

# kcare - Konzept

**Team:**

Jäger Daniel: (*csat2404*)

Kuen Dominik: (*csat2284*)

Lupoaie Valerica: (*csar6357*)

Perteneder Thomas: (*csam2910*)

Schmidinger Lukas: (*csat2148*)

**Proseminar-Gruppe:**

2

**Team:**

3

**Datum:**

30. März 2017

# Inhalt

---

<b>1 Systemüberblick</b>	<b>3</b>
<b>2 Use Cases</b>	<b>4</b>
2.1 Akteure	4
2.1.1 Supervisor	4
2.1.2 Parents	4
2.1.3 Administratoren	4
2.2 Use-Cases - Übersicht	5
2.2.1 Supervisor	5
2.2.2 Parents	5
Administrator	6
2.3 Use-Case Diagramm	6
<b>3 Klassendiagramm</b>	<b>7</b>
<b>4 Softwarearchitektur</b>	<b>8</b>
4.1 Komponentendiagramm	8
4.2 Beschreibung ausgewählter Technologien	9
<b>5 GUI Prototyp</b>	<b>11</b>
5.1. Login Page	11
5.2. Kinderkrippe Personal Welcome Page	11
5.3. Kinderkrippe Personal Welcome Page (2)	12
5.4. Eltern Welcome Page	12
5.5. Eltern Welcome Page (2)	13
<b>6 Projektplanung</b>	<b>14</b>
6.1 Planung des Projekt-Aufbaus	14
6.1.1 Aufstellung der Akteure	14
6.1.2 Ausarbeitung der Use-Cases	14
6.1.3 Erstellung eines abstrakten Klassen- und Komponentendiagramms	14
6.1.4 Auflistung aller Aufgaben	15
6.1.5 Aufteilung der Aufgaben auf einzelne Verantwortungsträger	15
6.1.6 Erstellung eines Projekt-Konzepts sowie eines Projekthandbuchs	15
6.2 Planung Projektentwicklung	16
6.2.1 Planung Basisfunktionalitäten	16
6.2.2 Planung der Weiterentwicklung von Basisfunktionalitäten	16
6.3 Meilensteine des Projekts	17

# 1 Systemüberblick

---

Das Ziel des Projekts ist es eine webbasierte Verwaltungssoftware für Kinderkrippen zu entwickeln. Es soll von allen modernen Endgeräten (PC, Tablet, Smartphone) darauf zugegriffen werden können.

Die Eltern der Kinder können sich über die Seite anmelden um beispielsweise aktuelle Nachrichten abzurufen, ihnen zugewiesene Jobs anzuzeigen oder um mit anderen Eltern und dem Kinderkrippen-Personal zu kommunizieren. Die Angestellten der Kinderkrippe können über die Seite die Maximalbelegung an Kindern für jeden Wochentag einstellen, da diese von der Anzahl der Betreuer am jeweiligen Tag abhängt.

Neue Eltern und Kinder werden ausschließlich von den Angestellten registriert. Zusätzlich zu den Eltern können weitere Kontakte angegeben werden, die befugt sind die Kinder von der Kinderkrippe abzuholen. Diese müssen dem Personal persönlich vorgestellt werden um freigeschaltet zu werden. Eltern können sowohl ihre Daten, als auch die ihrer Kinder jederzeit online bearbeiten.

Im Tagesplaner informiert das Personal unter anderem über die für heute angemeldeten Kinder, die jeweiligen Bring- und Abholzeiten und wer sie abholt. Der Plan kann auch als PDF exportiert bzw. ausgedruckt werden. In eingeschränkter Form ist auch eine Wochen-, Monats- und Jahresübersicht über den Tagesplaner möglich. Ebenfalls im Tagesplaner enthalten sind (geplante) Abwesenheiten einzelner Kinder.

# 2 Use Cases

---

## 2.1 Akteure

Zunächst soll kurz auf das Profil der Benutzergruppen, welche die Applikation später auch tatsächlich in der Praxis einsetzen werden, eingegangen werden.

### 2.1.1 Supervisor

Bei diesem Akteur handelt es sich um das Kinderkrippen-Personal, also um jene Personengruppe, welche die entwickelte Applikation am intensivsten nutzen wird. Aufgrund dessen verfügt dieser Akteur auch über das größte Repertoire an Funktionen (siehe Use-Case-Diagramm). Viele der administrativen und organisatorischen Arbeiten, die im Zusammenhang mit einer Kinderkrippe entstehen, können mit Hilfe der Applikation erledigt werden.

