Semesterprojekt Webapplikationen
Dokumentation inkl. Installationsanleitung

3. Juni 2016

Autor: Samuel Loepfe
Abgabetermin: 03.06.2016
Dozenten: Martin Studer / Norman Süsstrunk

Inhaltsverzeichnis

1 Projektidee

Der Turnverein Kirchberg SG führt jedes zweite Jahr eine Turnunterhaltung durch, dabei ist jeweils der Durchführungsort immer gleich. Dadurch sind auch die Dimensionen der Bühne gegeben. Dennoch wird jedes Mal eine enorme Zeit beansprucht bis jede Gruppe ihr Material und ihren Aufbau dem Aufbauleiter verständlich gemacht hat. Die Idee des Webapplikationsprojekt greift hier an. Um alle Parteien zu entlasten, Zeit einzusparen wo möglich und um auch nachträglich die Aufstellungen noch mal ansehen zu können, kommt die Applikation ins Spiel. Mittels diesem Webauftritt soll eine einheitliche Darstellung des gesamten Ablaufs möglich sein. Mit den üblichen Werkzeugen für Administratoren (Login, erstellen von neuen Aufbauten, löschen von Einträgen und einer Diashow der bestehenden Aufbauten) und einer einfachen Oberfläche für nicht technik-affine soll der Ablauf der Unterhaltung rasch und simpel er- und dargestellt werden können. Es soll eine Liste mit dem benötigten Material geben, die beliebig verändert werden kann, die Startzeit, Reihenfolge, Verantwortliche Person mit Telnr, Dauer und Fotos aus verschiedenen Winkeln werden angegeben.

Bsp: Aktivriege -> Darbietung mit Trampolin und Ringe.

- Person (M.Müller 078 123 45 67)
- Reihenfolge (1)
- Startzeit (20.13)
- Dauer (4.50min)
- Lied (Move It)
- Material
    1. 3x Ringe 1.2m ab Boden
    2. 2x Minitramp
    3. 2x grosse Matten
    4. 15 kleine Matten
    5. Magnesium
- Bilder
    1. Bild1 (Front)
    2. Bild2 (Links)
    3. Bild3 (Rechts)

2 Dokumentation

Die Webapplikation *Stagebuilder* soll den Aufbauaufwand für Turnunterhaltungen minimieren. Die Applikation benötigt nur Datenbanken, je nach Sicherheitskonzept sind dies verschiedene für Benutzer und Inhalt. Die Applikation soll auf einem gewöhnlichen Server lauffähig sein. Die folgende Beschreibung wird am Besten unter Zuhilfenahme des Datenmodell gelesen:

Main View Die Main_View bietet zwei Möglichkeiten weiterzufahren:

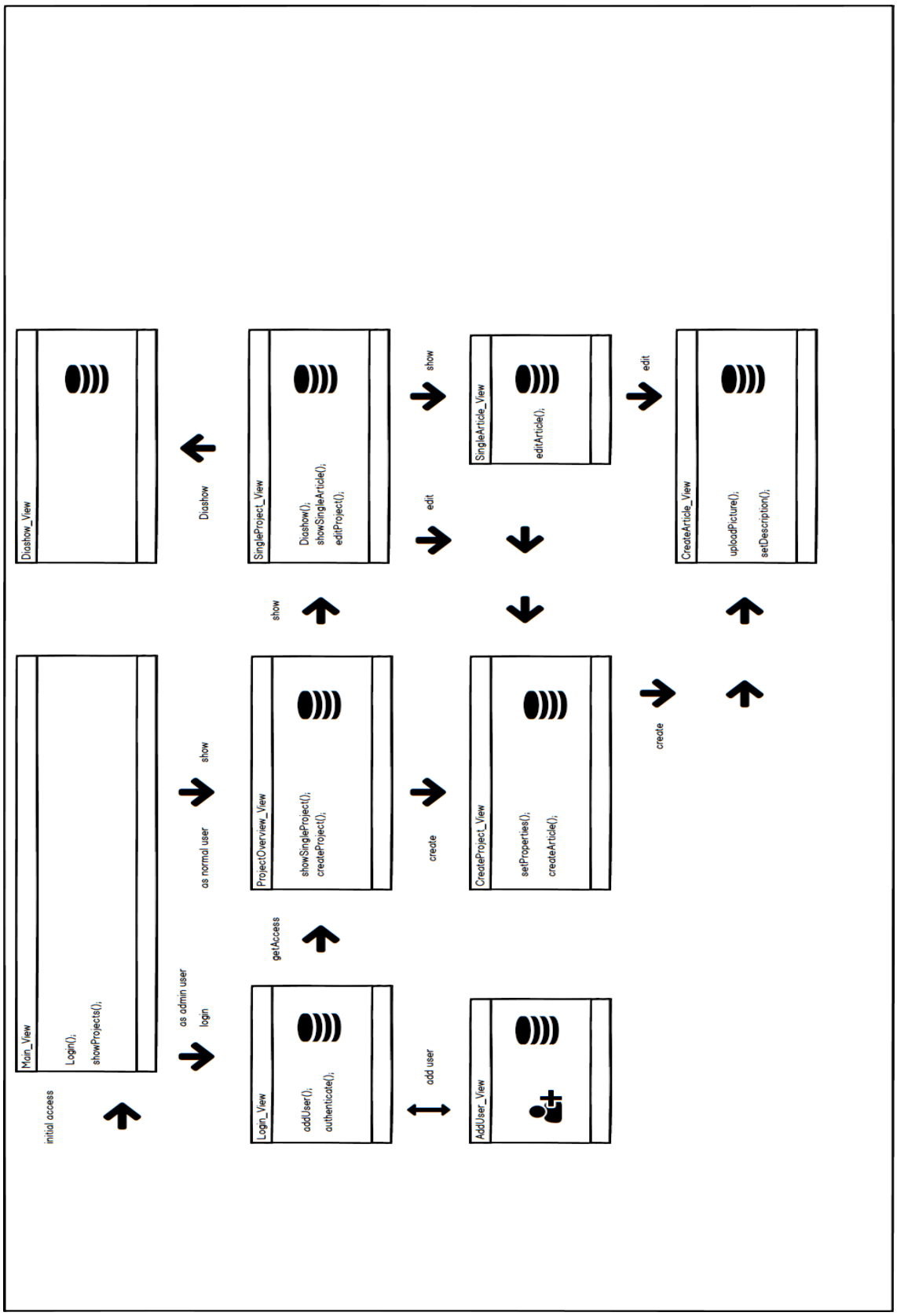
Fall 1 User Für alle interessierten Benutzer gibt es die Möglichkeit die Projekte anzusehen. Sie können aber keine erstellen oder editieren. Von der Main_View gelangen diese User auf die ProjectOverview_View und sehen dort alle bestehenden Projekte. Wählen sie eines aus, gelangen sie auf die SingleProject_View, wo sie das Projekt ansehen können. Ebenso können sie auch einzelne Beiträge ansehen, dafür kommen sie auf die SingleArticle_View. Damit ist der Funktionsumfang für nicht registrierte Benutzer erschöpft.

