten abgespeichert |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter |
| Auslöser | Mitarbeiter ist nicht aktuell |
| Beschreibung | Der Mitarbeiter ist mit alten oder falschen Daten vorhanden, deshalb wird er angepasst. |
| Alternativen | Neuen Mitarbeiter anlegen |
| Erweiterungen | - |
| Priorität | Niedrig, Workaround mit neuem Mitarbeiter anlegen |
| Häufigkeit | Niedrig |
| Offene Punkte | - |
| Sonstiges | - |



### Aktivitäten verwalten

|  |  |
| --- | --- |
| Anwendungsfall | Aktivitäten verwalten |
| Ziel | Aktivitäten sind abgespeichert |
| Vorbedingungen | Projektphase vorhanden |
| Nachbedingungen Erfolg | Die Aktivitäten sind mit den aktuellen Daten abgespeichert. |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter oder Mitarbeiter |
| Auslöser | Neue Aktivität muss erfasst werden oder bestehende muss bearbeitet werden. |
| Beschreibung | Eine neue Aktivität wird erfasst oder eine bestehende bearbeitet. |
| Alternativen | - |
| Erweiterungen | - |
| Priorität | Hoch, Kernprozess |
| Häufigkeit | Hoch |
| Offene Punkte | - |
| Sonstiges | - |



### Meilensteine verwalten

|  |  |
| --- | --- |
| Anwendungsfall | Meilensteine verwalten |
| Ziel | Meilensteine sind abgespeichert |
| Vorbedingungen | Projektphase vorhanden |
| Nachbedingungen Erfolg | Die Meilensteine sind mit den neuen Daten abgespeichert |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter |
| Auslöser | Neuer Meilenstein muss erfasst werden oder bestehender muss bearbeitet werden. |
| Beschreibung | Für eine Projektphase wird ein neuer Meilenstein wird erfasst oder eine bestehender bearbeitet. Fixe Meilenstein können nicht bearbeitet werden. |
| Alternativen | - |
| Erweiterungen | - |
| Priorität | Mittel |
| Häufigkeit | Niedrig |
| Offene Punkte | - |
| Sonstiges | - |



### Ressourcen verwalten

|  |  |
| --- | --- |
| Anwendungsfall | Ressourcen verwalten |
| Ziel | Ressourcen sind abgespeichert |
| Vorbedingungen | Aktivitäten sind vorhanden |
| Nachbedingungen Erfolg | Die Ressourcen sind mit den neuen Daten abgespeichert |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter oder Mitarbeiter |
| Auslöser | Neue Ressource muss erfasst werden oder bestehende muss bearbeitet werden. |
| Beschreibung | Für eine Aktivität wird eine Ressource bearbeitet. Die Ressource beinhaltet entweder Mitarbeiteraufwände oder externe Kosten. |
| Alternativen | - |
| Erweiterungen | - |
| Priorität | Hoch, Kernprozess |
| Häufigkeit | Hoch |
| Offene Punkte | - |
| Sonstiges | - |



### Dokumentreferenzen verwalten

|  |  |
| --- | --- |
| Anwendungsfall | Dokumentreferenzen verwalten |
| Ziel | Dokumentreferenzen sind abgespeichert |
| Vorbedingungen | Projekt, Phase oder Aktivität vorhanden |
| Nachbedingungen Erfolg | Die Dokumentreferenzen sind mit den neuen Daten abgespeichert |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter oder Mitarbeiter |
| Auslöser | Dokumentreferenz muss erfasst oder bearbeitet werden |
| Beschreibung | Für ein Projekt, Phase oder eine Aktivität wird eine Dokumentreferenz erfasst oder bearbeitet. |
| Alternativen | - |
| Erweiterungen | - |
| Priorität | Mittel |
| Häufigkeit | Mittel |
| Offene Punkte | - |
| Sonstiges | - |



### Projektmanagement spezifische Informationen verwalten

|  |  |
| --- | --- |
| Anwendungsfall | Projektmanagement spezifische Informationen verwalten |
| Ziel | Projekt mit aktuellen Daten abgespeichert |
| Vorbedingungen | Projekt vorhanden |
| Nachbedingungen Erfolg | Das Projekt ist mit den neuen Daten abgespeichert |
| Nachbedingungen Fehlschlag | Der Benutzer bricht den Vorgang ab |
| Akteur | Projektleiter |
| Auslöser | Projekt ist nicht mehr aktuell |
| Beschreibung | Das Projekt wird mit den neusten Daten aktualisiert. |
| Alternativen | - |
| Erweiterungen | - |
| Priorität | Mittel |
| Häufigkeit | Niedrig |
| Offene Punkte | - |
| Sonstiges | - |



# Design

## Architektur

Die Architektur richtet sich stark nach den Model View Controller Prinzip (MVC).

Benutzt

Verändert

Benutzer

Sieht

Stellt Daten zur Verfügung

Weitere Details zu den drei Schichten sind in den folgenden Kapiteln erläutert.

## Model

Das Model hat die Verantwortlichkeit der Daten und der Geschäftslogik. Es besteht im Wesentlichen aus zwei Arten von Klassen.