### 2.1.2 Parents

Die zweite wichtige Nutzergruppe der Applikation sind die Eltern der Kinder, welche die Kinderkrippe in Anspruch nehmen. Sie können die Applikation im wesentlichen für zweierlei Dienste nutzen:

1. Bekanntgabe wichtiger Information für das Kinderkrippen-Personal (Abwesenheit des Kindes, Anmeldung zum Mittagessen, etc.)
2. Erhalt spezifischer Informationen, die vom Kinderkrippen-Personal entweder direkt (Erledigung von Aufgaben, Kosten für das Mittagessen, etc.) an die jeweiligen Eltern übermittelt werden oder vom Kinderkrippen-Personal allgemein zur Verfügung gestellt werden (Bildern, etc.) und von Eltern abgerufen werden können.

### 2.1.3 Administratoren

Der Administrator übernimmt in dieser Applikation rein die Verwaltung aller Benutzer und verfügt dementsprechend über umfangreiche Nutzerrechte. Er ist als einziger Akteur befähigt Benutzer zu löschen.

## 2.2 Use-Cases - Übersicht

Bevor die Use-Cases im Detail diskutiert werden, soll hier zunächst eine allgemeine Auflistung über die dem jeweiligen Akteur zur Verfügung stehenden Funktionen gegeben werden.

Die Dienste der einzelnen Akteure wurden für eine übersichtlichere Darstellung in drei Kategorien - **Verwaltungstätigkeiten**, **Tagesgeschäft** und **Serviceleistungen** - gegliedert, die auch im späteren Use-Cases Diagramm durch eine farbliche Kennzeichnung ersichtlich sind. Zu beachten gilt es, dass diese Übersicht Funktionen nennt, welche gleich mehrere Use-Cases beinhalten. Alle elementaren Use-Cases finden sich im Use-Cases Diagramm.

### 2.2.1 Supervisor

#### **Verwaltungstätigkeiten**

- Kinder anmelden/abmelden
- Eltern anmelden
- Bearbeitung aller erfassten Nutzerdaten (Eltern, Kinder, Kontakte)

#### **Tagesgeschäft**

- Login
- Maximalbelegung eintragen
- Betreuungstage eintragen
- Ferienzeiten eintragen
- Eintragen allgemeiner Bring- und Abholzeiten
- Bezugsperson freischalten
- Buchungszahlen für Mittagessen abrufen
- Kosten-Reports für Mittagessen anzeigen/exportieren
- Aufgaben (Jobs) für Eltern erstellen
- Aufgaben als erledigt markieren

#### **Serviceleistungen**

- Kontaktliste anzeigen/exportieren
- Kinder-Stammbblatt anzeigen/exportieren
- Tagesplaner anzeigen/exportieren

### 2.2.2 Parents

#### **Verwaltungstätigkeiten**

- Personendaten bearbeiten (Kind, Eltern)

#### **Tagesgeschäft**

- Login
- Eintragung alternativer Abholperson
- Bekanntgabe von Abwesenheit
- Mittagessen anmelden/stornieren