Fall 2 Admin Für registrierte Nutzer gibt es die Login_View. Falls der Admin schon registriert ist, wird er nach erfolgreichem Login direkt zur ProjectOverview_View weitergeleitet. Falls der Admin noch nicht in der Datenbank (DB) existiert, kommt er auf die AddUser_View. Dort wird ein Admin angelegt und nach erfolgreicher Speicherung in der DB wieder die Login_View angezeigt. Dieser Schritt wird in einer späteren Version durch das manuelle Eintragen von User in die Datenbank ersetzt. Um jetzt aber die Applikation komfortabel bedienen zu können und um vom Lerninhalt der besagten Aufgabe profitieren zu können, wird dies aber noch so implementiert. Nach erfolgreichem Login wird die ProjectOverview_View angezeigt. Anders als der gewöhnliche User kann der Admin hier auch Projekte erstellen. Wird diese Möglichkeit gewählt, gelangt der Admin auf die CreateProject_View. Für ein Projekt braucht es die Properties sowie einen oder mehrere Beiträge. In den Properties werden Metadaten wie Zeit, Datum, Ersteller und Titel erfasst. Ein Beitrag besteht gemäss Projektidee aus 3 Fotos, einer Materialliste und Metadaten:

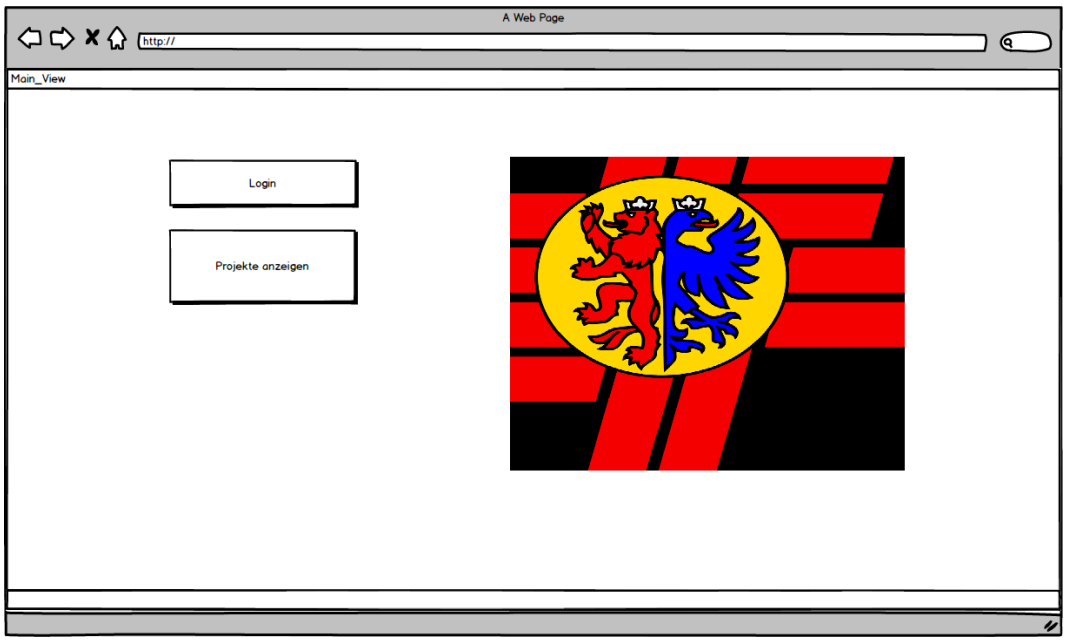
- Verantwortliche Person (M.Müller 078 123 45 67)
- Reihenfolge (1)
- Startzeit (20.13)
- Dauer (4.50min)
- Lied (Move It).