* Fachklassen
  + ProcessModel
  + Employee
  + Project
  + Phase
  + Milestone
  + Activity
  + Resource
  + PersonalResource
  + FinanceResource
  + DocumentReference
* Hilfsklassen
  + ISummedResources
  + ITimeBoxed
  + PhaseChild
  + ResourceCollector
  + TimeBoxedData

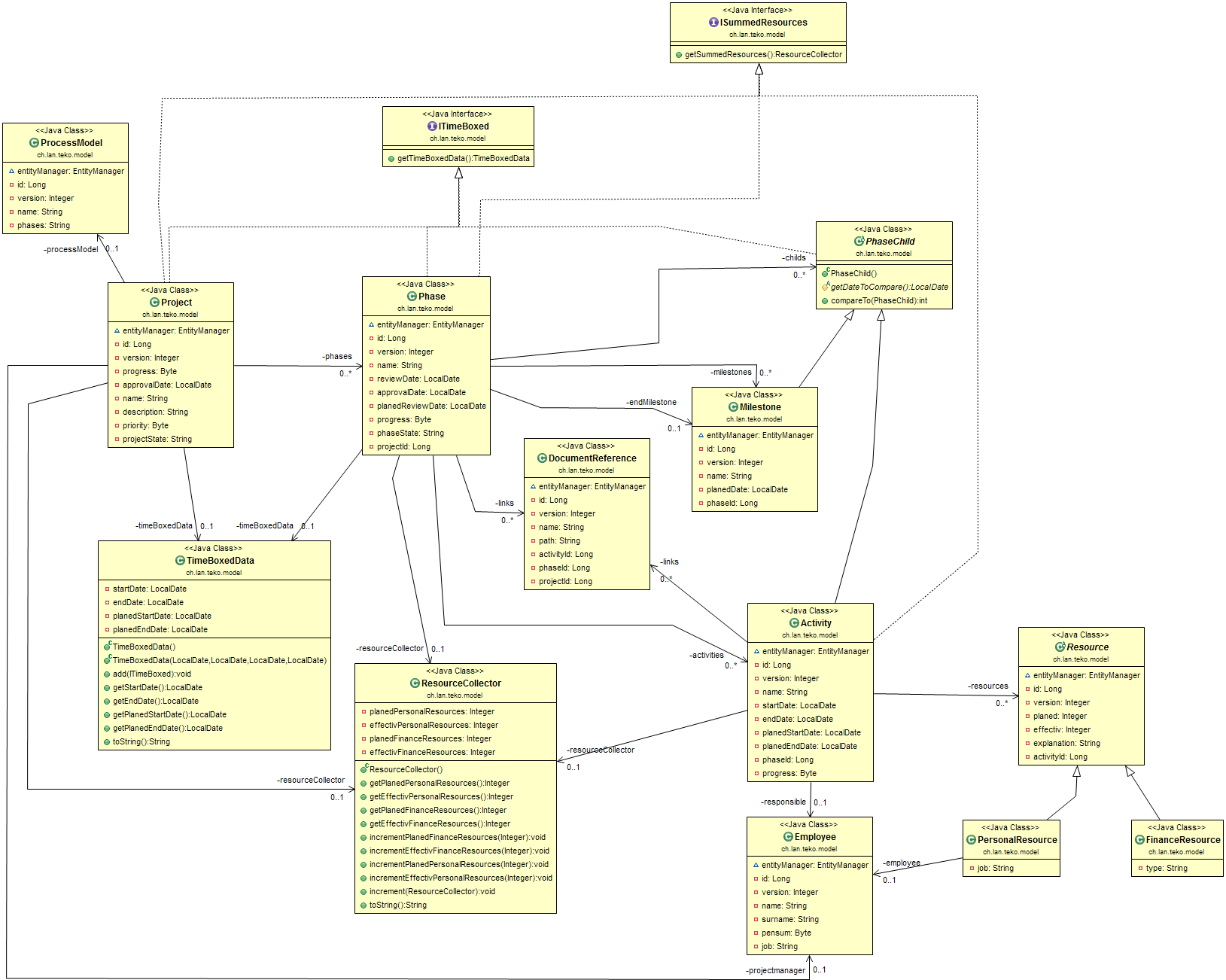
Die Fachklassen machen den deutlich grösseren Anteil aus. Sie haben alle gemeinsam, dass sie direkt in der Datenbank abgespeichert werden. Deshalb besitzen alle einen EntityManager welcher die Verbindung zur Datenbank darstellt. Er befindet sich in „javax.persistence.EntityManager“ und kommt aus dem Plugin „hibernate-jpa-2.1-api“. Eine detailliert Beschreibung findet man unter: <http://docs.oracle.com/javaee/6/api/javax/persistence/EntityManager.html>. Um die entsprechen Fachklassen in der Datenbank zu speichern haben alle folgende drei Methoden:

* persist()
* merge()
* remove()

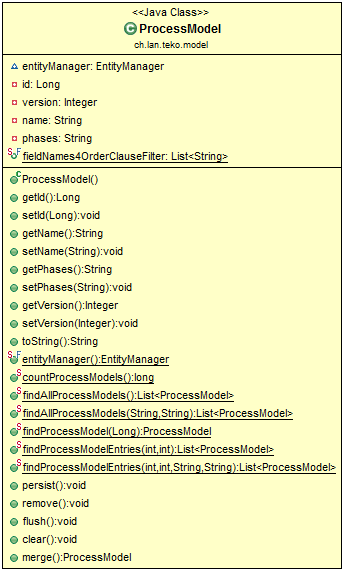
Diese speichert, aktualisieren oder löschen die Fachklassen aus der Datenbank. Anders als diese drei Methoden sind die Methoden um die Fachklassen aus der Datenbank zu laden alle statisch. Dies macht auch Sinn, weil man das entsprechende Objekt ja erst laden will. Wegen der Verbindung zur Datenbank haben alle Fachklassen eine Id, eine Version und eine statische Liste. Die Id representiert der Primary-Key. Die Version wird vom EntityManager für ein Optimistic Locking verwendet (<https://de.wikipedia.org/wiki/Optimistic_Concurrency>). Die statische Liste wird für die Reihenfolge bei einer Sortierung verwendet.

Wie der Name schon sagt unterstützen die Hilfsklassen, die Fachklassen bei der Arbeit.

Folgendes Klassendiagramm zeigt alle Klassen. Für die Übersichtlichkeit wurden bei den Fachklassen die Methoden weggelassen. Diese sind im entsprechenden Kapiteln der Klassen einzeln ersichtlich.

