

HOCHSCHULE OSNABRÜCK
UNIVERSITY OF APPLIED SCIENCES

Hochschule Osnabrück
University of Applied Sciences

Fakultät
Ingenieurwissenschaften und Informatik

Schriftliche Projektarbeit zum Thema:

**Entwicklung einer PC-Konfigurator-App mit externer Datenintegration
aus der 'PcPartApi' (SWA) unter Verwendung von Capacitor**

im Rahmen des Moduls
,Webanwendungen',
des Studiengangs Informatik-Medieninformatik
im Sommersemester 2023

Autoren: Daniel Graf
Matr.-Nr.: 914743
E-Mail: daniel.graf@hs-osnabrueck.de
Dozent: Dipl.-Inf. (FH) Björn Plutka

Abgabedatum: 28.09.2023

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Ausgangssituation	1
1.3	Ziele 1	
2	Stand des Wissens.....	2
2.1	Vergleichbare Ansätze	2
2.2	Eingesetzte Technologien	2
3	Anforderungen	4
3.1	Funktionale Anforderungen	4
3.1.1	Anwendungsfälle	4
3.1.2	Minimal angestrebte Systemfunktionen	4
3.2	Nichtfunktionale Anforderungen.....	5
3.2.1	Zielgruppen	5
4	Konzept.....	6
4.1	Gesamtkonzept	6
4.2	Kommunikation mit API	6
5	Implementierung.....	7
5.1	Funktionalitäten	7
5.1.1	Navigation	7
5.1.2	Login 7	
5.1.3	Dashboard.....	9
5.1.4	Konfiguration Erstellen	10
5.1.5	Öffentliche Konfigurationen	11
5.2	Technische Besonderheiten	11
5.2.1	API Service	11
5.2.2	Reactive Forms	11
5.3	Installation & Inbetriebnahme	12
6	Fazit	13
	Erklärung	15

1 Einleitung

Die vorliegende Projektarbeit entstand im Zuge des Bachelor-Studiums im fünften Semester im Modul ,Webanwendungen‘ bei Dipl.-Inf. (FH) Björn Plutka an der Hochschule Osnabrück. Diese berichtet über die Entwicklung einer *Angular- & Ionic*-App (gennant ,*PcPartPicker*‘) unter Verwendung von *Capacitor*, welche eine Anwendung zur PC-Konfiguration bereitstellen soll. Zusätzlich wird eine konsistente Datenhaltung mittels einer im Modul Software Architektur entwickelten API bereitgestellt (genannt ‘*PcPartAPI*’).

1.1 Motivation

Die Idee ist es, eine Webanwendung zu entwickeln, welche mithilfe der Datenspeicherung aus der *PcPartAPI* es dem Nutzer ermöglicht, Computerteile- und Komponenten in einer Datenbank zu speichern und diese für andere Nutzer sichtbar zu machen. Diese Komponenten kann man dann innerhalb einer individuellen PC-Konfiguration zusammenfügen und speichern. Motiviert wird diese Idee durch die große Anzahl an Optionen von Computerteilen, welche man in seinen Computer verbauen kann. Diese Anwendung zielt deshalb darauf hinaus, eine einheitliche und zentrale Plattform für das Zusammenstellen und Veröffentlichen von Pc-Konfigurationen zu ermöglichen. Komponenten als auch Konfigurationen können so eingesehen und verglichen werden sowie kommentiert und bewertet werden. Dies soll dem Nutzer beim Entscheidungsprozess, welche Komponenten in den Computer verbaut werden, unterstützen

1.2 Ausgangssituation

Im Rahmen des fünften Semesters an der Hochschule Osnabrück, wird nebenher das Modul ‘Software Architektur’ bearbeitet, wo eine entsprechende API für diese Webanwendung entwickelt wird. Diese API ist lediglich Entwicklerzentriert und bietet keinerlei graphische Oberfläche für eine nutzerfreundliche Anzeige der Daten. Diese bildet hier das *Backend* und wirkt somit als Grundlage für die Webanwendung.