## Serviceleistungen

- Anzeige/Export des Kinder-Stammblatts
- Bildergalerie anzeigen

Administrator

## Verwaltungstätigkeiten

- Benutzerverwaltung

## Tagesgeschäft

- Login

## 2.3 Use-Case Diagramm

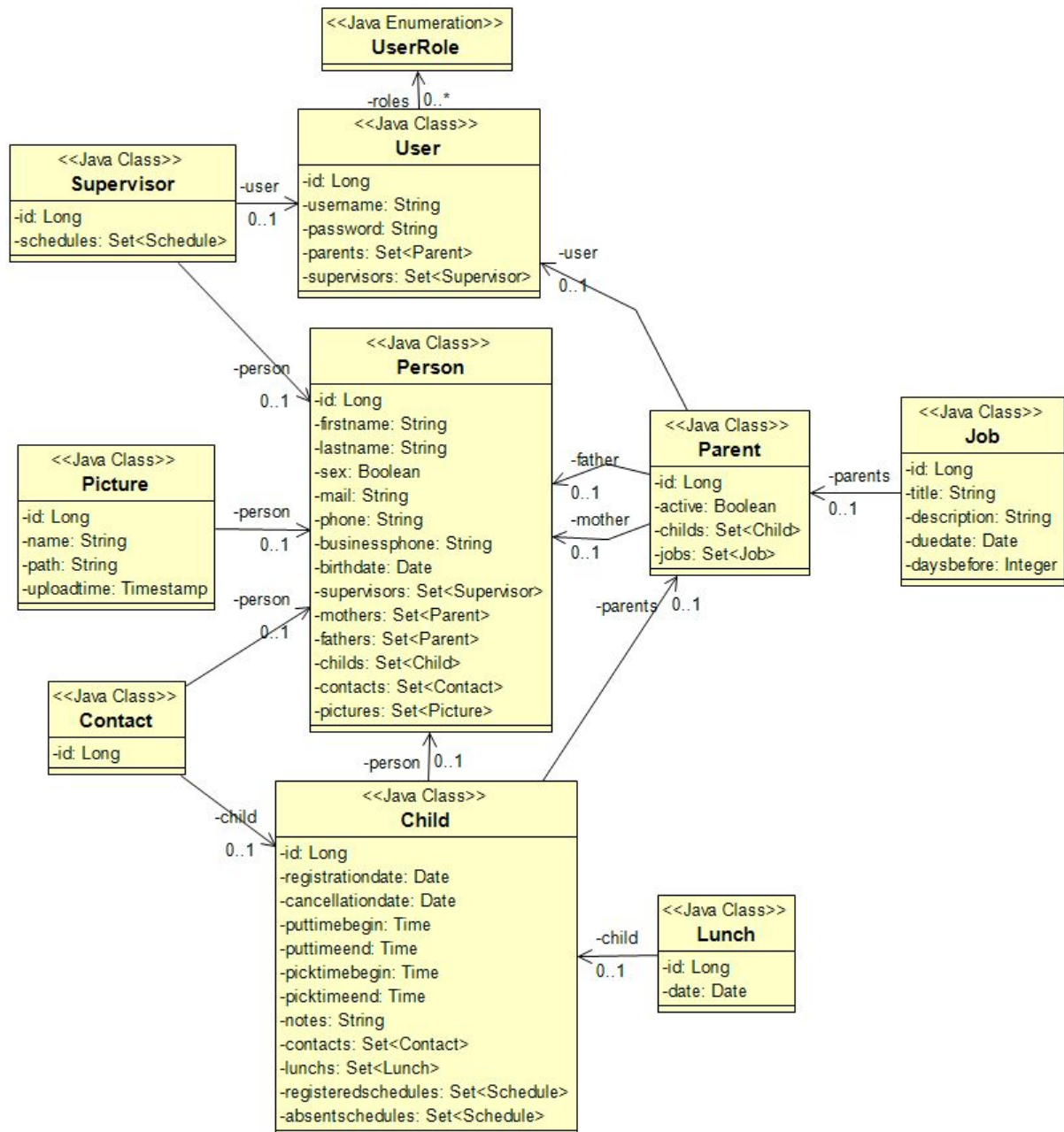
Hier sind die oben genannten Use-Cases noch einmal grafisch dargestellt:



Legende: Verwaltungstätigkeiten ||||| Tagesgeschäft ||||| Serviceleistungen |||||

# 3 Klassendiagramm

Das Klassendiagramm dient zur vereinfachten Veranschaulichung der wesentlichen Klassen/Objekte dieses Projekts. Die für das Klassendiagramm verwendete Sprache ist [UML](#) und wird bis heute verwendet.



# 4 Softwarearchitektur

## 4.1 Komponentendiagramm

Dieses Projekt basiert hauptsächlich auf dem heute bekannten MVC-Pattern.

Die einzelnen Komponenten des [MVC-Patterns](#) werden Model, View und Controller genannt.