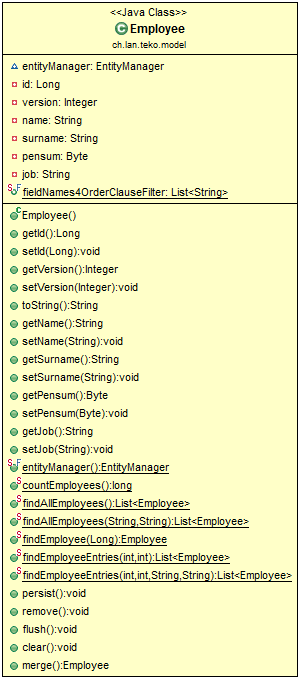


### ProcessModel



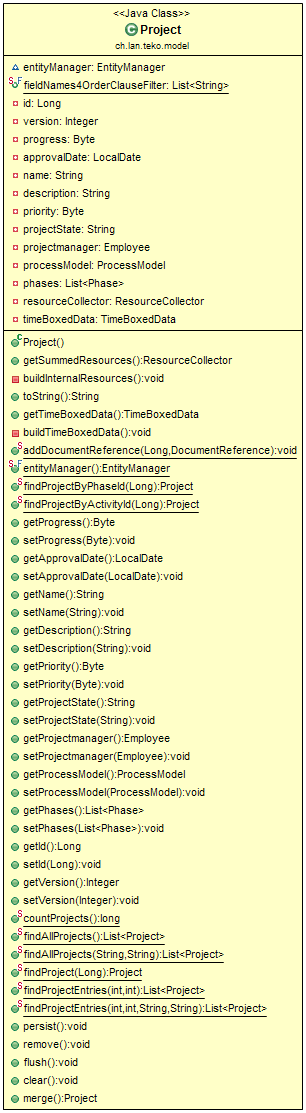
Hiermit wird ein Vorgehensmodell abgebildet. Dadurch hat sie einen Namen und Phasen. Die Phasen werden zu diesen Zeitpunkt in einem String gespeichert, getrennt durch ein Semicolon.

### Employee



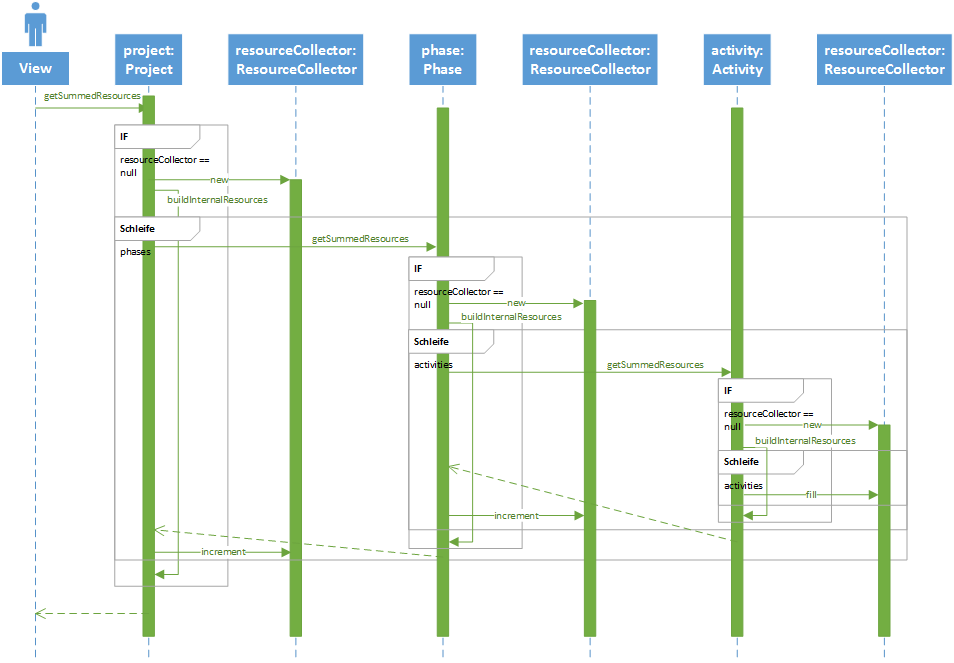
Hiermit wird ein Mitarbeiter abgebildet. Namen und Pensum dürften klar sein. Mit Job ist die Funktion gemeint, function kann man aber nicht mit allen Datenbanken verwenden.

