**Architekturdokumentation**

<Ilbec>

arc42 Template (Version 4.0) zur Dokumentation von Software- und Systemarchitekturen.

erstellt von

*Template Revision: 6.0 DE (Release Candidate)  
19. März 2012*

|  |  |
| --- | --- |
| We acknowledge that this document uses material from the arc 42 architecture  template, <http://www.arc42.de>. Created by Dr. Peter Hruschka & Dr. Gernot Starke. For additional contributors see arc42.de/about/contributors.html |  |

**Änderungsübersicht**

| **Version** | **Datum** | **Bearbeiter** | **Beschreibung** |
| --- | --- | --- | --- |
| 1.0 | 21.05.2014 | Philipp Streicher | Initiale Erstellung |
|  |  |  |  |
|  |  |  |  |

Inhaltsverzeichnis

1. Einführung und Ziele 4

1.1 Aufgabenstellung 4

1.2 Qualitätsziele 4

2. Randbedingungen 4

2.1 Technische Randbedingungen 4

2.2 Konventionen 5

3. Kontextabgrenzung 5

3.1 Fachlicher Kontext 5

3.2 Externe Schnittstellen 5

4. Lösungsstrategie 7

5. Bausteinsicht 8

5.1 Ebene 1 8

5.2 Ebene 2 8

5.3 Ebene 3 12

6. Laufzeitsicht 12

6.1 Laufzeitszenario 1 12

6.2 Laufzeitszenario 2 12

6.3 ... 12

6.4 Laufzeitszenario n 12

7. Konzepte 12

7.1 Fachliche Strukturen und Modelle 12

7.2 Typische Muster und Strukturen 12

7.3 Persistenz 12

7.4 Benutzungsoberfläche 13

7.5 Testbarkeit 13

7.6 Skalierung, Clustering 13

7.7 Buildmanagement 13

8. Entwurfsentscheidungen 13

8.1 Entwurfsentscheidung 1 13

8.2 Entwurfsentscheidung n 13

Anmerkung: In der Microsoft-Word-Variante enthält dieses Template Anleitungen und Ausfüllhinweise als „ausgeblendeten Text“. Durch den Befehl „Formate ein-/ausblenden“ können Sie die Anzeige dieser Texte bestimmen.

# Einführung und Ziele

Wer stand nicht schon einmal in der Stadt und hat sich in einer Gruppe gefragt: Wo können wir jetzt noch auf eine Party gehen? Oder wo können wir jetzt zum Mittagessen gehen? Kennt jemand eine gute Lokalität in der es vielleicht ein Angebot oder ähnliches gibt? Dafür möchte unser Projektteam eine Lösung anbieten: LLBEC. Mit unserem Projekt versuchen wir mit Hilfe einer App den Leuten zu zeigen, ob und wo es in Ihrer Nähe eine Lokalität gibt, die ein bestimmtes Angebot, wie eine Bar mit einer Happy Hour, einen Club mit einer Party oder auch nur ein Restaurant das einen Mittagstisch anbietet. Der Fokus der App liegt dabei, spontane Entscheidungen zu unterstützen und die Benutzer dorthin zu bringen, wo auch etwas los ist. Benutzer können auch Kommentare und Bewertungen abgeben. Dabei werden alle Lokalitäten auf einer GoogleMaps Karte angezeigt, wobei die, die ein Event oder ein spezielles Angebot haben, hervorgehoben.

## Aufgabenstellung

## Kurzbeschreibung der fachlichen Aufgabenstellung, Extrakt (oder Abstract) der Anforderungsdokumente.

#### Innovation

Die Innovation des Projekts liegt darin, dass nicht einfach nur eine Liste von Lokalitäten angezeigt wird, sondern es möglich ist, eine Liste mit Events zu bekommen die eine Art "Live-Ansicht" darstellen. Abends in der Stadt suchen die Leute einen Club oder eine Bar, wo sie als nächstes hingehen können bzw. wo etwas los sein könnte. Durch Kommentare ist es Benutzern möglich anderen Benutzern mitzuteilen, wie eine Feier ist, auf der die grade sind. Dadurch können andere Benutzer einschätzen, ob es sich lohnt, zu einer Veranstaltung zu gehen oder nicht.