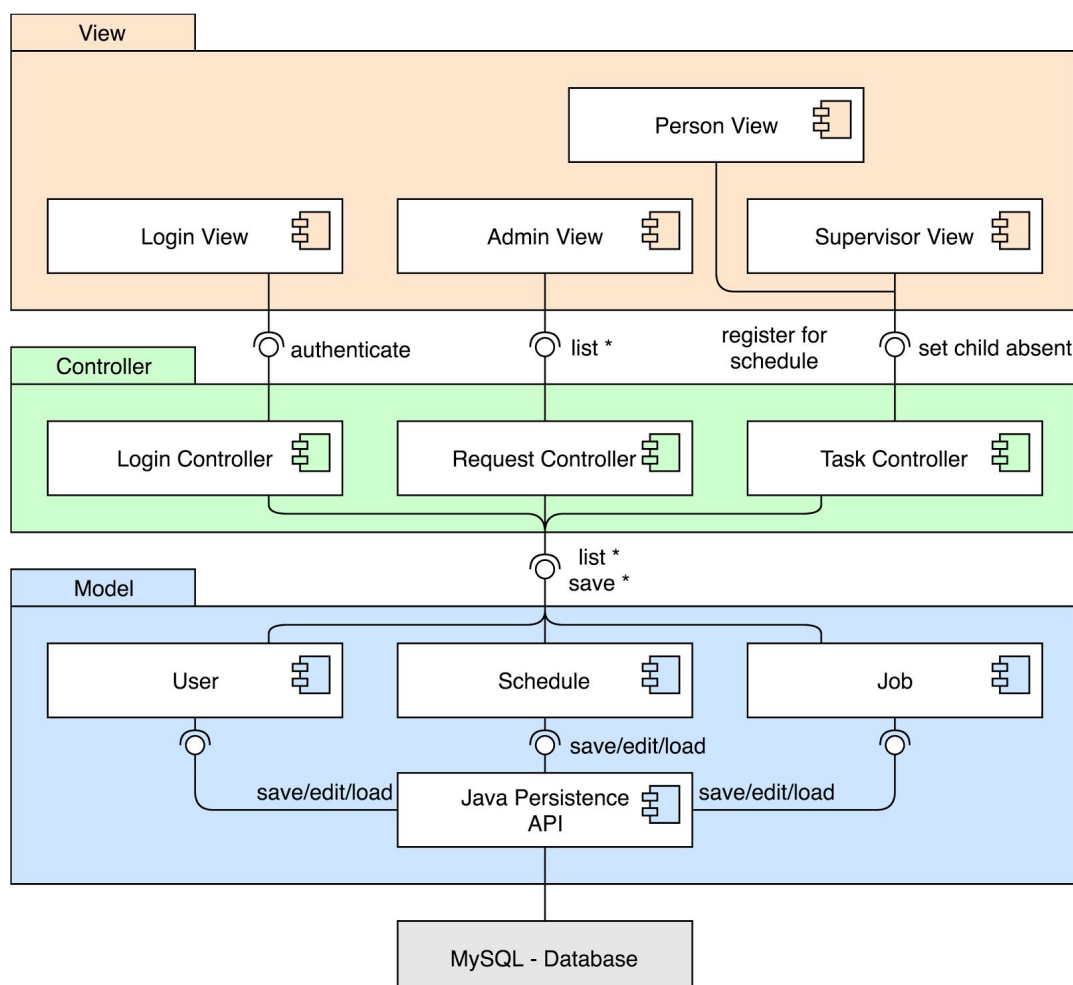
Models bearbeiten Daten auf fachlicher/technischer/logischer Ebene und geben diese an den Controller weiter. Sie sind auch für die Datensicherung verantwortlich.

Ein Controller bereitet diese eventuell noch für die Oberfläche ein wenig vor und gibt die Daten anschließend weiter.

Views stellen eine Oberfläche für den Nutzer zur Verfügung. Dazu zählen Text-Blöcke, Formulare, Bilder, sowie Grafiken aller Art und vieles mehr.

Das MVC-Pattern kann wie folgt dargestellt werden:

(Die Grafik wird nochmals verdeutlichen, dass ausschließlich direkte Kommunikation mit benachbarten Komponenten/Schichten möglich ist)





## 4.2 Beschreibung ausgewählter Technologien

Für die Entwicklung des Projekts wurden sorgfältig Technologien ausgewählt, welche sich bereits in der Praxis bewährt haben.

Diese sollten auch unterstützend für Projektentwickler und Endbenutzer wirken.

### **Java Spring Framework**

Grundsätzlich wird das Spring Framework verwendet. Es befolgt durchgehend dem oben beschriebenen MVC-Patterns.

Dabei werden einzelne Arbeits-Pakete in sogenannte "Beans" aufgeteilt, welche unterschiedlich lange über ein bequemes Annotations-Interface existieren können.

JSF ist sehr modular aufgebaut und kann deshalb mit weiteren, sehr nützlichen Modulen aufgebaut werden. Beispielsweise die

### **Java Persistence API**

oder auch kurz JPA. Diese sorgt dafür, dass einfache Klassen vollautomatisch über ein Annotations-Interface im Hintergrund als SQL-Tabellen, mit all ihren Attributen gespeichert werden.