### Project

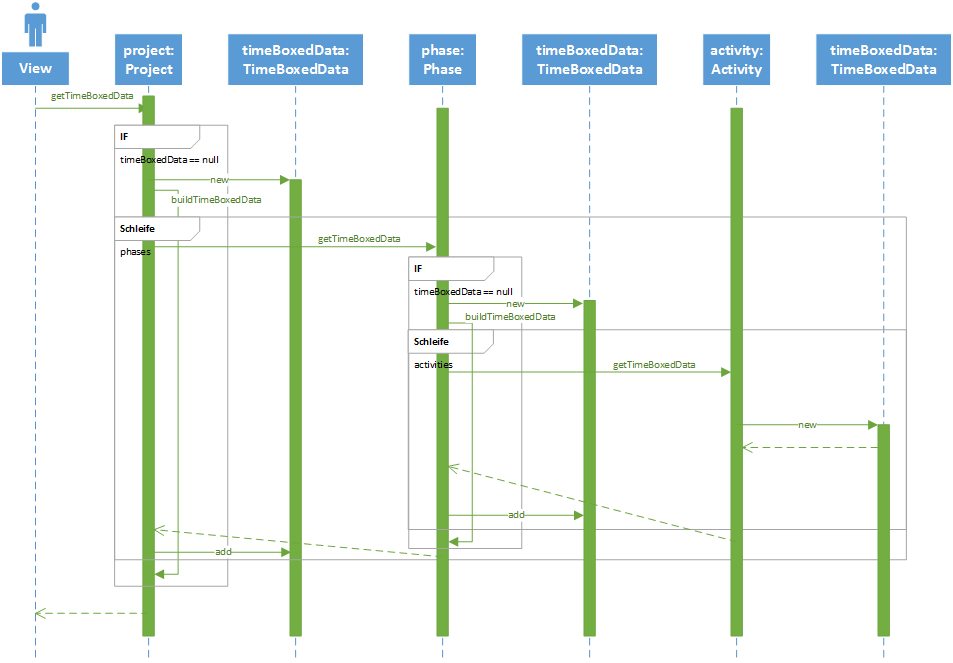


Hiermit wird ein Projekt abgebildet. Die meisten Attribute dürften klar sein, denn sie wurden aus den Anforderungen abgeleitet. Richtig interessant sind resourceCollector und timeBoxedData. Beide werden nicht in die Datenbank gespeichert sondern zur Laufzeit berechnet. In resourceCollector werden die Aufwände (PersonalResource & FinanceResource) für das gesamte Projekt zusammen gezählt. In timeBoxedData wird die Dauer des Projekts berechnet. Wie das gemacht wird zeigen folgende Sequenzdiagramme.

