Duale Hochschule Baden-Württemberg Mosbach
Projekt I + II

# Konzeption und prototypische Implementierung eines Geolocation-Portals für Stakeholder der Großen Kreisstadt Mosbach auf der Grundlage von Open Data

Verfasser
**Alexander Ciravegna, Frederik Dangel, Jule Gernet, Rico Horsinka, Luca Kersting, Sebastian Schmierer, Laura Schwind, Raphael Ullmer, Martin Wegele**

Studiengang Wirtschaftsinformatik
Profil: Handel
Bearbeitungszeitraum: 25. November 2019 – 05. Februar 2020

Kurs: WI17C
E-Mail: lauraschw0108@gmail.com

Gruppe 12

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| DHBW | Duale Hochschule Baden-Württemberg |
| ERM | Entity-Relationship-Modell |
| API | Application Programming Interfaces |
| DB | Datenbanken |
| REST Controller | Representational State Transfer Controller |
| JSON | JavaScript Object Notation |
| JSF | JavaServer Faces |
| JAR-Datei | Java Archive Dateiformat |
| XHTML | Extensible Hypertext Markup Language |
| CSS | Cascading Style Sheets |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einleitung

Die vorliegende Arbeit wurde im Rahmen der Vorlesung "Neue Konzepte Projekt I +II" erstellt und beschreibt die Vorgehensweise im Projektverlauf. Gegenstand dieser Arbeit ist die Planung, "Konzeption und prototypische Implementierung eines Geolocation-Portals für Stakeholder der Großen Kreisstadt Mosbach auf der Grundlage von Open Data"[1]. Der Fokus liegt hierbei auf dem Inhalt der technischen Dokumentation.

## 1.1 Vision des Projektes

Das gestaltete Geolocation-Portal soll alle Veranstaltungen in und um Mosbach anzeigen. Die Beschränkung auf den Bereich Veranstaltungen fand statt, da alle Anforderungen abgebildet werden können und eine Ausdehnung auf weitere Themengebiete im Nachhinein möglich ist. Veranstaltungen sind nicht nur für Bewohner der Kreisstadt Mosbach relevant, sondern auch für viele weitere Interessengruppen – wie Touristen, Studenten, Pendler, usw. Darüber hinaus werden nicht nur die Veranstaltungsorte (z.B. kulturelle, kulinarische und öffentliche Einrichtungen) als Geolocation abgebildet, sondern auch weitere öffentliche Plätze wie öffentliche Toiletten, Parkplätze, Bushaltestellen und Bahnhöfe.

Veranstaltungen haben zum einen eine Bedeutung für die Wochen- und Wochenendplanung von Studenten, Touristen, Einwohnern und Bewohnern der Nachbarortschaften, zum anderen sind sie relevant für Anwohner, um sich auf eine eventuell belebtere Stadt einzustellen.

Wichtig hierbei ist die Kategorisierung der Veranstaltungen, da Veranstaltungen meistens bestimmte Personengruppen ansprechen. Mit der Hilfe des Geolocation-Portals soll es dem Nutzer ermöglicht werden auf dem schnellsten und einfachsten Weg die Veranstaltungen zu finden, die für ihn relevant sind.

## 1.2 Vorgehensmethodik

Aufgrund der offen gehaltenen Aufgabenstellung wurde sich für eine agile Vorgehensmethodik entschieden. In der Konzeptionsphase kamen vor allem klassische Methoden zum Einsatz. Durch diese Trennung von Konzeption und Entwicklung wurde ein flexibler Projektablauf garantiert, um auf sich verändernde Anforderungen reagieren zu können. Für das agile Vorgehen erschien SCRUM als die geeignetste Methode. Denn mit Hilfe von SCRUM ist es möglich in einem kurzen

[1] Vgl. Aufgabenstellung, 2019, S. 2

Zeitraum ein präsentationsfähiges und verkaufsfähiges Produkt zu erarbeiten. Darüber hinaus ist es vor allem für Entwicklerteams von geringer Größe und räumlicher Nähe geeignet.

Folgende Punkte sind für SCRUM zu berücksichtigen:

1. Product Backlog
   Enthält alle Produktanforderungen in Form von User Stories. Formuliert werden diese als Antwort auf die Frage "Wer will was warum?" (vgl. Kapitel 3.3 Product Backlog).

2. Komplexität je Story in Story Points
   Die Komplexität der Storys wurde während dem "Estimation Game"[2] mit Hilfe von Karteikarten festgelegt. Jede Karteikarte war mit einer Story bedruckt und verdeckt auf einem Tisch ausgelegt. Es wurde eine Karteikarte nach der anderen aufgedeckt. Anschließend wurde beurteilt, an welche Stelle die aktuelle Story ihrer Komplexität nach eingeordnet werden soll und dort einsortiert. Alle Anwesenden hatten entweder die Chance eine neue Karte aufzudecken und einzuordnen oder eine bereits einsortierte Story an eine andere Stelle zu legen. Nachdem alle Storys aufgedeckt waren und keiner mehr eine Änderung der Reihenfolge vornehmen wollte, stand die Reihenfolge, welche die Komplexität aufzeigt, fest. Eine detailliertere Beschreibung folgt in Kapitel 3.3 Product Backlog.

3. Minimum Viable Product
   Als Minimum Viable Product wird der minimale Produktumfang beschrieben, welcher vom Kunden genutzt oder verkauft werden kann. Entsprechend dieses Konzepts wurden Storys, die für das Minimum Viable Product nicht von Relevanz sind, im Product Backlog (vgl. Kapitel 3.3 Product Backlog) rot markiert und in der Entwicklung entsprechend gering priorisiert.

4. Definition of Done
   In einer Definition of Done werden Kriterien festgelegt, die erfüllt werden müssen, damit eine Story als abgeschlossen gilt. Diese Kriterien fallen folgendermaßen aus:

- Die Story ist auf mögliche Fehler getestet und die dazugehörigen Testfälle sind angelegt
- Alle Akzeptanzkriterien wurden umgesetzt
- Es hat ein Review von einem zweiten Entwickler stattgefunden
- Der Code ist entsprechend der Aufgabenstellung ordentlich geschrieben und kommentiert
- Die Funktion ist auch auf mobilen Geräten nutzbar
- Die technische Dokumentation, Benutzerdokumentation und Angaben von Benutzernamen und Kennwörtern sind aktualisiert

[2] Vgl. Meindl 2012.

5. Sprints, Retrospektiven und Daily SCRUM

   Zunächst wird die Länge der Sprints definiert. Je nach Länge wird eine Anzahl von Story Points festgelegt, welche in einem Sprint umgesetzt werden kann. Dementsprechend wählen sich die Entwickler die Storys aus, an welchen sie im Sprint arbeiten werden. Der SCRUM-Master kümmert sich darum, dass alle für die Entwicklung benötigten Dinge zur Verfügung stehen und stets die Abläufe von SCRUM eingehalten werden. Nach jedem abgeschlossenen Sprint wird eine Retrospektive abgehalten zur Sammlung von Feedback in vier Kategorien, auf dessen Basis der nächste Sprint gestartet werden kann. In der Kategorie "Blumen" werden Teammitglieder oder Teilgruppen genannt, welche während des Sprints einen besonders wertvollen Beitrag geleistet haben. Die Retrospektiven zu jedem Sprint sind in Tabelle 1 aufgeführt.

| | **Das war gut** | **Das können wir besser** | **Neue Ideen** | **Blumen** |
|---|---|---|---|---|
| **Sprint 1** | - Zuverlässigkeit<br>- Arbeitsbereitschaft<br>- Zusammenarbeit Front-End Entwickler<br>- Organisation & Vorbereitung | - Anwesenheit<br>- Zeit (in der Weihnachtszeit)<br>- Weniger Detail-Diskussionen | - Regelmäßige Dailys | - Entwickler<br>- Organisation |
| **Sprint 2** | - Technische Probleme gelöst<br>- Teamwork<br>- „Dailys“<br>- Idee Authentifizierungsprozess | - Arbeit mit Primefaces<br>- Board aktuell halten<br>- Excel Berechtigungskonzept | - Ordner für Zukunftsvisionen in OneNote | - Luca<br>- Entwickler<br>- Martin |
| **Sprint 3** | - Zusammenarbeit<br>- Präsentationsentwurf<br>- Endergebnis | - Zeitplan einhalten (Entwicklerteam) | - Zukunftsvisionen | - Laura<br>- alle |

Tabelle 1 Retrospektiven

# 2 Projektplan

## 2.1 Zeitplan mit Meilensteinen

Zur zeitlichen Strukturierung des Projekts wurde ein Zeitplan erstellt (vgl. Tabelle 2), welcher in vier Spalten gegliedert ist. Die Meilensteinplanung erfolgte in der zweiten Spalte, alle Termine und wurden in dem dritten Tabellenabschnitt festgehalten und die Sprintzeiträume in der letzten Spalte.

Die Termine und die Teamtreffen sind mit zwei unterschiedlichen Farben gekennzeichnet, wobei der hellere Braunton auf dokumentierte Treffen des Entwicklungsteams, Treffen mit dem wissenschaftlichen Betreuer und kurze Absprachetermine hinweisen soll. Der dunklere Braunton stellt Termine dar, welche von allen Mitgliedern wahrgenommen werden sollten.

| Datum | Meilenstein | Teamtreffen/ Termine | Sprints |
|---|---|---|---|
| 25.11.2019 | | | |
| 26.11.2019 | Erstes Treffen mit Herr Dillinger | | |
| 27.11.2019 | | Teamtreffen | |
| 28.11.2019 | | | |
| 29.11.2019 | | Teamtreffen | |
| 30.11.2019 | | | |
| 01.12.2019 | | | |
| 02.12.2019 | | | |
| 03.12.2019 | | | |
| 04.12.2019 | | | |
| 05.12.2019 | | | |
| 06.12.2019 | | | |
| 07.12.2019 | | | |
| 08.12.2019 | | | |
| 09.12.2019 | fertiger Projektplan | Teamtreffen | |
| 10.12.2019 | | Teamtreffen | |
| 11.12.2019 | | | |
| 12.12.2019 | | Treffen Herr Dillinger | |
| 13.12.2019 | | | |
| 14.12.2019 | | | |
| 15.12.2019 | | | |
| 16.12.2019 | Sprint Planning/ Start | | Sprint 1 |
| 17.12.2019 | | Teamtreffen | |
| 18.12.2019 | | | |
| 19.12.2019 | fertiges Konzept/ Bereit für Implementierung/ Entwurfs- & Designphase abgeschlossen | | |
| 20.12.2019 | | | |
| 21.12.2019 | | | |
| 22.12.2019 | | Treffen Dev Team | |
| 23.12.2019 | | | |
| 24.12.2019 | | | |
| 25.12.2019 | | | |
| 26.12.2019 | | | |
| 27.12.2019 | | | |
| 28.12.2019 | | Treffen Dev Team | |
| 29.12.2019 | | Teamtreffen & Backlog Refinement | |
| 30.12.2019 | | | |
| 31.12.2019 | | | |
| 01.01.2020 | | | |
| 02.01.2020 | | | |
| 03.01.2020 | | Teamtreffen (Sprint Retro & Backlog Refinement) | |

| | | | |
|---|---|---|---|
| 04.01.2020 | | | Sprint 2 |
| 05.01.2020 | | | |
| 06.01.2020 | Treffen mit Herr Dillinger (KW2) | | |
| 07.01.2020 | | Backlog Refinement & Treffen Herr Dillinger | |
| 08.01.2020 | | | |
| 09.01.2020 | | Daily | |
| 10.01.2020 | | | |
| 11.01.2020 | | | |
| 12.01.2020 | | | |
| 13.01.2020 | | | |
| 14.01.2020 | | Teamtreffen (Sprint Retro & Sprint Planning) | |
| 15.01.2020 | | Daily | Sprint 3 |
| 16.01.2020 | | | |
| 17.01.2020 | | Daily | |
| 18.01.2020 | | | |
| 19.01.2020 | | | |
| 20.01.2020 | | | |
| 21.01.2020 | | Treffen Vorbereitung auf Treffen mit Herr Ferch | |
| 22.01.2020 | Treffen mit Herr Ferch | Treffen Doku-Team danach + Teamtreffen | |
| 23.01.2020 | | | |
| 24.01.2020 | | Daily | |
| 25.01.2020 | Entwicklungsphase beenden | | |
| 26.01.2020 | | | |
| 27.01.2020 | | Treffen (Review und Testen) - Testen des Prototypen | |
| 28.01.2020 | Testphase beenden | | |
| 29.01.2020 | | | |
| 30.01.2020 | | Treffen Herr Dillinger | |
| 31.01.2020 | Projektdoku fertigstellen | | |
| 01.02.2020 | | | |
| 02.02.2020 | Präsentation fertigstellen | | |
| 03.02.2020 | | Treffen (Abschlussbesprechung) | |
| 04.02.2020 | | | |
| 05.02.2020 | Abgabe des Projekts | | |
| 06.02.2020 | Präsentation des Projekts | | |
| 07.02.2020 | | | |

Tabelle 2 Zeit- und Meilensteinplan

## 2.2 Rollenverteilung

Die folgenden Tabellen 3 und 4 zeigen zum einen die Scrum-Rollen und zum anderen die Aufgabenverteilung innerhalb der Projektgruppe.

| Rolle | Besetzung |
|---|---|
| **Product Owner** | Stadt Mosbach |
| **Scrum Master** | Laura |
| **Team** | Alexander, Frederik, Jule, Luca, Martin, Raphael, Rico, Sebastian |

Tabelle 3 Scrum-Rollen

| Aufgabe | Verantwortliche |
|---|---|
| **Organisation** | Laura, Jule |
| **Entwicklung (Back-End)** | Frederik |
| **Entwicklung (Front-End)** | Luca, Raphael, Sebastian |
| **Oberflächendesign** | Alexander, Rico |
| **Dokumentation** | Laura, Jule, Alexander, Rico, Martin |
| **Diagramme** | Alexander, Frederik, Rico, Martin |
| **Stakeholderanalyse & Actor-Goal-Liste** | Jule, Rico |
| **Berechtigungskonzept** | Rico |
| **Entwicklertests** | Frederik, Luca, Raphael, Sebastian |
| **Anwendertests** | Laura, Jule, Alexander, Rico, Martin |

Tabelle 4 Aufgabenverteilung

# 3 Konzeption

## 3.1 Stakeholderanalyse

In der folgenden Tabelle 5 sind die einzelnen Stakeholder aufgelistet, die mit dem System in Kontakt kommen.

| Name | | | |
|---|---|---|---|
| **Einwohner** | | | |
| | Mosbach direkt | | |
| | Teilortschaften | | |
| | Nachbarortschaften | | |
| | Unterteilung in Altersklassen | | |
| | | Kinder (6-12 J.) | |
| | | Jugendliche (12-17 J.) | |
| | | Erwachsene (ab 18 J.) | |
| | | | Mit Kind |
| | | | Ohne Kind |
| **Touristen** | | | |
| | Unterteilung in Altersklassen | | |
| | | Kinder (6-12 J.) | |
| | | Jugendliche (12-17 J.) | |
| | | Erwachsene (ab 18 J.) | |
| | | | Mit Kind |
| | | | Ohne Kind |
| **Veranstalter** | | | |
| **DHBW Mosbach** | | | |
| | Mitarbeiter und Dozenten | | |
| | Studenten | | |
| | | Wohnhaft in Mosbach | |
| | | Pendler | |
| **Stadtverwaltung** | | | |
| | EDV Mitarbeiter/-in | | |
| | Sachbearbeiter/-in | | |
| **Unternehmen** | | | |

Tabelle 5 Liste der Stakeholder

## 3.2 Actor-Goal-List

Basierend auf den in Kapitel 3.1 identifizierten Stakeholdern wurden die in Tabelle 6 aufgeführten Akteure und deren Ziele ermittelt.

| Typ | Actor | | Goal |
|---|---|---|---|
| Primär | Benutzer | Einwohner | Schnelle und einfache Übersicht über die Veranstaltungen in und um Mosbach |
| Primär | Benutzer | Touristen | Schnelle und einfache Übersicht über die Veranstaltungen in und um Mosbach |
| Primär | Benutzer | DHBW Mosbach | Schnelle und einfache Übersicht über die Veranstaltungen in und um Mosbach |
| Primär | Veranstalter | Veranstalter | Unkomplizierte Termineinreichung<br>Hohe Teilnehmerzahl<br>Geringe Werbekosten |
| Sekundär | Administrator | Stadtverwaltung<br>EDV-Mitarbeiter/-in | Geringer Wartungsaufwand |
| Primär | Sachbearbeiter | Stadtverwaltung<br>Sachbearbeiter/-in | Einheitlicher Prozess für alle Veranstaltungen<br>Leichtere Pflege von Terminen |

Tabelle 6 Actor-Goal-Liste

## 3.3 Product Backlog

Sämtliche Anforderungen und Wünsche der zukünftigen Nutzer wurden in Form von sogenannten „Stories“ festgehalten. Zu jeder Story gibt es wiederum Akzeptanzkriterien, die definieren, wann die Funktion zufriedenstellend umgesetzt ist. Um den Aufwand für die einzelnen Stories möglichst realistisch einschätzen zu können, existieren verschiedene Bewertungsansätze. Im Rahmen dieser Arbeit wurde in einer Teamsitzung das „Estimation Game“ durchgeführt. Hierbei wird der Aufwand der Stories relativ zueinander geschätzt, d.h. jedes Teammitglied ordnet nacheinander eine Story ein.

Ein beispielhafter Ablauf sei hier kurz dargestellt:

1. Aus einem Stapel mit den Stories wird zunächst eine Karte aufgedeckt und in der Mitte platziert.
2. Teammitglied A zieht Story 1 von dem Stapel und legt diese entweder über (höherer Aufwand), neben (ähnlicher Aufwand) oder unter (geringerer Aufwand) die vorhandene Karte.
3. Teammitglied B kann nun entweder die Lage einer Karte gemäß seiner eigenen Einschätzung korrigieren oder eine neue Karte vom Stapel ziehen und diese wieder in die Hierarchie einordnen.

Beendet ist das „Estimation Game“, wenn alle Karten einsortiert wurden und niemand mehr Änderungen vornehmen will. Im Anschluss können noch Punktzahlen („Story Points“) vergeben werden, wobei hohe Punktzahlen einen hohen Aufwand symbolisieren. Häufig wird eine angepasste Fibonacci-Zahlenfolge zur Bewertung verwendet (1, 2, 3, 5, 8, 13, 20, 40, 100).

All diese Informationen wurden schlussendlich im „Product Backlog“ zusammengeführt. Dieses Product Backlog ist in Tabelle 7 dargestellt.

| Nr. | Story | Nähere Erläuterung | Akzeptanzkriterien | Front-End(F)/ Back-End (B) | Story Points |
|---|---|---|---|---|---|
| 1 | Bearbeitungsfunktion Veranstaltungen | Als Administrator/ Sachbearbeiter/ Veranstalter benötige ich die Funktion, Veranstaltungen zu bearbeiten, um die Veranstaltung auf dem aktuellsten Stand zu halten oder Fehler beim Anlegen der Veranstaltung zu beheben. | Alle Daten, die auch beim Anlegen der Veranstaltung angegeben werden können, können bearbeitet werden | F | 1 |
| 2 | Funktion Veranstaltung anlegen und speichern | Als Veranstalter/ Sachbearbeiter benötige ich die Funktion, Veranstaltungen anzulegen und einzureichen, um diese dem Datenpool des Veranstaltungsportals hinzuzufügen und nach Prüfung publik zu machen. | Eine Veranstaltung kann mit allen Daten, die im Konzept festgelegt wurden, angelegt werden; nur berechtigte Benutzer können Veranstaltungen anlegen | F | 13 |
| 3 | Speichern einer Veranstaltung | Als Front-End-Entwickler benötige ich die Funktion der Speicherung einer Veranstaltung, um zu gewährleisten, dass eine neu angelegte Veranstaltung oder Änderungen an einer Veranstaltung bei Betätigung eines Buttons in die Datenbank mitaufgenommen und in Zukunft entsprechend dem neuen Stand zur Verfügung gestellt werden können. | Nach Speichern einer Veranstaltung ist diese in der Datenbank vorhanden und kann weiterverwendet werden | B | 20 |
| 4 | Entfernen einer Veranstaltung | Als Front-End-Entwickler benötige ich die Funktion, eine Veranstaltung aus der Datenbank zu entfernen, um die Löschfunktion im Front-End auch im Datenbankmodell umzusetzen. | Nach Löschen einer Veranstaltung ist diese in der Datenbank nicht mehr vorhanden | B | 1 |
| 5 | Löschfunktion Veranstaltungen | Als Veranstalter/ Sachbearbeiter benötige ich die Funktion, eine Veranstaltung zu löschen, um nicht mehr stattfindende, fehlerhafte oder abgesagte Veranstaltungen aus dem Veranstaltungsportal zu entfernen. | Nach Löschen einer Veranstaltung wird diese nirgendwo in der Anwendung mehr angezeigt | F | 1 |
| 6 | Volltextsuche nach Veranstaltungen | Als Benutzer des Veranstaltungsportals benötige ich die Funktion, nur über einen Suchbegriff, ohne die Auswahl von Filtern nach einer Veranstaltung zu suchen, um so simpel wie möglich eine bestimmte Veranstaltung finden zu können. | Die passenden Ergebnisse werden entsprechend der Suche angezeigt | F | 8 |
| 7 | Gefilterte Suche von Veranstaltungen | Als Benutzer des Veranstaltungsportals benötige ich die Möglichkeit, nach einer Veranstaltung gefiltert zu suchen, um eine Veranstaltung schneller finden zu können. | Die passenden Ergebnisse werden entsprechend der Suche angezeigt | F | 8 |

| | | | | | |
|---|---|---|---|---|---|
| 8 | Detailansicht der Veranstaltung | Als Benutzer des Veranstaltungsportals benötige ich eine Detailansicht für Veranstaltungen, um alle nötigen Information über die Veranstaltung beziehen zu können. | Alle in der Konzeption festgelegten Details der Veranstaltung werden angezeigt | F | 2 |
| 9 | Route zur Veranstaltung anzeigen | Als Benutzer des Veranstaltungsportals benötige ich die Funktion, mir die Route zu einem Veranstaltungsort anzeigen zu lassen, um zu wissen wie der Weg zu der Veranstaltung aussieht. | Die Route wird benutzerfreundlich und korrekt in Form einer Karte angezeigt | F | 5 |
| 10 | Designrahmen aufsetzen | Als Benutzer des Veranstaltungsportals benötige ich einen einheitlichen Designrahmen, um benutzerfreundlich durch das Portal zu navigieren. | Menü, Startseite und Navigation werden entsprechend der Konzeption vollständig angezeigt | F | 13 |
| 11 | Ticketverkauf Weiterleitung | Als Benutzer des Veranstaltungsportals benötige ich die Funktion einer Weiterleitung zum Ticketverkauf oder Veranstalter der Veranstaltung, damit ich nach Erhalt der Infos zur Veranstaltung Tickets kaufen oder mich weiter informieren kann. | Die Weiterleitungen zum Ticketverkauf bzw. dem Veranstalter funktioniert bei allen Veranstaltungen | F | 1 |
| 12 | Newsletter Abonnierbutton einrichten | Als Benutzer des Veranstaltungsportals benötige ich die Funktion den Newsletter des Veranstaltungsportals zu abonnieren, um in Zukunft darüber weitere Informationen zu erhalten. | Der Benutzer kann den Abonnierbutton betätigen und erhält daraufhin eine Bestätigung und in Zukunft folgende Newsletter per Mail | F | 1 |
| 13 | Parkplatzanzeige für die Veranstaltungen | Als Benutzer des Veranstaltungsportals benötige ich die Funktion, mir die passenden Parkplätze für die Veranstaltung anzeigen zu lassen, um Informationen über die Parkplätze zu erhalten. | Die Parkplätze einer Veranstaltung werden korrekt auf einer Karte mit allen festgelegten Informationen angezeigt | F | 5 |
| 14 | Berechtigungskonzept umsetzen | Als Administrator/IT Mitarbeiter brauche ich ein umgesetztes Benutzerkonzept im Back-End, um die Zugriffe und Bearbeitungsrechte einzugrenzen und zu organisieren. | Verschiedene Benutzer haben entsprechend eingegrenzte und unterschiedliche Verhaltensmöglichkeiten | B | 20 |
| 15 | Benutzerkontenregistrierung im Front-End aufnehmen | Als Benutzer des Veranstaltungsportals benötige ich eine Registrierfunktion um meine benutzerspezifischen Informationen und Funktionen freizuschalten. | Man kann sich auf verschiedenen Seiten der Anwendung registrieren | F | 8 |
| 16 | Benutzerabhängige Favoritenauswahl einrichten | Als Front-End-Entwickler benötige ich die Funktion die Favoritenauswahl eines Benutzers speichern und nach Anmeldung des Nutzers abrufen zu können. | Allen Benutzern sind die passenden Favoriten zugewiesen | B | 20 |

