Web-Services 2016

Einzelbericht

Team: SparklyUnicorns
Name: Franziska Haaf

https://github.com/nelo112/UnicornMaker

1 Ideenfindung

Die Grundidee basierte auf der Serie „My little Pony“ und sollte dem Nutzer die Möglichkeit geben sein eigenes Pony zu erstellen. Oder die Hauptcharaktere der Serie zu individualisieren.
Sogenannte „Bronies“ sind ein Internetphänomen und genauer gesagt menschliche „Anhänger“ ebendieser Serie. Das „Fandom“, zu Deutsch „Fangemeinde“, entstand ursprünglich auf der bekannten Plattform „4Chan“ und hat sich mittlerweile zu einer internationalen Szene mit Conventions, Fanartikeln und einem Dokumentarfilm entwickelt.
Dementsprechend würde ein Produkt wie der „UnicornMaker“ wohl durchaus Zuspruch finden.
Im Verlauf der Planung wurde mir jedoch klar, dass es patentrechtlich gesehen wohl einfacher wäre eigene Grafiken anzufertigen.
Passend zum Namen „UnicornMaker“ entschied ich mich also dem Nutzer ein Einhorn bereit zu stellen, dass ihm dann wie geplant Gestaltungsmöglichkeiten offen lässt.
Ein „Unicorn“, also ein „Einhorn“, ist ein Fabelwesen das man bereits seit der Antike kennt.
Im modernen Internetzeitalter hat es so viel Popularität wie wohl noch nie erlangt.
Es gibt Videos, Musik, Fanartikel, Blogs, und weitere Gebiete rund um das Fabeltier.
Auch hier würde die Idee des „UnicornMakers“ also durchaus Anklang finden.

Abbildung 1 Die Serie "My little Pony"

2 Use Cases

In unserer ersten Präsentation, entschied ich mich für die Use-Cases und erstellte die Grafik für die Use-Cases des Nutzers.
Diese Use-Cases umfassten „Erstellen“ und „Teilen“. Welches auch unsere beiden wichtigsten Funktionalitäten umfasst. Für die nächste Präsentation hatten wir uns dazu entschieden, die Use-Cases etwas detaillierter darzustellen. Darum erstellte ich die neue Grafik mit den **Use-Cases** „*Unicorn anlegen*“, „*Unicorn speichern*“, „*Unicorn teilen*“ und „*Unicorn aufrufen*“.
Da wir uns des Weiteren dafür entschieden hatten, dass eine Admin Funktionalität zwecks Einstellungen der Anwendung im späteren Verlauf unumgänglich ist, erstellte ich zusätzlich die Grafik

mit den **Admin Use-Cases** „*Optionen*[1] *anlegen*“, „*Optionen ändern*“ und „*Optionen löschen*“.

Die den Benutzer betreffenden Use-Cases basierten dabei auf den von mir erstellten **User-Stories**

1) Als Nutzer möchte ich individuelle Einhörer erstellen
2) Als Nutzer möchte ich mein Einhorn mit meinen Freunden teilen
3) Als Nutzer möchte ich mein Einhorn erneut aufrufen können
4) Als Anbieter möchte ich, dass meine Social-Media Auftritte Beachtung bekommen

Wobei die 4. User-Story in den Use-Case „Unicorn teilen“ einfließt.