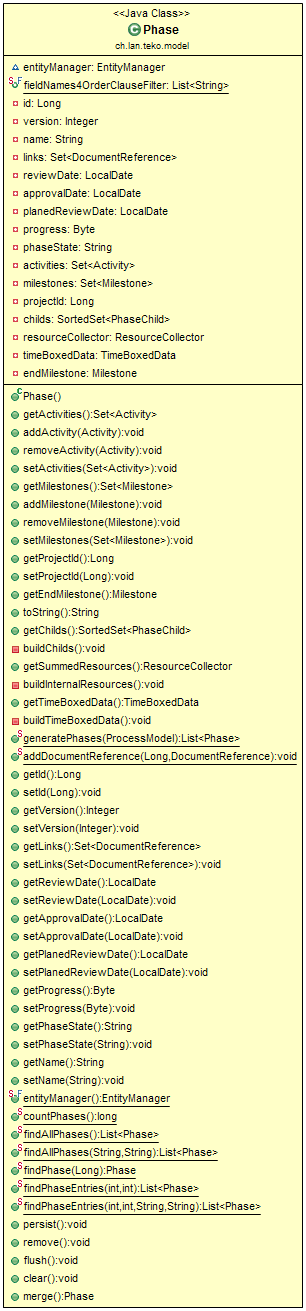
**Aufwände sammeln**



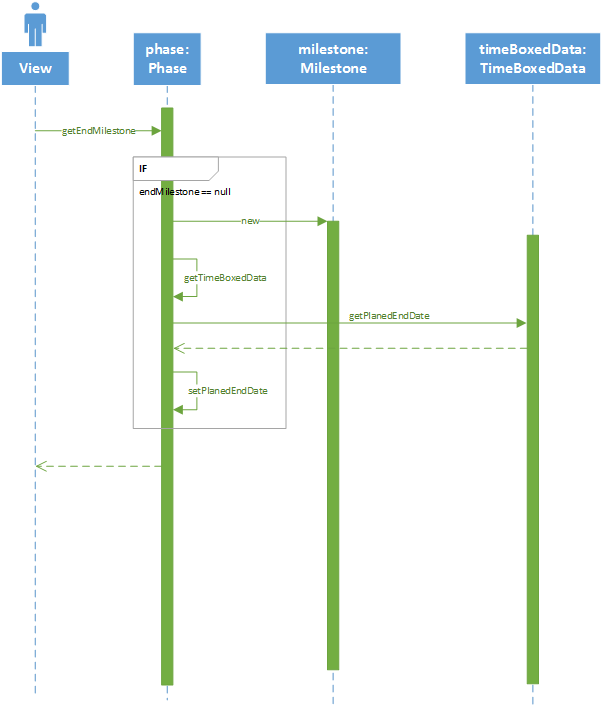
**Dauer berechnen**



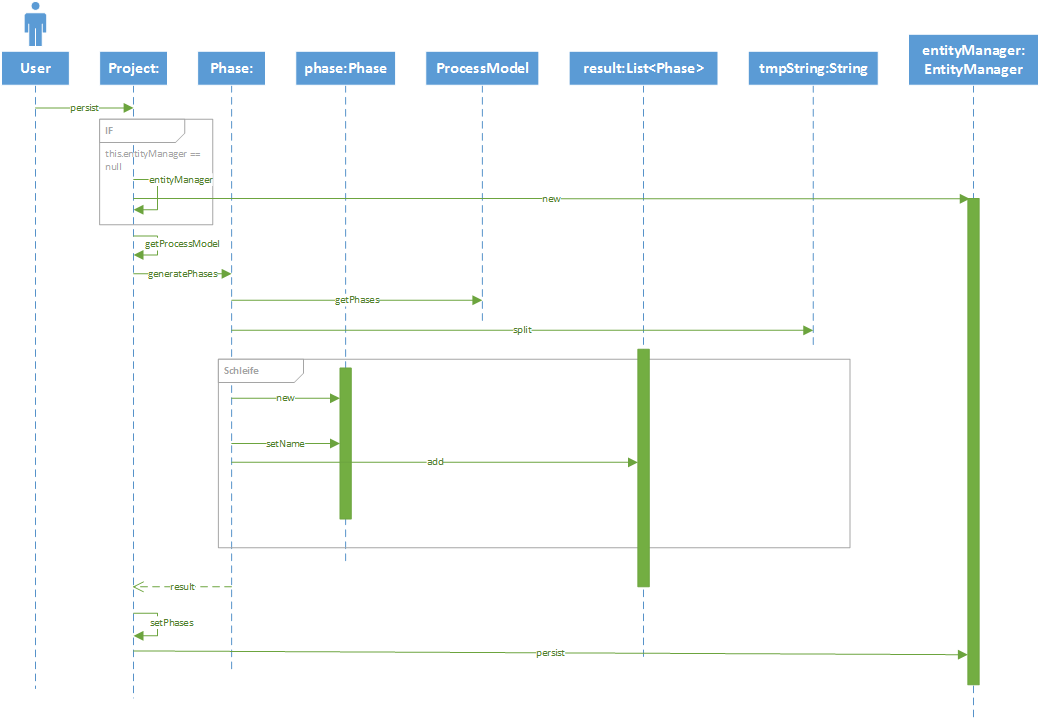
### Phase



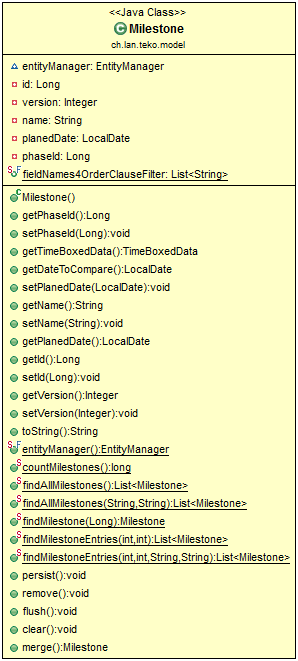
Hiermit wird eine Projektphase abgeleitet. Die meisten Attribute sind auch hier aus der Anforderungen abgeleitet und werden hier nicht näher beschrieben. Da auch die Phase eine Sammlung an Aufwänden und eine Dauer hat, findet man hier ebenfalls resourceCollector und timeBoxedData. Ihr Zweck und Verhalten ist gleich wie beim Projekt, nur halt auf Ebene der Phase. Erwähnenswert ist der endMilestone. Er wird zur Laufzeit berechnet und nicht in der Datenbank gespeichert. Sein Zweck ist es die Anforderung zu erfüllen, dass jede Phase mit einem Meilenstein endet. Sobald genügend Daten vorhanden sind um eine Dauer zu berechnen wird ein solcher Meilenstein erstellt. Hierzu folgendes Sequenzdiagramm.



Weiter gibt es eine statische Methode generatePhases() welche ein ProcessModel erwartet. Diese wird verwendet um aufgrund eines Vorgehensmodells Phase zu erstellen. Dies geschieht immer beim Anlegen eins Projekts wie folgendes Sequenzdiagramm zeigt.

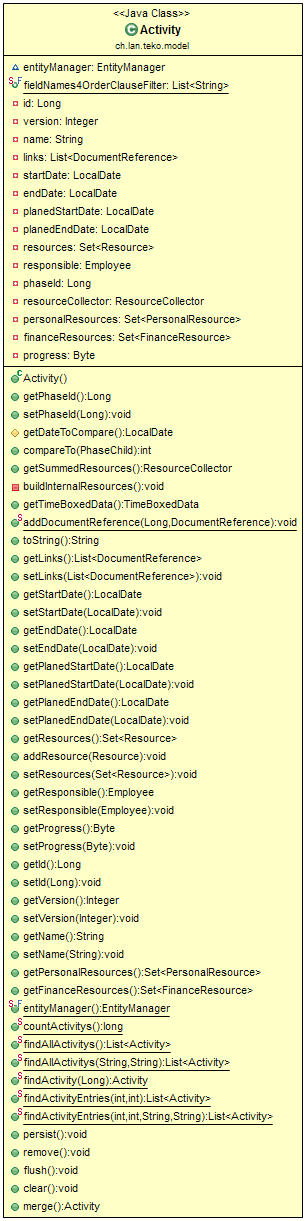


### Milestone



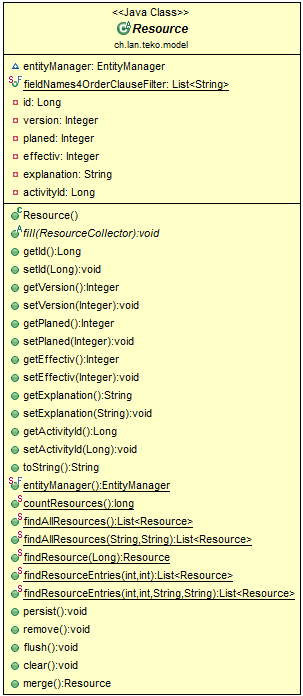
Hiermit wird ein Meilenstein abgebildet. Diese Klasse ist genauso wie Activity von PhaseChild abgeleitet. Deshalb muss getDateToCompare() implementiert werden. Wozu die Klasse PhaseChild ist, steht im entspechenden Kapitel. Hier wird getDateToCompare() so implementiert, dass planedDate zurückgegeben wird.

### Activity



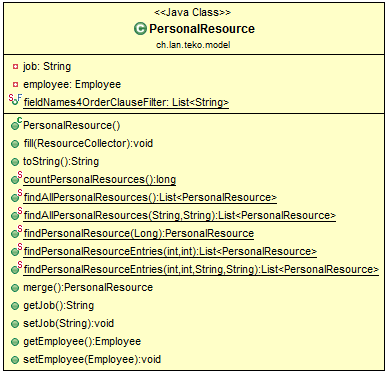
Hiermit wird eine Aktivität abgebildet. Die zwei Arten von Resourcen, PersonalResource und FinanceResource, werden über die gemeinsame Super-Klasse in einer HashSet gespeichert. Das sah beim Design gut aus bereitete in der UI aber Probleme. Deshalb gibt es zwei zusätzliche Sets (personalResources und financeResource) beide werden erst zur Laufzeigt gefüllt und nicht abgespeichert. Würde das Projekt länger gehen wäre dies eine der ersten Stellen die umgebaut würden. Die getDateToCompare() Methode wird hier so implementiert, dass planedStartDate zurückgegeben wird. Mehr dazu unter PhaseChild.