### **Java Server Faces**

oder kurz JSF sorgt auch für Vorteile des Endbenutzers. Dieses Modul ermöglicht eine vollautomatische Kommunikation zwischen Server und Client. Dafür sorgen komfortable Schnittstellen.

Weiters können einzelne Eigenschaften für Formularfelder aktiviert werden, um dem Benutzer bei der Eingabe zu helfen.

### **Prime Faces**

wird verwendet, um schnell und einfach bewährte Web-Konstrukte darzustellen. Dabei beginnt man mit einfachen Frames, einem Grid-System, bis zu komplexen Chart-Tools, welche über eine leicht zu verstehende API verwendet werden können.

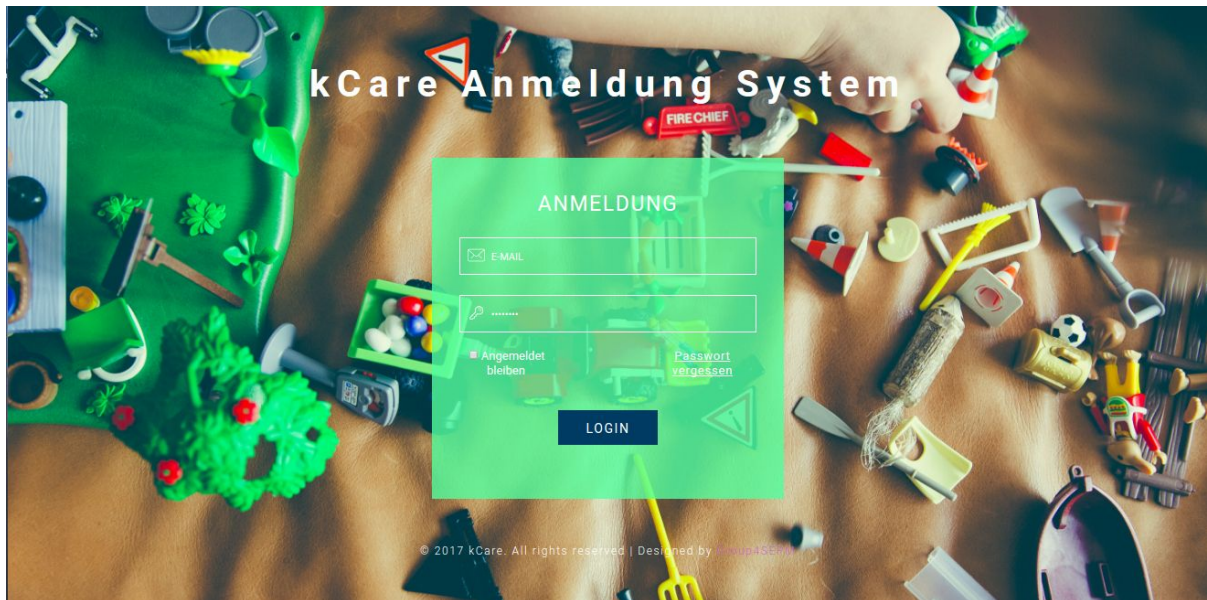
### **Spring-Boot mit Maven**

Zuguterletzt verwenden wir Spring-Boot für einen erleichterten Server-Start sowie für das Laden der einzelnen Module.