1.3 Ziele

Das Ziel der Projektarbeit ist die Entwicklung einer nativen App mithilfe von *Capacitor* auf Basis einer *Angular- & Ionic* Anwendung. Es gilt nun, die in der *PcPartApi* bereitgestellten Funktionalitäten sinnvoll mit einer Anwendungsoberfläche abzubilden. Die hier zu entwickelnde *Angular/Ionic* App soll diese Aufgabe übernehmen und letztendlich eine nutzerfreundliche Interaktion mit der API ermöglichen.

2 Stand des Wissens

Im Folgenden wird nun der Entwicklungskontext der PC-Konfigurator App näher beschrieben.

2.1 Vergleichbare Ansätze

Anwendungen bezüglich eines PC-Konfigurators gibt es schon teilweise öffentlich nutzbar im Netz. Einer der größten solchen Implementierungen heißt zum Beispiel ‘*pcpartpicker.com*’. Hier hat man als Nutzer die Möglichkeit, sich durch einen ausgiebigen Katalog von Computerkomponenten zu navigieren. Diese Komponenten haben zusätzlich umfassende Informationen bezüglich der Art und Spezifikation. Auch hat man hier die Möglichkeit diese Komponenten zu Bewerten und Kommentare zu verfassen. Ein weiteres Feature, ist das Hochladen von fertig gebauten Computer-Projekten. Diese Computer (im weiteren Verlauf genannt: ‘*Konfigurationen*’) können zusammen mit der Komponentenliste sowie Bildern hochgeladen werden, so dass andere Nutzer diese anschauen und kommentieren können. Eine beispielhafte Konfiguration sieht man in Abbildung 1.

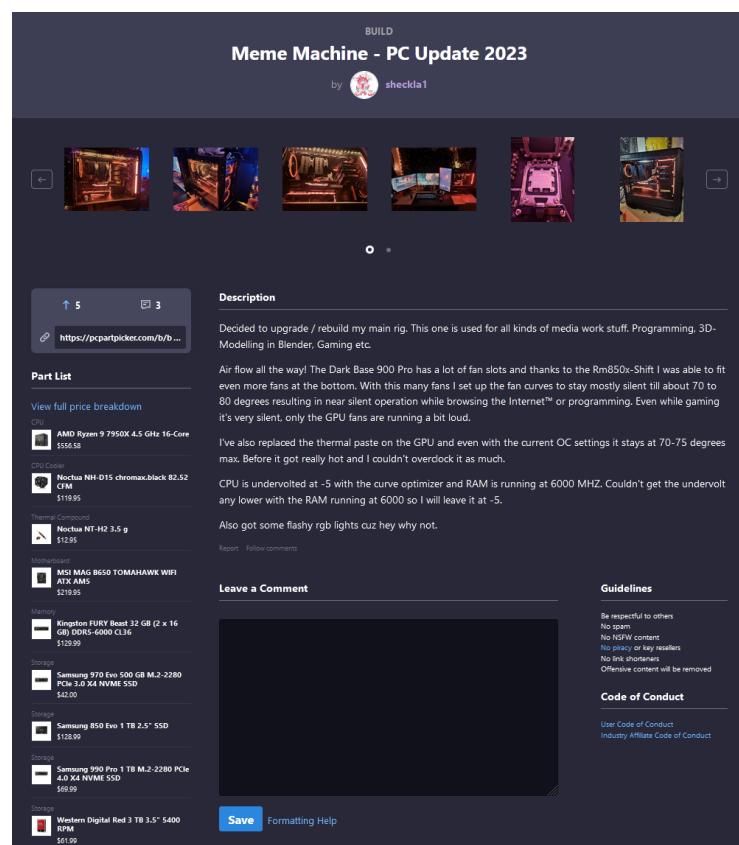


Abbildung 1: Öffentliche PC-Konfiguration (<https://pcpartpicker.com/b/bH7XsY>)

2.2 Eingesetzte Technologien

Um eine solche Konfigurator-App zu ermöglichen ist eine grundlegender Backend & Frontend Ansatz vorteilhaft. Das Backend kümmert sich um die Datenspeicherung- und Aufbereitung

und bietet angemessene Schnittstellen an, so dass diese Daten angefragt, abgespeichert, verändert und gelöscht werden können.