### Resource



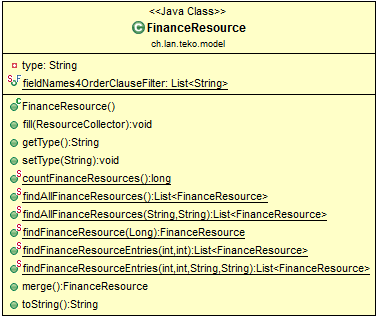
Hiermit werden die Gemeinsamkeiten von PersonalResource und FinanceResource abgebildet. Auffällig ist die abstrakte Methode fill() welche einen ResourceCollector erwartet. Damit wird sichergestellt, dass Aufwände von Ableitungen immer über einen ResourceCollector gesammelt werden können.

### PersonalResource



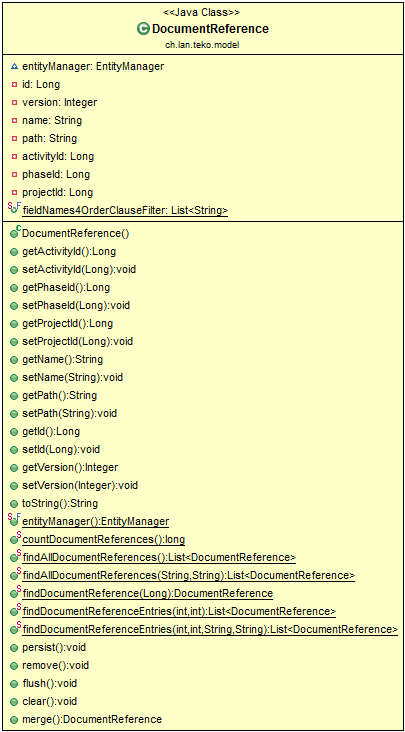
Hiermit werden Personenaufwände abgebildet. Die fill() Methode ist so implementier, dass die Personenaufwände spezifisches Daten im ResourceCollector gefüllt werden.

### FinanceResource



Hiermit werden Externe Kosten abgebildet. Die fill() Methode ist so implementiert, dass die „Externen Kosten Aufwände“ spezifischen Daten im ResourceCollector gefüllt werden.

### DocumentReference



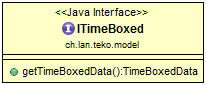
Hiermit werden die Dokumentreferenzen abgebildet. Diese Klasse hat nichts ausgewöhnliches, was Gelegenheit bietet die Methoden flush() und clear() zu erläutern. Man findet diese auch in einigen andern Fachklassen. Bei flush() handelt es sich um das Pendant zu einem Commit in einer Datenbank. Clear() hingegen löscht alle Änderungen welche nicht mit einem flush() zur Datenbank gesendet worden sind. Ausserdem gibt es drei Attribute activityId, phaseId und projectId auch diese findet man in anderen Fachklassen. Wenn man genauer hinsieht verbirgt sich dahinter immer die entgegengesetzte Richtung einer OneToMany Beziehung. Da die DocumentReference in Project, Phase und Activity eine ManyToOne Beziehung hat, sind hier alle drei vertreten.

### ISummedResources



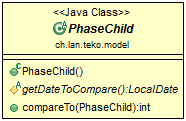
Die erste Hilfsklasse in der Aufzählung, wobei eigentlich ist es nur ein Interface. Alle Ableitungen sollen einen ResourceCollector zur Verfügung stellen. Dieser kann bereits gefüllt sein.

### ITimeBoxed



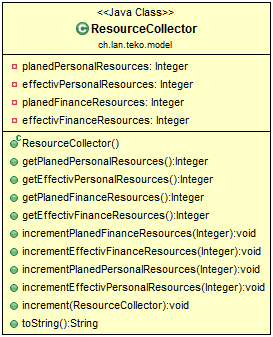
Zweites „Hilfs-Interface“. Alle Ableitungen sollen ein TimeBoxedData zur Verfügung stellen. Dieser kann bereits gefüllt sein.

### PhaseChild



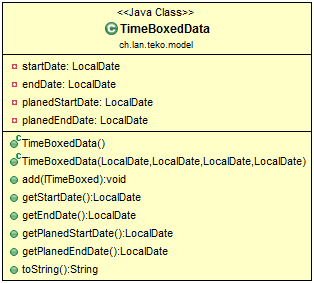
Hilfsklasse für die Sortierung von „Kinder“ einer Phase (bisher Milestone und Activity). Der Inhalt einer Phase soll im UI nach einem Datum sortiert angezeigt werden. Dazu implementiert diese Klasse das Comparable Interface und verlangt von seinen Ableitungen über getDateToCompare() ein Datum zum Sortieren.

### ResourceCollector



Hilfsklasse um die Aufwände zusammen zu zählen. Die Attribute entsprechen den möglichen Aufwänden einmal als Geplant und einmal als Effektiv. Die Aufwände können entweder separat über die increment…() Methoden hochgezählt werden oder es werden gleich die Aufwände von einem anderen ResourceCollector hinzugefügt. So ist es einfach möglich die Aufwände für ein gesamtes Projekt zu sammeln. Siehe das „Aufwände sammeln“ Sequenzdiagramm unter Project.

