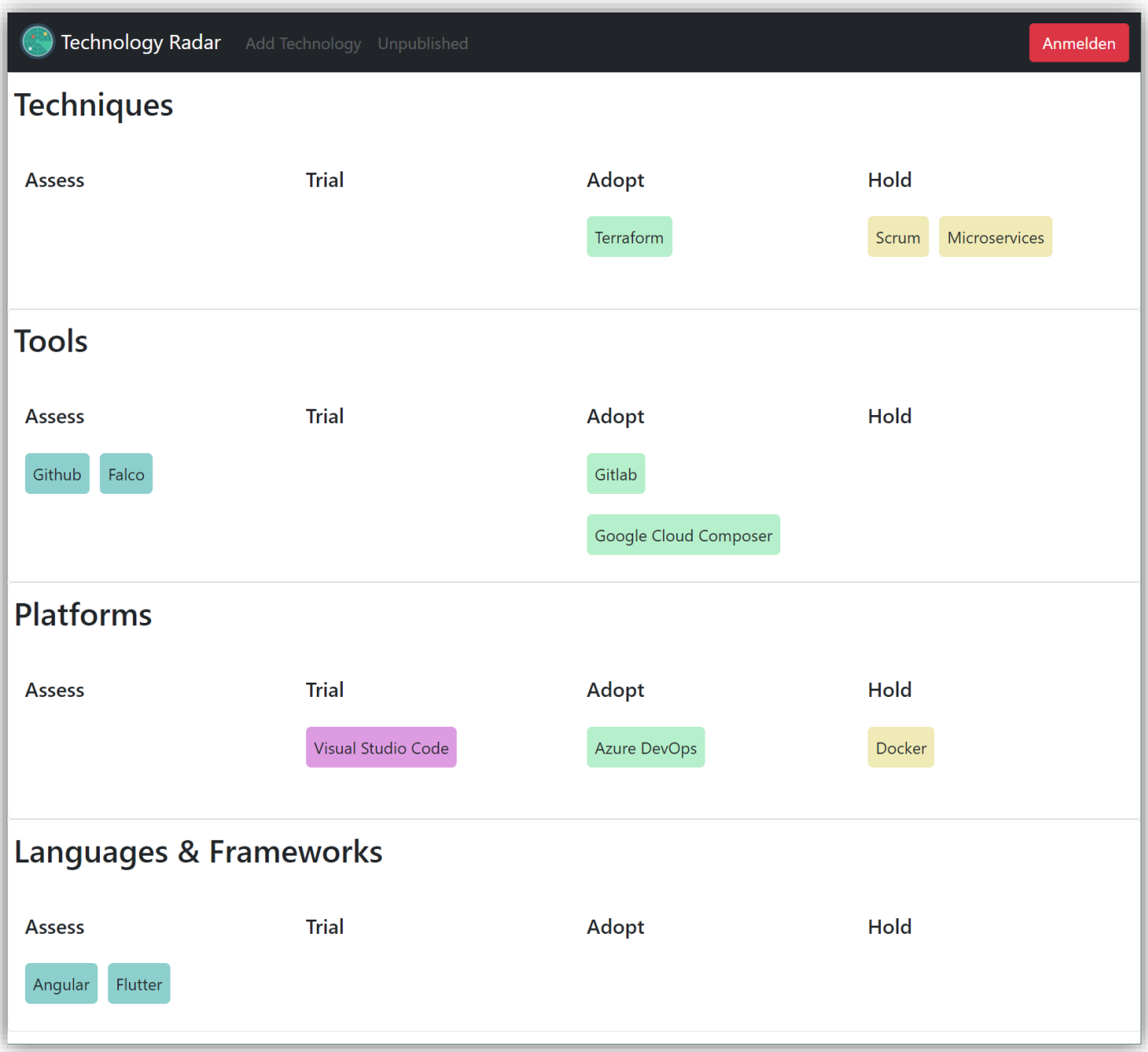
Dokumentation Technologieradar

Web Programming Lab (Blockwoche) HS2021

Dave B, 8. März 2022



Inhalt

Einleitung

Während der Web Programming Lab Blockwoche HS2021 wurde den Teilnehmenden vertieftes Wissen in Themen wie Architekturansätze von Web Anwendungen, JavaScript, Angular, Authentification und weiteres vermittelt. Als Abschluss wird ein eigenes Webprojekt mit Angular und Node.js umgesetzt.