| | | | | | |
|---|---|---|---|---|---|
| 17 | Anzeige der Favoritenauswahl | Als Benutzer des Veranstaltungsportals benötige ich die Funktion, nach Anmeldung mit einem Benutzerkonto favorisierte Veranstaltungen zu setzen und anzeigen zu lassen. | Die Favoriten eines Benutzers werden benutzerfreundlich angezeigt | F | 2 |
| 18 | Veranstaltungsvorschläge für Benutzer generieren (Machine Learning) | Als Front-End-Entwickler benötige ich benutzerspezifische Daten um diese dem Benutzer auf der Startseite anzuzeigen. | Veranstaltungsvorschläge werden für die Benutzer passend generiert (abhängig von der Art des Benutzers) | B | 40 |
| 19 | Veranstaltungsvorschläge für Benutzer auf Veranstaltungsstartseite anzeigen | Als Benutzer benötige ich die Funktion, mir Veranstaltungsvorschläge anzeigen zu lassen, so dass mir automatisch interessante und aktuelle Informationen angezeigt werden können. | Veranstaltungsvorschläge werden benutzerfreundlich dargestellt | F | 2 |
| 20 | Datenbanktabellen erstellen | Als Entwickler benötige ich Datenbanktabellen gemäß des konzipierten Modells um Daten speichern und abrufen zu können. | Alle Datenbanktabellen wurden gemäß dem konzipierten Modell angelegt | B | 2 |
| 21 | Datenzugriff implementieren | Als Front-End-Entwickler benötige ich die Möglichkeit, auf Daten aus dem Datenmodell zuzugreifen, um andere Funktionen aus dem Product Backlog umsetzen zu können. | Datenzugriff aus dem Front-End funktioniert | B | 13 |
| 22 | Verarbeitung der Änderungen vom Front-End | Als Front-End-Entwickler benötige ich die Funktion Änderungen an Daten im Front-End im Back-End weiterzugeben, um diese auch weiterhin zur Verfügung stellen zu können. | Änderungen aus dem Fontend werden im Back-End gespeichert | B | 40 |
| 23 | Oberfläche für Sachbearbeiter | Als Sachbearbeiter benötige ich eine Oberfläche, um wichtige Aktionen, wie das Freigeben von Veranstaltungen durchführen zu können. | Freischalten von Veranstaltungen (& evtl. Sperren von Benutzern) | F | 13 |
| 24 | Anzeigen des Benutzerprofils | Als Benutzer des Veranstaltungsportals möchte ich, dass mir eine Profilübersicht angezeigt wird über die ich bspw. mein Passwort oder meine E-Mail Adresse ändern kann. | Seite zur Anzeige des Profils; Anzeigen der Ergebnisse aus dem Machine Learning | F | 8 |
| 25 | Machine Learning für Mitarbeiter der Stadt Mosbach | Als Sachbearbeiter der Stadt Mosbach benötige ich Machine Learning Funktionen, um Vorhersagen treffen zu können und den Verlauf der Veranstaltungen zu analysieren. | Zusätzliche Funktionen mit Auswertungen auf der Oberfläche für Sachbearbeiter der Stadt | F/B | 40 |

Tabelle 7 Product Backlog

## 3.4 Use-Case Diagramm

Abbildung 1 zeigt die identifizierten Use-Cases sowie die beteiligten Akteure.

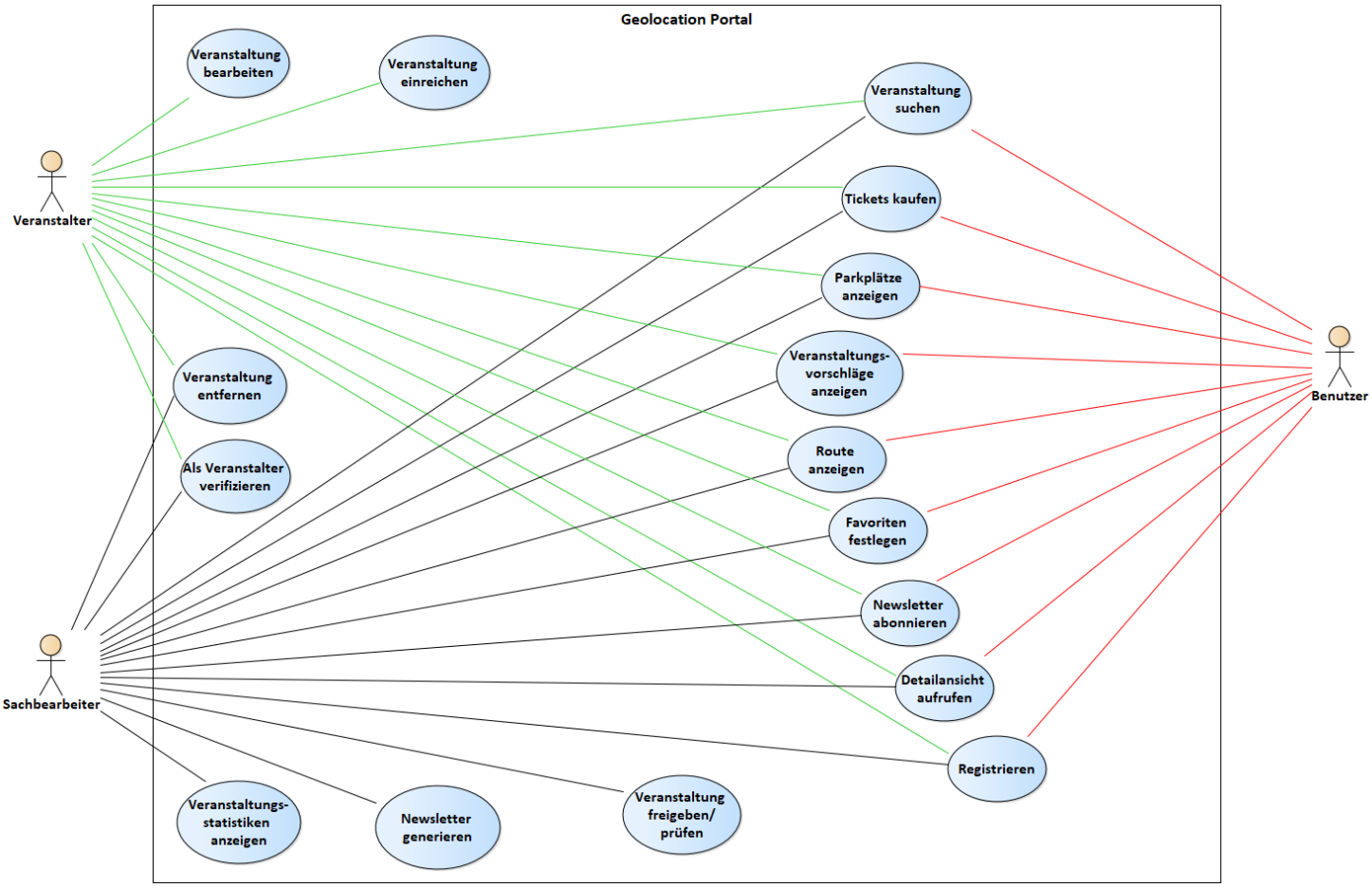



Abbildung 1 Use-Case-Diagramm

Im Abschnitt „Anwendungsfälle“ des Anhangs sind die einzelnen Use-Cases näher erläutert.

## 3.5 Domänenmodell

In Abbildung 2 ist ein Domänenmodell dargestellt. Dieses zeigt den Anwendungszusammenhang des Prototyps. Entsprechend der Projektvision bezieht sich das Modell vor allem auf das Zusammenspiel der Veranstaltungen mit verschiedenen Akteuren. Hierbei ist beispielsweise die Domäne Veranstaltung zu erkennen. Sie kann von mehreren Personen besucht werden und gehört zu einer oder auch mehreren Kategorien, die von einer Person als Favorit gekennzeichnet werden können. Zu einer Veranstaltung gehören die jeweiligen Veranstaltungsdetails und jede Veranstaltung findet an einem bestimmten Veranstaltungsort statt. Dieser befindet sich in einem Ort, der wiederum über diverse Parkplätze verfügen kann. In diesem Modell gibt es unterschiedliche Ausprägungen der Domäne Person. Abgeleitet von der Stakeholder Analyse und der Actor-Goal-List erkennt man Sachbearbeiter, Veranstalter, Benutzer und Administrator als Spezialisierungen.

Der Sachbearbeiter generiert Newsletter und prüft Veranstaltungen, die von einem Veranstalter erstellt wurden. Benutzer registrieren sich im System des Prototyps und nehmen entsprechend der geerbten Eigenschaften der Person an einer oder mehreren Veranstaltungen teil.

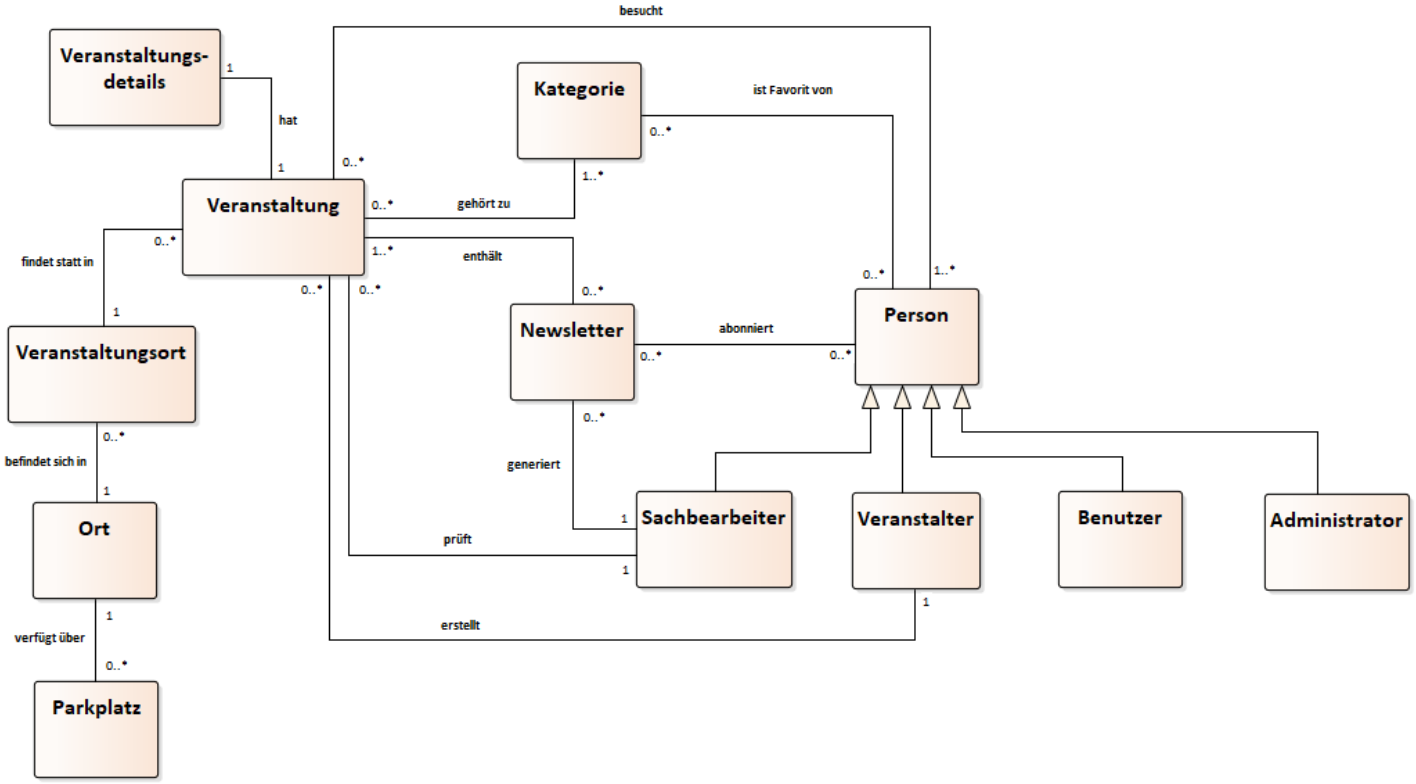



Abbildung 2 Domänenmodell

## 3.6 Klassendiagramm

Basierend auf dem Domänenmodell wird mit dem Klassendiagramm in Abbildung 3 die Struktur des Prototyps weiter vertieft. Hier sind die Klassen, ihre Eigenschaften und ihre Beziehungen untereinander detailliert dargestellt.

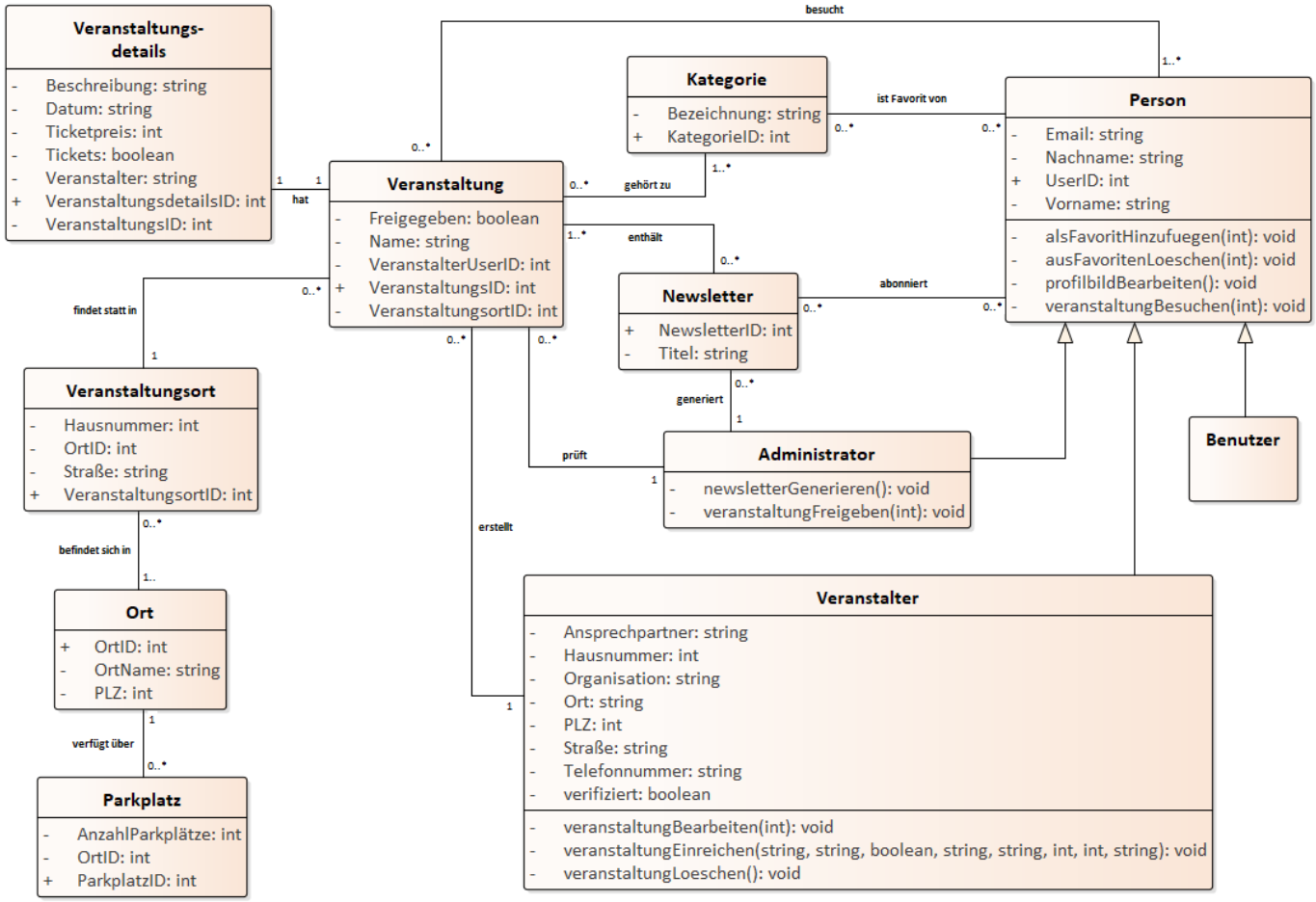



Abbildung 3 Klassendiagramm

## 3.7 Oberflächendesign

Das Design des Prototyps wurde mit der Software Balsamiq verwirklicht[3] (vgl. Anhang Kapitel Oberflächendesign). Durch die Software wurden Wireframes erstellt, die dem Entwicklungsteam als Grundlage für das Front-End dienten. Bei der Erstellung erster graphischer Entwürfe standen die Gestaltungsprinzipien der Norm 9241-110 im Mittelpunkt. Folgende Faktoren werden darüber definiert[4]:

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit

[3] Vgl. Balsamiq Wireframes | Balsamiq 2020.

[4] Vgl. Grundsätze der Dialoggestaltung nach DIN EN ISO 9241-110 - Grundsätze der Dialoggestaltung nach DIN EN ISO 9241-110 2020.

- Erwartungskonformität
- Lernförderlichkeit
- Steuerbarkeit
- Fehlertoleranz
- Individualisierbarkeit

Hierbei gilt es zu beachten, dass jedes Prinzip ein anderes Prinzip abschwächt. Somit können nicht alle sieben Prinzipien in gleichem Maße angewendet werden. Bei dem Prototyp lag der Fokus auf der Aufgabenangemessenheit, der Steuerbarkeit und der Fehlertoleranz. Wird beispielsweise das erstgenannte Prinzip, Aufgabenangemessenheit, betrachtet, so wurde versucht, die Dialoge entsprechend der Aufgaben des Benutzers zu gestalten. Des Weiteren wird mit Fehlermeldungen gearbeitet. Ein Beispiel dafür ist das Anzeigen einer Meldung, wenn der Benutzer vergisst, Felder, die als Pflichtfelder gekennzeichnet sind, auszufüllen. Durch die dauerhafte Möglichkeit, einen Prozess abzubrechen, hat der Benutzer des Prototyps immer die Kontrolle. Dies erfüllt das Prinzip der Steuerbarkeit.

## 3.8 Berechtigungskonzept

Der Prototyp enthält verschiedene Funktionen, die durch ein Berechtigungskonzept geplant und kontrolliert werden. Durch dieses wird sichergestellt, dass jeder Benutzer die Zugriffsrechte bekommt, die er benötigt, um vorgesehene Aufgaben erfüllen zu können[5]. Das Konzept (vgl. Tabelle 8) unterteilt die verschiedenen Benutzer in fünf Benutzertypen.

| **Benutzertyp** | **Beschreibung** |
|---|---|
| Gast | Ruft die Webseite oder App auf, ohne sich anzumelden. Kann nur die Suche benutzen. |
| Benutzer | Ruft die Webseite oder App auf und loggt sich ein. Kann Favoriten anlegen und weitere Features nutzen. |
| Veranstalter | Ruft die Webseite oder App auf und loggt sich ein. Kann Veranstaltungen erstellen, bearbeiten und entfernen. |
| Sachbearbeiter | Ein/-e Mitarbeiter/-in der Stadt. Zuständig für Benutzerpflege und die Veranstaltungen. |
| Administrator | Ein/-e Mitarbeiter/-in der Stadt. Zuständig für die Wartung, Updates oder Fehlerbehebung. |

Tabelle 8 Beschreibung der Benutzertypen

[5] Vgl. Berechtigungskonzept | ISDSG - Institut für Sicherheit und Datenschutz im Gesundheitswesen.

Beispielhaft werden die Benutzertypen „Gast“ und „Veranstalter“ näher beschrieben:
Ein Gast bezeichnet jede Person, die den Prototyp nutzt, ohne sich zu registrieren. Dies können Personen sein, die sich über Veranstaltungen informieren möchten oder an diesen teilnehmen wollen. Sobald ein Gast eine Veranstaltung besuchen möchte, für die der Kauf von Tickets erforderlich ist, muss er sich registrieren. Durch eine Registrierung ändert sich der Benutzertyp von Gast zu Benutzer.
Als Veranstalter werden diejenigen Anwender bezeichnet, die mithilfe des Systems Veranstaltungen erstellen und durchführen. Jeder Veranstalter war vorher vom Typ Benutzer. Durch eine Verifizierung kann er ein Veranstalter werden (vgl. Kapitel 3.9). Dies berechtigt ihn, für eine bestimmte Organisation (z.B. Fußballverein, Tennisverein, etc.) Veranstaltungen zu erstellen.

## 3.9 Verifizierung als Veranstalter

In der folgenden Abbildung 4 ist ein Aktivitätsdiagramm für den Verifizierungsprozess eines Veranstalters dargestellt.