Außer Kommentare können auch Fotos hochgeladen werden, um einen genaueren Eindruck zu vermitteln.

Dabei sind die Kommentare nicht auf einer Lokalität basiert, sondern auf ein bestimmtes Event und somit nicht dauerhaft. Die Benutzer können sich bestimmte Lokalitäten auch als Favoriten markieren, um so immer aktuelle Informationen für das nächste Event zu bekommen.

#### Anforderungen

Eventveranstalter haben die Möglichkeit, über eine eigene Schnittstelle Events einzutragen, um somit potentielle Kunden zu gewinnen. Zudem haben sie die Möglichkeit ihre Veranstaltung zu aktualisieren um Änderungen bekannt zu geben, wie bspw. dass das Event voll ist oder es kein Mittagsangebot mehr gibt. Um dies zu ermöglichen muss es verschiedene Nutzergruppen geben.

Für den normalen Benutzer soll es nicht nötig sein, sich bei dem System zu registrieren, um dieses nutzen zu können.

Beim Start der Anwendung muss der Client mit dem Internet verbunden sein. Nach dem Start soll es dem Client aber auch möglich sein, die wesentlichen Aktionen auch offline durchzuführen (Filtern, …)

Für den Anfang soll das System lediglich für die Stadt Ulm existieren. Es ist aber offen, ob das System für weitere Städte implementiert wird oder nicht. Auf die Effizienz muss am Anfang des Projektes kein besonderer Fokus liegen, da die Nutzerzahlen beschränkt sind.

Wichtig ist, dass die Filterfunktionen getestet werden. Das heißt jede Filterfunktion soll durch Unit-Tests abgedeckt sein.

Das System soll Plattformunabhängig implementiert werden. Das heißt es soll auf jedem Betriebssystem laufen. Kleine Unterschiede in der Darstellung sind nicht entscheidend, die wesentliche UI soll aber plattformübergreifend übereinstimmen. Die Anwendung soll jedoch für Mobile Clients optimiert sein.

# Randbedingungen

## Laufzeitumgebung

* Browser: Es sollen möglichst viele Browser unterstützt werden. Die Anwendung soll für Mobile Clients optimiert sein.

## Eingesetzte Frameworks

Das System soll mit HTML5, CSS3 und JavaScript entwickelt werden, um eine Plattformunabhängigkeit zu gewährleisten. Die Anwendung soll über den Browser auf dem Client ausgeführt werden.

Weiterhin soll nicht Plain JavaScript entwickelt werden, sondern das Framework AngularJS zum Einsatz kommen.

Wichtig ist, dass nach Start der Anwendung die Kommunikation mit dem Server begrenzt werden soll, damit der Client auch offline arbeiten kann und nicht immer eine Internetverbindung braucht, die die Anwendung ausbremsen könnte. Als Beispiel können hier die Filteroperationen angesehen werden. Sobald die relevanten Daten beim Start vom Server geladen wurden, soll es möglich sein die Daten auf dem Client zu filtern.

## Eingesetzte APIs

* GoogleMaps API for Angular: Eine Bibliothek, die für AngularJS bereitgestellt wird und mit denen es möglich ist, die Features von Angular zu nutzen. Es gibt eine eigene GoogleMaps und Marker Komponente die eingesetzt werden sollen

## Technische Randbedingungen

|  |  |
| --- | --- |
| Bedingung | Beschreibung |
| Programmiersprachen | HTML5, CSS3, JavaScript (AngularJS), PHP auf Server Seite |
| Datenbank | MySQL DB |
| Server | XAMPP |

# Lösungsstrategie

## Model-View-Controller

Das Model-View-Controller Pattern wird durch das AngularJS Framework impliziert.

* Es muss für jede View ein Controller existieren, welcher auf Benutzereingaben reagiert. Je nach Größe einer View können auch mehrere Controller existieren. Beispielsweise sollte für die GoogleMaps Komponente ein eigener Controller implementiert werden.
* Der Controller kommuniziert mit dem Modell. Das Modell wird über Data binding mit der View verknüpft, damit keine Inkonsistenz der Daten herrscht.
* Um mit dem Server zu kommunizieren, sollen auf JavaScript heraus Php Skripte auf dem Server aufgerufen werden. Diese liefern ein Ergebnis, was in JavaScript weiterverarbeitet wird. Alle anderen Möglichkeiten werden von dem Browser aus sicherheitstechnischen Gründen blockiert.
* Um zwischen Controllern zu kommunizieren, sollten Services genutzt werden, um nicht von anderen Komponenten abhängig zu sein.