Dazu wird der Admin auf die CreateArticle_View geleitet. Hier wird einer oder mehrere Beiträge erstellt. Der Admin hat auch die Möglichkeit aus der SingleProject_View oder der SingleArticle_View die einzelnen Projekte und Beiträge zu editieren. Dazu wird er entweder auf die CreateProject_View oder die CreateArticle_View geleitet.

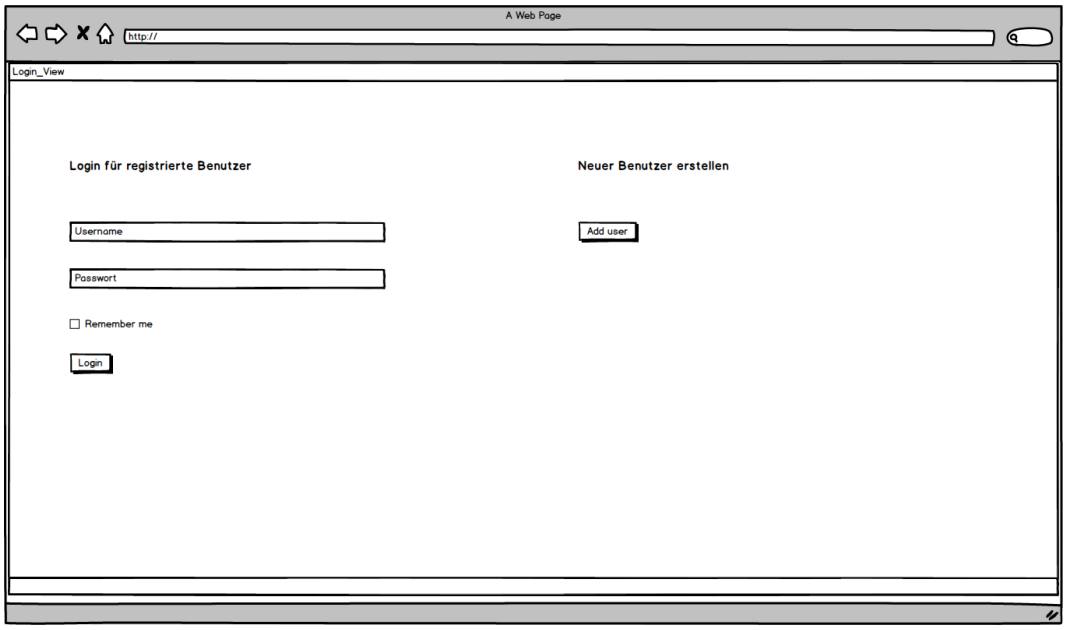
2.1 Pageview


Main_View
Login();
showProjects();
initial access
as admin user
login
Login_View
addUser();
authenticate();
getAccess
add user
AddUser_View
as normal user
show
ProjectOverview_View
showSingleProject();
createProject();
create
CreateProject_View
setProperties();
createArticle();
create
Diashow
Diashow_View
show
SingleProjekt_View
Diashow();
showSingleArticle();
editProject();
edit
show
SingleArticle_View
editArticle();
edit
CreateArticle_View
uploadPicture();
setDescription();