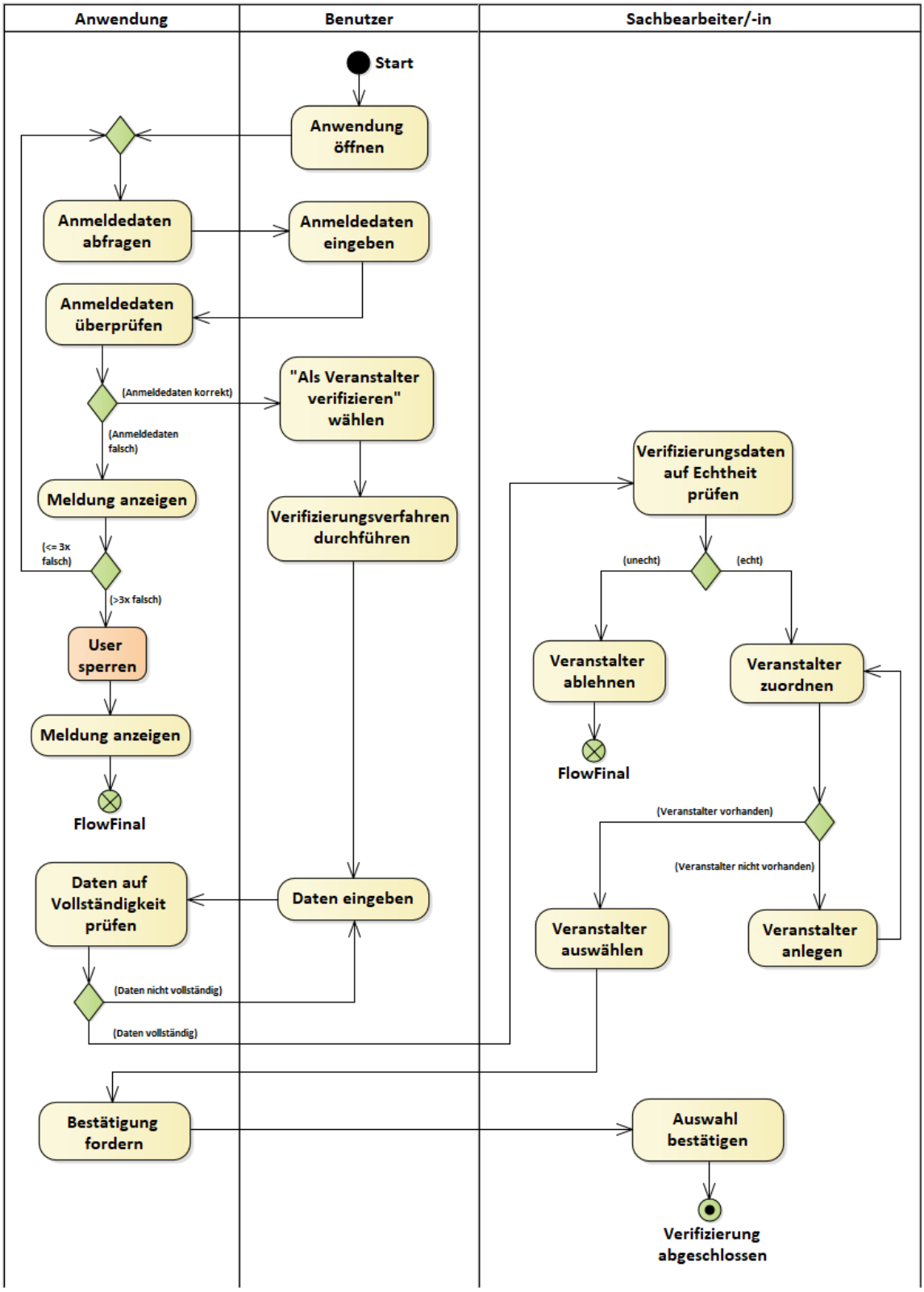



Abbildung 4 Aktivitätsdiagramm Veranstalterverifizierung

Jeder neue Benutzer, der zu einem bereits vorhandenen oder neuen Veranstalter gehört, muss zunächst verifiziert werden. Wie dies abläuft ist im Aktivitätsdiagramm schematisch dargestellt. Wie zu erkennen ist, muss ein Benutzer bereits registriert sein, um sich als Veranstalter verifizieren zu können. Für das Verifizierungsverfahren wurden verschiedene Varianten anderer Städte evaluiert. Teilweise ist es kostenpflichtig, Veranstaltungen einzustellen, während bei anderen eine redaktionelle Prüfung oder eine Bestätigung der Veranstaltung per E-Mail durchgeführt wird. Im Fall des Geolocation-Portals wurde sich für die Hinterlegung eines Fotos des Personalausweises entschieden. Hinter dem Vorgang "Verifizierungsverfahren durchführen" des Aktivitätsdiagramm verbirgt sich demnach die Angabe verschiedener Daten in einem Formular. Bei diesen Daten handelt es sich um den Namen, einen Veranstalter, dem der Benutzer zugeordnet werden soll und den Personalausweis, den der Sachbearbeiter überprüfen kann. Hat der Benutzer diese Daten eingegeben wird das Verfahren entsprechend der Abbildung weitergeführt.

# 4 Entwicklung des Prototyps

In diesem Kapitel werden technische Details zur Entwicklung des prototypischen Geolocation Portals näher erläutert. Hierzu wird zuerst die Datenbankarchitektur anhand eines Entity Relationship Modells (ERM) dargestellt, bevor auf die Implementierung eingegangen wird. Bei der Entwicklung wird zwischen Front-End und Back-End Entwicklung unterschieden. Dies liegt darin begründet, dass die Entwicklung des Front-End auf das Layout und die clientseitige Entwicklung beschränkt ist, während das Back-End die Server- und Datenbankentwicklung übernimmt.

## 4.1 Datenbankarchitektur

Als Grundlage für die Implementierung des Prototyps dient eine MySQL Datenbank. Diese bietet sich als kostenlose relationale Datenbank für den Verlauf der Entwicklung an. Außerdem steht den Entwicklern hier ein Arbeitsbereich zur Verfügung, der die Administration der Datenbank und ihrer Tabellen erleichtert. Welche Tabellen die Datenbank umfasst ist in folgendem ERM zu sehen (vgl. Abbildung 5). Es handelt sich um die verschiedenen Entitäten und deren Zwischentabellen.

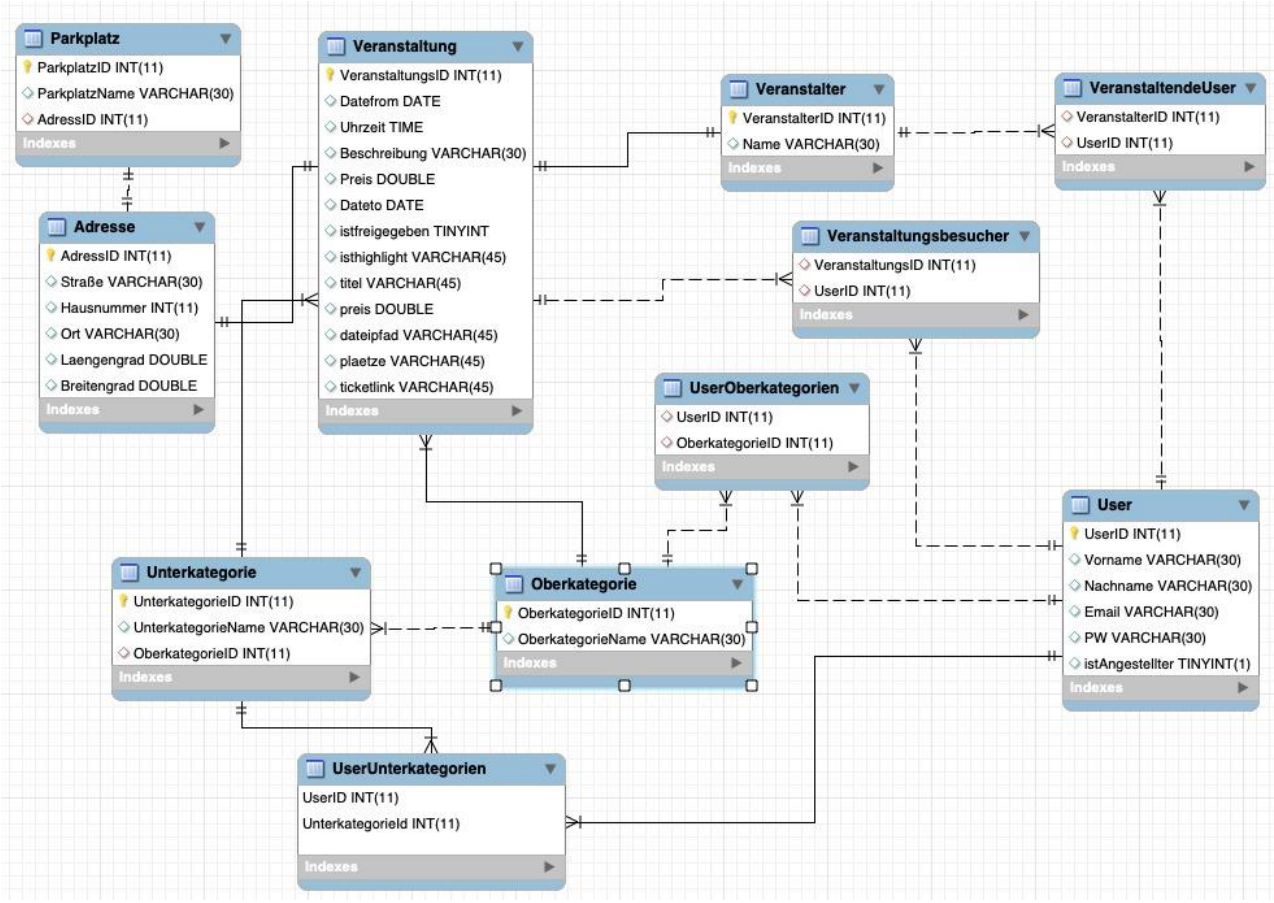



Abbildung 5 Entity Relationship Modell

## 4.2 Weiterführende Implementierung des Back-Ends

Neben der Einrichtung der Datenbank und deren Administration ist auch die Bereitstellung der Daten für das Front-End eine Aufgabe des Back-Ends. Hierbei spielt zum Beispiel die Einrichtung des Servers eine wichtige Rolle. Um diese Aufgaben zu bewältigen und dem Front-End die benötigten Informationen zur Verfügung zu stellen, wurden verschiedene Technologien verwendet, die im Folgenden näher erläutert werden.

Zum einen wird das Framework „Hibernate“, speziell „Hibernate 5.2“ verwendet, das bei der Verbindung zur Datenbank unterstützt. Es wird auch als „Object Relational Mapping Tool“ bezeichnet[6].

Zum anderen kommt bei der Implementierung im Back-End „Spring 5“ zum Einsatz. Dieses Framework vereinfacht vor allem das Ausführen der Applikation[7] durch die Simplifikation des Umgangs mit verschiedenen Bibliotheken und dem entsprechenden Webserver. Bei dem Webserver handelt es sich um einen Tomcat Server[8].

In Form von verschiedenen Application Programming Interfaces (API) werden die Daten dann dem Front-End zur Verfügung gestellt. Das heißt durch den Aufruf eines Links mit variierendem Anhang können unterschiedliche Informationen abgerufen werden. Welche APIs das Back-End dem Front-End zur Verfügung stellt, ist im Anhang zu finden (vgl. Anhang Kapitel „API Liste“).

Für jede Entität gibt es einen Rest-Controller, für den die APIs der entsprechenden Entität gepflegt sind, dazu zählen die CRUD-Operationen, „findAll()“, APIs um vorhandene APIs zu verknüpfen, APIs um neue Entitäten anzulegen und direkt mit einer vorhandenen Entität zu verknüpfen, sowie spezielle APIs, um die verknüpften Entitäten abzurufen. Verwendet werden sowohl Request Body als auch Path Variablen. Den Weg der Verarbeitung im Back-End, wie es also von der Tabelle in einer Datenbank zur Bereitstellung der Daten durch eine API kommt, zeigt Abbildung 6.

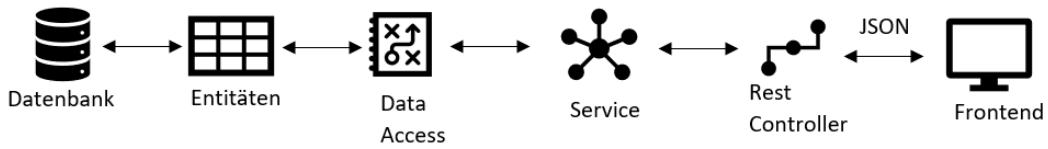



Abbildung 6 Schichtenmodell Back-End

Hierbei werden verschiedene Packages benötigt beziehungsweise Schichten durchlaufen (vgl.

[6] Vgl. Hibernate :: hibernate :: ITWissen.info.

[7] Vgl. Spring Boot - was ist das, was kann das?, 2013.

[8] Vgl. Project - Apache Tomcat, 2020.

*Abbildung* 6 Schichtenmodell Back-End6):

- com.frederik.springboot.cruddemo
- com.frederik.springboot.cruddemo.dao
- com.frederik.springboot.cruddemo.entity
- com.frederik.springboot.cruddemo.service
- com.frederik.springboot.cruddemo.rest

Wird im Front-End also eine API aufgerufen, wird im Back-End im Package „rest“ der entsprechende Rest Controller aufgerufen. Außerdem sind hier spezifische Fehlerklassen, mit speziellen Fehlermeldungen und entsprechenden Fehlerbehandlungsklassen zu finden. Vom Rest Controller geht der Zugriff auf den Service, wobei hier erste Prüfungen und Fehlerkontrollen stattfinden. In dem jeweiligen Service werden Funktionen aus den Data Access Objects transaktional aufgerufen. In diesen Data Access Objects wird das Framework Hibernate das erste Mal verwendet, um auf die Entitäten entsprechend der Datenbanktabellen zuzugreifen. Im folgenden Package „entity“ werden dann Entitäten (Adresse, Login, Oberkategorie, Parkplatz, Request, Unterkategorie, User, Veranstalter, Veranstalterverifizierung und Veranstaltung) mit ihren Attributen und Methoden, sowie ihren Abhängigkeiten, wie one-to-one, one-to-many oder many-to-many untereinander festgelegt. Jede Entität durchläuft diese beschriebene Package-Hierarchie.
In der Datei „Pom.xml“ werden wichtige Abhängigkeiten des Back-Ends definiert, während in der Datei „application.properties“ allgemeine Einstellungen des Back-Ends, wie zum Beispiel der Datenbanktreiber, gepflegt werden.

Ziel dieses Schichtenmodells ist es eine Micro Service Architektur bereitzustellen. Diese bietet den Vorteil, flexibler an den Services zu arbeiten und diese leichter durch neu ausgearbeitete Funktionen zu ersetzen, ohne dass die anderen Services oder das Front-End davon betroffen sind. Hierdurch ist auch die Datenbank leichter zu ersetzen oder zu überarbeiten.

## 4.3 Implementierung im Front-End

In diesem Unterkapitel wird die Vorgehensweise, sowie die dazu verwendete Software und ihre Zusammenhänge bei der Entwicklung des Front-Ends näher beschrieben.
Das Ziel bei der Entwicklung des Front-Ends ist es, eine webbasierte Oberfläche bereitzustellen, die sowohl im Browser als auch auf mobilen Endgeräten darstellbar ist. Dies wird auch als Responsive Design bezeichnet.
Zur Erreichung dieses Ziels verwendet das Entwicklungsteam die Programmiersprache Java. Die Entwickler stützen sich dabei auf das „JavaServer Faces“ Framework, kurz JSF, für Java-basierte Webanwendungen[9]. Ein weiteres Framework namens „PrimeFaces“ dient hierbei als Unterstützer, da es auf bewährten Technologien und Standards wie „Ajax“, dem „Atmosphere“ Framework und „JQuery“ basiert und diese in einer Java Bibliothek zusammenführt[10]. Mit Hilfe von „Maven“, welches auch im Back-End verwendet wird kann diese Bibliothek als unabhängige JAR-Datei eingebunden werden. Zur Implementierung von JSF wird „MyFaces“ der Apache Software Foundation benutzt.
Auf Grundlage der vorhandenen Wireframes werden die verschiedenen Seiten der Anwendung gestaltet. Die Views sind in XHTML programmiert. Aufbauend auf dem XHTML Code wurde CSS-Code, zur farblichen Gestaltung der Views, implementiert. Die Logik der Seiten wird mit Java und JavaScript umgesetzt.

[9] Vgl. Salvanos 2018, S. 757.

[10] Vgl. PrimeFaces - Webanwendungen für Desktop und mobile Geräte auf Basis von Java und JSF - Business -Software- und IT-Blog - Wir gestalten digitale Wertschöpfung 2016.

# 5 Testen

Um die Qualität des Prototyps gewährleisten zu können, müssen die Funktionen sowohl während als auch nach der Entwicklung regelmäßig getestet werden.

Während der Entwicklungsphasen beziehungsweise der Sprints werden die Funktionen deshalb bei verschiedenen Treffen mit der Projektgruppe getestet. In diesem Zusammenhang werden Fehler in der Logik und wichtige Abweichungen vom Konzept besprochen. Außerdem wird unter den Entwicklern das "Vier-Augen-Prinzip" angewendet. Dieses beschreibt, dass die Entwickler untereinander die implementierten Funktionen der jeweils anderen auf mögliche Fehler testen[11]. Die APIs des Back-Ends beispielsweise wurden durch das Programm „Postman“ auf die richtige Ausgabe der Daten getestet. Das Front-End wurde vor allem anhand des typischen Ablaufs, aber auch anhand möglicher Abweichungen der gesammelten Anwendungsfälle überprüft.

Zusätzlich zu diesen Verfahren werden die Funktionen von Projektmitgliedern getestet, die nicht direkt an der Entwicklung beteiligt sind. Auf diese Weise ist es möglich, die Anwendung aus einer weiteren Sichtweise zu überprüfen.

Sämtliche Testprotokolle finden sich im Abschnitt „Testprotokolle“ des Anhangs.

**11** Vgl. Bitkom e.V.: Entwicklung erfolgreicher Webanwendungen, 2015

# 6 Schlussbemerkungen

## 6.1 Fazit

Letztendlich konnte im Projekt, rund um die Aufgabenstellung der Vorlesung "Neue Konzepte Projekt I + II" ein funktionierender Prototyp eines Geolocation Portals für die Stadt Mosbach umgesetzt werden. Der Prototyp basiert auf Open Data und stellt in diesem Zusammenhang eine Menge von veranstaltungsrelevanten Daten bereit, die von Anwendern frei genutzt werden können. Darüber hinaus stellt der Prototyp interaktive Elemente zur Verfügung, durch die Nutzer neue Daten hinzufügen und verbreiten können.

## 6.2 Kritische Betrachtung

Die Konstellation von Projektgruppen bringt Vor- und Nachteile mit sich. Während es zum einen zu innovativen neuen Idee kommt, bietet eine Projektarbeit in Gruppen zum anderen auch stets Potential für Konflikte, sowohl in zwischenmenschlicher als auch in organisatorischer und technischer Hinsicht.

Die Organisation und Vorbereitung der Termine war immer gut durchdacht und geplant. Trotzdem kam es vor allem während der Weihnachtszeit zu höheren Abwesenheiten als vermutet. Dies führte wiederum zu mehr Aufwand der Anwesenden bei der Koordination der Aufgabenverteilung. Allerdings wurden alle Aufgaben stets termingetreu und gewissenhaft erarbeitet. Dies verhalf allen Projektbeteiligten zu einer guten Zusammenarbeit.

Des Öfteren kam es zu Detail-Diskussionen, die für das Endergebnis eher zweitrangig waren, was zu unnötigem Zeitverlust und Unstimmigkeiten in der Gruppe führte. Jedoch konnten diese durch das Einführen der Daily-Meetings vermieden bzw. umgangen werden. Dieses Vorgehen führte außerdem zu einer besseren Kooperation im Team.

Auch die Aufteilung in ein Entwickler- und ein Dokumentationsteam führte dazu, dass jedem Gruppenmitglied Aufgaben zugeteilt werden konnte und jeder genau wusste was er zu erarbeiten hat. Darüber hinaus konnte das Entwicklerteam durch die räumliche Nähe sehr gut und harmonisch zusammenarbeiten.

## 6.3 Ausblick und Weiterentwicklungsmöglichkeiten

Der aktuelle Stand des Prototyps bietet verschiedene Möglichkeiten zur Weiterentwicklung und Nutzung über die bereits konzipierten Anwendungsfälle hinaus.

Zum einen bietet die Webanwendung das Potenzial Abläufe für Mitarbeiter der Stadt Mosbach zu vereinfachen, so dass der interne Prozess rund um die Beantragung und Organisation einer Veranstaltung mittels der hier vorgestellten Anwendung digitalisiert und automatisiert werden kann.

Zum anderen lässt der Prototyp Raum für die Entwicklung weiterer Funktionen:

- Nutzung der Funktionen des Veranstaltungsportals für weitere Anwendungsgebiete (zum Beispiel: Übersicht kulinarische Einrichtungen – Restaurants)
- Suchen vergangener Veranstaltungen
- Periodische Veranstaltungen in regelmäßigen Abständen
    - Regelmäßige Abstände beim Anlegen angeben
    - Damit es bei der Suche nach Veranstaltungen nicht öfter angezeigt wird, werden Veranstaltungen mit der gleichen ID (wenn sie also periodisch sind) nur einmal angezeigt
- Algorithmus, um ähnliche Veranstaltungen vorzuschlagen, muss verbessert werden
- Möglichkeiten, für Veranstaltungen zuzusagen (Kalkulation für Veranstalter & Benutzerprofil mit Erinnerung)
- Bewertungsmöglichkeiten für Veranstaltungen
- Nicht umgesetzte Anwendungsfälle und Storys:
    - Exportieren von Veranstaltungen in gängige Dateiformate (Bsp.: Excel) und teilen beispielsweise von Suchergebnissen durch Weitergabe des Links
    - Bearbeitungsfunktion Veranstaltungen (als Administrator/Sachbearbeiter/Veranstalter wird die Funktion Veranstaltungen zu bearbeiten benötigt, um die Veranstaltung auf dem aktuellsten Stand zu halten oder Fehler beim Anlegen der Veranstaltung zu beheben)
    - Parkplatzanzeige der Veranstaltungen
    - Benutzer sollen Veranstaltungen als Favoriten abonnieren können
    - Machine Learning (aus vorliegenden Daten sollen Veranstaltungsvorschläge für Benutzer und Statistiken für Sachbearbeiter der Stadt generiert werden)
    - Benutzerverwaltung über Sachbearbeiteroberfläche
    - Route zur Veranstaltung anzeigen
    - Eingeben der Adresse beim Anlegen der Veranstaltung, statt den Standort ausschließlich auf der Karte auswählen zu können

# Literaturverzeichnis

Balsamiq Wireframes | Balsamiq (2020). Online verfügbar unter https://balsamiq.com/wireframes/, zuletzt aktualisiert am 24.01.2020, zuletzt geprüft am 25.01.2020.

Berechtigungskonzept | ISDSG - Institut für Sicherheit und Datenschutz im Gesundheitswesen. Online verfügbar unter https://www.isdsg.de/beratung/konzepterstellung/berechtigungskonzept, zuletzt geprüft am 24.01.2020.

Grundsätze der Dialoggestaltung nach DIN EN ISO 9241-110 - Grundsätze der Dialoggestaltung nach DIN EN ISO 9241-110 (2020). Online verfügbar unter https://www.ergo-online.de/ergonomie-und-gesundheit/software/dialoggestaltung/artikel/grundsaetze-der-diashylogshygestaltung-nach-din-en-iso-9241-110/grundsaetze-der-dialoggestaltung-nach-din-en-iso-9241-110/, zuletzt aktualisiert am 17.01.2020, zuletzt geprüft am 25.01.2020.

Hibernate :: hibernate :: ITWissen.info. Online verfügbar unter https://www.itwissen.info/Hibernate-hibernate.html, zuletzt geprüft am 24.01.2020.

Meindl, Claudia (2012): Schätzen in agilen Projekten. Online verfügbar unter https://alphanodes.com/de/schaetzen-agilen-projekten, zuletzt aktualisiert am 24.01.2020, zuletzt geprüft am 25.01.2020.

PrimeFaces - Webanwendungen für Desktop und mobile Geräte auf Basis von Java und JSF - Business -Software- und IT-Blog - Wir gestalten digitale Wertschöpfung (2016). Online verfügbar unter https://blog.doubleslash.de/primefaces/, zuletzt aktualisiert am 23.01.2020, zuletzt geprüft am 24.01.2020.

Prof. Dr. Dirk Palleduhn (2019): Aufgabenstellung DHBW.

Project, Apache Tomcat (2020): Apache Tomcat® - Welcome! Online verfügbar unter https://tomcat.apache.org/index.html, zuletzt aktualisiert am 06.01.2020, zuletzt geprüft am 24.01.2020.

Salvanos, Alexander (2018): Professionell entwickeln mit Java EE8. Das umfassende Handbuch. 2., aktualisierte und erweiterte Auflage. Bonn: Rheinwerk (Rheinwerk Computing).

Spring Boot - was ist das, was kann das? (2013). Online verfügbar unter https://jaxenter.de/spring-boot-2279, zuletzt aktualisiert am 25.01.2020, zuletzt geprüft am 25.01.2020.