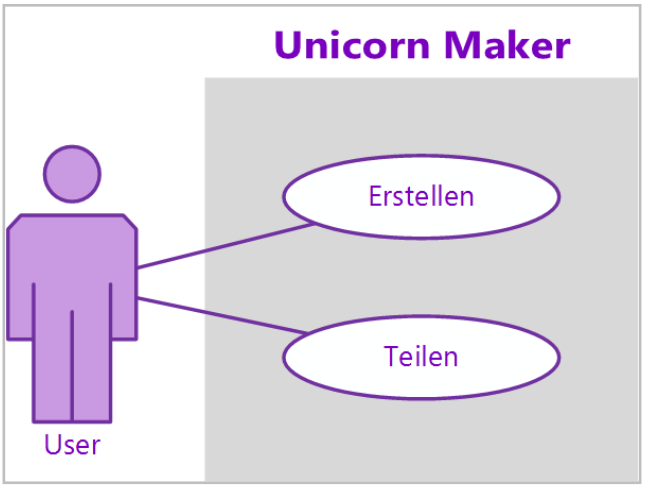

Abbildung 2 Use Cases

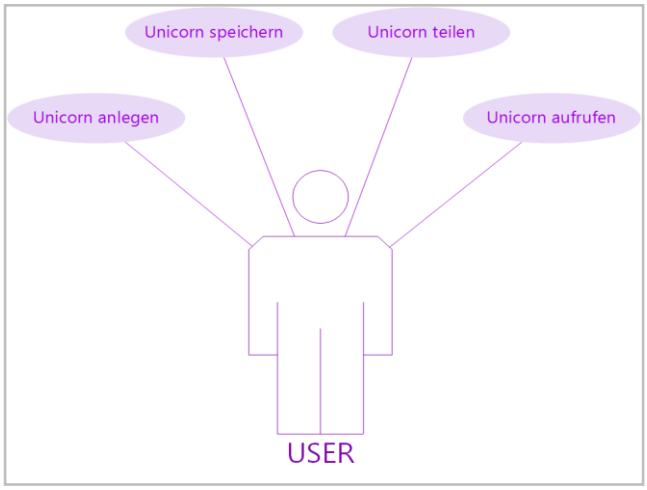

Abbildung 3 Use Cases – detaillierter

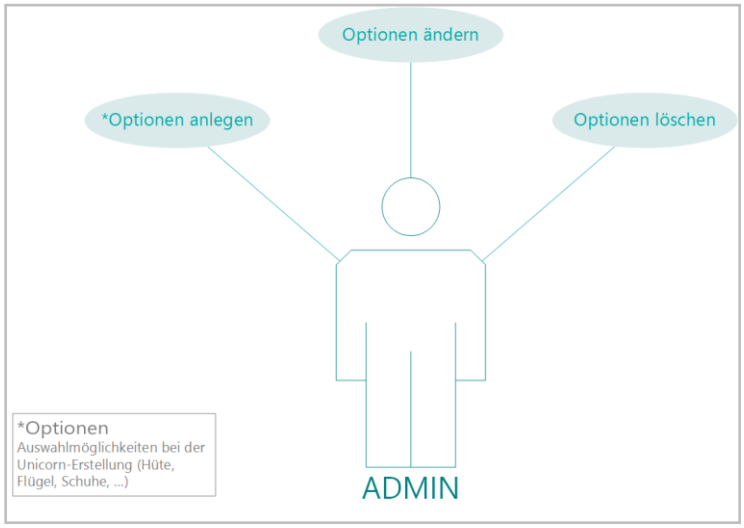

Abbildung 4 Admin Use Cases

Von mir im Code umgesetzt wurde der Use-Case „Teilen“ (s. Abbildung 2) sowie der „Welcome-Screen“ (Bezug auf User-Stories 2) und 4)).

[1] Optionen umfasst die verschiedenen Möglichkeiten ein Einhorn zu individualisieren. Dies umfasst aktuell Farbe, Accessoires und die Möglichkeit das Einhorn ohne oder mit Flügeln auszustatten.

3 Usability und User-Interface

3.1 Foliendesign und Erstellung

Die Folien und das damit hergehende Design wurden von mir entworfen und erstellt.
Ich entschied mich dabei für ein junges und witziges Gesamtkonzept, dass zu der Idee des „Unicorn-Makers“ und dem Markenkonzeptes passt.
Ich wollte zudem minimalistische und klare Folien präsentieren, welche die jeweilige Information deutlich vermitteln.

3.2 Mockups

Die von mir angefertigten Mockups sind die Vorgaben für die programmatische Umsetzung.
Und legen zeitgleich das Design der Seite fest.