Im folgenden Kapitel "Architekturdokumentation", wird die Architektur des Webprojekts dokumentiert, danach folgt das "Fazit & Reflexion" und Anhänge wie das Arbeitsjournal.

Architekturdokumentation

Einführung und Ziele

Die Aufgabenstellung ist, eine Softwarelösung zu entwickeln, die ein intuitives Betrachten und Verwalten von Technologien, Tools, Plattform und Sprachen & Frameworks ermöglicht, welche in einem Unternehmen eingesetzt werden. Die volle Anforderungsliste befindet sich hinter diesem Link.

Mitarbeiter und Kunden können anhand der Technologieradars interessante Informationen über die Art und Weise erfahren wie das Unternehmen arbeitet.

User Story	Prio
Anmelden Technologie-Radar-Administration	Could
Technologie publizieren	Should
Technologie ändern	Should
Technologie-Einordnung ändern	Should
Anmelden am Technologie-Radar-Viewer	Could
Technologien anzeigen	Must

Qualitätsziele

Die Qualitätsziele alias die nicht funktionale Anforderungen umfassen folgendes:

Qualitätsziele
Der Technologie-Radar-Viewer soll neben der Desktop-Ansicht, auch für die Mobile-Ansicht optimiert sein.
Der Technologie-Radar-Viewer soll innert 1s geladen sein.
Sämtliche Änderungen an Technologie-Einträgen sollen historisiert sein.
Sämtliche Anmeldungen an die Technologie-Radar-Administration werden aufgezeichnet.

Randbedingungen

Die Rahmenbedingungen sind eher locker und ermöglichen einen grossen Spielraum in der Umsetzung. Die Softwarelösung soll allgemeine UI und UX -Guidelines befolgen.

Lösungsstrategie

Aufgrund der genannten Qualitätsziele werden folgende Lösungsansätze verfolgt.

Qualitätsziele	Lösungsansätze
Der Technologie-Radar-Viewer soll neben der Desktop-Ansicht, auch für die Mobile-Ansicht optimiert sein.	• Single Page Architektur • Frontend mit Angular und Bootstrap • Responsive Entwicklung
Der Technologie-Radar-Viewer soll innert 1s geladen sein.	• Single Page Architektur • Einsatz performanter Infrastruktur: Apache2 & MongoDB
Sämtliche Änderungen an Technologie-Einträgen sollen historisiert sein.	• Einsatz von MongoDB • Einsatz von JavaScript mit node.js • Änderungen werden serverseitig geloggt
Sämtliche Anmeldungen an die Technologie-Radar-Administration werden aufgezeichnet.	• Einsatz von MongoDB • Anmeldungen werden serverseitig geloggt

Bausteinsicht

Die Entwicklung der Software wurde in zwei unabhängige Teile zerlegt. Das eine das Front-End (Angular) und das Backend (node.js).