Das Frontend hat im Gegensatz dazu die Aufgabe, diese Daten sinngemäß und nutzerzentriert darzustellen. Das heißt, dass die rohen Datensätze aus der API ansprechend für den Nutzer der App angezeigt werden. Ein solcher Ansatz wird somit auch in diesem Projekt genutzt. Hierbei ist es wichtig, auf die Konnektivität bezüglich *Backend* und *Frontend* zu achten.

3 Anforderungen

Im Folgenden Kapitel werden die Nutzungsszenarien des Projektes als Use-Cases im Kontext der funktionalen- & nichtfunktionalen Anforderungen beschrieben.

3.1 Funktionale Anforderungen

Zu Beginn werden die essenziellen Anforderungen zum Betreiben des angestrebten Systems mithilfe von Use-Cases erläutert, welche die Grundlage für die davon abgeleiteten Systemfunktionen sind. Diese Anforderungen stellen das technische Fundament für die Anwendung dar und sind in ihrer Implementierung kritisch und gelten daher als Muss-Kriterium.

3.1.1 Anwendungsfälle

Um die Sinnhaftigkeit und Nutzbarkeit des Systems im Kontext der vorherigen Kapitel zu verdeutlichen, werden nun beispielhafte Anwendungsfälle als Use-Cases beschrieben:

- **Anzeige & Hinzufügen von Komponenten:** Ein Nutzer, der über Computerteile informiert werden möchte, kann die Listenansicht der App öffnen, um alle aktuell eingetragenen Komponenten zu sehen. Diese beinhalten Detailinformationen bezüglich der Spezifikation.
- **Anzeige & Erstellen einer Konfiguration:** Ein Nutzer, der sich die PC-Konfigurationen von anderen Nutzern anschauen möchte, um diese mit seiner eigenen zu vergleichen, kann sich die öffentlich eingetragenen Konfigurationen anzeigen lassen. Des Weiteren besteht die Möglichkeit, selbst eine Konfiguration mit den zur Verfügung stehenden Komponenten zu erstellen.
- **Verfassen von Kommentaren & Bewertungen:** Ein Nutzer, welcher eine positive oder negative Erfahrung mit der Handhabung einer Computerkomponente hat, kann diese aus dem Komponentenkatalog auswählen und eine entsprechende Bewertung sowie Kommentar verfassen.

3.1.2 Minimal angestrebte Systemfunktionen

Aus dem vorherigen Use-Case lassen sich nun folgende grundlegende Systemfunktionen ableiten.

- **Komponentenliste:** Die App beinhaltet eine Ansicht, welche alle verwalteten Komponenten anzeigt.

- **Komponenten Hinzufügen:** Die App bietet die Funktion, über Formulare neue PC-Komponenten einzureichen und dieser der Datenbank hinzuzufügen.
- **Konfigurationen ansehen:** Die App bietet die Möglichkeit, alle öffentlich hochgeladenen PC-Konfigurationen anzeigen zu lassen.
- **Konfiguration zusammenstellen:** Es wird ein System bereitgestellt, welche es dem Nutzer ermöglicht anhand der verfügbaren Komponenten eine individuelle Konfiguration zu erstellen.
- **Anmeldesystem:** Die App bietet die Möglichkeit, sich als bestehender Nutzer anzumelden oder sich als neuer Nutzer zu registrieren.
- **Bewerten & Kommentieren:** Es wird für Komponenten und Konfigurationen eine Funktion zur Bewertung sowie Kommentierung bereitgestellt.