## Benutzergruppen

Es muss verschiedene Benutzergruppen in der Anwendung geben.

### Systemadmin

Der Systemadmin hat die Möglichkeit auf alle Seiten der Anwendung zuzugreifen und alle Funktionen der Anwendung zu benutzen. Seine Aufgabe ist es, die Benutzer zu verwalten, Locations zu verwalten und ihre Veranstalter dazu zu verwalten.

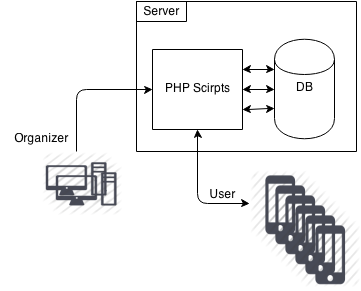
### Veranstalter

Veranstalter sind immer einer oder mehreren Locations zugeordnet. Dabei haben sie die Aufgabe Events für eine Location zu verwalten, d.h. Events anzulegen und zu löschen.

### User

Die normalen Benutzer müssen nicht an dem System angemeldet sein und haben daher auch keine Nutzergruppe in der Datenbank zugeteilt. Sie können nicht auf die Adminseiten zugreifen.

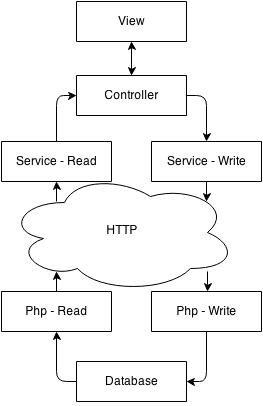
# Bausteinsicht



Das Diagramm zeigt den generellen Aufbaut der Applikation.

## Systemarchitektur

Die generelle Systemarchitektur wird in dem folgenden Diagramm dargestellt. Hier wird gezeigt, welche Schichten in der Applikation existieren und wie diese miteinander kommunizieren.



Wichtig ist, dass read und write Zugriffe getrennt sind. Es gibt Services, die von den Controllern aufgerufen werden und genutzt werden, um mit der Datenbank zu kommunizieren. Diese Services führen http Aufrufe aus und kommunizieren so mit dem Server. Der Controller ist immer mit einer View verbunden.

## Server Komponenten

Für jede Komponente die in dem Diagramm aufgelistet ist, existiert eine View und ein Controller, welcher das Model enthält.

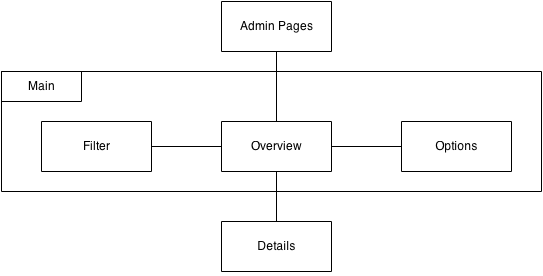
### Php Skripte

Die Php Skripte dienen zur Kommunikation des Clients mit dem Server. Die Skripte greifen auf die Datenbank zu, fragen Daten ab und liefern die gewünschten Daten an die Clients zurück.

## Client Komponenten

### Komponenten Aufbau

Insgesamt besteht die Applikation aus drei Haupt-Komponenten, wobei zwei davon für den unangemeldeten Nutzer zur Verfügung stehen und eine Komponente, welche für die Administration nötig ist. Diese drei Komponenten sind hierarchisch aufgebaut und bestehen aus weiteren Subkomponenten, welche später noch erläutert werden. Starten tut die Applikation in der Main Komponente, auf der Overview Page



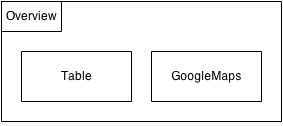
### Overview

Die Anwendung wird mit der Overview Komponente initialisiert. Von hier aus kann man durch die Anwendung navigieren und alle anderen Komponenten erreichen.

Die Overview Komponente bietet dem Benutzer eine Übersicht über alle Veranstaltungen und Locations. Dabei werden die Locations mit einem Event, welches den Filtern entspricht anders dargestellt als Locations, dessen Events nicht den Filtern entsprechen.