### TimeBoxedData

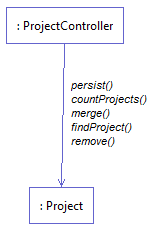


Hilfsklasse um die Dauer zu berechnen. Die Attribute zeigen eine Geplante und eine Effektive Dauer. Die Attribute können entweder mit Hilfe des Konstruktors gesetzt werden oder über die add() Methode. Die add() Methode nimmt jeweils das frühste Start und späteste End Datum und merkt sich diese. Damit ist es ähnlich wie beim ResourceCollector einfach die gesamte Dauer eines Projekts zu berechnen. Siehe das „Dauer berechnen“ Sequenzdiagramm unter Project.

## Controller

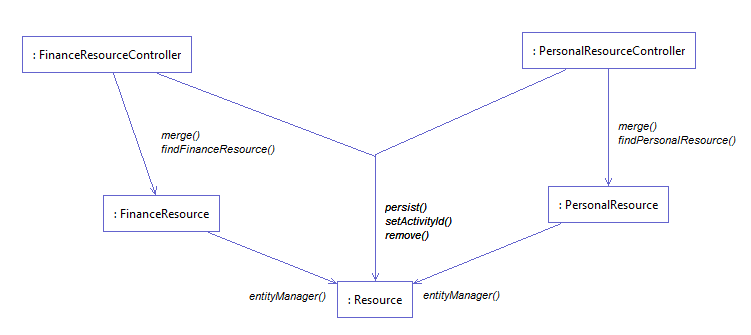
Die Verantwortung der Controller besteht darin eine Benutzer Anfrage entgegen zu nehmen, im Model die entsprechende Logik auf zu rufen und dann auf eine View zu verweisen.

Grundsätzlich gibt es für jede Fachklasse einen Controller welcher die Create, Read, Update und Delete (CRUD) Funktionen bereitstellt. Hier stellvertretend mit dem Project als Kommunikationsdiagramm dargestellt.



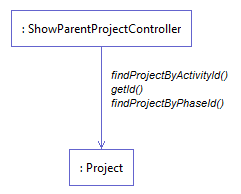
So oder so ähnlich sehen alle Diagramme zwischen Controller und Fachklasse aus. Es gibt jedoch zwei besondere Fälle, die hier weiter beschrieben werden.

Für PersonalResource und FinanceResource wird zum Anlegen und Löschen nicht die entsprechende Fachklasse aufgerufen sondern nur Resource. Folgendes Diagramm zeigt dies.



Man sieht auch gleich, dass sich beide Fachklassen den EntitiyManager teilen und somit auch die Controller gleich auf Resource zugreifen können.

Der ShowParentProjectController wird verwendet um in der UI einen Link auf das übergeordnete Projekt bereit zu stellen. Deshalb such er sich je nach Aufrufer das Projekt und macht eine Weiterleitung an den ProjectController. Folgendes Diagramm zeigt die möglichen Such-Aufrufe auf das Projekt.



## View

## Datenbank

Für den Prototypen wird auf eine In-Memory Datenbank gesetzt. Das Mapping zwischen Java und der Datenbank wird mit Hibernate und JPA realisiert. Hibernate und JPA übernehmen weiter auch den Aufbau des Datenbankschemas, deshalb wird es hier nicht näher beschrieben. Bei einer Änderung des Datenmodells wird die Datenbank automatisch angepasst. Allfällige Datenmigrationen werden nicht unterstützt.

# Qualitätsmanagement

## Testkonzept

### Ziel

Ziel eines Tests ist es z.B. einen Anwendungsfall so zu prüfen, dass man davon ausgehen kann, dass er unter allen Voraussetzungen einwandfrei funktioniert.

### Testarten

|  |  |  |
| --- | --- | --- |
| Nr. | Testart | Beschreibung |
| 1 | Unittests | Für jede Klasse wird ein Unittest erstellt. Ausnahmen bilden nur Klassen die keine Logik beinhalten z.B. einfach Container-Klassen. Im Unittest wird nur die Funktionen der zu testenden Klasse überprüft, andere benötigten Funktionen werden gemockt. |
| 2 | Integrationstests | Alle Klassen für eine bestimmte Funktion werden zusammen getestet. |
| 3 | Systemtests | Die gesamte Applikation wird automatisierten Tests überprüft. Dabei startet der Tests einen Webserver und prüft dessen ausgaben. Alle Testfälle werden als automatisierte Tests angelegt. |
| 4 | Manuelle Tests nach Testplan | Die Testfälle werden von einem Tester ausgeführt. |

### Fehlerklassen

|  |  |  |
| --- | --- | --- |
| Nr. | Fehlerklasse | Beschreibung |
| 1 | Fehlerfrei | Es wurden keine Fehler entdeckt. Das getestete Modul bzw. System arbeitet 100% fehlerfrei. |
| 2 | Minor | Es wurden Fehler entdeckt, welche aber den Betrieb der Software nicht stören. Fehler können z. B. Schreibfehler sein. |
| 3 | Major | Es wurden Fehler entdeckt, welche noch nicht als kritisch angesehen werden müssen. Diese Fehler können aber nicht bestehen und müssen in absehbarer Zeit behoben werden. |
| 4 | Critical | Es wurden Fehler entdeckt, welchen den Betrieb der Software nicht mehr sicherstellen. Es sind Fehler die nicht heute auf morgen gelöst werden können und nochmals eine intensive Planung im fehlerhaften Modul herbeiführen. |
| 5 | Blocker | Es wurden Fehler entdeckt, welche den Betrieb der Software verunmöglichen. Solche Fehler müssen mit grösster Priorität angegangen werden. Sie können das unter Umständen das ganze Projekt in Gefahr bringen. |