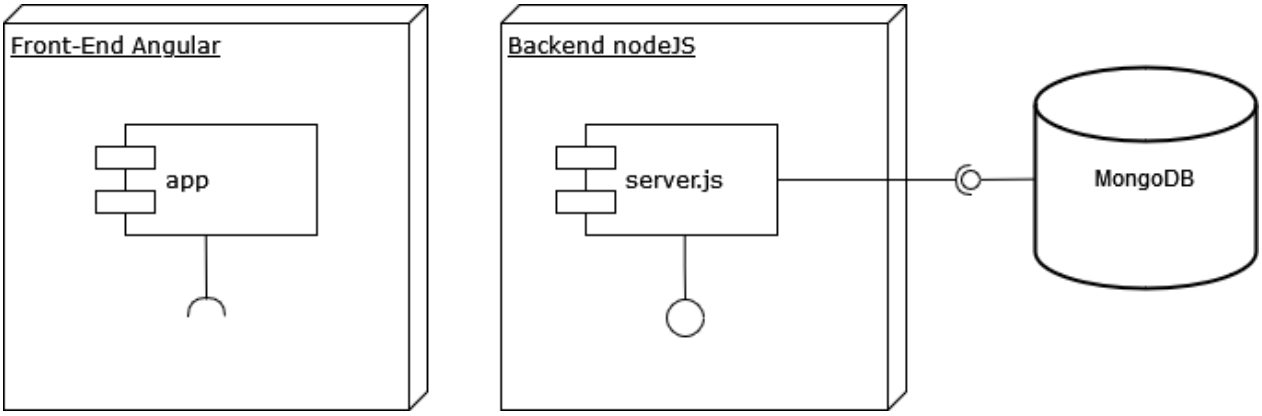


Abbildung 1 - Visualisierung der Modulsicht

Front-End

Das Front-End ist ein Angular Projekt, welches in einem Modul aufgebaut ist.

Modul	app (nur 1 Modul)
Komponenten	Components: • app.component.ts • unpublished.component.ts • technology-Panel.component.ts • login.component.ts • header.component.ts • edit-technology.component.ts • detail-view.component.ts • add-technology.component.ts Services: • tech-data.service.ts • auth.service.ts • auth.guard.ts • token-interceptor.service.ts
Dependencies	• angular 13.2.0v • bootstrap 5.1.3v • rxjs 7.5.0v • typescript 4.5.2v • karma 6.3.0v

Back-End

Das Back-End besteht aus einem JavaScript Programm, welches verschiedene API anbietet. Der Service ist via IP und den vorkonfigurierten Port 4566 erreichbar. Das Programm ist mit einem lokalen MongoDB verbunden, in der die Daten gehalten und gepflegt werden.

Modul	Technology-radar-backend
Komponenten	• server.js
Dependencies	• nodemon 2.0.15v • mongodb 4.3.1v • express 4.17.2v • dotenv 16.0.0v • jsonwebtoken 8.5.1v

API im Backend

Aufruf	Beschreibung
"/getByName/:name"	GET-Schnittstelle, welche mit dem key (:name) das zugehörige JSON zur Technologie zurück gibt.
"getAllPublishedByCategory/:category"	GET-Schnittstelle, welche mit dem key (:category) alle zur Category gehörenden Technologien zurückgibt, welche beim Property "published" den Wert auf true gesetzt haben.
"/addNewTechnology"	POST-Schnittstelle, bei welcher neue Technologien erfasst werden können. Um diese Schnittstelle zu verwenden, muss sich der Client mit einem JSON-Webtoken erfolgreich authentifizieren.
"/editTechnology"	POST-Schnittstelle, bei welcher eine neue Version von Technologien erfasst werden können. Um diese Schnittstelle zu verwenden muss sich der Client mit einem JSON-Webtoken erfolgreich authentifizieren.
"/login"	POST-Schnittstelle, bei welcher sich ein User anmelden kann. Beim Aufruf dieser Schnittstellen werden im Body ein username und password erwartet. Wird der User erfolgreich authentifiziert, so erhält er als Response ein JWT.
"/getAllUnpublished"	GET-Schnitstelle, die alle Technologien zurückgibt, welche beim Property "published" false gesetzt haben. Um diese Schnittstelle zu

	verwenden, muss sich der Client mit einem JSON-Webtoken erfolgreich authentifizieren.

MongoDB

Auf der MongoDB in der 5.0.6 Community Version werden die Technologie-Documents gehalten, sowie die CTO-User Credentials und Login-Logs.