Grundsätzlich hat der Benutzer zwei verschiedene Arten sich die Daten anzeigen zu lassen:

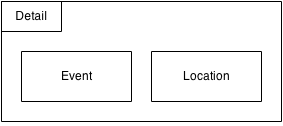
1. Über eine Karte: Es wird eine Karte dargestellt, auf der die Position des Benutzers markiert ist, damit sich dieser leichter orientieren kann. Alle Locations werden dann auf der Karte angezeigt, wobei die Locations mit Events auffälliger gestaltet sind.
2. Über eine Tabelle: Dem Benutzer wird eine Tabellarische Übersicht gegeben, auf der alle Locations aufgelistet sind. Dabei kommen die Locations mit einem Event als erstes. Hier hat der Benutzer die Möglichkeit die Tabelleneinträge zu filtern, um nach bestimmten Events oder Locations zu suchen. Die verschiedenen Arten der Locations werden farblich voneinander unterschieden.



Der Benutzer hat die Möglichkeit Locations, die ihm angezeigt werden anzuklicken und somit auf eine Detail Seite zu gelangen, welche ihm detaillierte Informationen über die Location über das Event, welches den Filtern entspricht anzuzeigen. Dabei wird, sofern ein Event für die gewählte Location existiert, das Event angezeigt und falls nicht, wird die Location angezeigt.

### Detail-Seiten

Auf der Detail-Seite einer Location wird dem Benutzer eine Tabelle mit allen Events angezeigt, die für die Location existieren. Die Einträge werden nach Datum sortiert, sodass der Benutzer das neueste Event zuerst sieht. Er hat die Möglichkeit auf die Detail-Page des Events zu gelangen, indem er einen Eintrag aus der Tabelle durch anklicken auswählt. Weiterhin wird auf den Detail-Seiten Information über die Location oder das Event angezeigt, die für den Benutzer interessant sind. Bei Locations, sowie Events hat er die Möglichkeit Kommentare zu hinterlassen (inklusive Bildern) und Kommentare von anderen Benutzern werden ihm ebenfalls angezeigt.



### Filter

Die Filter Komponente beinhaltet alle Filter. Die Filter werden auf die in der Overview dargestellten Locations angewendet. Wichtig hierbei ist, dass sobald ein Filter geändert wird, die Locations in der Overview aktualisiert werden. Wie die Kommunikation zwischen den beiden Komponenten stattfindet, wird im nächsten Punkt erläutert.

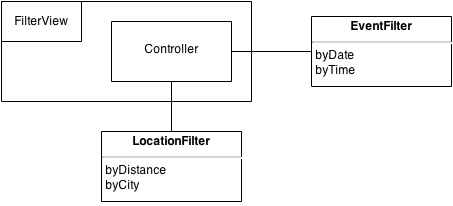
Der Benutzer hat die Möglichkeiten nach vier verschiedenen Kriterien zu Filtern:

1. Radius: Der Benutzer kann einen Radius wählen, in dem alle Locations angezeigt werden. Als Zentrum des Radius wird der aktuelle Standort des Benutzers gewählt.
2. Ort: Der Benutzer kann einen Ort auswählen in dem alle Locations ausgewählt werden. Wenn ein Ort ausgewählt wird, dient dieser als neues Zentrum des Radius.
3. Datum: Alle Locations die an dem ausgewählten Datum ein Event haben werden besonders angezeigt.
4. Uhrzeit: Alle Locations die an der ausgewählten Uhrzeit ein Event haben werden besonders angezeigt.

Hier sieht man bereits, dass es zwei verschiedene Filtertypen gibt. Die Filter Radius und Ort bestimmten, ob eine Location überhaupt auf der Overview angezeigt wird oder nicht. Somit werden hier die Locations gefiltert

Auf der anderen Seite gibt es noch die Filter Datum und Uhrzeit, bei denen die Events gefiltert werden. Hierbei ist es so, dass alle Locations die durch die Filter kommen anders bzw. auffälliger für den Benutzer angezeigt werden als die Locations die nicht diesen beiden Filtern entsprechen.