Als die Idee unseres Projektes fest stand, fertigte ich die Mockups für die jeweiligen Seiten an.
Eine „Welcome-Page“ als Einstiegsseite, soll den Benutzer willkommen heißen und ihn neugierig machen weiter auf der Seite zu verweilen.

Bei der Gestaltung der „Unicorn-Erstell-Page“, unserer Hauptfunktionalität und somit der wichtigsten Seite, orientierte ich mich an bekannten Charakter-Editoren.
Der Grundaufbau solcher Charakter-Editoren (z.B. World of Warcraft oder Skyrim) folgt immer demselben Schema, und ist somit bekannt. Um dem Nutzer eine bestmögliche User-Experience zu ermöglichen hielt ich mich an dieses bekannte Schema.

Die „Unicorn teilen“ Seite, sollte dem Nutzer einen Link zur Verfügung stellen, der ihm beim Aufruf die Möglichkeit gibt sein von ihm erstelltes Einhorn zu präsentieren.
Durch die Social-Media-Buttons auf der Seite, bekommt der Nutzer zudem die Möglichkeit direkt einen Post auf der jeweils gewählten Seite mit vorgefertigtem Text und Link zum Einhorn zu teilen.
Mit dieser Social-Media-Funktion wird zeitgleich die User-Story 4) erfüllt.

Abbildung 5 Welcome Page

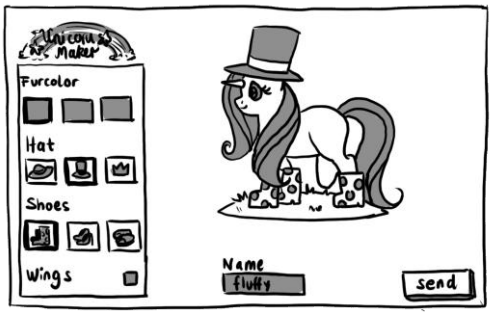

Abbildung 6 Unicorn Erstellung

Abbildung 7 Unicorn Teilen

3.3 Grafiken

Sämtliche Grafiken (Logo, Einhörner, Hüte, Schuhe, Flügel, Roadmap, etc.) wurden von mir entworfen und umgesetzt. Ich wollte hierbei ein niedliches, simples und „comichaftes „Design verfolgen. Dadurch liegen die Rechte an den Bildern bei uns, und können ohne Probleme öffentlich genutzt werden.

4 Implementierung der App

Wir entschieden uns dafür die Backend-Seite mit C# umzusetzen, und das Frontend mit Angular2. Angular2 ist eine einfache, und schnell umsetzbare Plattform die uns genau das bietet was wir für unsere Anwendung wollten.

„Unicorn-Server“ : C#
Implementierung der

- „HatsController.cs“,
- „ShoesController.cs“,
- „UnicornController.cs“
- „WingsController.cs“.

Diese dienen als REST-Endpunkte für die Services der Angular2-Anwendung.
Sie sind somit Ansprechpartner für das jeweils Vorgesehene. (Bilder für Hüte/Schuhe/… laden und im zukünftigen Release Neue hinzufügen)

Zudem Implementierung der dazugehörigen „Connector“-Klassen

- „HatsConnector.cs“,
- „ShoesConnector.cs“,
- „UnicornConnector.cs“
- „WingsConnector.cs“.

Die Connector-Klassen stellen die Verbindung zur Datenbank her, und werden den Controllern übergeben und von Ihnen für den Zugriff verwendet.

Schlussendlich die Implementierung der Methoden „DbInitializer.AddShoes()“ und „DbInitializer.AddHats()“, wodurch die Bilder in den Datenbank-Kontext geladen werden. Somit sind sie für die Connector-Klassen zugänglich.

„Unicorn-App“ : Angular2

Umsetzung des „Welcome-Screens“ sowie des „Save-Screens“. Implementierung der jeweiligen Html, Less und Typescript Dateien, und wo nötig Einpflegung von CSS Klassen in die Haupt-Less-Datei (z.B. Buttondesign).