Aufbau

DB: techradar

Collections: loginLogs, technologies, users

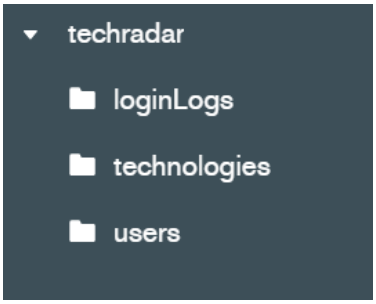


Abbildung 2 - Visualisierung der Datenbankstruktur

Schema eines Technologieeintrags

```
{
    "name": {
        "type": "string"
    },
    "category": {
        "type": "string"
    },
    "id": {
        "type": "string",
        "format": "integer"
    },
    "status": {
        "type": "string"
    },
    "author": {
        "$ref": "string"
    },
    "description": {
        "type": "string"
    },
    "published": {
        "type": "boolean"
    },
    "created": {
        "type": "string"
    },
    "history": {
        "type": "array"
    }
}
```

Abbildung 3 - Abbildung des Schema eines Technology-document

Laufzeitsicht

Inbetriebnahme der Applikation

Das folgende Flussdiagramm beschreibt die initiale Inbetriebnahme des Systems.

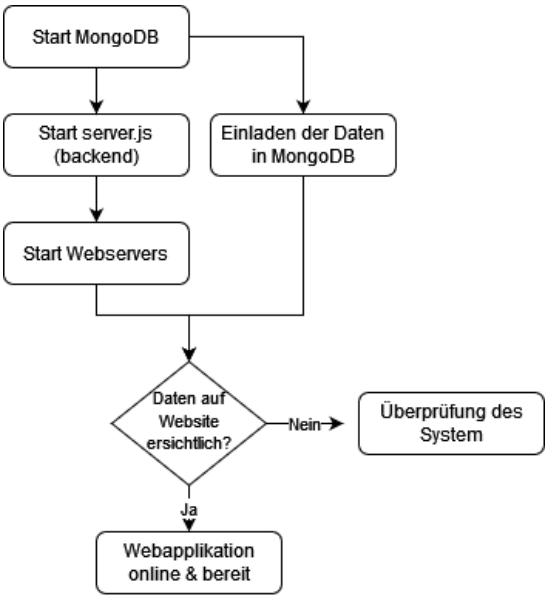


Abbildung 4 - Visualisierung des Inbetriebnahmeprozesses

Benutzung der Applikation

Das folgende Diagramm beschreibt, die Interaktionsmöglichkeiten eines Users mit der Single Page Applikation.