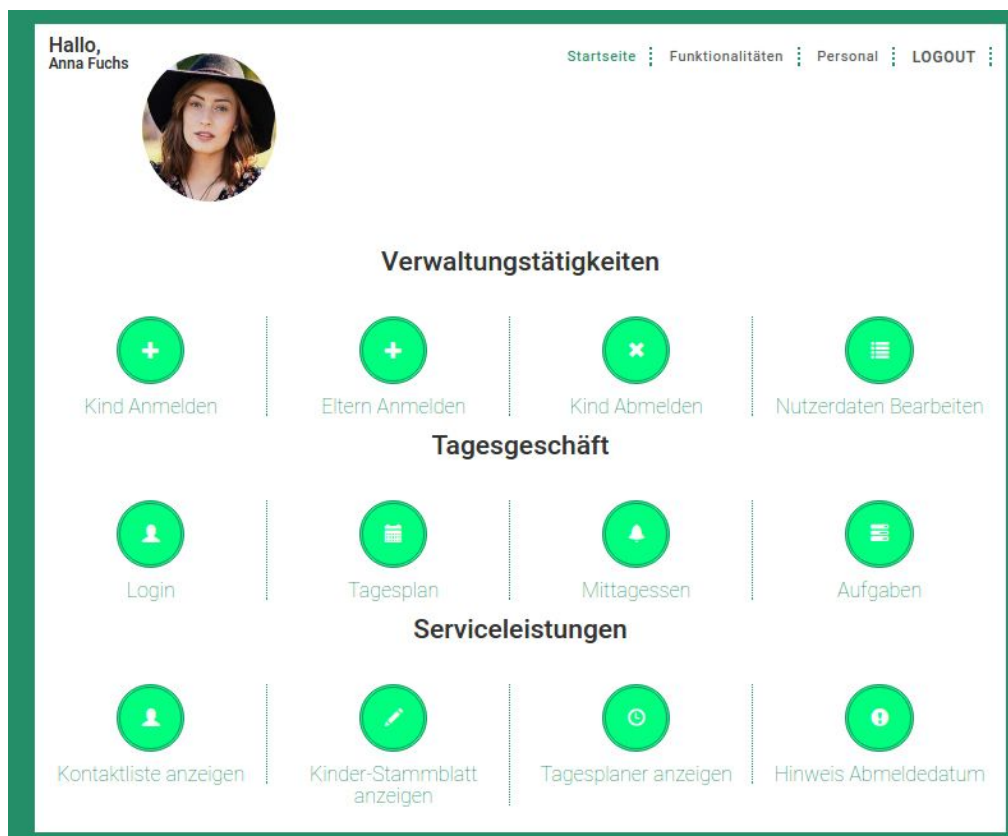
Maven stellt über eine XML-Datei automatische Modul-Downloads zur Verfügung, über welche auch beispielsweise Prime Faces und viele weitere Module vorbereitet werden.