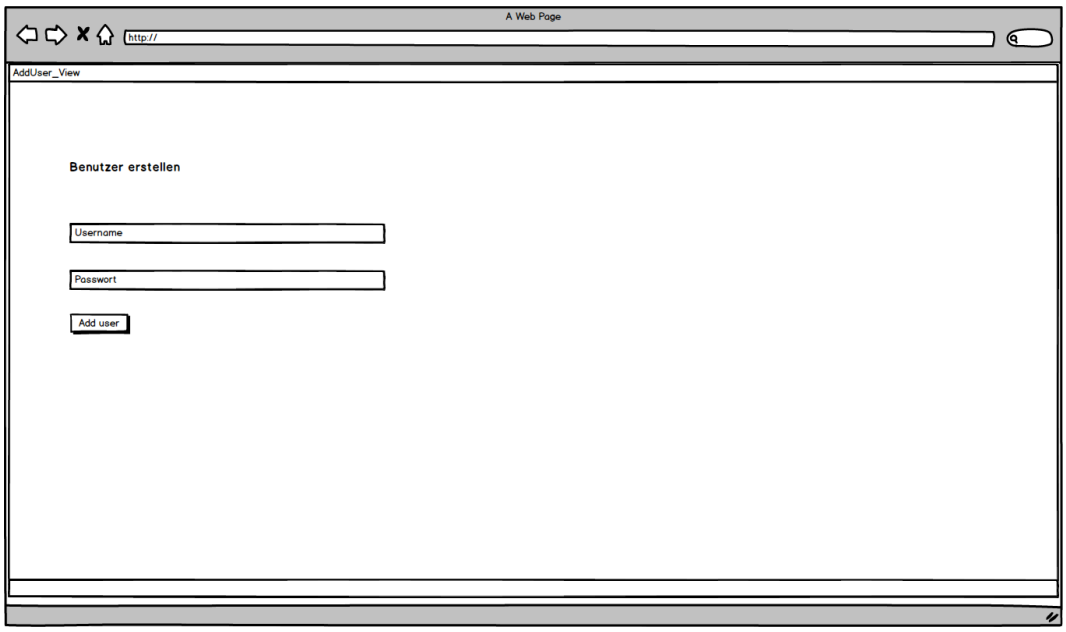
2.2 Mockup Main_View



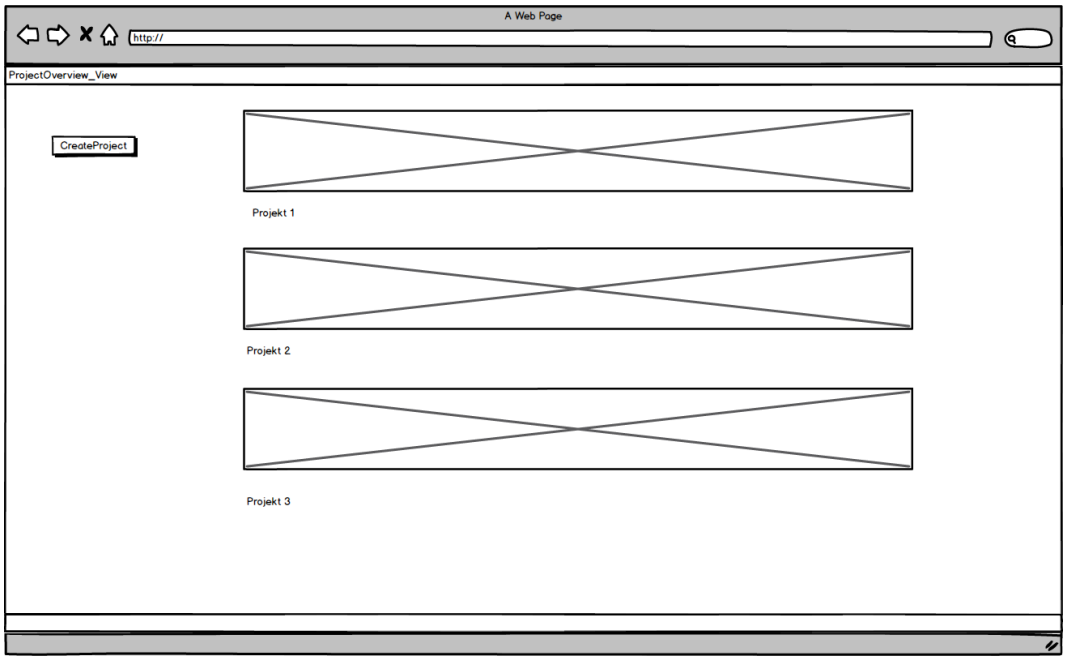
2.3 Mockup Login_View



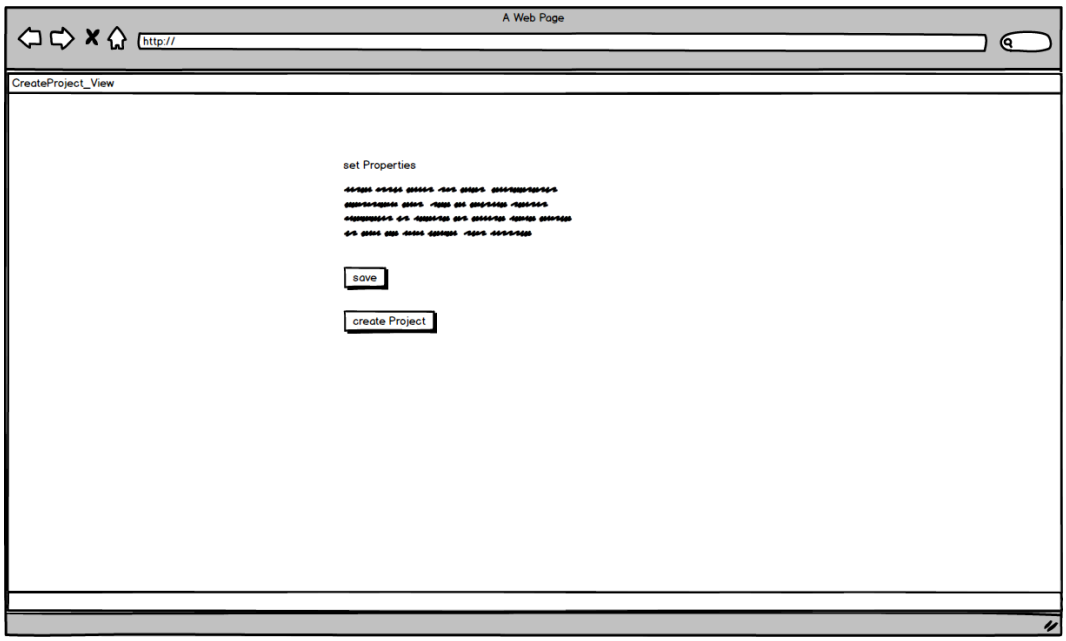
2.4 Mockup AddUser_View



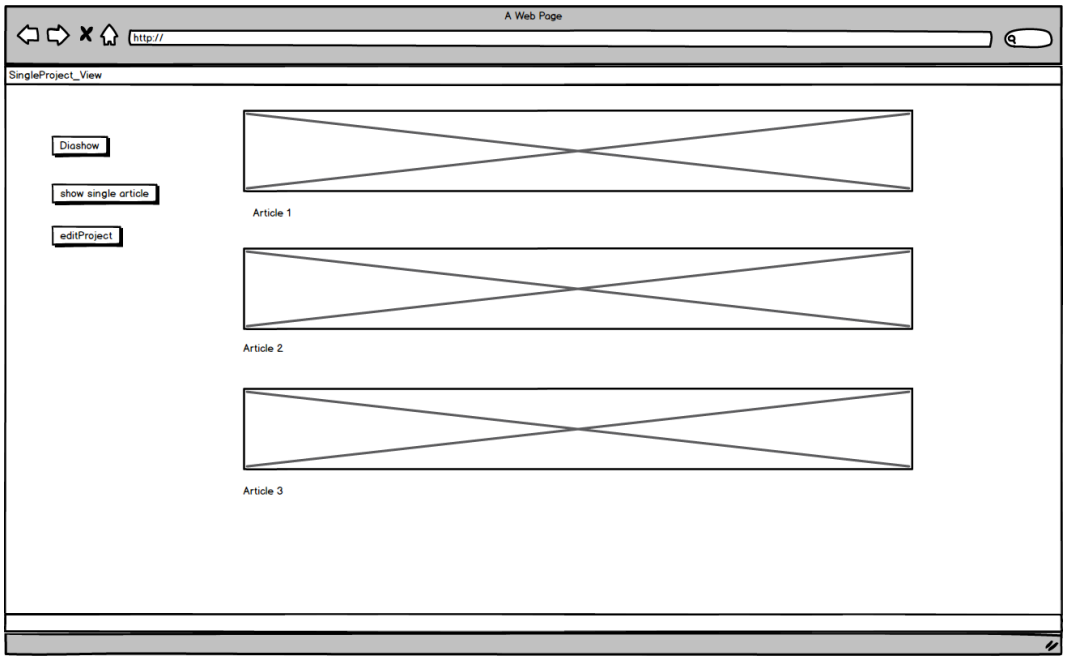
2.5 Mockup ProjectOverview_View



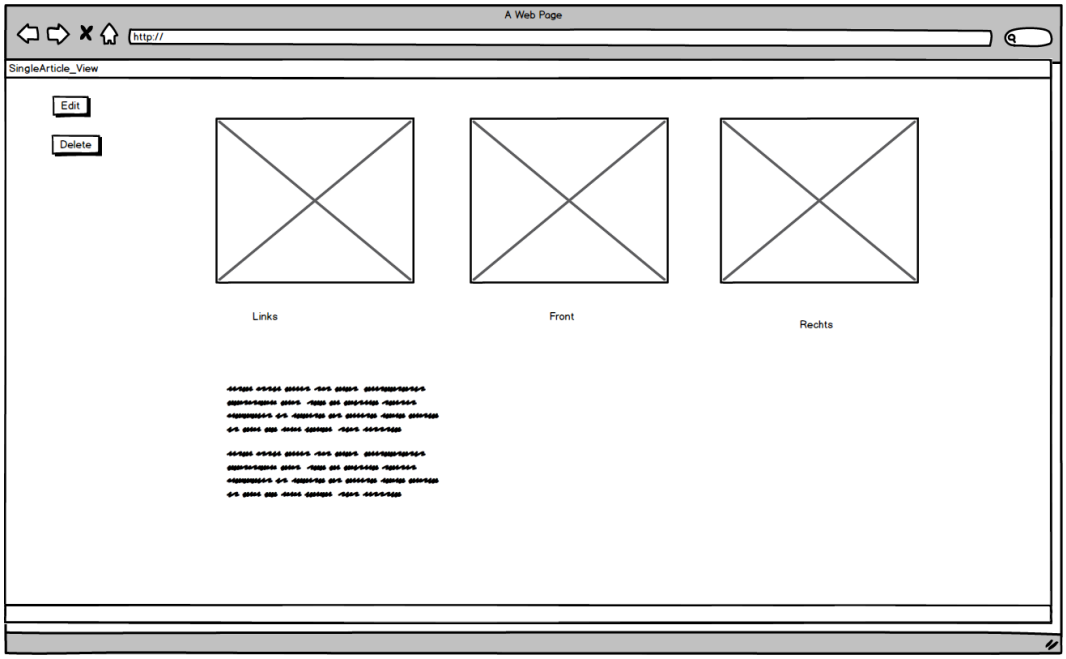
2.6 Mockup CreateProject_View