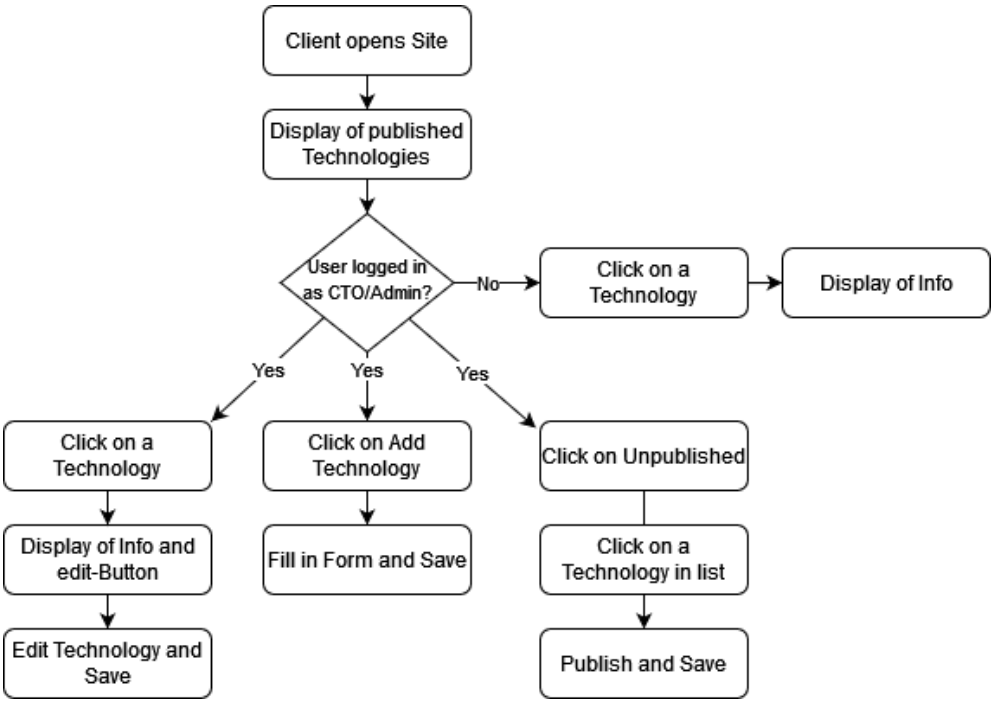


Abbildung 5 - Visualisierung der Benutzungsmöglichkeiten

Sequenzdiagramm

Im folgenden Diagramm wird gezeigt, wie welche Komponenten miteinander kommunizieren.

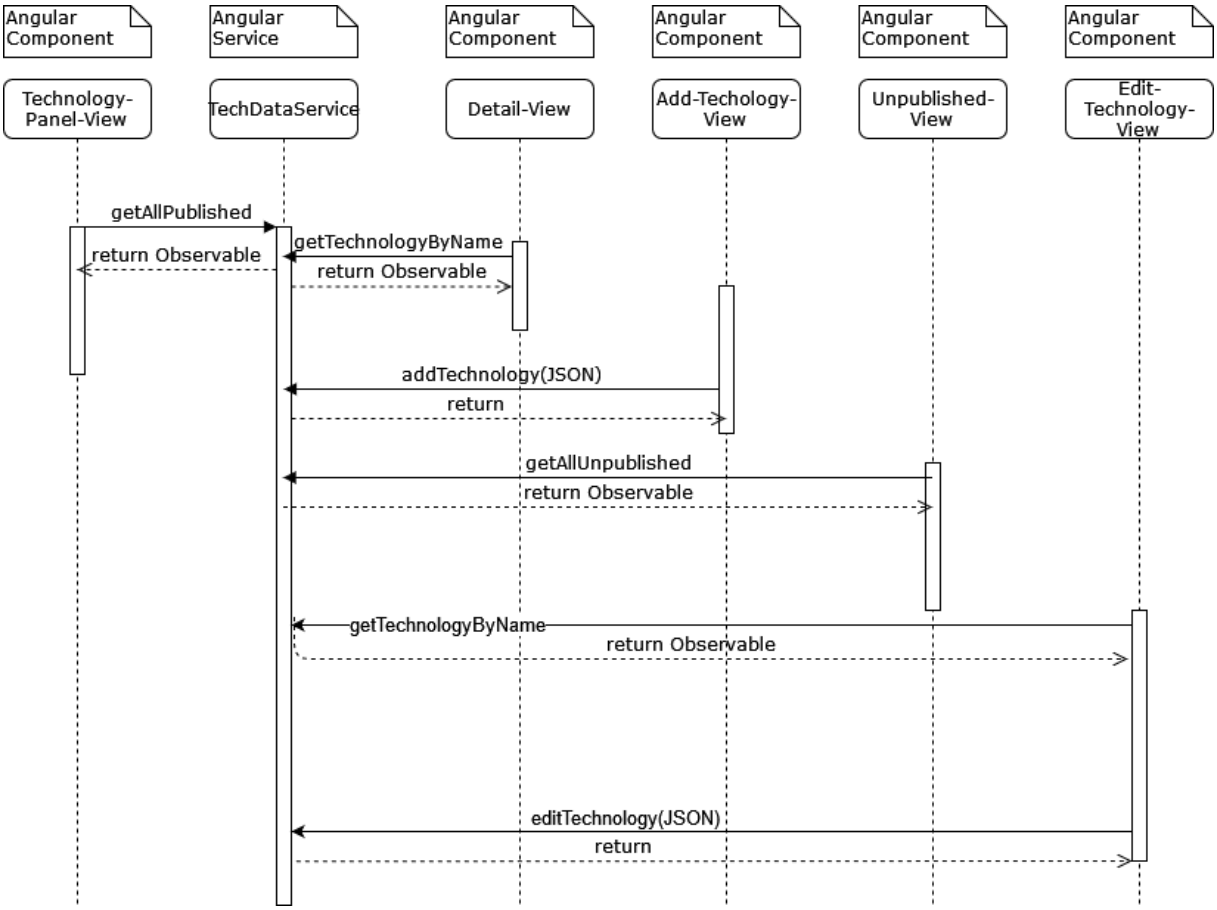
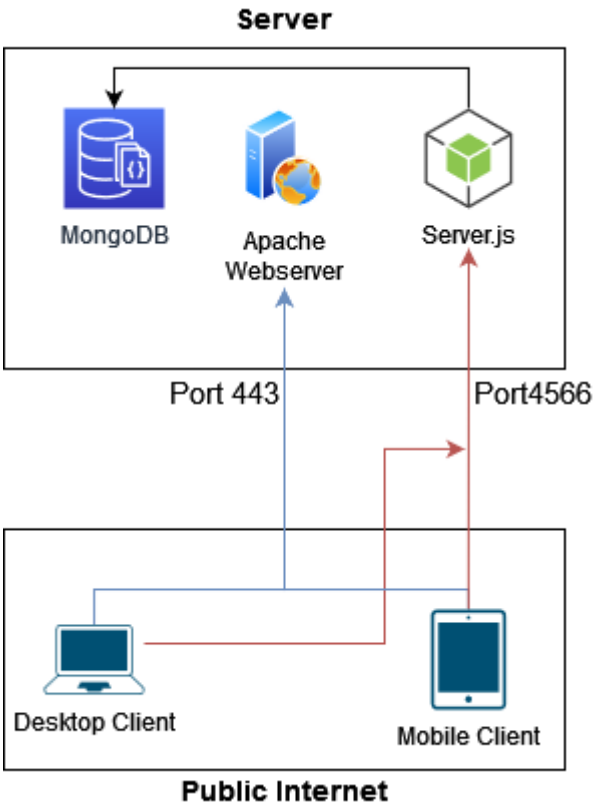