# 5 GUI Prototyp

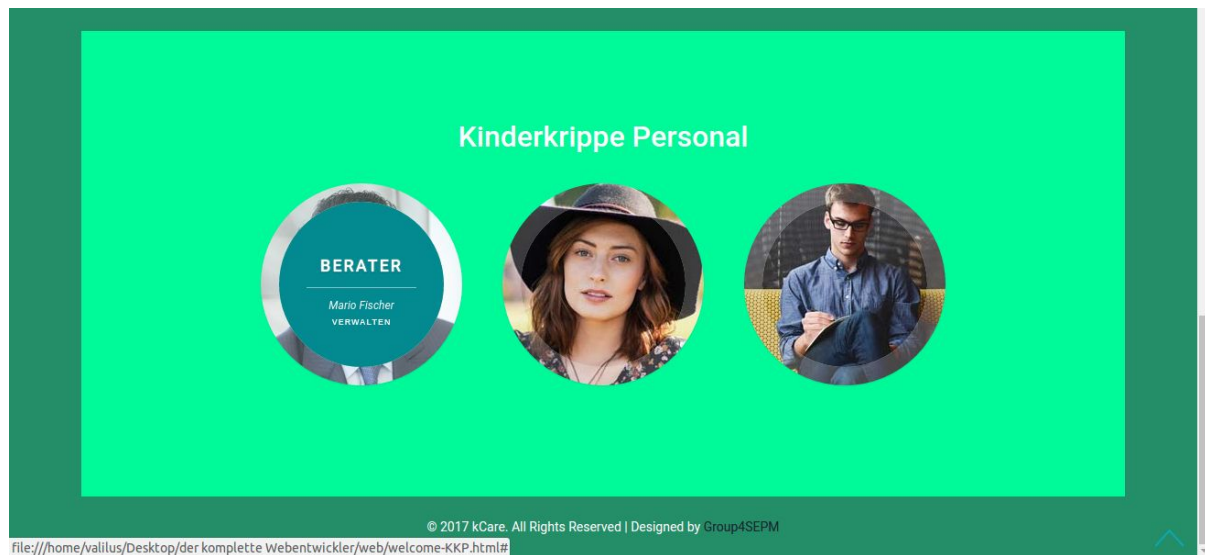
## 5.1. Login Page



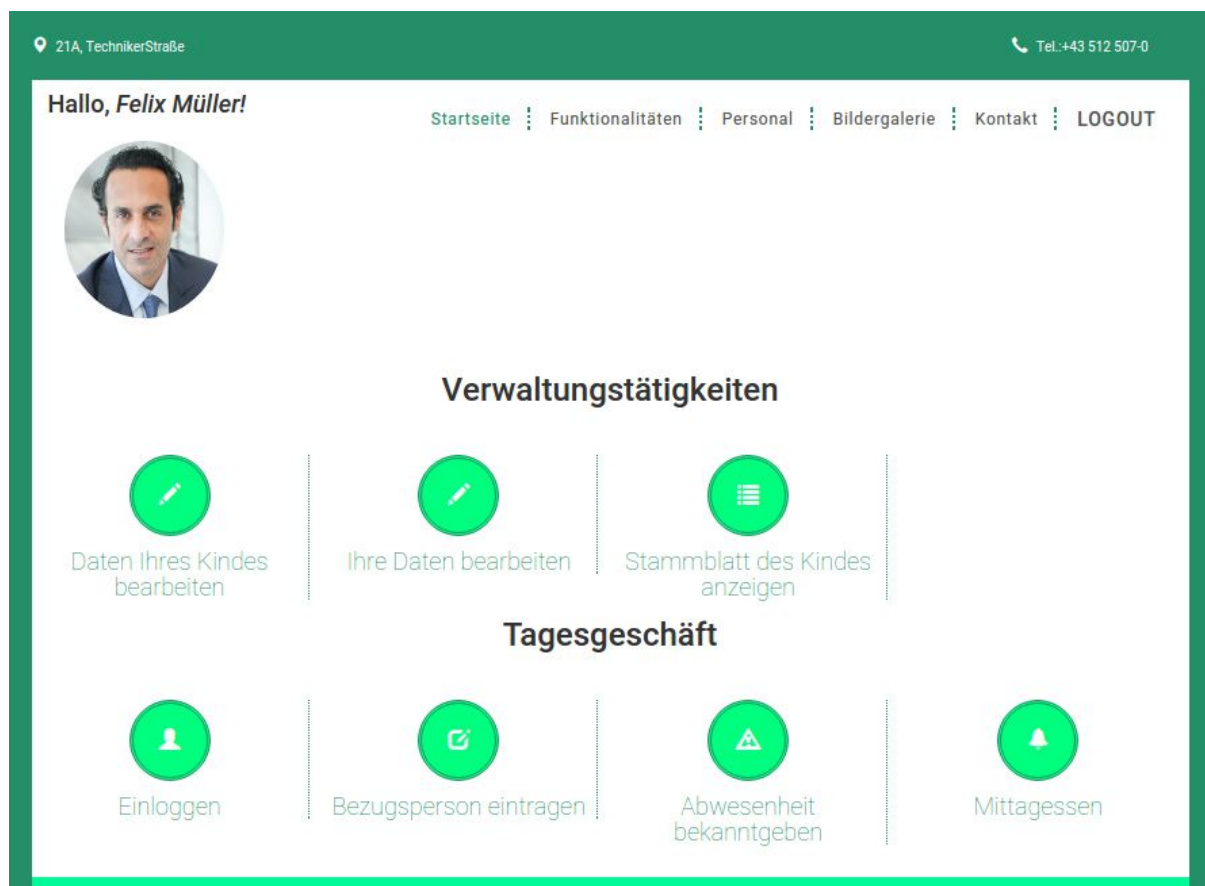
## 5.2. Kinderkrippe Personal Welcome Page



### 5.3. Kinderkrippe Personal Welcome Page (2)









### 5.4. Eltern Welcome Page







## 5.5. Eltern Welcome Page (2)

### Bildergalerie



### Kinderkrippe Personal



### Kontakt

**Adresse:**  
Technikerstraße 21A,  
6020 Innsbruck,

**Anrufen:**  
Telefonnummer: +43 512 507-0  
FAX: +43 512 507-0

**E-mail:**  
kcare@example.com

© 2017 kCare. All Rights Reserved | Designed by Group4SEPM

# 6 Projektplanung

---

## 6.1 Planung des Projekt-Aufbaus

Eine ordentliche Planung ist ein sehr bedeutender Schritt für den Verlauf der späteren Entwicklung bis hin zur erfolgreichen Fertigstellung dieses Projekts. Dazu zählen die Aufstellung der Akteure, wichtige Use Cases, Diagramme und vieles mehr, welche vor allem in Punkt 2 aufgelistet sind.

### 6.1.1 Aufstellung der Akteure

Die Aufstellung stellt bereits viele Fragen zur Verwirklichung des Projektes auf und zeigt gleichzeitig Wegweiser, wie einzelne Basisstrukturen im Hintergrund aussehen können oder sogar müssen. Außerdem geben sie einen Einblick in mögliche Use Cases.