V, Bitkom e.: Entwicklung erfolgreicher Webanwendungen, zuletzt geprüft am 24.01.2020.

# Anhang

## Anhang-Verzeichnis

# Anhang-Abbildungsverzeichnis

# Anwendungsfälle

In folgenden Tabellen werden die Anwendungsfälle des Prototyps näher erläutert:

Veranstaltung bearbeiten

| **Name:** | **Veranstaltung bearbeiten** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Die Daten einer bestehenden Veranstaltung werden durch den Veranstalter bearbeitet. |
| **Auslöser:** | Aufruf des Links „Veranstaltung bearbeiten“ |
| **Akteure:** | Veranstalter |
| **Eingehende Information:** | - Veranstaltungsname<br>- Adresse (Straße, Hausnummer, PLZ, Ort)<br>- Tickets benötigt ja/nein<br>- Datum<br>- Kategorie<br>- evtl. zusätzliche Anmerkungen |
| **Vorbedingungen:** | - Benutzer ist angemeldet<br>- Benutzer ist als Veranstalter verifiziert<br>- Benutzer hat den Link „Veranstaltung bearbeiten“ angeklickt |
| **Nachbedingungen:** | - Alle Pflichtfelder des Formulars wurden ausgefüllt<br>- Benutzer hat den Button „bestätigen“ betätigt |
| **Essenzielle Schritte:** | 1. Benutzer klickt Link „Veranstaltung bearbeiten“ an<br>2. Benutzer ändert Daten<br>3. Benutzer betätigt Button „bestätigen“ |

Veranstaltung einreichen

| **Name:** | **Veranstaltung einreichen** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Es wird ein Antrag für eine Veranstaltung vom Veranstalter zur Genehmigung/Bestätigung eingereicht. |
| **Auslöser:** | Aufruf des Links „Veranstaltung einreichen“ |

| **Name:** | **Veranstaltung einreichen** |
|---|---|
| **Akteure:** | Veranstalter |
| **Eingehende Information:** | - Veranstaltungsname<br>- Adresse (Straße, Hausnummer, PLZ, Ort)<br>- Tickets benötigt ja/nein<br>- Datum<br>- Kategorie<br>- evtl. zusätzliche Anmerkungen |
| **Vorbedingungen:** | - Benutzer ist angemeldet<br>- Benutzer ist als Veranstalter verifiziert<br>- Benutzer hat den Link „Veranstaltung einreichen“ angeklickt |
| **Nachbedingungen:** | - Alle Pflichtfelder des Formulars wurden ausgefüllt<br>- Benutzer hat den Button „bestätigen“ betätigt |
| **Essenzielle Schritte:** | 1. Benutzer klickt Link „Veranstaltung einreichen“ an<br>2. Benutzer gibt benötigte Informationen ein<br>3. Benutzer betätigt Button „bestätigen“ |

Veranstaltung entfernen

| **Name:** | **Veranstaltung entfernen** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Eine bestehende Veranstaltung wird gelöscht. |
| **Auslöser:** | Aufruf des Links „Veranstaltung entfernen“ |
| **Akteure:** | Veranstalter, Sachbearbeiter |
| **Eingehende Information:** | - Veranstaltungsname |
| **Vorbedingungen:** | - Benutzer ist angemeldet<br>- Benutzer ist als Veranstalter verifiziert<br>- Benutzer hat den Link „Veranstaltung entfernen“ angeklickt |
| **Nachbedingungen:** | - Benutzer hat Sicherheitsabfrage bestätigt |

| **Name:** | **Veranstaltung entfernen** |
|---|---|
| **Essenzielle Schritte:** | 1. Benutzer ruft Liste mit seinen Veranstaltungen auf<br>2. Benutzer wählt zu löschende Veranstaltung aus<br>3. Benutzer klickt auf Link „Veranstaltung entfernen“<br>4. Benutzer bestätigt Sicherheitsabfrage<br>5. Benachrichtigung wird an Mitarbeiter der Stadt versandt |

Veranstaltung freigeben/prüfen

| **Name:** | **Veranstaltung prüfen/freigeben** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Ein Mitarbeiter der Stadt prüft neu eingereichte Veranstaltungen, bevor diese veröffentlicht werden. |
| **Auslöser:** | Neue/geänderte Veranstaltung wurde eingereicht |
| **Akteure:** | Sachbearbeiter |
| **Eingehende Information:** | - Veranstaltungsname<br>- Adresse (Straße, Hausnummer, PLZ, Ort)<br>- Tickets benötigt ja/nein<br>- Datum<br>- Kategorie<br>- evtl. zusätzliche Anmerkungen |
| **Vorbedingungen:** | - Sachbearbeiter ist angemeldet<br>- Neue/geänderte Veranstaltung ist eingegangen<br>- Sachbearbeiter hat den Link „Veranstaltung prüfen“ der entsprechenden Veranstaltung angeklickt |
| **Nachbedingungen:** | - Alle Informationen zu der Veranstaltung sind korrekt<br>- Sachbearbeiter hat den Button „Veranstaltung freigeben“ betätigt |
| **Essenzielle Schritte:** | 1. Sachbearbeiter wählt Veranstaltung aus Liste zu prüfender Veranstaltungen aus<br>2. Sachbearbeiter klickt Link „Veranstaltung prüfen“ an<br>3. Sachbearbeiter überprüft die Informationen zu der eingereichten Veranstaltung auf Richtigkeit<br>4. Sachbearbeiter bestätigt Richtigkeit durch Betätigung des Buttons „Veranstaltung freigeben“ |

Veranstaltung(en) suchen

| **Name:** | **Veranstaltung(en) suchen** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Es wird mit verschiedenen Kriterien nach Veranstaltungen gesucht und die Ergebnisse werden in einer Listen- und Mapansicht dargestellt. |
| **Auslöser:** | Aufruf des Links „Veranstaltung suchen“ |
| **Akteure:** | Benutzer, Veranstalter, Sachbearbeiter |
| **Eingehende Information:** | Suchkriterien |
| **Vorbedingungen:** | - Es sind Veranstaltungen vorhanden<br>- Beliebige Kriterien wurden ausgewählt<br>- Button „Suchen“ wurde betätigt |
| **Nachbedingungen:** | - Ergebnisse wurden in einer Liste und auf einer Karte ausgegeben |
| **Essenzielle Schritte:** | 1. Link „Veranstaltung suchen“ wird aufgerufen<br>2. Beliebige Suchkriterien werden ausgewählt<br>3. Datenbank wird anhand der eingegebenen Kriterien nach passenden Veranstaltungen durchsucht<br>4. Suchergebnisse werden in einer Liste und auf einer Karte ausgegeben |

Detailansicht aufrufen

| **Name:** | **Detailansicht aufrufen** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Die Detailansicht einer Veranstaltung wird angezeigt. |
| **Auslöser:** | Benutzer wählt aus der Liste/der Map eine Veranstaltung aus und klickt Link „Details anzeigen“ an |
| **Akteure:** | Benutzer, Veranstalter, Sachbearbeiter |
| **Eingehende Information:** | Ausgewählte Veranstaltung (ID) |
| **Vorbedingungen:** | - Veranstaltung wurde aus Liste der Suchergebnisse ausgewählt<br>- Link „Details anzeigen“ wurde angeklickt |

| **Name:** | **Detailansicht aufrufen** |
|---|---|
| Nachbedingungen: | - Detailinformationen zu der Veranstaltung wurden angezeigt |
| Essenzielle Schritte: | 1. Veranstaltung wird aus Liste der Suchergebnisse ausgewählt<br>2. Link „Details anzeigen“ wird angeklickt<br>3. Detailinformationen zu der Veranstaltung werden auf einer neuen Seite angezeigt |

Veranstaltungsstatistiken anzeigen

| **Name:** | **Veranstaltungsstatistiken anzeigen** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Statistische Daten (aktuell und historisch) zu den Veranstaltungen werden angezeigt. |
| **Auslöser:** | Aufruf des Links „Statistiken anzeigen“ |
| **Akteure:** | Sachbearbeiter |
| **Eingehende Information:** | |
| **Vorbedingungen:** | - Sachbearbeiter ist angemeldet<br>- Sachbearbeiter hat Link „Statistiken anzeigen“ angeklickt |
| **Nachbedingungen:** | - Filterbare Tabelle mit Statistiken zu allen Veranstaltungen wurde angezeigt |
| **Essenzielle Schritte:** | 1. Sachbearbeiter klickt Link „Statistiken anzeigen“ an<br>2. Aktuelle und historische Statistiken zu den Veranstaltungen werden angezeigt |

Route anzeigen

| **Name:** | **Route anzeigen** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Die Route zu der gewünschten Veranstaltung wird ermittelt und angezeigt. Zudem besteht die Möglichkeit, unterschiedliche Verkehrsmittel auszuwählen. |
| **Auslöser:** | Betätigung des Buttons „Route anzeigen“ |
| **Akteure:** | Benutzer, Veranstalter, Sachbearbeiter |
| **Eingehende Information:** | - Zielort (ausgewählte Veranstaltung)<br>- Standort des Nutzers oder eingegebene Adresse<br>- Verkehrsmittel |
| **Vorbedingungen:** | - Veranstaltung wurde aus Liste der Suchergebnisse ausgewählt<br>- Button „Route anzeigen“ wurde betätigt<br>- Standortzugriff ist erlaubt oder Adresse wurde eingegeben<br>- Verkehrsmittel wurde ausgewählt |
| **Nachbedingungen:** | - Eine oder mehrere mögliche Routen wurden angezeigt |
| **Essenzielle Schritte:** | 1. Veranstaltung wird aus Liste der Suchergebnisse ausgewählt<br>2. Button „Route anzeigen“ wird betätigt<br>3. Benutzer wird nach Standort gefragt (aktueller Standort oder Adresse)<br>4. Benutzer wird nach Verkehrsmittel gefragt<br>5. Es wird mindestens eine mögliche Route angezeigt |

Parkplätze zur Veranstaltung anzeigen

| **Name:** | **Parkplätze zur Veranstaltung anzeigen** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Die umliegenden Parkplätze zu der gewünschten Veranstaltung werden angezeigt. |
| **Auslöser:** | Betätigung des Buttons „Parkplätze anzeigen“ |
| **Akteure:** | Benutzer, Veranstalter, Sachbearbeiter |
| **Eingehende Information:** | Zielort (ausgewählte Veranstaltung) |

| **Name:** | **Parkplätze zur Veranstaltung anzeigen** |
|---|---|
| **Vorbedingungen:** | - Veranstaltung wurde aus Liste der Suchergebnisse ausgewählt<br>- Button „Parkplätze anzeigen“ wurde betätigt |
| **Nachbedingungen:** | - Umliegende Parkplätze wurden angezeigt |
| **Essenzielle Schritte:** | 1. Veranstaltung wird aus Liste der Suchergebnisse ausgewählt<br>2. Button „Parkplätze anzeigen“ wird betätigt<br>3. Benutzer wird nach gewünschtem Umkreis gefragt<br>4. Öffentliche Parkplätze im gewählten Umkreis werden angezeigt |

Tickets kaufen

| **Name:** | **Tickets kaufen** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Ein Benutzer kann Tickets kaufen, falls dies für eine Veranstaltung notwendig ist. |
| **Auslöser:** | Betätigung des Buttons „Tickets kaufen“ |
| **Akteure:** | Benutzer, Veranstalter, Sachbearbeiter |
| **Eingehende Information:** | Ausgewählte Veranstaltung |
| **Vorbedingungen:** | - Veranstaltung wurde aus Liste der Suchergebnisse ausgewählt<br>- Zum Besuch der Veranstaltung sind Tickets erforderlich<br>- Button „Tickets kaufen“ wurde betätigt |
| **Nachbedingungen:** | - Benutzer wurde auf Drittanbieterseite zum Kauf von Tickets umgeleitet |
| **Essenzielle Schritte:** | 1. Veranstaltung wird aus Liste der Suchergebnisse ausgewählt<br>2. Button „Tickets kaufen“ wird betätigt<br>3. Benutzer wird auf Drittanbieterseite zum Kauf von Tickets umgeleitet |

Newsletter abonnieren

| **Name:** | **Newsletter abonnieren** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Ein Benutzer abonniert den Newsletter, in dem er über aktuelle und zukünftige Veranstaltungen informiert wird. |
| **Auslöser:** | Benutzer gibt seine E-Mail Adresse ein und betätigt Button „Newsletter abonnieren“ |
| **Akteure:** | Benutzer, Veranstalter, Sachbearbeiter |
| **Eingehende Information:** | E-Mail Adresse |
| **Vorbedingungen:** | - Benutzer hat seine E-Mail Adresse in das dafür vorgesehene Feld eingegeben<br>- E-Mail Adresse ist noch nicht in der Datenbank hinterlegt<br>- Benutzer betätigt Button „Newsletter abonnieren“ |
| **Nachbedingungen:** | - E-Mail Adresse wurde in Datenbank aufgenommen<br>- Benutzer hat Mail mit Bestätigungslink erhalten<br>- Benutzer hat Newsletter-Anmeldung bestätigt |
| **Essenzielle Schritte:** | 1. Benutzer gibt seine E-Mail Adresse in das dafür vorgesehene Feld ein<br>2. Benutzer betätigt Button „Newsletter abonnieren“<br>3. Mail mit Bestätigungslink wird an Benutzer versendet<br>4. Benutzer bestätigt Newsletter-Anmeldung über den Link in der Mail |

Newsletter generieren

| **Name:** | **Newsletter generieren** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Aus aktuellen und zukünftigen Veranstaltungen können einige ausgewählt und daraus ein Newsletter generiert werden. |
| **Auslöser:** | Aufruf des Links „Newsletter erstellen“ |
| **Akteure:** | Sachbearbeiter |
| **Eingehende Information:** | Ausgewählte Veranstaltung(en) |
| **Vorbedingungen:** | - Mindestens eine Veranstaltung wurde ausgewählt<br>- Button „Newsletter generieren“ wurde betätigt |

| **Name:** | **Newsletter generieren** |
|---|---|
| **Nachbedingungen:** | - Inhalt für Newsletter (Bilder und Infos zu gewählten Veranstaltungen) wurde generiert |
| **Essenzielle Schritte:** | 1. Sachbearbeiter klickt Link „Newsletter erstellen“ an<br>2. Sachbearbeiter wählt mindestens eine Veranstaltung aus, die in dem Newsletter aufgeführt werden soll<br>3. Newsletter-Inhalt wird aus Bildern und Infos zu Veranstaltung(en) generiert<br>4. Sachbearbeiter passt Inhalt bei Bedarf an |

Als User registrieren

| **Name:** | **User-Registrierung** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Ein Benutzer erstellt sich ein Profil für die Webseite. |
| **Auslöser:** | Aufruf des Links „Registrieren“ |
| **Akteure:** | Benutzer, Veranstalter |
| **Eingehende Information:** | - E-Mail Adresse<br>- Passwort |
| **Vorbedingungen:** | - Benutzer hat Link „Registrieren“ angeklickt<br>- Benutzer ist noch nicht registriert |
| **Nachbedingungen:** | - Alle Pflichtfelder des Formulars wurden ausgefüllt<br>- Benutzer hat den Button „bestätigen“ betätigt<br>- Benutzer erhält Mail mit Bestätigungslink<br>- Benutzer bestätigt Registrierung |
| **Essenzielle Schritte:** | 1. Benutzer klickt Link „Registrieren“ an<br>2. Benutzer gibt seine E-Mail Adresse und gewünschtes Passwort ein<br>3. Mail mit Bestätigungslink wird an Benutzer versendet<br>4. Benutzer bestätigt Registrierung über den Link in der Mail |

Als Veranstalter verifizieren

| **Name:** | **Als Veranstalter verifizieren** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Ein Benutzer bestätigt die Echtheit seines Profils durch einen Verifizierungsprozess. |
| **Auslöser:** | Aufruf des Links „Profil verifizieren“ |
| **Akteure:** | Veranstalter, Sachbearbeiter |
| **Eingehende Information:** | - Benutzerdaten<br>- weiterführende Veranstalterdaten |
| **Vorbedingungen:** | - Benutzer ist noch nicht verifiziert<br>- Profil wurde vollständig ausgefüllt<br>- Benutzer hat Button „Verifizierung beantragen“ betätigt |
| **Nachbedingungen:** | - Antrag auf Verifizierung ist bei der Stadt eingegangen<br>- Sachbearbeiter hat Antrag geprüft<br>- Sachbearbeiter hat mit Veranstalter Kontakt aufgenommen (z.B. per Videochat)<br>- Sachbearbeiter hat Echtheit des Profils bestätigt (Ausweis-Prüfung) |
| **Essenzielle Schritte:** | 1. Benutzer klickt Link „Profil verifizieren“ an<br>2. Antrag zur Verifizierung geht bei der Stadt ein<br>3. Sachbearbeiter prüft Antrag<br>4. Sachbearbeiter nimmt Kontakt zu (potentiellem) Veranstalter auf (z.B. per Videochat)<br>5. Sachbearbeiter prüft Echtheit des Profils, indem der die Daten mit dem Personalausweis abgleicht<br>6. Sachbearbeiter bestätigt Echtheit des Profils |

Veranstaltungsvorschläge anzeigen

| **Name:** | **Veranstaltungsvorschläge anzeigen** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Dem Benutzer werden Veranstaltungen angezeigt, die ihn interessieren könnten (auf Basis bereits besuchter Veranstaltungen und Favoriten). |
| **Auslöser:** | Aufruf der Veranstaltungs-Seite |
| **Akteure:** | Benutzer, Veranstalter, Sachbearbeiter |
| **Eingehende Information:** | - In der Vergangenheit besuchte Veranstaltungen<br>- Favoriten |
| **Vorbedingungen:** | - Benutzer ist angemeldet<br>- Benutzer hat bereits Veranstaltungen besucht oder Favoriten festgelegt |

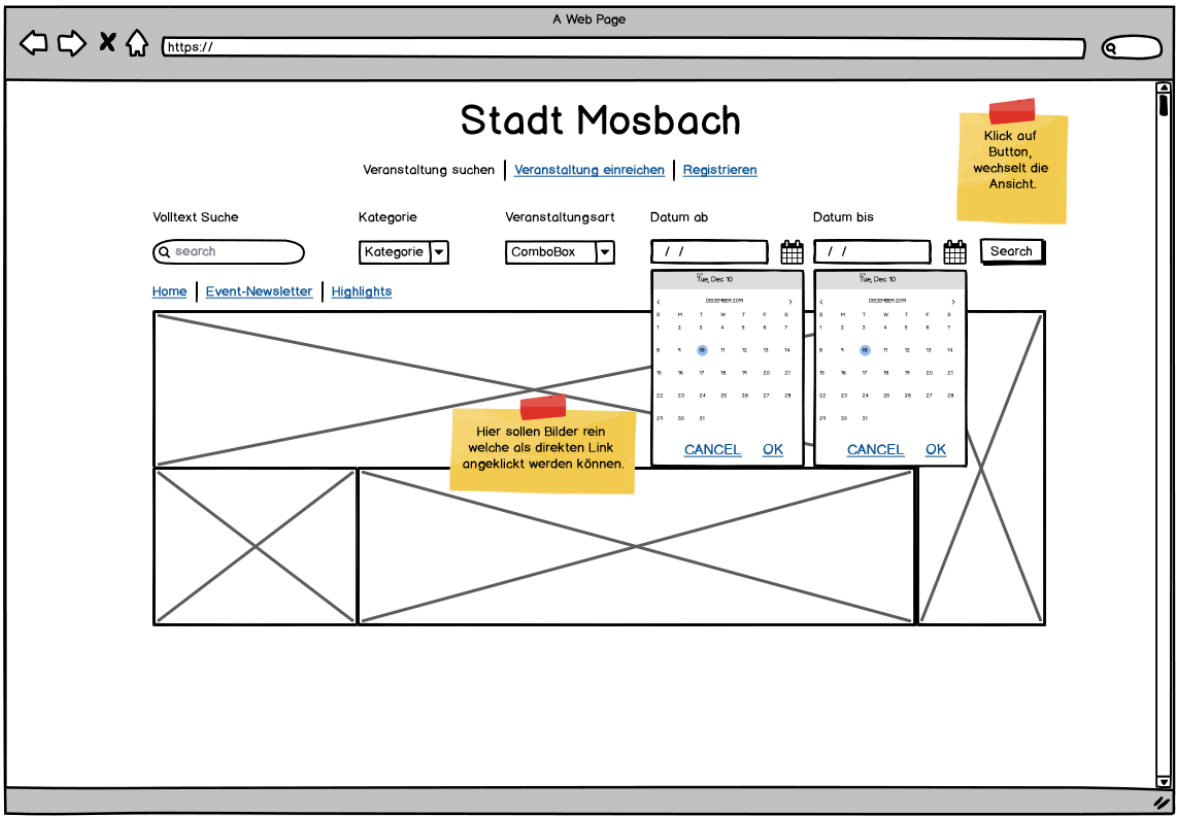
| **Name:** | **Veranstaltungsvorschläge anzeigen** |
|---|---|
| **Nachbedingungen:** | - Ähnliche Veranstaltungen wie bereits besuchte oder Veranstaltungen aus Favoriten-Kategorien wurden angezeigt |
| **Essenzielle Schritte:** | 1. Benutzer ruft Seite „Veranstaltungen“ auf<br>2. In einem Bereich „Vorschläge“ werden Veranstaltungsvorschläge angezeigt |

Favoriten festlegen

| **Name:** | **Favoriten festlegen** |
|---|---|
| **Art:** | Anwendungsfall |
| **Kurzbeschreibung:** | Ein Benutzer kann eine oder mehrere Kategorien und/oder Veranstaltungen als Favoriten markieren. |
| **Auslöser:** | Betätigung des Buttons „zu Favoriten hinzufügen“ |
| **Akteure:** | Benutzer, Veranstalter, Sachbearbeiter |
| **Eingehende Information:** | Ausgewählte Veranstaltung oder Kategorie |
| **Vorbedingungen:** | - Benutzer ist angemeldet<br>- Benutzer hat eine oder mehrere Veranstaltung(en)/Kategorie(n) ausgewählt<br>- Benutzer hat Button „zu Favoriten hinzufügen“ betätigt |
| **Nachbedingungen:** | - Veranstaltung(en)/Kategorie(n) wurden zu Favoriten hinzugefügt |
| **Essenzielle Schritte:** | 1. Benutzer wählt bei der Suche nach Veranstaltungen eine oder mehrere Veranstaltung(en)/Kategorie(n) aus<br>2. Benutzer betätigt Button „zu Favoriten hinzufügen“<br>3. Ausgewählte Veranstaltung(en)/Kategorie(n) werden zu Favoriten des Benutzers hinzugefügt |

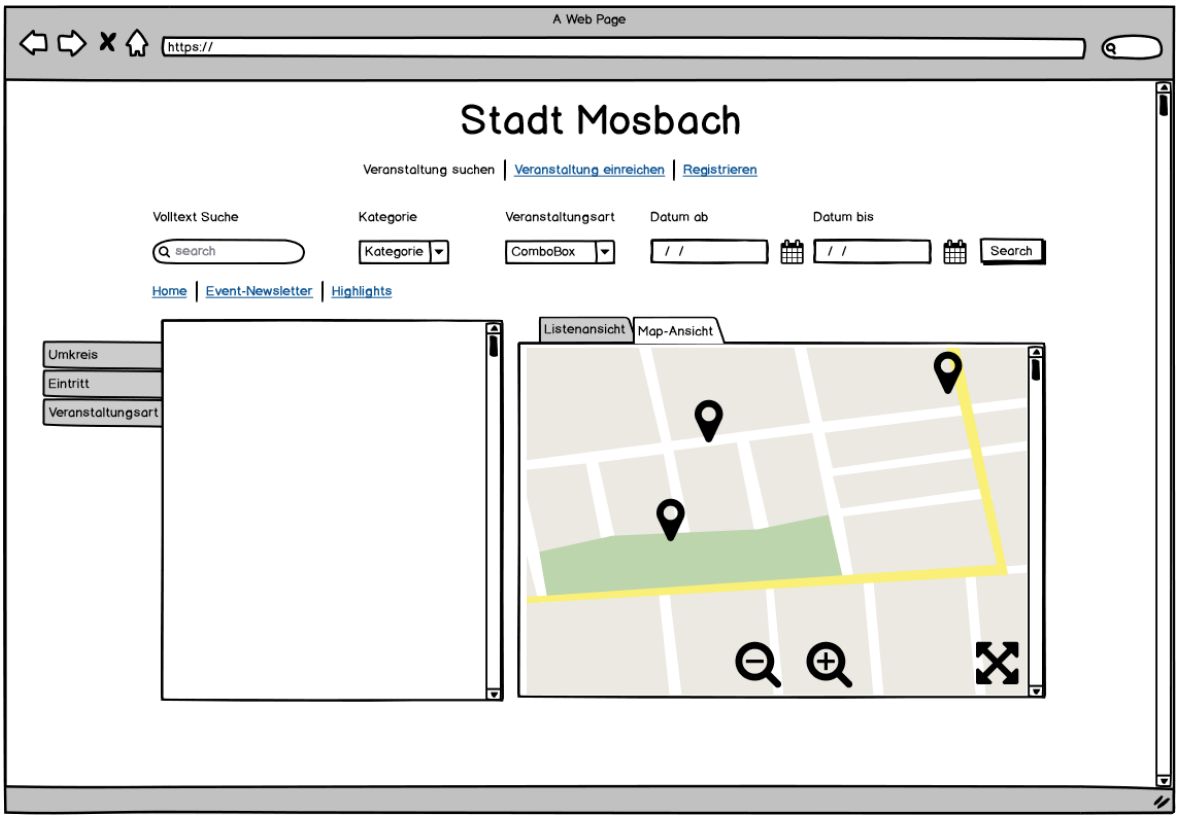
## Oberflächendesign

In den folgenden Abbildungen sind die Mockups des Oberflächendesigns dargestellt.