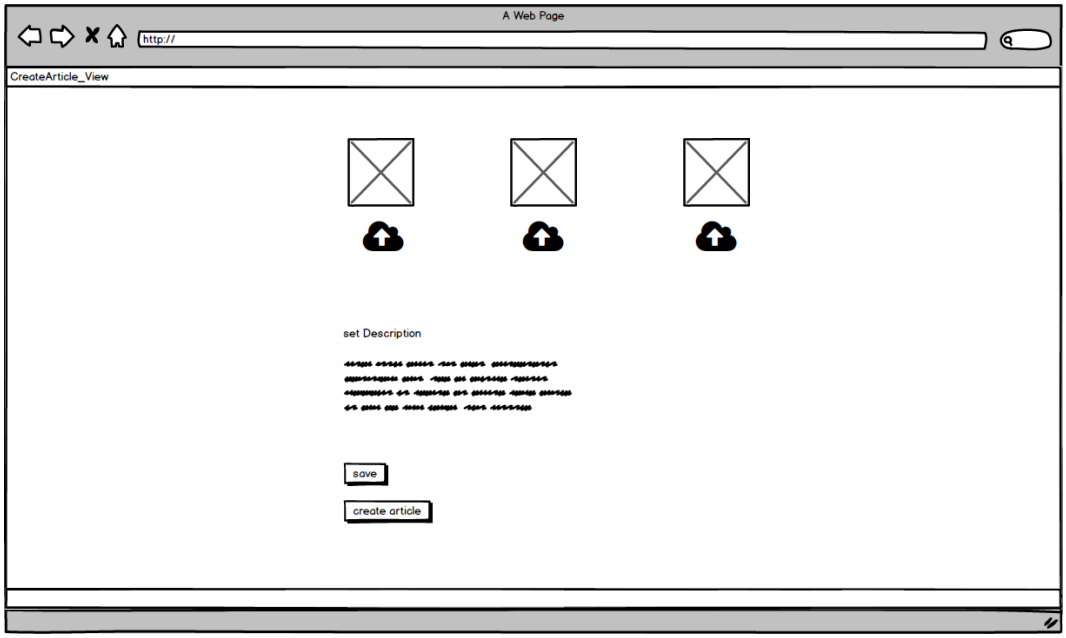
2.7 Mockup SingleProject_View



2.8 Mockup SingleArticle_View

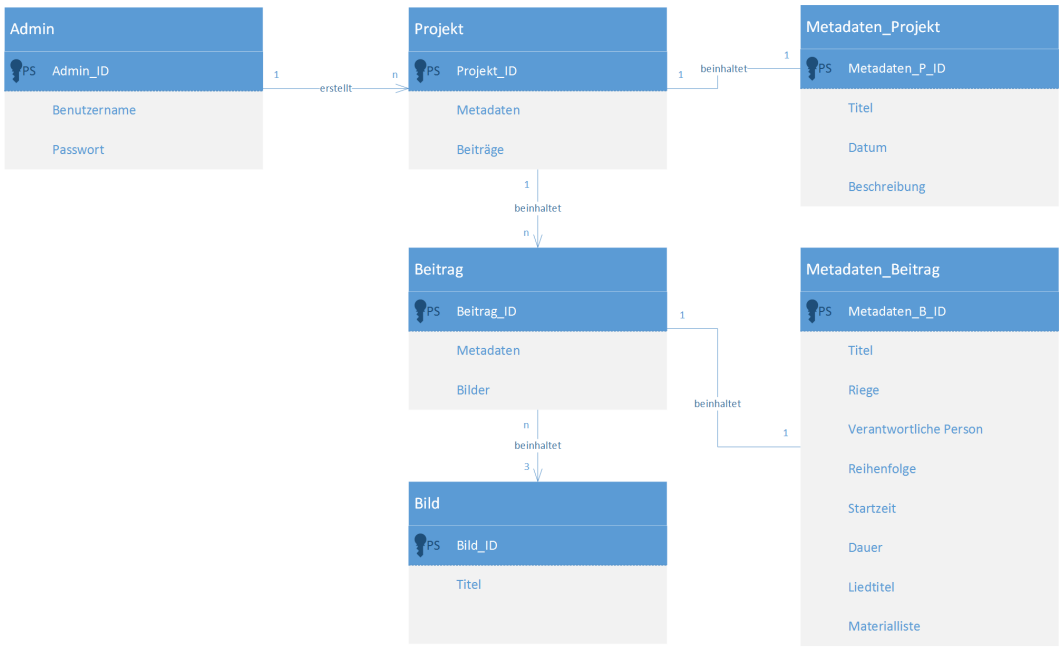


2.9 Mockup CreateArticle_View

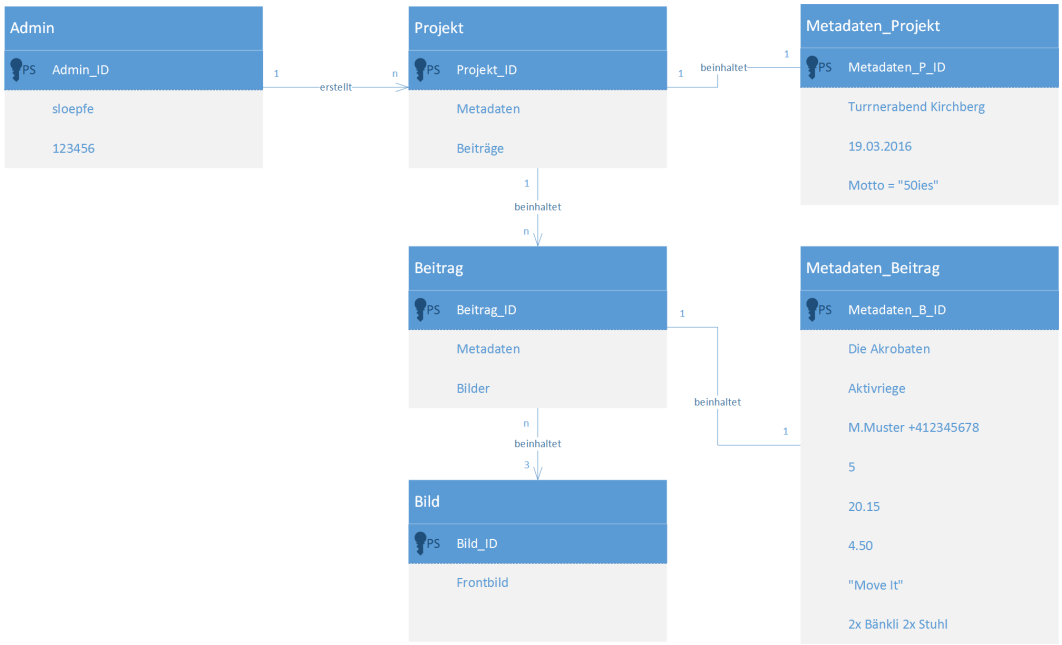


2.10 Datenmodelle Datenbank

2.10.1 Klassendiagramm



2.10.2 Objektdiagramm



2.11 Architektur

2.11.1 Einführung

StageBuilder basiert auf dem PHP-Framework Symfony. Symfony folgt dem MVC Paradigma, was sich sehr positiv auf den Arbeitsfluss und die Übersichtlichkeit auswirkt. Mittels Controller lässt sich die Logik klar gliedern und jederzeit ändern, ohne dass an der View grosse Veränderungen nötig gemacht werden. Als Beispiel soll die Erstellung eines Projektes dienen:

2.11.2 Login

Damit ein Projekt erstellt werden kann, ist ein Login notwendig. Der Securitycontroller erfüllt hier die Aufgaben Authentifizierung und Verifizierung. Mit einem Formular werden die Nutzerdaten entgegengenommen:

Diese werden dann in der Logik ausgewertet und überprüft. Stimmen sie mit einem registrierten Benutzer überein, wird auf die Projektübersichtsseite gewechselt und der Link *Neues Projekt erstellen* freigeschaltet. Nachfolgend der Code des Securitycontrollers und der entsprechenden View:

```
public function loginAction(Request $request)
{
    $authenticationUtils = $this-> get('security.authentication_utils');

    // get login error
    $error = $authenticationUtils->getLastAuthenticationError();

    //last username entered by user
    $lastUsername = $authenticationUtils->getLastUsername();

    return $this-> render(
            'security/login.html.twig',
            array(
                //last username entered by user
                'last_username'=> $lastUsername,
                'error'=> $error,
                )
            );

}
```

```
{% block body %}



    <div id="top" class="jumbotron container-fluid">
        <div class="container">
        <h2>Loggen Sie sich bitte ein:</h2>
        </div>

        {% if error %}
            <div class="container">
                {{ error.messageKey|trans(error.messageData, 'security') }}
            </div>
        {% endif %}
    <div class="container">
    <form action="{{ path('login') }}" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="_username" value="{{ last_username }}" />

    <label for="password">Password:</label>
    <input type="password" id="password" name="_password" />

    {#
        If you want to control the URL the user
        is redirected to on success (more details below)
        <input type="hidden" name="_target_path" value="/account" />
    #}
        <input type="hidden" name="_target_path" value="/projektUebersicht" />

    <button type="submit" class="btn btn-success">login</button>
    </form>
    </div>
    </div>


{% endblock %}
```

Um ein Projekt zu erstellen braucht es die unter Abschnitt *Fall 2 Admin* genannten Angaben, diese werden in der View dem Template übergeben:

```
<div id="wrapper">
    <div id="container">
        <div id="welcome">
            <h1>Erstelle ein neues Projekt</h1>
        </div>

        <div id="next">
            <p>
            <h2>Was wird benötigt?</h2>
            <h3>Titel</h3>
            <h3>Verantwortliche Person</h3>
            <h3>Datum</h3>
            <h3>Einen oder mehrere Beiträge</h3>

            {% form_theme form 'bootstrap_3_layout.html.twig' %}
            {{ form_start(form) }}
            {{ form_row(form.titel) }}
            {{ form_row(form.ersteller) }}
            {{ form_row(form.datum) }}
            {{ form_row(form.brochure) }}
```

Die Angaben werden wiederum vom Controller verarbeitet und in die Datenbank geschrieben. Jedes Projekt wird als eine Entität der Klasse Entity/Projekt erstellt:

```php
/**
 * @ORM\Entity
 * @ORM\Table(name="projekte")
 * @ORM\Entity(repositoryClass="AppBundle\Entity\ProjektRepository")
 */
class Projekt {

    /**
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=100)
     */
    private $titel;

    /**
     * @ORM\Column(type="string", length=100)
     */
    private $ersteller;


    /**
     *@ORM\Column(type="string", length=100)
     *
     */
    private $datum;
```

Die Instanzen der Klasse Projekt oder Beitrag werden durch den ORM verwaltet. Dieser stellt sicher, dass sie konsistent sind. Ebenfalls einen Auszug aus dem Projektcontroller:

Symfony bietet sehr viele vorbereitete Hilfe in Form eines Cookbooks, welches sehr einfach gehalten und äusserst übersichtlich gegliedert wurde. Zu jedem Problem findet man schnell einen Lösungsansatz. Ebenfalls existiert eine grosse Community, welche schon viele Probleme erfolgreich gelöst hat.

```php
public function NeuesProjektAction(Request $request)
{
    // build the form
    $projekt = new Projekt();
    $form = $this->createForm(ProjektType::class, $projekt);

    //handle the submit
    $form->handleRequest($request);
    if($form->isSubmitted() && $form ->isValid()){

        // $file stores the uploaded PDF file
        /** @var Symfony\Component\HttpFoundation\File\UploadedFile $file */
        $file = $projekt->getBrochure();

        // Generate a unique name for the file before saving it
        $fileName = md5(uniqid()).'.'.$file->guessExtension();

        // Move the file to the directory where brochures are stored
        $brochuresDir = $this->container->getParameter('kernel.root_dir').'/../web/uploads/brochures';
        $file->move($brochuresDir, $fileName);

        // Update the 'brochure' property to store the PDF file name
        // instead of its contents
        $projekt->setBrochure($fileName);

        // ... persist the $product variable or any other work


        $em = $this->getDoctrine()->getManager();
        $em->persist($projekt);
        $em->flush();
        //return $this->redirect($this->generateUrl('app_product_list'));
        //$this->get('session')->getFlashBag()->add('project','Neues Projekt erstellt');

        return $this->redirectToRoute("neuer_Beitrag");
    }

    return $this -> render('neuesProjekt/neuesProjekt.html.twig',array('form' => $form->createView()));
```