Daher ist die Aufstellung der Akteure sorgfältig durchzuführen.

### 6.1.2 Ausarbeitung der Use-Cases

Use Cases sind ein sehr wichtiger Teil der Projektplanung. Vor allem, wenn, wie in diesem Projekt, viele Akteure vorhanden sind. Anhand der Use Cases kann man bereits schätzen, wie lange die Projektentwicklung dauern wird und wie viel Aufwand die Verwirklichung mit sich Trägt.

Zudem bieten sie die Möglichkeit, Sicherheitsaspekte des Projektes zu klären sowie die Möglichkeit, das Projekt anhand der Use Cases auf einzelne Entwickler aufzuteilen.

### 6.1.3 Erstellung eines abstrakten Klassen- und Komponentendiagramms

Das abstrakte Klassendiagramm bietet für jeden Team-Entwickler einen schnellen und vereinfachten Einblick auf etwas komplexeres.

Sie bieten die Möglichkeit, unabhängig von Implementierungen von Teammitgliedern zu arbeiten.

Die Erstellung eines abstrakten Klassendiagramms gibt bereits erste Hinweise, wie die Basisstruktur des eigentlichen Programms aussehen wird. Außerdem werden Schnittstellen im Komponentendiagramm klar für beide Seiten definiert.

#### 6.1.4 Auflistung aller Aufgaben

Nach der Ausarbeitung der Use Cases, des Klassendiagramms sollten einzelne Bedingungen klar definiert werden.

Dabei hinterfragen wir, welche Aufgaben wie zu erledigen sind, wie diese dargestellt werden, und welche Möglichkeiten erreichbar sind. (siehe Use Cases)

#### 6.1.5 Aufteilung der Aufgaben auf einzelne Verantwortungsträger

Nachdem (die wichtigsten) Aufgaben bekannt sind folgt die genaue Aufteilung der einzelnen Arbeitspakete.

Im Optimalfall tätigt jeder den Aufgabenteil, den er am besten kann.  
Die Verantwortungsbereiche sehen derzeit folgendermaßen aus:

##### Datenbank:

- Dominik

##### Repositories:

- Daniel
- Dominik

##### Controller, Services, Beans:

- Thomas
- Lukas
- Daniel und Dominik (unterstützend)

##### GUI und Design:

- Valerica
- Alle restlichen Teammitglieder (unterstützend)

##### Tests:

- alle Teammitglieder
- fortlaufende Implementierung, um Code-Coverage aufrecht zu erhalten

#### 6.1.6 Erstellung eines Projekt-Konzepts sowie eines Projekthandbuchs

Das Projekt-Konzept sorgt für einen durchgehenden Leitfaden und sollte von jedem Teammitglied vollständig durchgelesen werden.

Außerdem sollten das Konzept sowie das Handbuch stets aktuell gehalten werden.

## 6.2 Planung Projektentwicklung

### 6.2.1 Planung Basisfunktionalitäten

Wir beginnen mit der Implementierung von Basisfunktionalitäten.

Genauer gesagt, werden grundlegende Models implementiert und daraufliegende Repository-Schnittstellen definiert.

Außerdem werden bereits, je nach priorität und deadline geordnet, einzelne Kernprobleme behandelt. (z.B.: Datensicherung, Authentication-Security, ...)

### 6.2.2 Planung der Weiterentwicklung von Basisfunktionalitäten

Nach Abschluss der Basis-Implementierung beginnen wir mit der fortlaufenden Verfeinerung und Erweiterung bestehender Kernfunktionalitäten.

Dies sollte so lange weitergeführt werden, bis alle Anforderung, welche im Pflichtenheft (oder im GIT-System) angegeben sind, erfolgreich implementiert wurden.

Das Programm sollte in dieser Entwicklungsstufe IMMER ausführbar sein. Genauer sollten alle Tests erfolgreich sein und alle Kernfunktionalitäten wie erwartet laufen.

Zusätzlich werden hier Feinheiten am GUI durchgeführt, welche unter Punkt 2 und 5 aufzufinden sind.

## 6.3 Meilensteine des Projekts

Dieses Projekt ist in 3 Haupt-Meilensteine und in wenige Unter-Meilensteine aufgeteilt. Dies funktioniert am Besten mithilfe einer grafischen Darstellung.