Alle Filter können und müssen teilweise auch kombiniert werden. Der Filter nach Datum ist immer aktiv. Es macht keinen Sinn, dass der Benutzer nur basierend auf einer Uhrzeit nach Events sucht, ohne dabei ein Datum anzugeben.



Jeder Filter hier ist eine Stand-Alone Filter-Komponente. AngularJS bietet ein Konstrukt für Filter an, welches vollkommen unabhängig ist und in der ganzen Anwendung genutzt werden kann[[1]](#footnote-1). Diese Filter arbeiten immer nur auf einem bestimmten Input-Objekt und entscheiden was zurückgegeben wird.

Bei zwei der Filter (Time, Date) ist ein wesentlicher Vorteil, dass die Filter auf dem Client ausgeführt werden und nicht auf dem Server. Dadurch ist es dem Benutzer möglich die Filter zu ändern, ohne dass er mit dem Server kommunizieren muss.

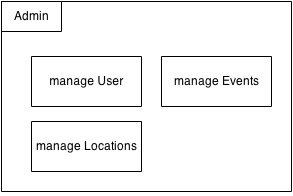
Bei den anderen beiden Filtern (Distance, City) wird nach einer Änderung die Datenbank erneut kontaktiert. Dies ist so, damit nicht alle Daten immer auf dem Client vorhanden sein müssen, da die meisten Nutzer den Standort nicht so häufig wechseln werden. So wird der Overhead minimiert.

### Admin Pages

Die Admin Pages haben drei verschiedene Aufgaben:

1. Benutzer verwalten
2. Locations verwalten
3. Events verwalten

Für jede Aufgabe steht dem Benutzer eine Seite zur Verfügung.



Auf alle Seiten hat nur die Benutzergruppe der Systemadministratoren Zugriff. Dann gibt es noch die Benutzergruppe der Veranstalter die nur auf die „manage Events“ Komponente Zugriff hat. Je nachdem, für welche Locations er verantwortlich ist, kann er Events für diese Locations eintragen, ändern und löschen.

# Laufzeitsicht

## Fehlerbehandlung

### Aktuelle Position nicht gefunden

Fehler: Es ist beim Start der Anwendung nicht möglich die Position des Benutzers zu ermitteln

Umgebung: Tritt auf, wenn die Anwendung gestartet wird.

Fehlerbehandlung: Der Benutzer bekommt die Information, dass seine Ortungsdienste deaktiviert sind. Es wird versucht eine alte Position des Benutzers aus den Cookies zu laden. Wenn dies fehlschlägt wird er aufgefordert einen Ort einzugeben, für den die Locations angezeigt werden sollen.

### Datenbankverbindung kann nicht aufgebaut werden

Fehler: Es ist nicht möglich eine Verbindung zur Datenbank aufzubauen.

Umgebung:

* Tritt auf, wenn die Anwendung gestartet wird
* Tritt auf, wenn Kommentare abgegeben werden
* Tritt auf, wenn Bewertungen abgegeben werden

Fehlerbehandlung: Der Benutzer wird informiert, dass keine aktuellen Daten angezeigt werden können. Es werden keine aktuellen Daten aus der Datenbank geladen und auf den alten Daten weitergearbeitet.

### Es kann keine aktuelle Uhrzeit ermittelt werden

Fehler: Die aktuelle Uhrzeit für die Filtereinstellungen kann nicht ermittelt werden.

Umgebung:

Fehlerbehandlung: Wenn keine Uhrzeit ermittelt werden kann, wird als default Uhrzeit 20:00 Uhr angenommen

### Es kann kein aktuelles Datum ermittelt werden

Fehler: Das aktuelle Datum für die Filtereinstellungen kann nicht ermittelt werden.

Umgebung:

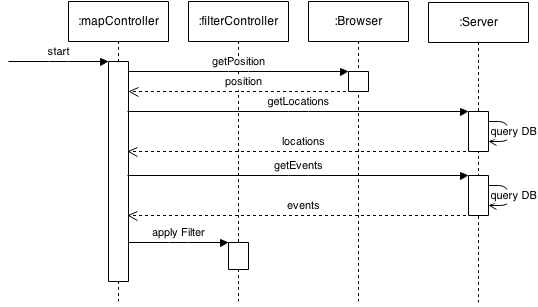
Fehlerbehandlung: Wenn kein Datum ermittelt werden kann, wird der Benutzer explizit aufgefordert ein Datum einzugeben.