Abbildung 6 - Visualisierung der Interkomponenten-Kommunikation

Verteilungssicht

Die MongoDB der Apache Webserver und das Backend-Programm (Server.js) laufen alle auf derselben VM auf einem Server. Die Clients werden dabei über Port 443 die Website aufrufen können und via 4566 die Aufrufe auf die API machen.



Querschnittliche Konzepte

Clean Code:

Bei der Entwicklung wurde auf "Single Responsibility Principle" (SRP) geachtet. So gibt es diverse Services in der Applikation, welche jede aber nur einen klar bestimmten Funktionsumfang hat.

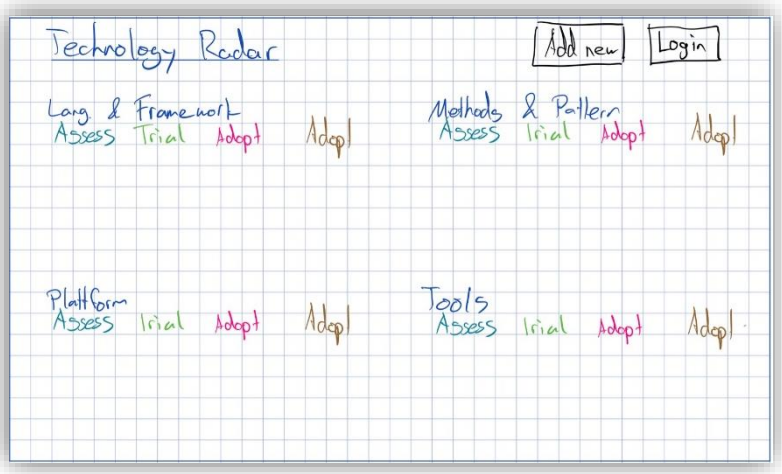
Auch das "Dont repeat yourself" Prinzip (DRY) wurde angewendet. So wird für das Editieren und das Publizieren der Technologien dieselbe View verwendet.