2.12 Anforderungen

1. Webapplikation ist mit Browser erreichbar
2. Admin kann sich einloggen
3. Superuser legt Admin in Datenbank an und verteilt Login
4. Admin kann Projekte anlegen, editieren und löschen
5. User und Admin können Projekte und Beiträge ansehen mittels Einzelansicht
6. Projekte und Beiträge sind selektierbar
7. User und Admin können Projekte mittels Diashow ansehen
8. Bilder in der SingleArticle_View können einzeln vergrössert dargestellt werden
9. Die Projekte werden in einer Liste dargestellt
10. Beiträge werden in einer Liste dargestellt
11. Die Projektübersicht zeigt Frontbild von erstem Beitrag
12. Die Beitragsübersicht zeigt Frontbild von Beitrag

13. Bilder können per Fileexplorer in Beitrag eingebettet werden
14. (Wunsch) Einzelne Beiträge können per Drag ’n Drop im Projekt nach oben und unten verschoben werden
15. (Wunsch) Diashow kann pausiert werden
16. (Wunsch) Diashowgeschwindigkeit wählbar
17. (Wunsch) Projekte werden alle nacheinander in Diashow angezeigt

2.12.1 Erfüllung der Anforderungen

1. erfüllt
2. erfüllt
3. erfüllt (superuser hat direkten Zugriff auf Datenbank)
4. erfüllt (editieren nur mit direktem Zugriff aud DB)
5. erfüllt
6. nicht erfüllt (Anforderung hinfällig, da Projekte direkt bearbeitet werden können)
7. nicht erfüllt (Zeit nicht vorhanden)
8. nicht erfüllt (Zeit nicht vorhanden)
9. erfüllt
10. erfüllt
11. nicht erfüllt (eigenes Hauptbild für Projekt)
12. erfüllt
13. erfüllt
14. nicht erfüllt (Aufwand zu hoch)
15. nicht erfüllt (siehe Punkt 7)
16. nicht erfüllt (siehe Punkt 7)
17. nicht erfüllt (siehe Punkt 7)

3 Reflektion

3.1 Benefits

Der Autor konnte anhand einem praktischen Beispiel die Vorzüge eines perfekt aufgebauten Framework erleben. Die Einarbeitung in Symfony ist aufwendig aber immer nach dem gleichen Schema aufgebaut. Die einzelnen Kapitel im [Symfony/cookbook] sind einfach und simpel gehalten, dennoch schnell auf eigene Wüsche adaptierbar. Die Grösse der Communitiy ist verantwortlich für unzählige Programmbeispiele, welche zum Teil direkt implementiert werden können. Als Abschluss steht die Portierung auf einen Server an. Mit diesem Punkt ist die Arbeit an Stagebuilder als Schulprojekt offiziell abgeschlossen.

3.2 Ausblick

Grundsätzlich ist der Autor mit dem Projekt sehr zufrieden. Einzelne Punkte werden auch nach Beendigung der offiziellen Laufzeit noch angepasst und/oder optimiert. Da die Intention einer praktischen Nutzung im Vordergrund steht, ist der Autor zuversichtlich, dies auch zu erreichen. Die Benutzerfreundlichkeit und die Interaktion wird mit praktischen Übungen einzelner Probanden angepasst und weiter optimiert.

4 Installation

Um Stagebuilder lokal in einer VM nutzen zu können, sind einige Vorarbeiten nötig. Um ein Symfony Projekt selber zu erstellen, befolgen Sie nachfolgende Punkte:

1. PHP und SQLite muss installiert sein
2. Symfony Installation unter symfony.com im cookbook befolgen
3. Im gewünschten Verzeichnis ein neues Symfony Projekt anlegen: symfony new *projektname*
4. Datenbankparameter im Symfonyprojekt anpassen (mit Netbeans/Eclipse öffnen)
5. In Projektverzeichnis den integrierten Server starten: *./bin/console server:run*
6. Im Browser auf 127.0.0.1:8000 die Ausgabe prüfen

Um Stagebuilder in Ihre Projekte zu integrieren, befolgen Sie folgende Punkte:

1. PHP und SQLite muss installiert sein
2. Symfony Paket von symfony.com
3. Klonen des Projektes von Github *git clone https://github.com/sloepfe/stagebuilder.git*
4. Im Verzeichnis web/stagebuilder *composer install* ausführen
5. Parameters.yaml im Projekt anpassen auf lokale DB
6. Mit Doctrine die Entitäten *Projekt* und *Beitrag* erstellen, die Tabelle *app_user* wird automatisch mit erstellt.
7. In Projektverzeichnis den integrierten Server starten: *./bin/console server:run*
8. Im Browser auf 127.0.0.1:8000 die Ausgabe prüfen

Falls die Anwendung auf einem Server laufen soll, sind einzelne Anpassungen nötig:

- Das Framework Bootstrap wird mittels Bower integriert
- Die Benutzerrechte für den Ordner Uploads müssen für die Nutzer angepasst werden
- Die Grösse für die Bilder Uploads muss angepasst werden (default = 2MB)
- Die Datenbanken müssen remote gepflegt werden können, und das Backup muss sichergestellt werden