## Anwendungsfälle

Im Folgenden werden die wichtigsten Anwendungsfälle beschrieben.

### Systemstart

Das folgende Diagramm stellt dar, wie der Systemstart abläuft und stellt die Kommunikation zwischen den beteiligten Komponenten dar.

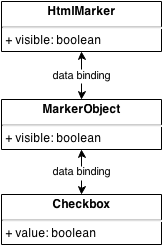


### Device Filter

Hier wird exemplarisch dargestellt, wie Filter ausgeführt werden, die auf dem Device ausgeführt werden.

#### Type Filter

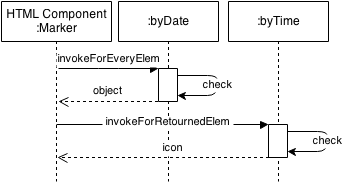
Der Type Filter zeigt, welche Art von Location angezeigt wird.



Das Filtern hier funktioniert über Data Binding. Sobald sich der Wert in der Checkbox ändert, wird die visible-Property in dem Marker Objekt aktualisiert. Dieses Objekt wiederrum aktualisiert die visible Property in der HtmlMarker-Komponente.

#### Time and Date Filter

Diese beiden Filter bestimmen, wie die Location auf der Karte angezeigt wird. Hier gibt es verschiedene Icons die von Ergebnis der Filter und Art der Location abhängen.



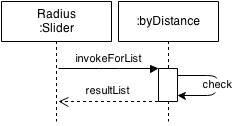
Das Diagramm beschreibt den Filteraufruf exemplarisch für alle Bars. Wichtig hier ist das Filterkonstrukt von AngularJS. Durch dieses Konstrukt ist es möglich, Funktionen zu definieren die einen Wert zurückgeben, welcher dann z.B. in einer HTML Komponente verwendet werden können. In diesem Fall ist es ein String der für das Icon, mit dem die Location angezeigt werden soll, benutzt wird.

Die Filter werden immer auf ein einzelnes Element angewendet, dass durch das „ng-repeat“-Tag definiert wird. Der erste Filter gibt dann ein Objekt zurück, wenn der Filter erfolgreich durchgelaufen ist. Auf dieses Objekt wird dann der zweite Filter angewendet und je nach Ergebnis dann ein Icon zurück geliefert.

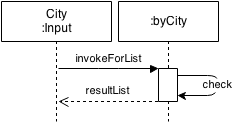
Wenn der erste Filter bereits fehlschlägt, wird null zurückgegeben. In diesem Fall gibt der zweite Filter das gleiche Icon zurück, wie wenn der zweite Filter fehlschlägt.

### Database Filter

Die beiden Filter byDistance und byCity kontaktieren die Datenbank und filtern die erhaltenen Informationen. Hier wird, anders als bei den beiden anderen Filtern nicht jedes Element gefiltert, sondern die ganze Ergebnis-Liste.



Sobald der Benutzer den Slider für den Radius ändert, wird der Filter angewendet.



Sobald der Benutzer den Ort ändert, wird der Filter angewendet.

Mit dem Ergebnis der Filter arbeitet der Rest der Anwendung weiter.

## Benutzerinteraktion Veranstalter

Veranstalter möchte ein neues Event für seine Location eintragen:

1. Anmelden (Login)
2. Auf Event verwalten Seite wechseln (Admin Page)
3. Veranstaltungsdetails eintragen (Admin Page)
4. Location auswählen (Admin Page)
5. Veranstaltung eintragen (Admin Page)
6. Veranstaltung auf der Karte suchen (Overview/Map)
7. Details anschauen (Detail Page) 🡪 Fehler entdeckt
8. Events verwalten (Admin Page)
9. Event auswählen (Admin Page)
10. Fehler beheben (Admin Page)
11. Änderung speichern (Admin Page)
12. Event suchen (Overview/Map)
13. Event kontrollieren (Detail)

## Benutzerinteraktion User: User sucht Event

Benutzer sucht ein Event für einen Abend:

1. Öffnen der App (Overview/Map)
2. Filter einstellen 🡪 Bars, Zeit: 20:00 Uhr, Radius: 10km
3. Locations anschauen (Overview/Map)
4. Location mit Event auswählen (Overview/Map)
5. Details anschauen (Details Page)
6. Event gefällt nicht 🡪 Neues Event auswählen (Overview/Map)
7. Kommentare anschauen (Details Page)
8. Event gefällt 🡪 Auf Karte anschauen, wo die Location ist

## Benutzerinteraktion User: User ist auf einem Event

Benutzer möchte ein Foto mit anderen Nutzern teilen:

1. Benutzer öffnet App
2. Benutzer sucht Event auf dem er ist über Filter(Overview/Tabelle)
3. Benutzer lädt Foto hoch und kommentiert dieses

## Laufzeitszenario Administrator

Ein neuer Veranstalter soll für eine Location angelegt werden:

1. Starten der App (Overview)
2. Anmelden als Administrator (Login)
3. Location erstellen (AdminPage)
4. Details eintragen (AdminPage)
5. Benutzer anlegen(Admin Page)
6. Location für Benutzer auswählen (Admin Page)
7. Benutzer eintragen (Admin Page)

# Konzepte

In diesem Kapitel werden einige grundlegende Konzepte dargestellt.

## Persistenz

Es gibt verschiedene Daten die persistiert werden. Dabei wird einmal zwischen den Benutzer- und den Anwendungsdaten unterschieden.

### Benutzerdaten

Als Benutzerdaten gelten die Filter und die Einstellungen des Benutzers. Auch soll es dem Benutzer möglich sein Locations als Favoriten zu kennzeichnen.

Diese Daten sollen in den Cookies persistiert werden. Die Daten müssen beim Beenden der Anwendung serialisiert werden und beim Start der Anwendung vorhandene Daten deserialisiert.

Folgende konkrete Daten fallen darunter:

* Filtereinstellungen
* Art der Anzeige
* Favorisierte Locations
* Optionen
* Art der Anzeige
* Standort des Benutzers

### Anwendungsdaten

Unter die Anwendungsdaten fallen:

* Locations
* Events
* Kommentare
* Bewertungen

Ein wesentlicher Unterschied ist, dass sobald Änderungen an einer dieser Daten stattfinden, werden diese sofort in die Datenbank geladen.

Beim Start der Anwendung werden alle Daten aus der Datenbank geholt. Erst nach erneutem starten, werden die vorhandenen Daten aktualisiert.

Persistenz (Dauerhaftigkeit, Beständigkeit) bedeutet, Daten aus dem (flüchtigen) Hauptspeicher auf ein beständiges Medium (und wieder zurück) zu bringen.

Einige der Daten, die ein Software-System bearbeitet, müssen dauerhaft auf einem Speichermedium gespeichert oder von solchen Medien gelesen werden:

1. Flüchtige Speichermedien (Hauptspeicher oder Cache) sind technisch nicht für dauerhafte Speicherung ausgelegt. Daten gehen verloren, wenn die entsprechende Hardware ausgeschaltet oder heruntergefahren wird.
2. Die Menge der von kommerziellen Software-Systemen bearbeiteten Daten übersteigt üblicherweise die Kapazität des Hauptspeichers.
3. Auf Festplatten, optischen Speichermedien oder Bändern sind oftmals große Mengen von Unternehmensdaten vorhanden, die eine beträchtliche Investition darstellen.

Persistenz ist ein technisch bedingtes Thema und trägt nichts zur eigentlichen Fachlichkeit eines Systems bei. Dennoch müssen Sie sich als Architekt mit dem Thema auseinander setzen, denn ein erheblicher Teil aller Software-Systeme benötigt einen effizienten Zugriff auf persistent gespeicherte Daten. Hierzu gehören praktisch sämtliche kommerziellen und viele technischen Systeme. Eingebettete Systeme (embedded systems ) gehorchen jedoch oft anderen Regeln hinsichtlich ihrer Datenverwaltung.

## Benutzungsoberfläche

IT-Systeme, die von (menschlichen) Benutzern interaktiv genutzt werden, benötigen eine Benutzungsoberfläche. Das können sowohl grafische als auch textuelle Oberflächen sein.

Für die meisten Systeme gibt es existierende Altsysteme, die durch die neuen Systeme abgelöst werden sollen. Denken Sie als Architekt nicht nur an Ihre neue, schöne Architektur, sondern rechtzeitig auch an alle organisatorischen und technischen Aspekte, die zur Einführung oder Migration der Architektur beachtet werden müssen.