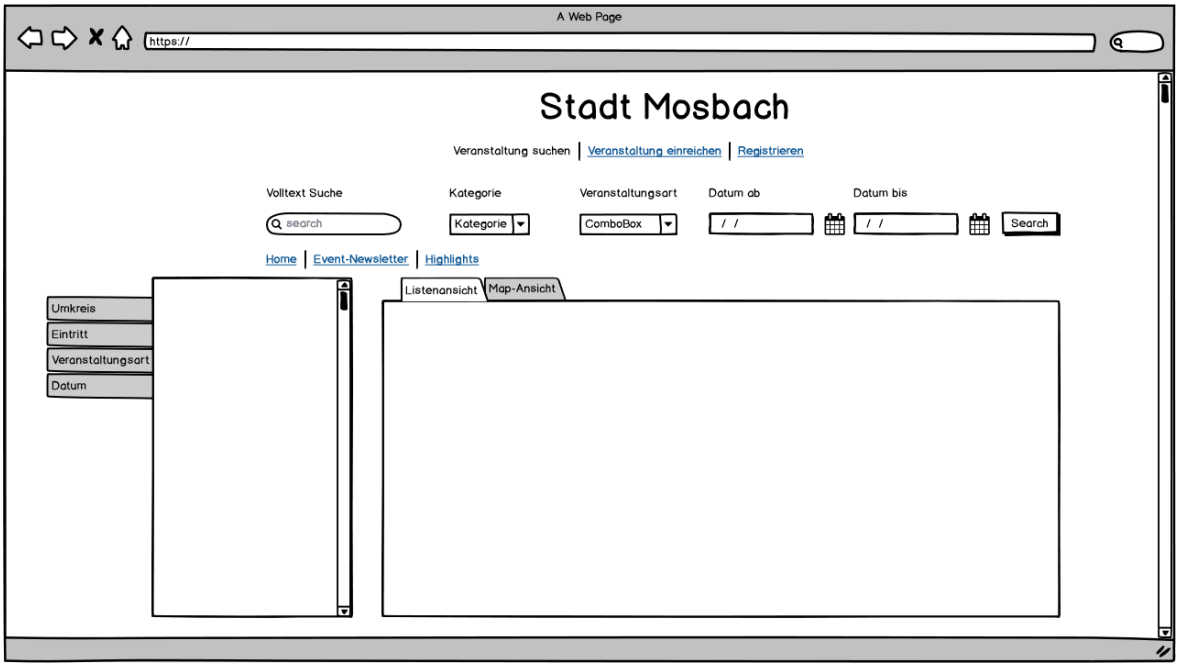



Anhang-Abbildung 1 Startseite

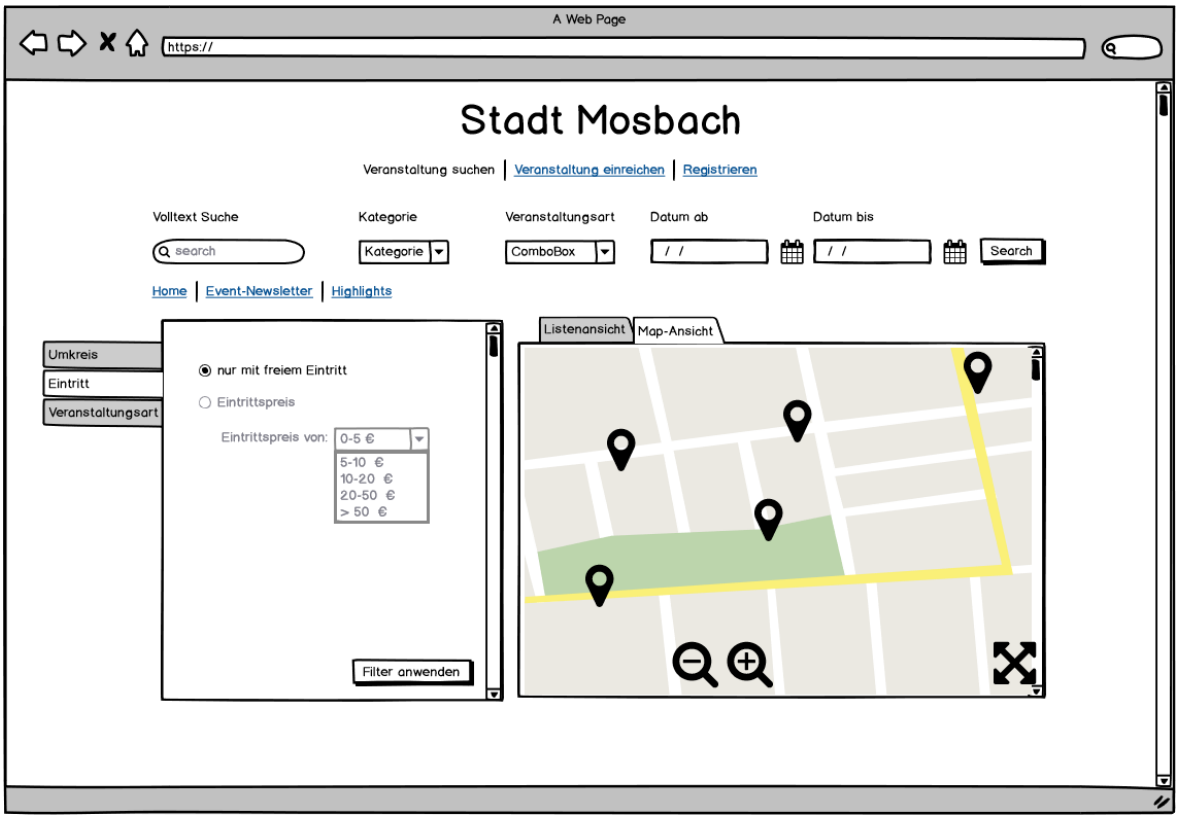
Nachdem auf der Startseite nach Veranstaltungen gesucht wurde, werden diese auf einer Karte oder in einer Liste angezeigt. Dort können diese Veranstaltungen durch Filter weiter eingegrenzt werden. Außerdem ist es möglich, Veranstaltungen zu abonnieren und diese in der Profilansicht einzusehen. Als Veranstalter lassen sich die selbst angelegten Veranstaltungen im Profil einsehen.