### Testfälle

#### Mitarbeiter verwalten

|  |  |
| --- | --- |
| Id / Titel | T-01 / Mitarbeiter verwalten |
| Beschreibung | Ein Mitarbeiter mit folgenden Daten anlegen:  Den Namen in T01 ändern |
| Testvoraussetzung | Applikation ist gestartet |
| Testschritte | -Ansicht: neuer Mitarbeiter öffnen  -Daten eingeben  -Mitarbeiter speichern  -Daten überprüfen  -Namen ändern  -Speichern  -Namen überprüfen |
| Erwartetes Ergebnis | Mitarbeiter wird mit eingegebenen Daten angezeigt. Der geänderte Name wurde übernommen. |

#### Vorgehensmodell anlegen

|  |  |
| --- | --- |
| Id / Titel | T-02 / Vorgehensmodell anlegen |
| Beschreibung | Ein Vorgehensmodell mit folgenden Daten anlegen:  Titel: VM01  Phasen: P1, P2, P3  Den Titel in T02 ändern. |
| Testvoraussetzung | Applikation ist gestartet |
| Testschritte | -Ansicht: neues Vorgehensmodell öffnen  -Daten eingeben  -Vorgehensmodell speichern  -Daten überprüfen  -Titel ändern  -Speichern  -Titel überprüfen |
| Erwartetes Ergebnis | Vorgehensmodell wird mit eingegebenen Daten angezeigt. Der geänderte Titel wurde übernommen. |

#### Projekt verwalten

|  |  |
| --- | --- |
| Id / Titel | T-03 / Projekt verwalten |
| Beschreibung | Ein Projekt mit folgenden Daten anlegen:  Den Namen in T03 ändern. |
| Testvoraussetzung | Applikation ist gestartet  Ein Mitarbeiter und ein Vorgehensmodell sind erfasst |
| Testschritte | -Ansicht: neues Projekt öffnen  -Daten eingeben  -Projekt speichern  -Daten überprüfen  -Namen ändern  -Speichern  -Namen überprüfen |
| Erwartetes Ergebnis | Projekt wird mit eingegebenen Daten angezeigt. Der geänderte Name wurde übernommen. |

#### Aktivität verwalten

|  |  |
| --- | --- |
| Id / Titel | T-04 / Aktivitäten verwalten |
| Beschreibung | Eine Aktivität mit folgenden Daten anlegen:  Den Titel in T04 ändern |
| Testvoraussetzung | Ein Projekt ist erfasst |
| Testschritte | -Ansicht: neue Aktivität anlegen  -Daten eingeben  -Aktivität speichern  -Daten überprüfen  -Titel ändern  -Speichern  -Titel überprüfen |
| Erwartetes Ergebnis | Aktivität wird mit eingegebenen Daten angezeigt. Der geänderte Titel wurde übernommen. |

#### Dokumentreferenzen verwalten

|  |  |
| --- | --- |
| Id / Titel | T-05 / Dokumentreferenzen anlegen |
| Beschreibung | Eine Dokumentreferenz mit folgenden Daten anlegen:  Name: Ref01  Pfad: C:/tmp/doc.txt  Den Namen in T05 ändern.  Die Dokumentenreferenz löschen. |
| Testvoraussetzung | Ein Projekt ist erfasst |
| Testschritte | -Ansicht: neue Dokumentreferenz  -Daten eingeben  -Dokumentreferenz speichern  -Daten überprüfen  -Namen ändern  -Speichern  -Namen überprüfen  -Löschen  -Prüfen ob die Referenz nicht mehr vorhanden ist |
| Erwartetes Ergebnis | Dokumentenreferenz wird mit eingegebenen Daten angezeigt. Der geänderte Name wurde übernommen. Nach dem Löschen ist der Meilenstein nicht mehr vorhanden. |

#### Meilensteine verwalten

|  |  |
| --- | --- |
| Id / Titel | T-06 / Meilensteine anlegen |
| Beschreibung | Ein Meilensteine mit folgenden Daten anlegen:  Name: M01  Phase: P2  Datum: ein Tag vor dem Ende von P2  Den Namen in T06 ändern.  Meilenstein löschen |
| Testvoraussetzung | Eine Phase P2 ist erfasst. Sie dauert min 2 Tage. |
| Testschritte | -Ansicht: neuer Meilenstein anlegen  -Daten eingeben  -Meilenstein speichern  -Daten überprüfen  -Namen ändern  -Speichern  -Namen überprüfen  -Löschen  -Prüfen ob Meilenstein nicht mehr vorhanden ist |
| Erwartetes Ergebnis | Meilenstein wird mit eingegebenen Daten angezeigt. Der geänderte Name wurde übernommen. Nach dem Löschen ist der Meilenstein nicht mehr vorhanden. |

### Testprotokolle

#### Testprotokoll vom xx.xx.16

|  |  |
| --- | --- |
| Testfall Id |  |
| Resultat |  |
| Bemerkungen |  |
| Fehlerklasse |  |

# Konfigurationsmanagement

## Hardware

Die Dokumentation und Applikation werden auf einem HP-Laptop erstellt. Dieser gehört dem Projektleiter und hat folgende Eckdaten:

## Software

### Dokumentation

Für die Dokumentation wird auf die Microsoft Produkte Word und Ecxel gesetzt. Für Uml-Diagramme wird die kostenlose Software AgroUml verwendet.

### Softwareentwicklung

Für die Softwareentwicklung wird die SpringSource Toolsuite eingesetzt. Diese beinhaltet neben diversen kleinen Plugins vor allem Spring Roo welches das Erstellen einer Webapplikation erleichtert. Hier eine Übersicht aller verwendeten Software von Dritt-Anbietern.

|  |  |  |
| --- | --- | --- |
| Plugin | Anbieter | Zweck |
|  |  |  |

## Daten

Zur Versionsverwaltung von Source-Code und Dokumentation wird Git eingesetzt. Dazu wurde auf github.com ein entsprechendes Repository eröffnet.

Die Daten werden somit auf dem oben erwähnten Laptop und auf den Servern von github.com gesichert.

# Änderungsmanagement

# Anhang