Beispiele

1. Konzept, Vorgehensweise oder Werkzeuge zur Datenübernahme und initialen Befüllung mit Daten
2. Konzept zur Systemeinführung oder zeitweiliger Parallelbetrieb von Alt- und Neusystem

Müssen Sie bestehende Daten migrieren? Wie führen Sie die benötigten syntaktischen oder semantischern Transformationen durch?

## Testbarkeit

Unterstützung für einfache (und möglichst automatische) Tests. Diese Eigenschaft bildet die Grundlage für das wichtige Erfolgsmuster "Continous Integration". In Projekten sollte mindestens täglich der gesamte Stand der Entwicklung gebaut und (automatisch) getestet werden - daher spielt Testbarkeit eine wichtige Rolle. Wichtige Stichworte hierzu sind Unit- Tests und Mock-Objekte.

## Skalierung, Clustering

Wie gestalten Sie Ihr System „wachstumsfähig“, so daß auch bei steigender Last oder steigenden Benutzerzahlen die Antwortzeiten und/oder Durchsatz erhalten bleiben?

Wie und wo verwenden Sie Codegeneratoren, um Teile Ihres Systems aus Modellen oder domänenspezifischen Sprachen (DSL’s) zu generieren?

## Buildmanagement

Wie wird das gesamte System aus Sourcecode Bausteinen gebaut? Welche Repositories (Versionsverwaltungssysteme) enthalten welchen Sourcecode, wo liegen Konfigurationsdateien, Testdaten und/oder Build-Skripte (make, ant, maven, gradle oder Ähnliche)?

# Entwurfsentscheidungen

Inhalt

Dokumentieren Sie hier alle wesentlichen Entwurfsentscheidungen und deren Gründe!

Motivation

Es ist wünschenswert, alle wichtigen Entwurfsentscheidungen geschlossen nachlesen zu können. Wägen Sie ab, inwiefern Entwurfsentscheidungen hier zentral dokumentiert werden sollen oder wo eine lokale Beschreibung (z.B in der Whitebox-Sicht von Bausteinen) sinnvoller ist. Vermeiden Sie aber redundante Texte. Verweisen Sie evtl. auf Kap. 4 zurück, wo schon zentrale Architekturstrategien motiviert wurden.

Form

informelle Liste, möglichst nach Wichtigkeit und Tragweite der Entscheidungen für den Leser aufgebaut.

Alternativ auch ausführlicher in Form von einzelnen Unterkapiteln je Entscheidung. Die folgende Mindmap (Quelle: Kolumne „Architekturen dokumentieren“ von S. Zörner im Java Magazin 3/2009) soll Sie dabei unterstützen, wichtige Entscheidungen zu treffen und festzuhalten. Die Hauptäste stellen dabei die wesentlichen Schritte dar. Sie können auch als Überschriften innerhalb eines Unterkapitels dienen (siehe Beispiel unten).



Die Fragen sind nicht sklavisch der Reihe nach zu beantworten. Sie sollen Sie lediglich leiten. In der Vorlage löschen Sie diese heraus, und lassen nur die Inhalte/Antworten stehen.

## Entwurfsentscheidung 1

### Fragestellung

Was genau ist das Problem?

Warum ist es für die Architektur relevant?

Welche Auswirkung hat die Entscheidung?

### Rahmenbedingungen

Welche festen Randbedingungen haben Sie einzuhalten?

Welche EInflussfaktoren sind zu beachten?

### Annahmen

Welche Annahmen haben Sie getroffen?

Welche Annahmen können wie vorab überprüft werden?

Mit welchen Risiken müssen Sie rechnen?

### Betrachtete Alternativen

Welche Lösungsoptionen ziehen Sie in die nähere Auswahl?

Wie bewerten Sie jede einzelne?

Welche Optionen schließen Sie bewusst aus?

### Entscheidung

Wer (wenn nicht Sie selbst) hat die Entscheidung getroffen?

Wie ist sie begründet?

Wann wurde entschieden?

## Entwurfsentscheidung n

1. https://docs.angularjs.org/api/ng/filter/filter [↑](#footnote-ref-1)