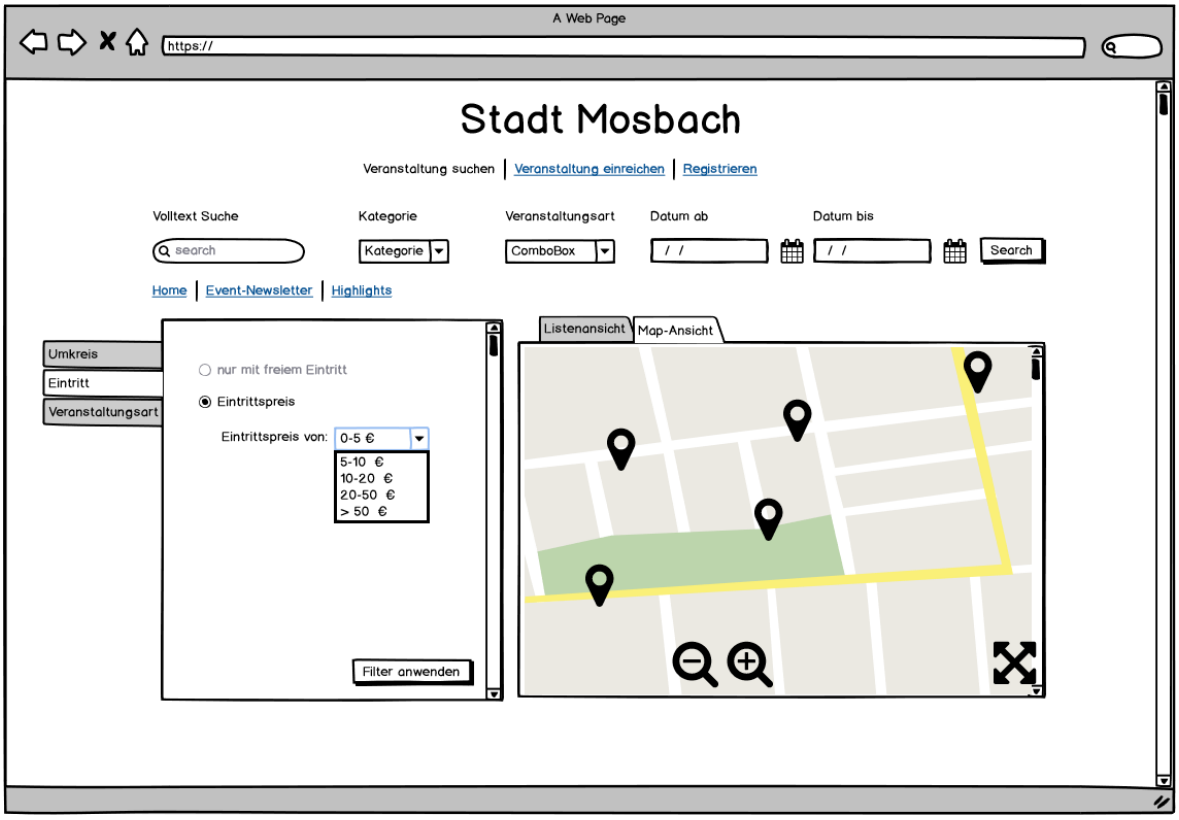

Anhang-Abbildung 2 Kartenansicht



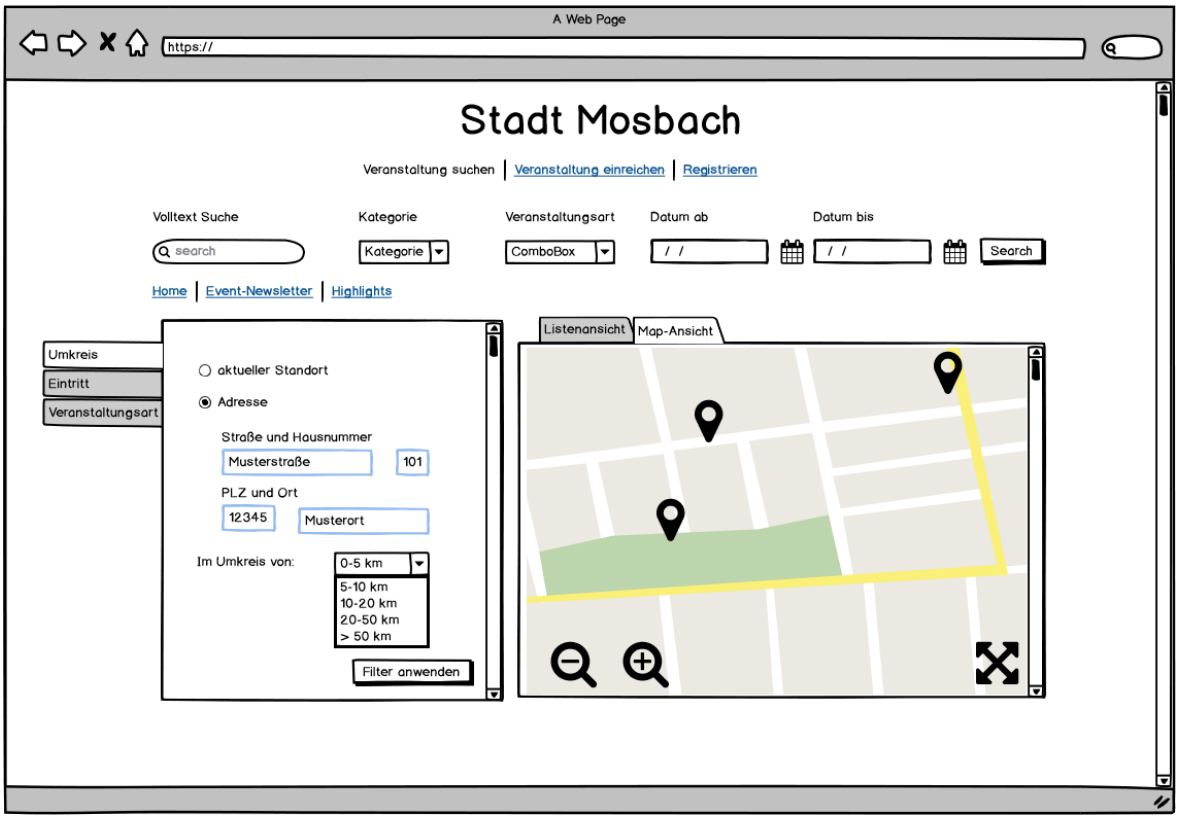

Anhang-Abbildung 3 Listenansicht



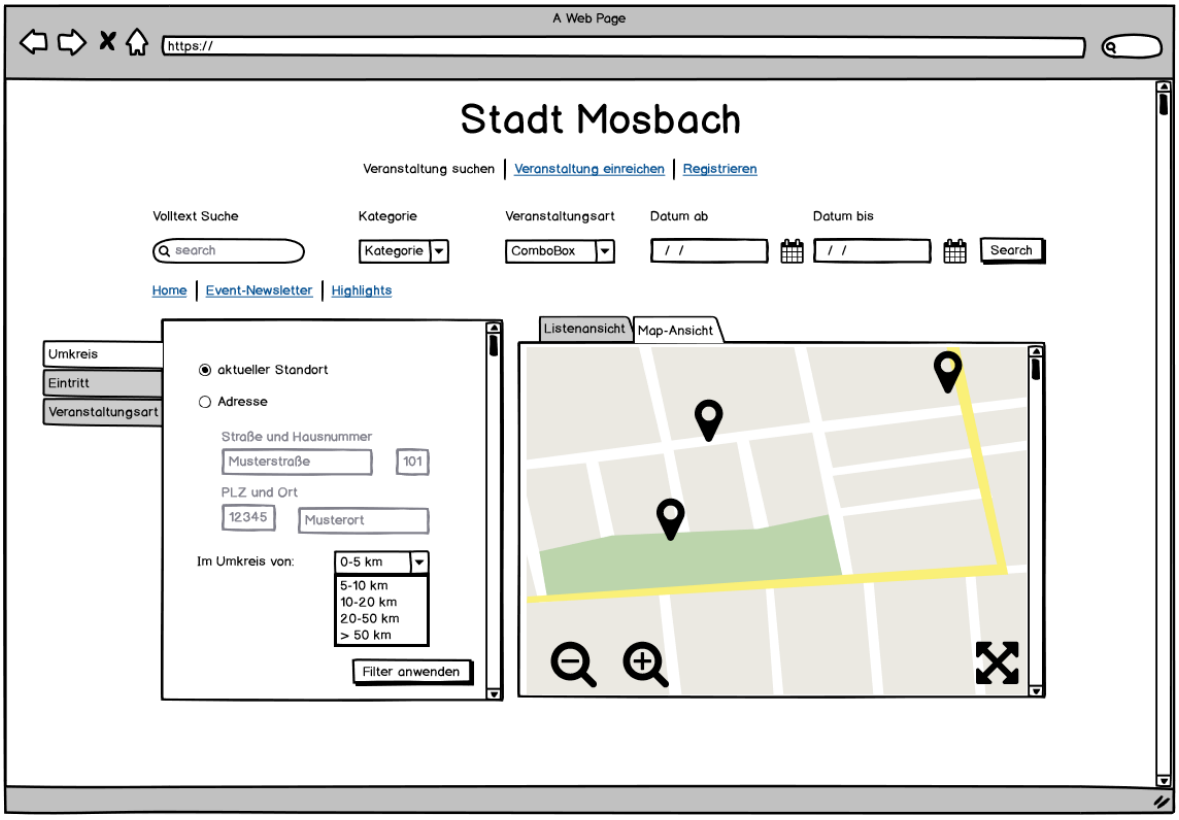

Anhang-Abbildung 4 Eintritt Filter freier Eintritt




Anhang-Abbildung 5 Eintritt Filter Eintrittspreis

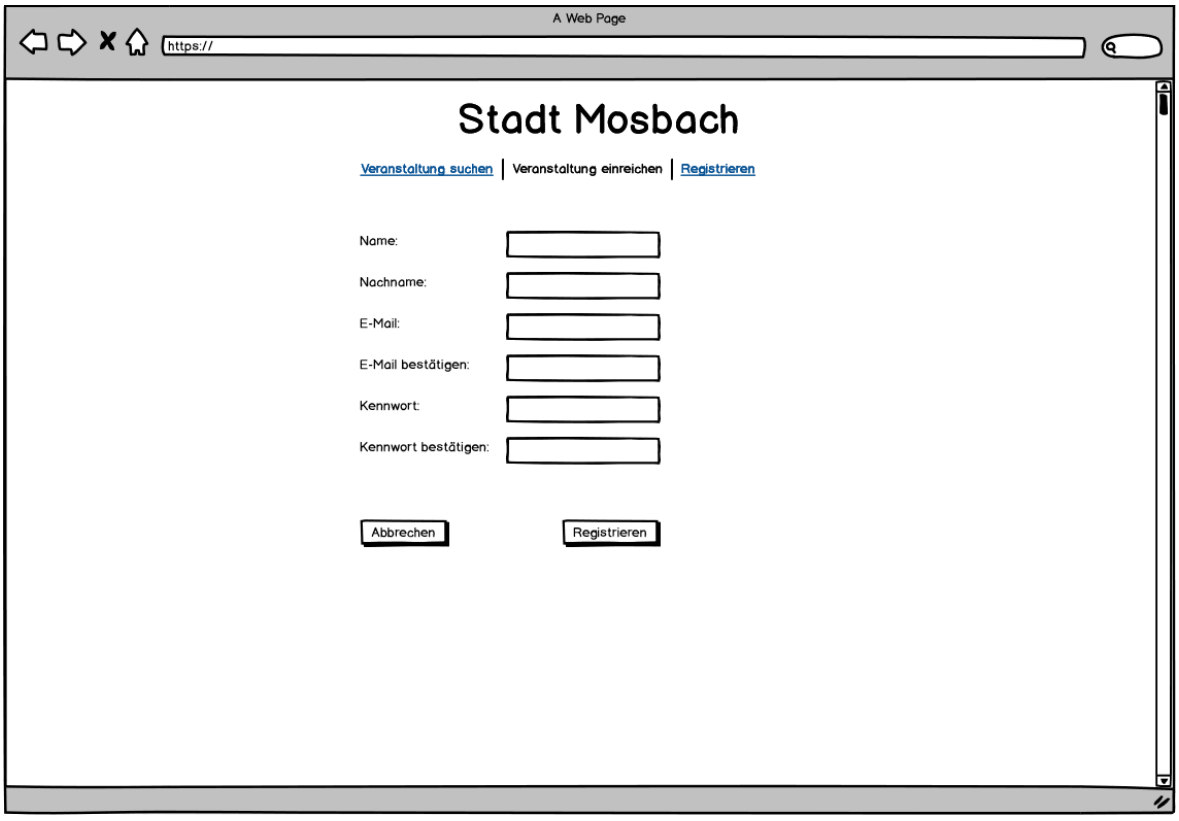



Anhang-Abbildung 6 Umkreis Filter Adresse

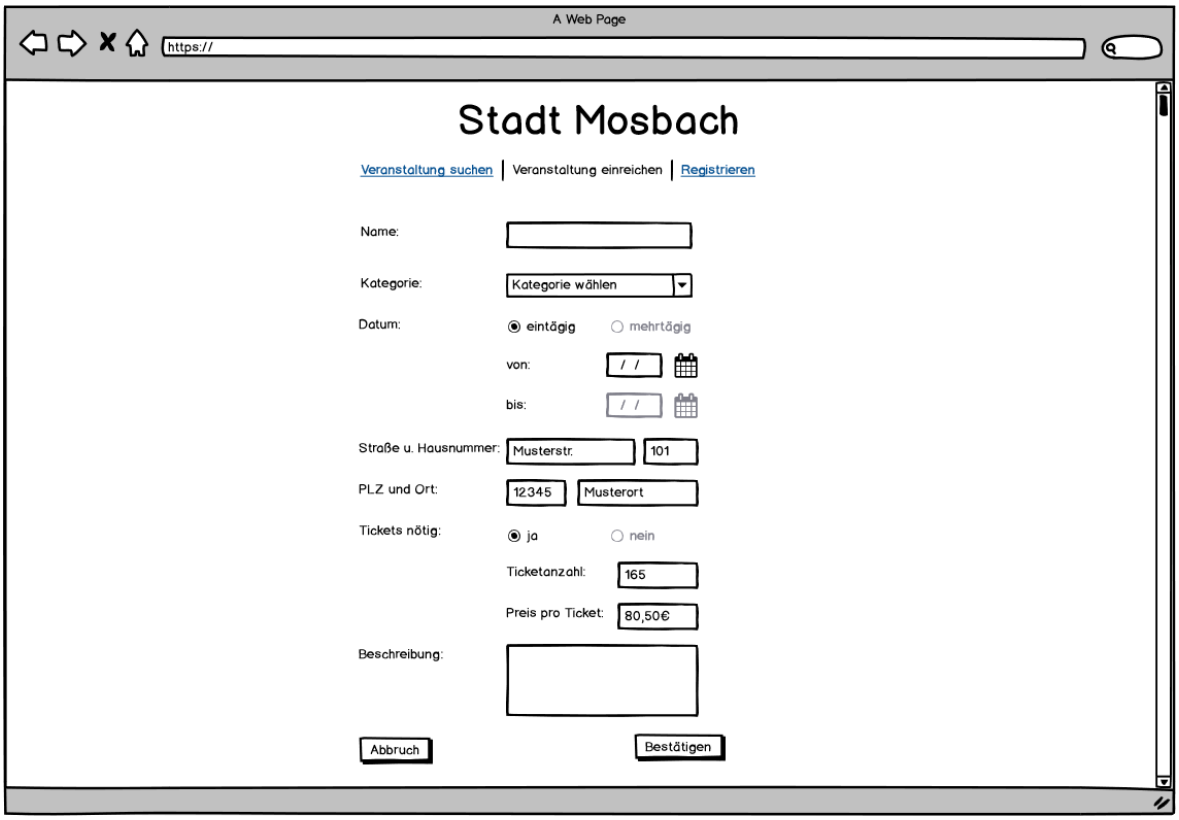



Anhang-Abbildung 7 Umkreis Filter Aktueller Standort

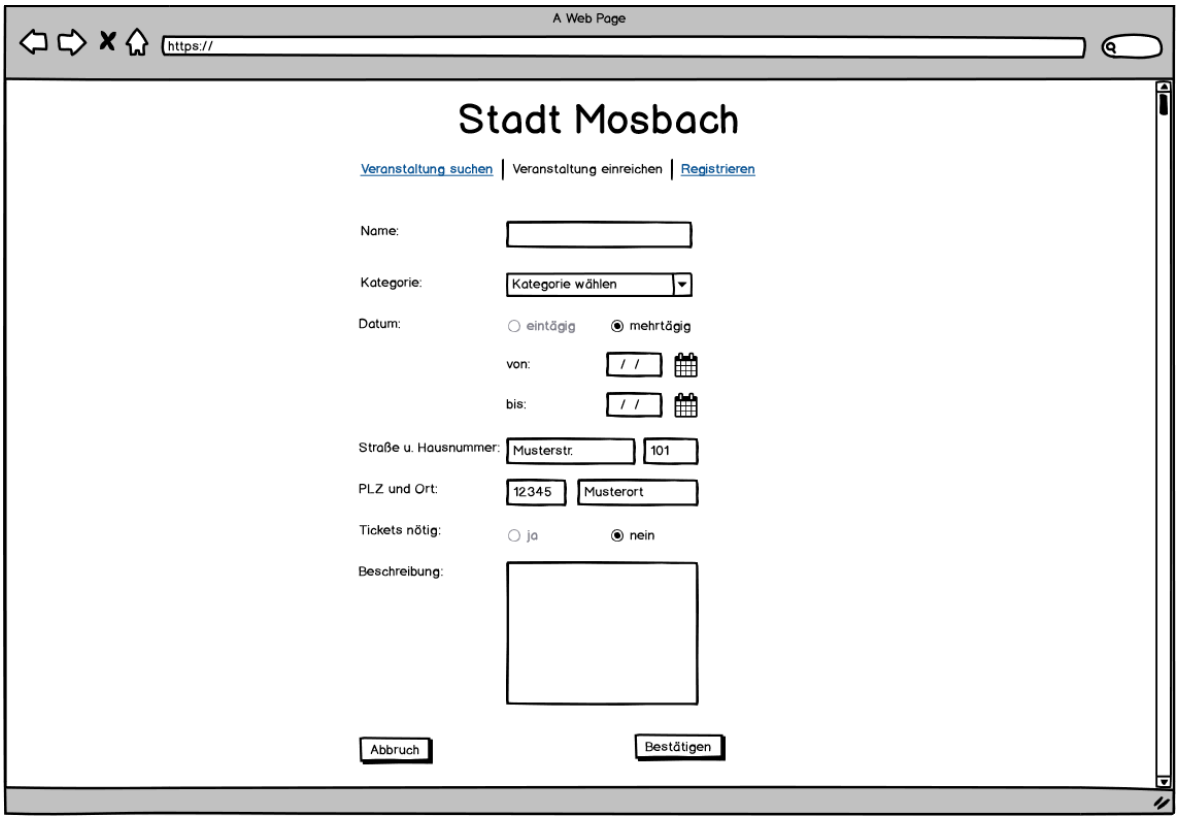

Anhang-Abbildung 8 Veranstaltungsart Filter



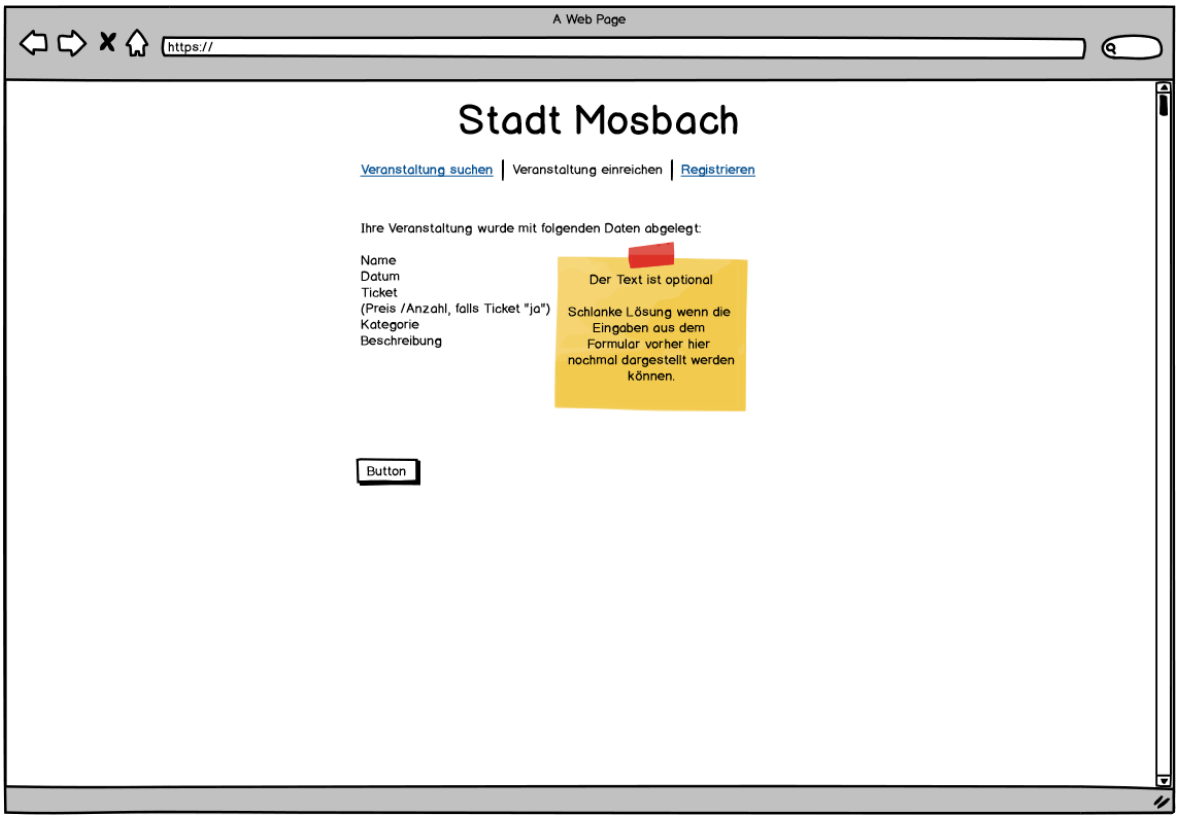

Anhang-Abbildung 9 Registrierung



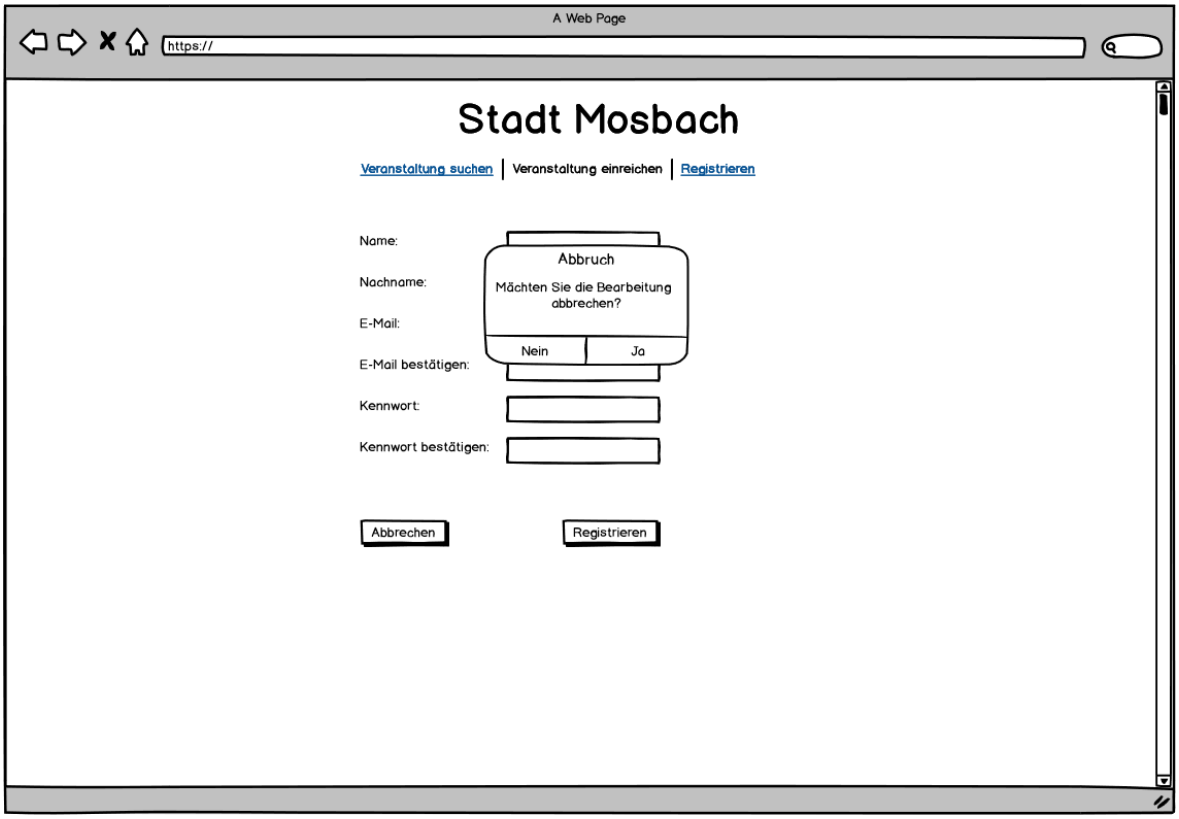

Anhang-Abbildung 10 Veranstaltung anlegen eintägig



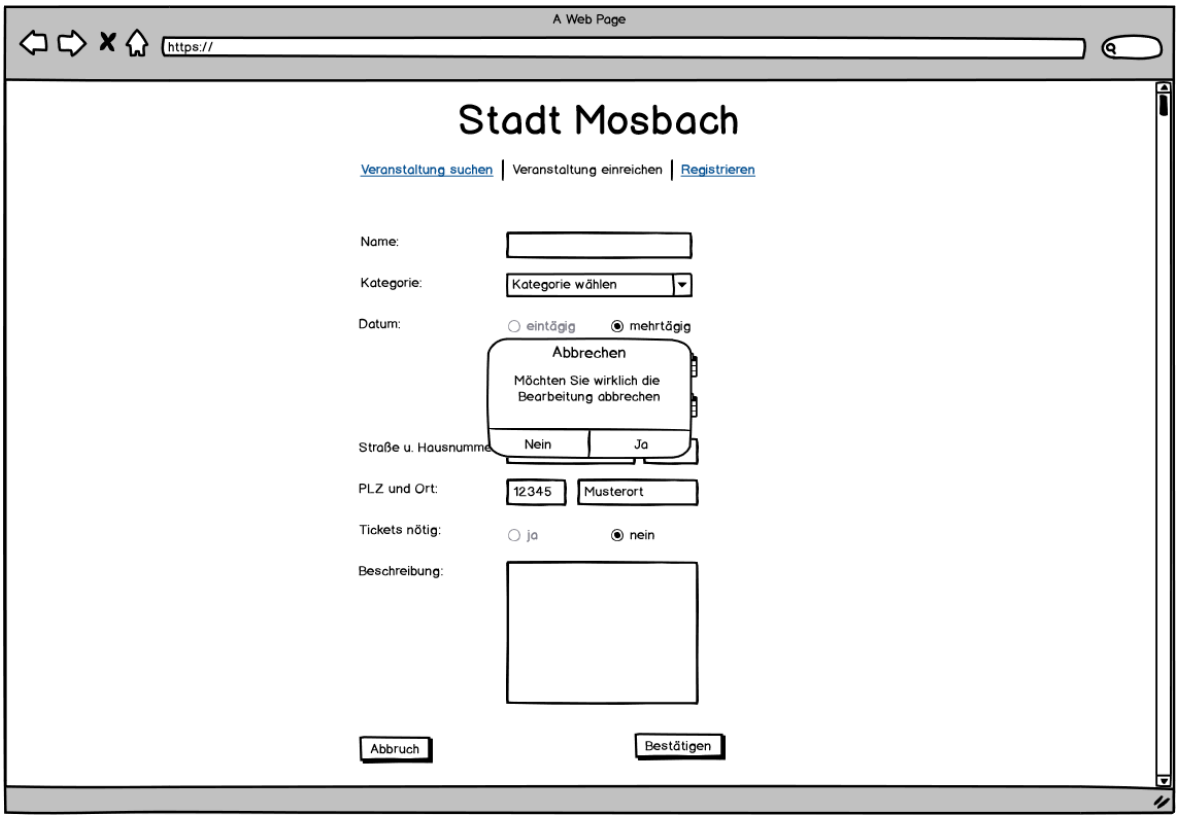

Anhang-Abbildung 11 Veranstaltung anlegen mehrtägig



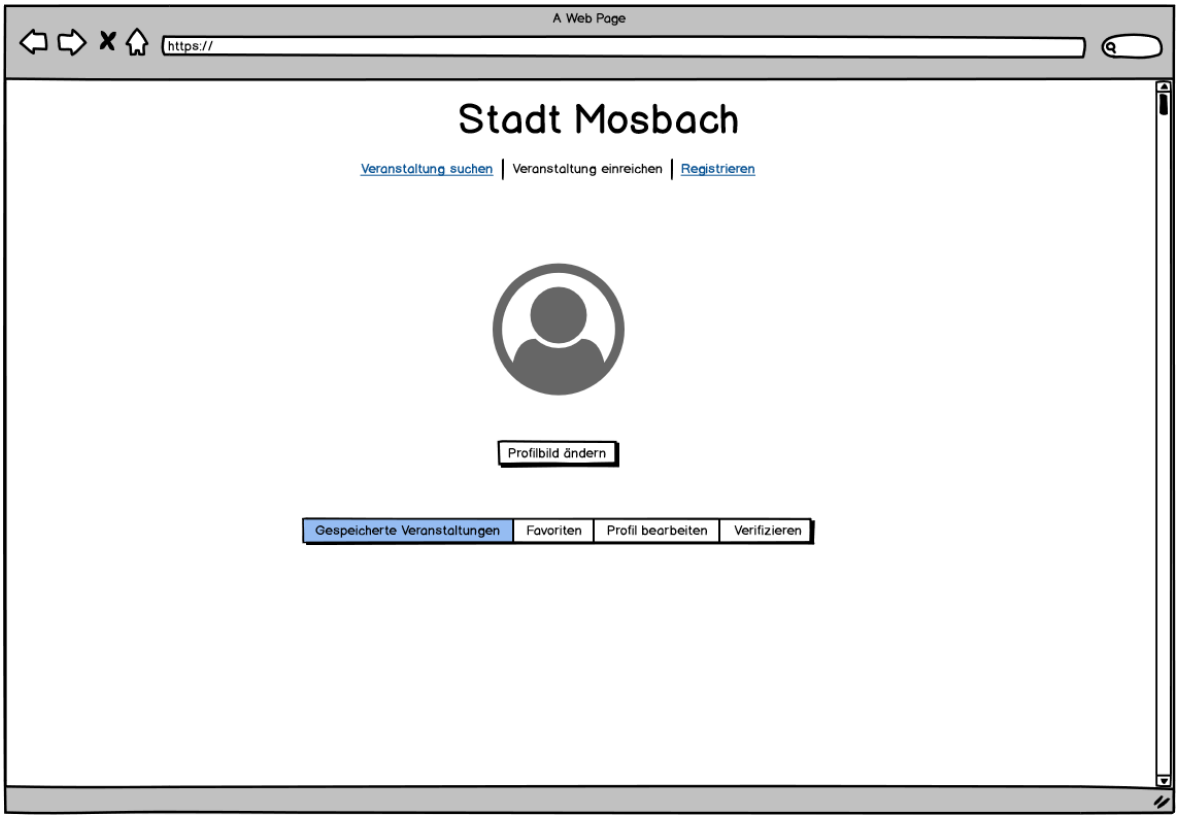

Anhang-Abbildung 12 Bestätigung des Anlegens einer Veranstaltung



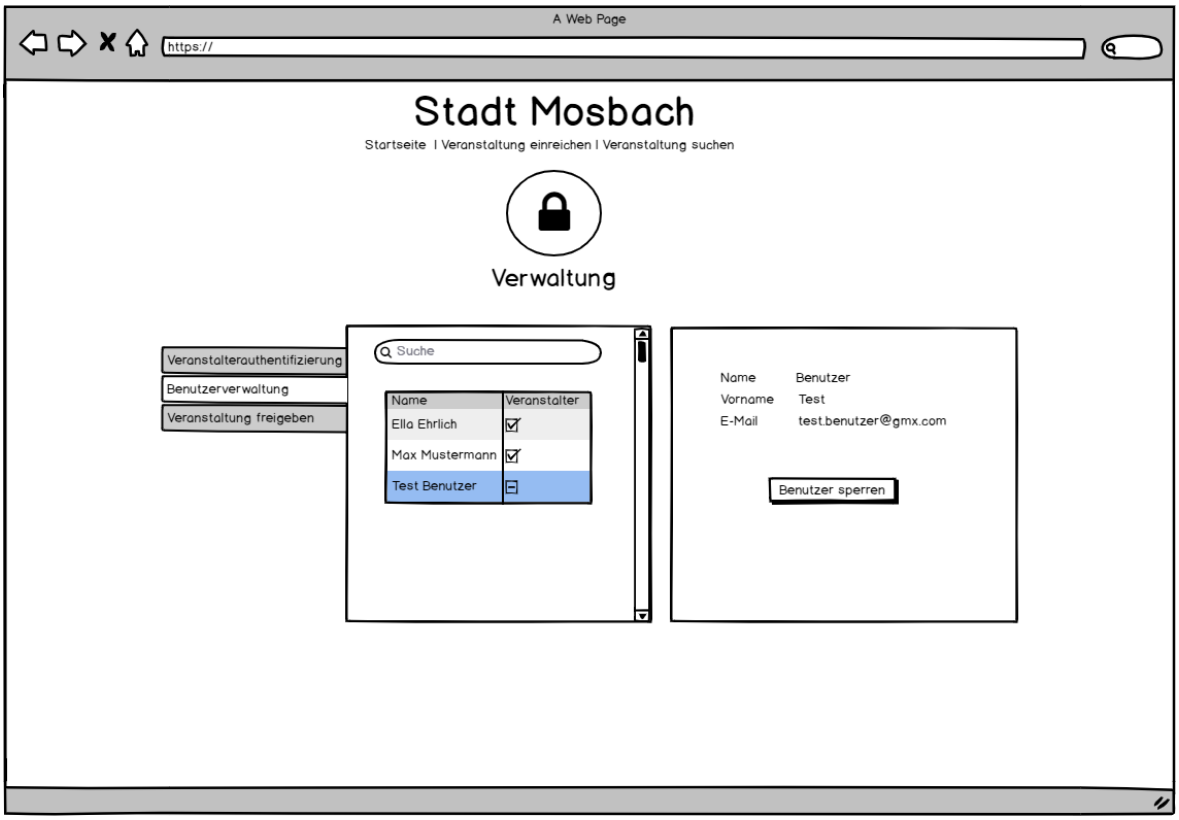

Anhang-Abbildung 13 Abbruch einer Registrierung



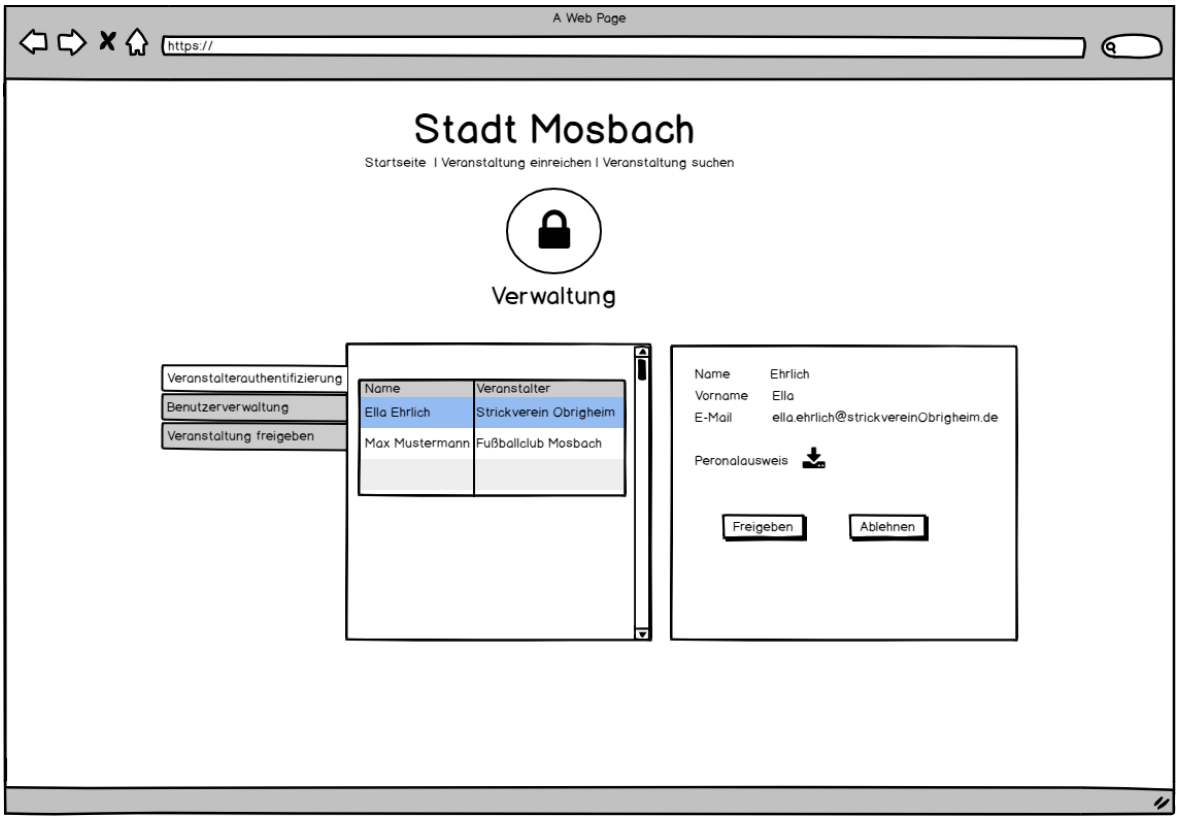

Anhang-Abbildung 14 Abbrechen des Anlegens einer Veranstaltung



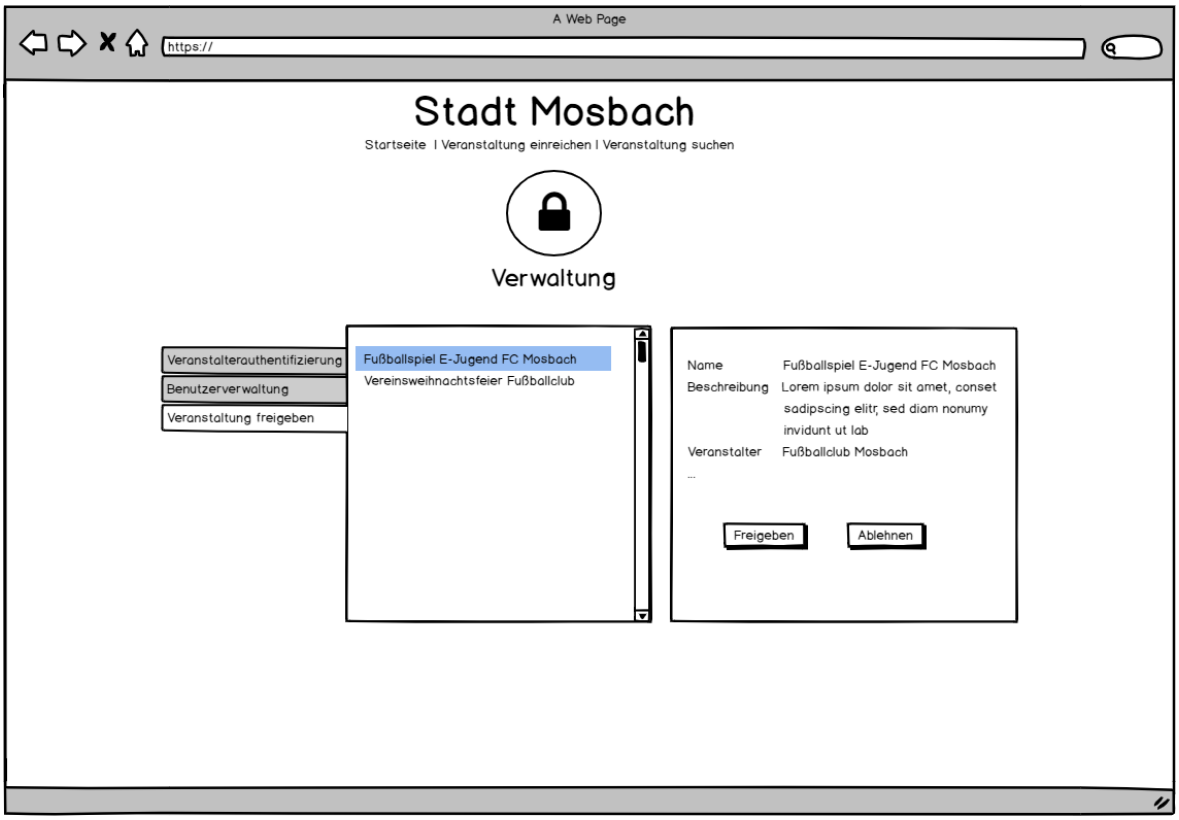

Anhang-Abbildung 15 Profilansicht nach Registrierung



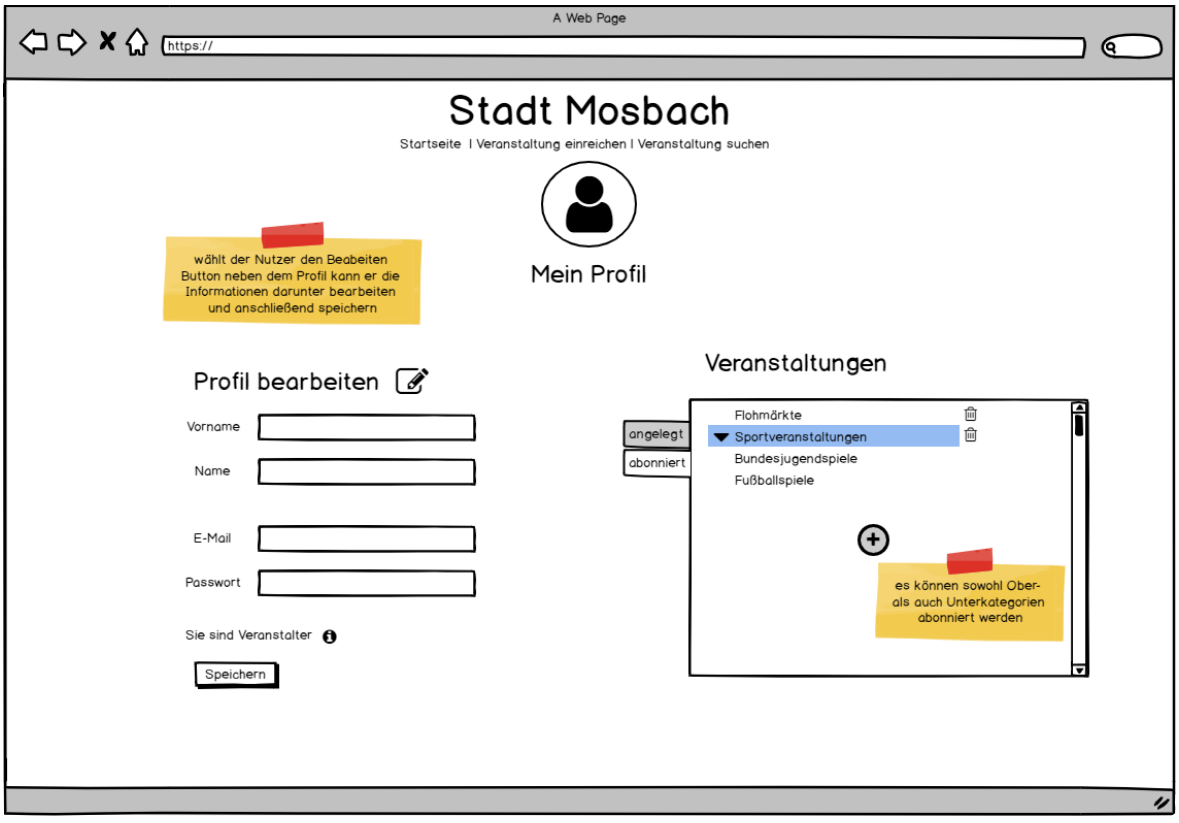

Anhang-Abbildung 16 Sachbearbeiter Oberfläche Benutzerverwaltung



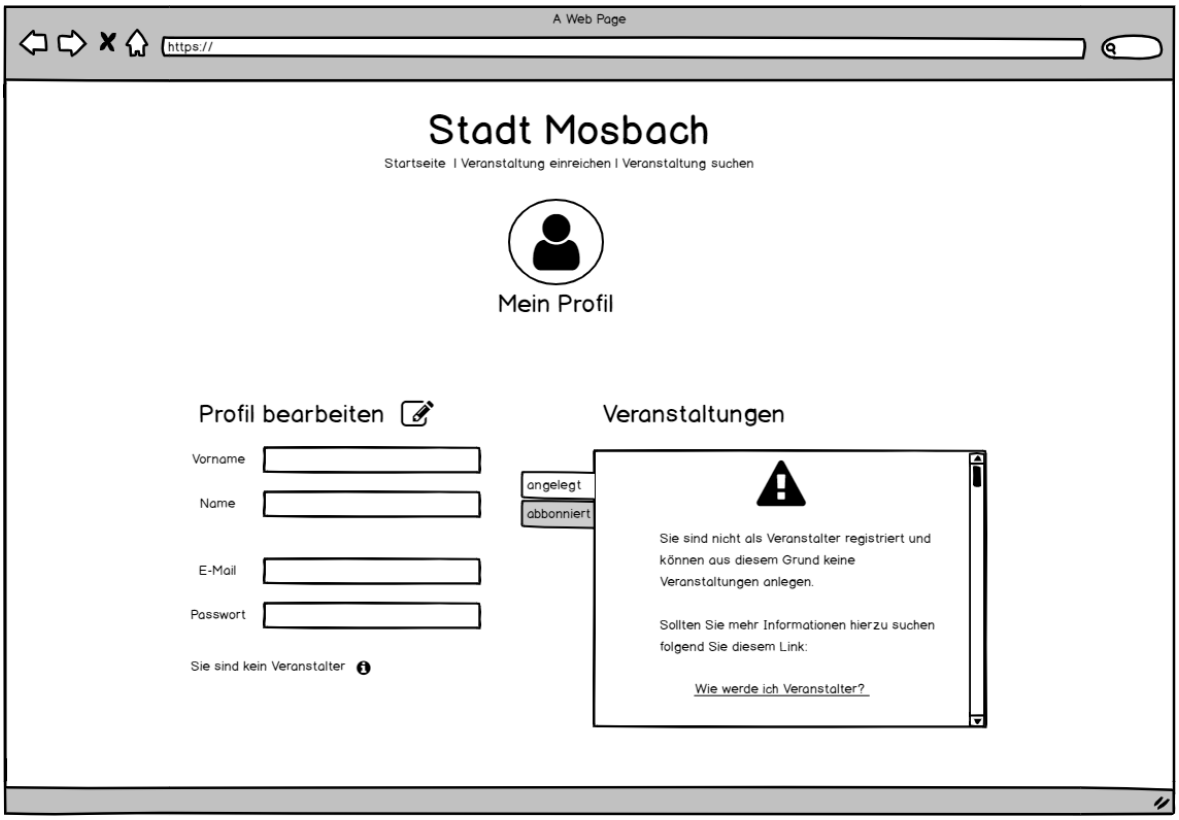

Anhang-Abbildung 17 Sachbearbeiter Oberfläche Veranstalterauthentifizierung



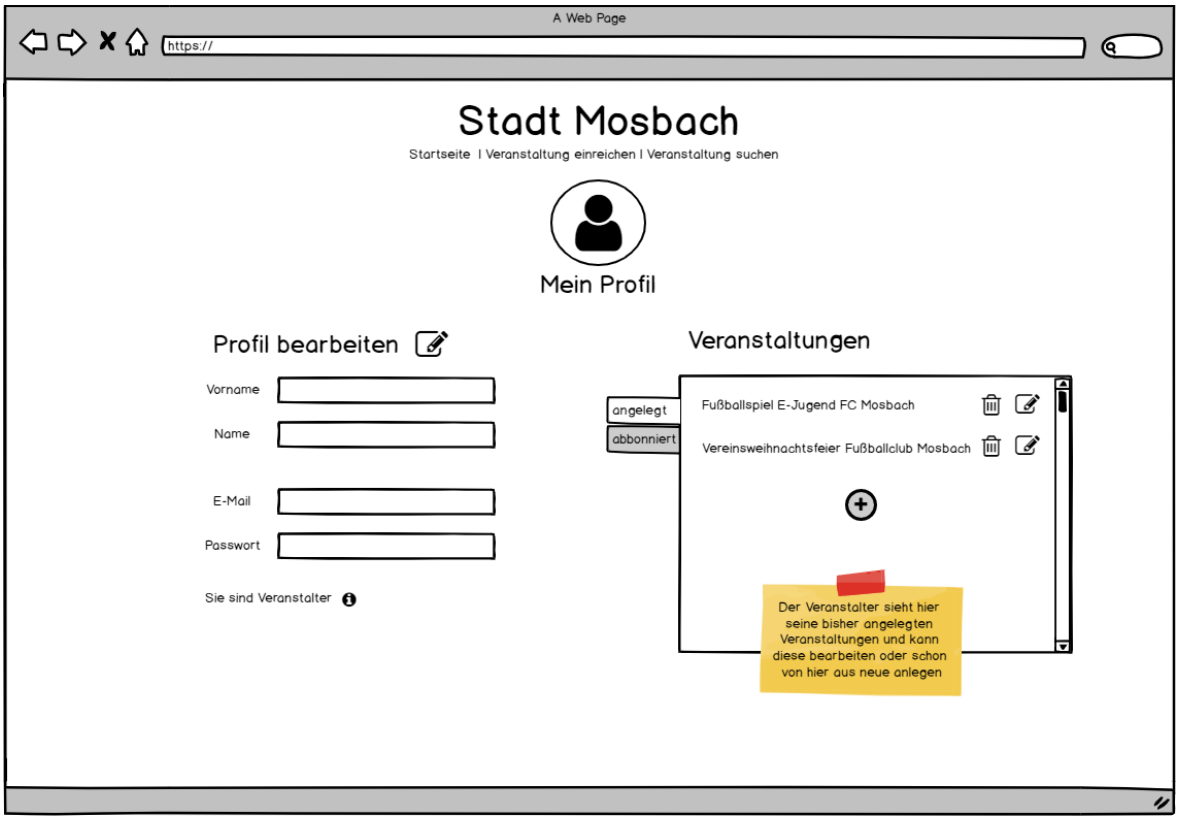

Anhang-Abbildung 18 Sachbearbeiter Oberfläche Veranstaltung freigeben




Anhang-Abbildung 19 Benutzerprofil abonnierte Veranstaltungen




Anhang-Abbildung 20 Benutzerprofil kein Veranstalter




Anhang-Abbildung 21 Benutzerprofil Veranstalter angelegte Veranstaltungen

# Berechtigungskonzept

In den folgenden Abbildungen ist das vollständige Berechtigungskonzept abgebildet. In Anhang-Abbildung 22 sind deshalb die Akteure im Hinblick auf das Berechtigungskonzept nochmal näher beschrieben.

| Benutzer | Beschreibung |
|---|---|
| Administrator | Host des Systems/Webseite |
| Sachbearbeiterin | Verantwortlich für die Pflege |
| Veranstalter | Benutzer, der Veranstaltung erstellen darf |
| Gast | Ruft die Seite der Stadt Mosbach auf |
| Reg. Benutzer | Hat sich schon registriert um womöglich Ticktes zu bestellen oder Newsletter zu erhalten |

Anhang-Abbildung 22 Benutzerbeschreibung

In Anhang-Abbildung 23 sind die Berechtigungen anhand der Anwendungsfälle angegeben. Diese sind links aufgelistet. Mit einem x markiert wird der Anwendungsfall, zugehörig zu einem bestimmten Benutzer, wenn der Benutzer diesen Use Case ausführen darf.

| Berechtigungskonzept | | | | | | |
|---|---|---|---|---|---|---|
| Berechtigungen | Nutzer | | | | | |
| | Administrator | Sachbearbeiterin der Stadt | Veranstalter* | Gast** | Registrierter Benutzer | |
| Veranstaltung suchen | x | x | x | x | x | |
| Veranstaltung anzeigen | x | x | x | x | x | |
| Veranstaltung bearbeiten | | x | x | | | |
| Veranstaltung entfernen | | x | x | | | |
| Veranstaltung einreichen | | x | x | | | |
| Veranstaltung freigeben | | x | | | | |
| Filter benutzen | x | x | x | x | x | |
| Newsletter abonnieren | x | x | x | x | x | |
| Newsletter schreiben | | x | | | | |
| Infoblatt aufrufen | x | x | x | x | x | |
| Infoblatt ausdrucken | x | x | x | x | x | |
| Registrieren | | | | x | | |
| Benutzer löschen (für seinen Benutzer) | | | x | | x | |
| Benutzer anlegen (alle) | x | | | | | |
| Benutzer ändern (für seinen Benutzer) | | | x | | x | |
| Benutzer ändern (alle) | x | | | | | |
| Benutzer sperren/freigeben | x | x | | | | |
| Passwort ändern (für seinen Benutzer) | x | x | x | | x | |
| Anmelden/Abmelden | x | x | x | | x | |
| Benutzer als Veranstalter anfordern | | | | | x | |
| Benutzer als Veranstalter verifizieren | | x | | | | |
| Favoriten verwalten | | x | x | | x | |
| Highlights festlegen | | x | | | | |

Anhang-Abbildung 23 Berechtigungskonzept

## API Liste

AdresseRestController

| | |
|---|---|
| @GetMapping("/Adresse") | public List<Adresse> findAll() |
| @GetMapping("/Adresse/{AdresseId}") | public Adresse getAdresse(@PathVariable int AdresseId) |
| @PostMapping("/Adresse") | public Adresse addAdresse(@RequestBody Adresse theAdresse) |
| @PutMapping("/Adresse") | public Adresse updateAdresse(@RequestBody Adresse theAdresse) |
| @DeleteMapping("/Adresse/{AdresseId}") | public String deleteAdresse(@PathVariable int AdresseId) |

OberkategorieRestController

| | |
|---|---|
| @GetMapping("/Oberkategorie") | public List<Oberkategorie> findAll() |
| @GetMapping("/Oberkategorie/{OberkategorieId}") | public Oberkategorie getOberkategorie(@PathVariable int OberkategorieId) |
| @PostMapping("/Oberkategorie") | public Oberkategorie addOberkategorie(@RequestBody Oberkategorie theOberkategorie) |
| @PutMapping("/Oberkategorie") | public Oberkategorie updateOberkategorie(@RequestBody Oberkategorie theOberkategorie) |
| @DeleteMapping("/Oberkategorie/{OberkategorieId}") | public String deleteOberkategorie(@PathVariable int OberkategorieId) |
| @GetMapping("/Oberkategorie/User/{OberkategorieId}") | public List<User> getOberkategorieUsers(@PathVariable int OberkategorieId) |
| @GetMapping("/Oberkategorie/Unterkategorie/{OberkategorieId}") | public List<Unterkategorie> getOberkategorieUnterkategorien(@PathVariable int OberkategorieId) |
| @GetMapping("/Oberkategorie/Veranstaltung/{OberkategorieId}") | public List<Veranstaltung> getOberkategorieVeranstaltungen(@PathVariable int OberkategorieId) |
| @PostMapping("/Oberkategorie/User/{OberkategorieId}") | public List<User> addOberkategorieUser(@PathVariable int OberkategorieId, @RequestBody User theUser) |
| @PostMapping("/Oberkategorie/Unterkategorie/{OberkategorieId}") | public List<Unterkategorie> addOberkategorieUnterkategorie(@PathVariable int OberkategorieId, @RequestBody Unterkategorie theUnterkategorie) |
| @PostMapping("/Oberkategorie/Veranstaltung/{OberkategorieId}") | public List<Veranstaltung> addOberkategorieVeranstaltung(@PathVariable int OberkategorieId, @RequestBody Veranstaltung theVeranstaltung) |