Auf Kommentare im Code wurde fast vollumfänglich verzichtet.

Fazit und Reflexion

Der Start des Weblab-Projekts ist mir sehr gut gelungen. Ich begann nicht sofort darauf loszuprogrammieren, sondern ich habe mir zuerst alle User Stories mehrmals genau durchgelesen und mir ein eigenes Kanban Board auf GitHub erstellt.

Dazu habe ich ein Mockup der Startseite gezeichnet, ein Schema definiert, wie die Technologie-Einträge als JSON Dokumente gespeichert werden sollen. Auch habe ich mir alle API-Methoden aufgeschrieben, die ich im Backend zur Verfügung stellen und im Frontend verwenden werde.



Diese Herangehensweise hat es mir erlaubt anschliessend effizient am Code zu schreiben, da ich mich mehr oder weniger voll auf die Implementierung und nicht um die Art und Weise fokussieren konnte.

Zu den jeweiligen Angular-Components habe ich meist jeweils eigenen scss-Code hinzugefügt. Das würde ich nächstes Mal anders machen. Damit ich nicht, vieles ähnliches in vielen Files mehrmals definiert habe.

Was mir am meisten Schwierigkeiten bereitet hatte, war das Handling der Authentifizierung und des Logins. Mit einem guten YouTube-Tutorial, welches alle nötigen Schritte beim Einsatz von JWT in Angular hervorragend erklärte, konnte ich die Rechteverwaltung für die jeweiligen Seiten und API-Requests aber anschliessend gut lösen.

Bei einem zukünftigen Web-Projekt würde ich mich vor Beginn mehr mit Bootstrap beschäftigen, damit ich mehr von den vielen Funktionalitäten von Bootstrap verwenden kann. In meinem Projekt habe ich einige Elemente von Bootstrap verwendet, aber es gäbe noch viele weitere.

Der Workflow war im Allgemeinen sehr effizient und ich konnte gut und fokussiert am Projekt arbeiten, da es mir auch sehr viel Spass gemacht hat. Zum Abschluss habe ich die Webapplikation und das Backend auf meinem Raspberry Pi deployed. Nun ist es unter diesem Link öffentlich aufrufbar.

Arbeitsjournal

Datum	Arbeit	Kategorie	Zeit (h)
14.02.2022	Initialisierung der Architekturdokumentation	Doc	2.0
14.02.2022	Erstellung Mockup & Planung	Doc	3.0
14.02.2022	Initialisierung des Projektcodes & Schreiben	Programmieren	6.0
15.02.2022	div. Components erstellt	Programmieren	6.0
16.02.2022	Backend programmieren	Programmieren	6.0
17.02.2022	Formular & Validierung	Programmieren	6.0
18.02.2022	Login und Authentifizierung	Programmieren	6.0
18.02.2022	Navigation und Guard	Programmieren	5.0
21.02.2022	Refactoring	Programmieren	4.0
22.02.2022	Architekturdokumentation	Doc	3.0
23.02.2022	Architekturdokumentation	Doc	3.0
24.02.2022	Fazit & Reflexion	Doc	1.5
24.02.2022	Präsentation	Doc	1.5
07.03.2022	Überarbeitung Doku	Doc	1.6
08.03.2022	Review und Abschluss der Doku & Präsentation	Doc	2.0
Total			**56.6**

Softwareartefakte

Die Softwareartefakte sind auf GitHub und können unter dem folgenden Link aufgerufen werden.

Front-End: https://github.com/dave1b/technology-radar

Back-End: https://github.com/dave1b/technology-radar-backend