| | |
|---|---|
| @PostMapping("/Oberkategorie/User/{OberkategorieId}/{userId}") | public List<User> addOberkategorieUserById(@PathVariable int OberkategorieId, @PathVariable int userId) |
| @PostMapping("/Oberkategorie/Unterkategorie/{oberkategorieId}/{unterkategorieId}") | public List<Unterkategorie> addOberkategorieUnterkategorieById(@PathVariable int oberkategorieId, @PathVariable int unterkategorieId) |
| @PostMapping("/Oberkategorie/Veranstaltung/{OberkategorieId}/{veranstaltungId}") | public List<Veranstaltung> addOberkategorieVeranstaltungById(@PathVariable int OberkategorieId, @PathVariable int veranstaltungId) |

ParkplatzRestController

| | |
|---|---|
| @GetMapping("/Parkplatz") | public List<Parkplatz> findAll() |
| @GetMapping("/Parkplatz/{ParkplatzId}") | public Parkplatz getParkplatz(@PathVariable int ParkplatzId) |
| @PostMapping("/Parkplatz") | public Parkplatz addParkplatz(@RequestBody Parkplatz theParkplatz) |
| @PutMapping("/Parkplatz") | public Parkplatz updateParkplatz(@RequestBody Parkplatz theParkplatz) |
| @DeleteMapping("/Parkplatz/{ParkplatzId}") | public String deleteParkplatz(@PathVariable int ParkplatzId) |
| @GetMapping("/Parkplatz/Adresse/{ParkplatzId}") | public Adresse getParkplatzAdresse(@PathVariable int ParkplatzId) |
| @PostMapping("/Parkplatz/Adresse/{ParkplatzId}/{AdressId}") | public Adresse addParkplatzAdresse(@PathVariable int ParkplatzId, @PathVariable int AdressId) |

RequestRestController

| | |
|---|---|
| @PostMapping("/Veranstaltung/Request") | public List<Veranstaltung> findFilterVeranstaltung(@RequestBody Request request) |

UnterkategorieRestController

| | |
|---|---|
| @GetMapping("/Unterkategorie") | public List<Unterkategorie> findAll() |
| @GetMapping("/Unterkategorie/{UnterkategorieId}") | public Unterkategorie getUnterkategorie(@PathVariable int UnterkategorieId) |

| | |
|---|---|
| @PostMapping("/Unterkategorie") | public Unterkategorie addUnterkategorie(@RequestBody Unterkategorie theUnterkategorie) |
| @PutMapping("/Unterkategorie") | public Unterkategorie updateUnterkategorie(@RequestBody Unterkategorie theUnterkategorie) |
| @DeleteMapping("/Unterkategorie/{UnterkategorieId}") | public String deleteUnterkategorie(@PathVariable int UnterkategorieId) |
| @GetMapping("/Unterkategorie/Oberkategorie/{UnterkategorieId}") | public Oberkategorie getOberkategorie(@PathVariable int UnterkategorieId) |
| @GetMapping("/Unterkategorie/User/{UnterkategorieId}") | public List<User> getUnterkategorieUsers(@PathVariable int UnterkategorieId) |
| @GetMapping("/Unterkategorie/Veranstaltung/{UnterkategorieId}") | public List<Veranstaltung> getUnterkategorieVeranstaltungen(@PathVariable int UnterkategorieId) |
| @PostMapping("/Unterkategorie/Veranstaltung/{UnterkategorieId}") | public List<Veranstaltung> addUnterkategorieVeranstaltung(@PathVariable int UnterkategorieId, @RequestBody Veranstaltung theVeranstaltung) |
| @PostMapping("/Unterkategorie/Veranstaltung/{UnterkategorieId}/{veranstaltungId}") | public List<Veranstaltung> addUnterkategorieVeranstaltungById(@PathVariable int UnterkategorieId, @PathVariable int veranstaltungId) |
| @PostMapping("/Unterkategorie/User/{UnterkategorieId}") | public List<User> addUnterkategorieUser(@PathVariable int UnterkategorieId, @RequestBody User theUser) |
| @PostMapping("/Unterkategorie/User/{UnterkategorieId}/{userId}") | public List<User> addUnterkategorieUserById(@PathVariable int UnterkategorieId, @PathVariable int userId) |

UserRestController

| | |
|---|---|
| @GetMapping("/User") | public List<User> findAll() |
| @GetMapping("/User/{UserId}") | public User getUser(@PathVariable int UserId) |
| @PostMapping("/User") | public User addUser(@RequestBody User theUser) |
| @PutMapping("/User") | public User updateUser(@RequestBody User theUser) |
| @DeleteMapping("/User/{UserId}") | public String deleteUser(@PathVariable int UserId) |
| @GetMapping("/User/Veranstaltung/{UserId}") | public List<Veranstaltung> getUserVeranstaltungen(@PathVariable int UserId) |

| | |
|---|---|
| @GetMapping("/User/Veranstalter/{UserId}") | public List<Veranstalter> getUserAlsVeranstalter(@PathVariable int UserId) |
| @GetMapping("/User/Oberkategorie/{UserId}") | public List<Oberkategorie> getUserOberkategorien(@PathVariable int UserId) |
| @GetMapping("/User/Unterkategorie/{UserId}") | public List<Unterkategorie> getUserUnterkategorien(@PathVariable int UserId) |
| @PostMapping("/User/Veranstaltung/{UserId}") | public List<Veranstaltung> addUserVeranstaltung(@PathVariable int UserId, @RequestBody Veranstaltung theVeranstaltung) |
| @PostMapping("/User/Veranstalter/{userId}") | public List<Veranstalter> addUserAlsVeranstalter(@PathVariable int userId, @RequestBody Veranstalter theVeranstalter) |
| @PostMapping("/User/Oberkategorie/{userId}") | public List<Oberkategorie> addUserOberkategorie(@PathVariable int userId, @RequestBody Oberkategorie theOberkategorie) |
| @PostMapping("/User/Unterkategorie/{userId}") | public List<Unterkategorie> addUserUnterkategorie(@PathVariable int userId, @RequestBody Unterkategorie theUnterkategorie) |
| @PostMapping("/User/Veranstaltung/{UserId}/{VeranstaltungId}") | public List<Veranstaltung> addUserVeranstaltungById(@PathVariable int UserId, @PathVariable int VeranstaltungId) |
| @PostMapping("/User/Veranstalter/{userId}/{veranstalterId}") | public List<Veranstalter> addUserAlsVeranstalterById(@PathVariable int userId, @PathVariable int veranstalterId) |
| @PostMapping("/User/Oberkategorie/{userId}/{oberkategorieId}") | public List<Oberkategorie> addUserOberkategorieById(@PathVariable int userId, @PathVariable int oberkategorieId) |
| @PostMapping("/User/Unterkategorie/{userId}/{unterkategorieId}") | public List<Unterkategorie> addUserUnterkategorieById(@PathVariable int userId, @PathVariable int unterkategorieId) |
| @PostMapping("/User/ByPW") | public User findUserByMaillAndPW (@RequestBody Login login ) |
| @PostMapping("/User/Existiert") | public boolean existiertMail ( @RequestBody Email email) |

VeranstalterRestController

| | |
|---|---|
| @GetMapping("/Veranstalter") | public List<Veranstalter> findAll() |
| @GetMapping("/Veranstalter/{VeranstalterId}") | public Veranstalter getVeranstalter(@PathVariable int VeranstalterId) |
| @PostMapping("/Veranstalter") | public Veranstalter addVeranstalter(@RequestBody Veranstalter theVeranstalter) |
| @PutMapping("/Veranstalter") | public Veranstalter updateVeranstalter(@RequestBody Veranstalter theVeranstalter) |
| @DeleteMapping("/Veranstalter/{VeranstalterId}") | public String deleteVeranstalter(@PathVariable int VeranstalterId) |
| @GetMapping("/Veranstalter/User/{VeranstalterId}") | public List<User> getVeranstalterUsers(@PathVariable int VeranstalterId) |
| @PostMapping("/Veranstalter/User/{veranstalterId}/{userId}") | public List<User> addVeranstalterUser(@PathVariable int veranstalterId, @PathVariable int userId) |
| @PostMapping("/Veranstalter/Veranstaltung/{veranstalterId}/{veranstaltungId}") | public List<Veranstaltung> addVeranstalterVeranstaltungById(@PathVariable int veranstalterId, @PathVariable int veranstaltungId) |
| @PostMapping("/Veranstalter/Veranstaltung/{veranstalterId}") | public List<Veranstaltung> addVeranstalterVeranstaltung(@PathVariable int veranstalterId, @RequestBody Veranstaltung theVeranstaltung) |
| @GetMapping("/Veranstalter/Veranstaltung/{veranstalterId}") | public List<Veranstaltung> findVeranstalterVeranstaltungen(int veranstalterId) |

VeranstalterverifizierungRestController

| | |
|---|---|
| @GetMapping("/Veranstalterverifizierung") | public List<Veranstalterverifizierung> findAll() |
| @GetMapping("/Veranstalterverifizierung/{VeranstalterverifizierungId}") | public Veranstalterverifizierung getVeranstalterverifizierung(@PathVariable int VeranstalterverifizierungId) |
| @PostMapping("/Veranstalterverifizierung") | public Veranstalterverifizierung addVeranstalterverifizierung( @RequestBody Veranstalterverifizierung theVeranstalterverifizierung) |
| @DeleteMapping("/Veranstalterverifizierung/{VeranstalterverifizierungId}") | public String deleteVeranstalterverifizierung(@PathVariable int VeranstalterverifizierungId) |

VeranstaltungRestController

| | |
|---|---|
| @GetMapping("/Veranstaltung") | public List<Veranstaltung> findAll() |
| @GetMapping("/Veranstaltung/{VeranstaltungId}") | public Veranstaltung getVeranstaltung(@PathVariable int VeranstaltungId) |
| @PostMapping("/Veranstaltung") | public Veranstaltung addVeranstaltung(@RequestBody Veranstaltung theVeranstaltung) |
| @PutMapping("/Veranstaltung") | public Veranstaltung updateVeranstaltung(@RequestBody Veranstaltung theVeranstaltung) |
| @DeleteMapping("/Veranstaltung/{VeranstaltungId}") | public String deleteVeranstaltung(@PathVariable int VeranstaltungId) |
| @GetMapping("/Veranstaltung/Veranstalter/{VeranstaltungId}") | public Veranstalter getVeranstaltungsVeranstalter(@PathVariable int VeranstaltungId) |
| @GetMapping("/Veranstaltung/Oberkategorie/{VeranstaltungId}") | public Oberkategorie getVeranstaltungOberkategorie(@PathVariable int VeranstaltungId) |
| @GetMapping("/Veranstaltung/Unterkategorie/{VeranstaltungId}") | public Unterkategorie getVeranstaltungUnterkategorie(@PathVariable int VeranstaltungId) |
| @GetMapping("/Veranstaltung/Adresse/{VeranstaltungId}") | public Adresse getVeranstaltungsAdresse(@PathVariable int VeranstaltungId) |
| @GetMapping("/Veranstaltung/User/{VeranstaltungId}") | public List<User> getVeranstaltungsBesucher(@PathVariable int VeranstaltungId) |
| @PostMapping("/Veranstaltung/Veranstalter/{VeranstaltungId}/{veranstalterId}") | public Veranstalter addVeranstaltungsVeranstalterById(@PathVariable int VeranstaltungId, @PathVariable int veranstalterId) |
| @PostMapping("/Veranstaltung/Veranstalter/{VeranstaltungId}") | public Veranstalter addVeranstaltungsVeranstalter(@PathVariable int VeranstaltungId, @RequestBody Veranstalter theVeranstalter) |
| @PostMapping("/Veranstaltung/User/{veranstaltungId}/{userId}") | public List<User> addVeranstaltungsBesucherById(@PathVariable int veranstaltungId, @PathVariable int userId) |
| @PostMapping("/Veranstaltung/User/{veranstaltungId}") | public List<User> addVeranstaltungsBesucher(@PathVariable int veranstaltungId, @RequestBody User theUser) |
| @PostMapping("/Veranstaltung/Oberkategorie/{veranstaltungId}/{oberkategorieId}") | public Oberkategorie addVeranstaltungsOberkategorieById(@PathVariable int veranstaltungId, @PathVariable int oberkategorieId) |

| | |
|---|---|
| @PostMapping("/Veranstaltung/Oberkategorie/{veranstaltungId}") | public Oberkategorie addVeranstaltungsOberkategorie(@PathVariable int veranstaltungId, @RequestBody Oberkategorie theOberkategorie) |
| @PostMapping("/Veranstaltung/Unterkategorie/{veranstaltungId}/{unterkategorieId}") | public Unterkategorie addVeranstaltungsUnterkategorieById(@PathVariable int veranstaltungId, @PathVariable int unterkategorieId) |
| @PostMapping("/Veranstaltung/Unterkategorie/{veranstaltungId}") | public Unterkategorie addVeranstaltungsUnterkategorie(@PathVariable int veranstaltungId, @RequestBody Unterkategorie theUnterkategorie) |
| @PostMapping("/Veranstaltung/Adresse/{veranstaltungId}/{adressId}") | public Adresse addVeranstaltungsAdresseById(@PathVariable int veranstaltungId, @PathVariable int adressId) |
| @PostMapping("/Veranstaltung/Adresse/{veranstaltungId}") | public Adresse addVeranstaltungsAdresse(@PathVariable int veranstaltungId, @RequestBody Adresse theAdresse) |
| @GetMapping("/Veranstaltung/Highlights") | public List<Veranstaltung> findHighlights() |
| @PostMapping("/Request") | public Request findRequest(@RequestBody Request request) |
| @GetMapping("/Veranstaltung/Freigeben") | public List<Veranstaltung> findFreizugebendeVeranstaltungen() |

VorschlagRestController

| | |
|---|---|
| @GetMapping("/User/Vorschlag/{userID}") | public List<Veranstaltung> findUserVeranstaltungVorschlaege(@PathVariable int userID) |

# Testprotokolle

| Zusammenfassung | 39 | 25 | 16 | 9 | 14 | 0 |
|---|---|---|---|---|---|---|
| | | | | | | |
| **Veranstaltungsportal für Mosbach Prototyp, Stand 27.01.2020** | | | | | | |
| | **Gesamtanzahl** | **getestet** | **OK** | **fehlerhaft** | **nicht testbar** | **TF unklar** |
| UC 1 Veranstaltung bearbeiten | 8 | 0 | 0 | 0 | 8 | 0 |
| UC 2 Veranstaltung einreichen | 7 | 7 | 1 | 6 | 0 | 0 |
| UC 3 Veranstaltung entfernen | 5 | 2 | 2 | 0 | 3 | 0 |
| UC 4 Veranstaltung freigeben | 3 | 0 | 0 | 0 | 3 | 0 |
| UC 5 Veranstaltungen suchen | 8 | 8 | 6 | 2 | 0 | 0 |
| UC 6 V-Details anzeigen | 1 | 1 | 1 | 0 | 0 | 0 |
| UC 7 Als User registrieren | 4 | 4 | 4 | 0 | 0 | 0 |
| UC 8 Login | 2 | 2 | 2 | 0 | 0 | 0 |
| UC 9 Logout | 1 | 1 | 0 | 1 | 0 | 0 |

Anhang-Abbildung 24 Übersicht Testfälle

| 8 | 0 | 0 | 0 | 8 | 0 | **Testprotokoll** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Testfälle | | | | | | **8** | **0 OK** | **8 NOTOK** | | **Testergebnisse** | |
| Gesamtanzahl | getestet | OK | fehlerhaft | nicht testbar | TF unklar | **Testfall#** | **Testfall** | **Auslöser/ Anweisung** | **Ziel/ Erwartung Bedingung** | **Bemerkung** | **Testdatum** |
| | | | | | | **TF 1** | Benutzer klickt Link "Veranstaltung bearbeiten" | | | | |
| x | | | | x | | **TF 01.01** | | Klick auf "Veranstaltung bearbeiten" | Änderungs-Formular öffnet sich | | |
| | | | | | | **TF 2** | Benutzer ändert Daten | | | | |
| x | | | | x | | **TF 02.01** | Datum/Uhrzeit | | Änderung des Eintrags ist möglich | | |
| x | | | | x | | **TF 02.02** | Tickets | | Änderung des Eintrags ist möglich | | |
| x | | | | x | | **TF 02.03** | Preis | | Änderung des Eintrags ist möglich | | |
| x | | | | x | | **TF 02.04** | Kategorie | | Änderung des Eintrags ist möglich | | |
| x | | | | x | | **TF 02.05** | Veranstaltungsort | | Änderung des Eintrags ist möglich | | |
| x | | | | x | | **TF 02.06** | Veranstaltungsbild | | Hochladen eines neuen Bilds ist möglich | | |
| | | | | | | **TF3** | Veranstaltung bestätigen | | | | |
| x | | | | x | | **TF 03.01** | Fehlermeldung bei Fehlern anzeigen | Klick auf "Geänderte Veranstaltung einreichen" | Anzeige von Fehlermeldung, wenn nicht alle Pflichtfelder ausgefüllt sind | | |
| | | | | | | | Rückführung zur Veranstaltungsseite | Klick auf "Geänderte Veranstaltung einreichen" | User wird auf Veranstaltungsseite zurückgeleitet | | |
| | | | | | | | Bestätigungsnachricht auf Veranstaltungsseite | Rückführung zur Veranstaltungsseite | Bestätigungsnachricht wird angezeigt | | |

Anhang-Abbildung 25 Testprotokoll Veranstaltung bearbeiten

| Gesamtanzahl | getestet | OK | fehlerhaft | nicht testbar | TF unklar | Testprotokoll | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 1 | 6 | 0 | 0 | 7 | 1 OK | 6 NOTOK | | Testergebnisse | |
| Testfälle | | | | | | Testfall# | Testfall | Auslöser/ Anweisung | Ziel/ Erwartung Bedingung | Bemerkung | Testdatum |
| | | | | | | TF 1 | Benutzer klickt Link "Veranstaltung einreichen" an | | | | |
| x | x | x | | | | TF 01.01 | | Klick auf "Veranstaltung einreichen" | Formular zum Einreichen einer Veranstaltung öffnet sich | | 27.01.2020 |
| | | | | | | TF 2 | Benutzer gibt benötigte Informationen ein | | | | |
| x | x | | x | | | TF 02.01 | Pflichtfelder und Falscheingaben prüfen | alle Daten eingeben | Fehlermeldung bei Klick auf "Veranstaltung einreichen" | bisher keine Prüfung | 27.01.2020 |
| x | x | | x | | | TF 02.02 | Datumseingabe prüfen | Datum aus Kalender auswählen | nur zukünftige Daten auswählbar | vergangenes Datum möglich | 27.01.2020 |
| x | x | | x | | | TF 02.03 | Veranstaltungsbild anhängen | Bild vom PC auswälen | Bild wird im Formular angezeigt | wird "hochgeladen" und verschwindet dann aus der Ansicht | 27.01.2020 |
| | | | | | | TF3 | Benutzer betätigt Button "bestätigen" | | | | |
| x | x | | x | | | TF 03.01 | Fehlermeldung bei Fehlern anzeigen | Klick auf "Veranstaltung einreichen" | Anzeige von Fehlermeldung, wenn nicht alle Pflichtfelder ausgefüllt sind | keine Fehlermeldungen | 27.01.2020 |
| x | x | | x | | | TF 03.02 | Rückführung zur Veranstaltungsseite | Klick auf "Veranstaltung einreichen" | User wird auf Veranstaltungsseite | keine Rückführung | 27.01.2020 |
| x | x | | x | | | TF 03.03 | Bestätigungsnachricht auf Veranstaltungsseite | Rückführung zur Veranstaltungsseite | Bestätigungsnachricht wird angezeigt | keine Bestätigung | 27.01.2020 |

Anhang-Abbildung 26 Testprotokoll Veranstaltung einreichen

| Gesamtanzahl | getestet | OK | fehlerhaft | nicht testbar | TF unklar | Testprotokoll | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 2 | 2 | 0 | 3 | 0 | 5 | 2 OK | 3 NOTOK | | Testergebnisse | |
| Testfälle | | | | | | Testfall# | Testfall | Auslöser/ Anweisung | Ziel/ Erwartung Bedingung | Bemerkung | Testdatum |
| | | | | | | TF 1 | Benutzer ruft Liste mit seinen Veranstaltungen auf | | | | |
| x | x | x | | | | TF 01.01 | | Klick auf "User-Profil" | User-Profil öffnet sich | | 27.01.2020 |
| | | | | | | TF 2 | Benutzer wählt zu löschende Veranstaltung aus | | | | |
| x | x | x | | | | TF 02.01 | | Klick auf gewünschte Veranstaltung | Detail-Seite öffnet sich | | 27.01.2020 |
| | | | | | | TF 3 | Benutzer klickt auf Link "Veranstaltung entfernen" | | | | |
| x | | | | x | | TF 03.01 | | Klick auf "Veranstaltung entfernen" | Sicherheitsabfrage öffnet sich | | 27.01.2020 |
| | | | | | | TF 4 | Benutzer bestätigt Sicherheitsabfrage | | | | |
| x | | | | x | | TF 04.01 | | Klick auf "Ok" | User-Profil soll sich öffnen | | 27.01.2020 |
| | | | | | | TF 5 | Benachrichtigung wird an Mitarbeiter der Stadt versandt | | | | |
| x | | | | x | | TF 05.01 | | Veranstaltung wurde gelöscht | Benachrichtigung über Löschung wird an MitarbeiterIn der Stadt versandt | | 27.01.2020 |

Anhang-Abbildung 27 Testprotokoll Veranstaltung entfernen

| Gesamtanzahl | getestet | OK | fehlerhaft | nicht testbar | TF unklar | Testprotokoll | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 0 | 0 | 0 | 3 | 0 | Testfälle | 3 | 0 OK | 3 NOTOK | | Testergebnisse |
| | | | | | | | **Testfall#** | **Testfall** | **Auslöser/ Anweisung** | **Ziel/ Erwartung Bedingung** | **Bemerkung** | **Testdatum** |
| | | | | | | **TF 1** | Sachbearbeiter wählt Veranstaltung aus Liste zu prüfender Veranstaltungen aus | | | | | |
| x | | | | x | | **TF 01.01** | | Klick auf gewünschte Veranstaltung | Detail-Seite wird geöffnet | | | |
| | | | | | | **TF3** | Sachbearbeiter überprüft die Informationen zu der eingereichten Veranstaltung auf Richtigkeit | | | | | |
| x | | | | x | | **TF 03.01** | | | Veranstaltung wird als "in Ordnung" bewertet | | | |
| | | | | | | **TF 4** | Sachbearbeiter betätigt Button "Veranstaltung freigeben" | | | | | |
| x | | | | x | | **TF 04.01** | | Klick auf "Veranstaltung freigeben" | Veranstaltung wird für alle User sichtbar angezeigt | | | |

Anhang-Abbildung 28 Testprotokoll Veranstaltung freigeben

| Gesamtanzahl | getestet | OK | fehlerhaft | nicht testbar | TF unklar | Testprotokoll | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 8 | 6 | 2 | 0 | 0 | Testfälle | 8 | 6 OK | 2 NOTOK | | Testergebnisse | |
| | | | | | | | **Testfall#** | **Testfall** | **Auslöser/ Anweisung** | **Ziel/ Erwartung Bedingung** | **Bemerkung** | **Testdatum** |
| | | | | | | **TF 1** | Link "Veranstaltung suchen" wird aufgerufen | | | | | |
| x | x | x | | | | **TF 01.01** | | Klick auf "Veranstaltung suchen" | Suchmaske wird angezeigt | | | 27.01.2020 |
| | | | | | | **TF 2** | Beliebige Suchkriterien werden ausgewählt | | | | | |
| x | x | x | | | | **TF 02.01** | Volltextsuche | Ein Text wird in das vorgesehene Feld eingegeben | Die Daten werden zur Datenbankabfrage übermittelt | | | 27.01.2020 |
| x | x | x | | | | **TF 02.02** | Datum von | Von-Datum wird eingegeben | Die Daten werden zur Datenbankabfrage übermittelt | | | 27.01.2020 |
| x | x | x | | | | **TF 02.03** | Datum bis | Bis-Datum wird eingegeben | Die Daten werden zur Datenbankabfrage übermittelt | | | 27.01.2020 |
| x | x | x | | | | **TF 02.04** | Oberkategorie | Eine oder mehrere Oberkategorien werden ausgewählt | Die Unterkategorien werden dementsprechend sortiert | | | 27.01.2020 |
| x | x | x | | | | **TF 02.05** | Unterkategorie | Eine oder mehrere Unterkategorien werden ausgewählt | Die Daten werden zur Datenbankabfrage übermittelt | | | 27.01.2020 |
| | | | | | | **TF3** | Datenbank wird anhand der eingegebenen Kriterien nach passenden Veranstaltungen durchsucht | | | | | |
| x | x | | x | | | **TF 03.01** | | Eingeben von Suchkriterien | "Live"-Anzeige der Treffer im Suchen-Button | "Live"-Anzeige funktioniert nicht wegen TF4 | | 27.01.2020 |
| | | | | | | **TF4** | Suchergebnisse werden in einer Liste und auf einer Karte ausgegeben | | | | | |
| x | x | | x | | | **TF 04.01** | | Klick auf den Button "Suchen" | Suchergebnisse werden dargestellt | Suchkriterien werden nicht berücksichtigt -> Backend-Problem | | 27.01.2020 |

Anhang-Abbildung 29 Testprotokoll Veranstaltungen suchen

| Gesamtanzahl | getestet | OK | fehlerhaft | nicht testbar | TF unklar | Testprotokoll | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | Testfälle | 1 | 1 OK | 0 NOTOK | | Testergebnisse | |
| | | | | | | | **Testfall#** | **Testfall** | **Auslöser/ Anweisung** | **Ziel/ Erwartung Bedingung** | **Bemerkung** | **Testdatum** |
| | | | | | | **TF 2** | Link "Details anzeigen" wird angeklickt & Detailinformationen zu der Veranstaltung werden auf neuer Seite angezeigt | | | | | |
| x | x | x | | | | **TF 02.01** | Benutzer klickt auf "Details anzeigen" | Klick auf Button "Details anzeigen" | Veranstaltungsdetails sollen angezeigt werden | | | 27.01.2020 |

Anhang-Abbildung 30 Testprotokoll Details anzeigen

| Gesamtanzahl | getestet | OK | fehlerhaft | nicht testbar | TF unklar |
|---|---|---|---|---|---|
| 4 | 4 | 4 | 0 | 0 | 0 |

**Testprotokoll**

| 4 | 4 OK | 0 NOTOK | | Testergebnisse | |
|---|---|---|---|---|---|
| **Testfall#** | **Testfall** | **Auslöser/ Anweisung** | **Ziel/ Erwartung Bedingung** | **Bemerkung** | **Testdatum** |
| **TF 1** | Benutzer klickt Link "Registrieren" an | | | | |
| **TF 01.01** | | Klick auf den Button "Registrieren" | Registrier-PopUp öffnet sich | | 27.01.2020 |
| **TF 2** | Benutzer gibt seine E-Mail-Adresse und gewünschtes Passwort ein | | | | |
| **TF 02.01** | Eingabe E-Mail-Adresse | Eingabe in vorgesehenes Feld | | | 27.01.2020 |
| **TF 02.02** | Eingabe Passwort | Eingabe in vorgesehenes Feld | | | 27.01.2020 |
| **TF 02.03** | Freigabe | Klick auf Button "Ok" | Benutzer wird im System registriert | | 27.01.2020 |

Anhang-Abbildung 31 Testprotokoll Registrierung

| Gesamtanzahl | getestet | OK | fehlerhaft | nicht testbar | TF unklar |
|---|---|---|---|---|---|
| 2 | 2 | 2 | 0 | 0 | 0 |

**Testprotokoll**

| 2 | 2 OK | 0 NOTOK | | Testergebnisse | |
|---|---|---|---|---|---|
| **Testfall#** | **Testfall** | **Auslöser/ Anweisung** | **Ziel/ Erwartung Bedingung** | **Bemerkung** | **Testdatum** |
| **TF 1** | Benutzer klickt "Login" | | | | |
| **TF 01.01** | | Klick auf Button "Login" | Popup zur Eingabe von Benutzername und Passwort erscheint | | 27.01.2020 |
| **TF 01.02** | | Klick auf "OK"-Button innerhalb des Popups | Benutzer wird bei korrekter Kombination aus Benutzername/Passwort angemeldet, ansonsten erscheint eine Fehlermeldung | | 27.01.2020 |

Anhang-Abbildung 32 Testprotokoll Login

| Gesamtanzahl | getestet | OK | fehlerhaft | nicht testbar | TF unklar |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 0 |

**Testprotokoll**

| 1 | 0 OK | 1 NOTOK | | Testergebnisse | |
|---|---|---|---|---|---|
| **Testfall#** | **Testfall** | **Auslöser/ Anweisung** | **Ziel/ Erwartung Bedingung** | **Bemerkung** | **Testdatum** |
| **TF 1** | Benutzer klickt "Logout" | | | | |
| **TF 01.01** | | Klick auf Button "Logout" | Ausloggen und Rückführung zur Startseite | Logout funktioniert; Rückführung nicht | 27.01.2020 |

Anhang-Abbildung 33 Testprotokoll Logout