3.2 Nichtfunktionale Anforderungen

In Folge der funktionalen Anforderungen werden nun zusätzlich die nichtfunktionalen Anforderungen betrachtet.

- Aufgrund der potenziell hohen Anzahl an Komponenten, sind diese graphisch sortiert und geordnet anzuzeigen. Dies soll die Komponentenliste überschaubarer gestalten.
- Bestimmte Funktionen sollen nur für bestimmte Nutzergruppen verfügbar sein. So können Administratoren neue Komponenten einreichen sowie Bilder hochladen, Moderatoren können Kommentare Löschen und angemeldete Nutzer können Konfigurationen erstellen. Dies soll im Betrieb, die Qualität der Daten erhöhen und unangemessene Kommentare sowie Komponenten mit Fehlinformationen vermeiden.

3.2.1 Zielgruppen

Anhand der technischen und Detailinformation basierten Natur dieser Anwendung, ist sie an Personen gerichtet, welche sich mit Computerhardware und Komponenten befassen. Dies können Hobby-Enthusiasten oder auch professionelle Hardwareshops sein, welche eine zentrale sowie einheitliche Sammlung von kuratierten Daten bezüglich verschiedenster Komponenten suchen. Gerade die öffentlich ansehbaren PC-Konfigurationen können als Schaustück für Enthusiasten dienen, so dass sie Feedback und Kritik zu ihrer Konfiguration und Komponentenauswahl erhalten können.

4 Konzept

Nachdem nun in den vorherigen Kapiteln die Technologien und Nutzungsszenarien vorgestellt wurden, wird nun näher auf das Gesamtkonzept des Systems eingegangen.

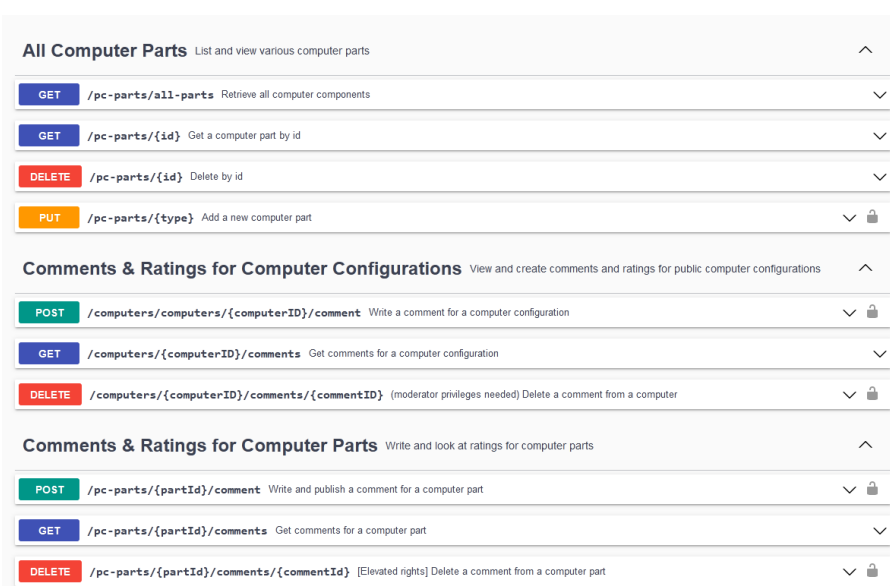
4.1 Gesamtkonzept

Da das System als ganzes eine *Frontend* sowie *Backend* Anwendung beinhaltet ist das Abstimmen dieser beiden Programme von großer Bedeutung. Die PcPartAPI ist ein Java Programm mit dem *Quarkus*-Framework und stellt mithilfe von *Docker*-Containern die Datenbanken zur Verfügung. Schnittstelle ist dabei eine REST-API, welche über entsprechende Pfade die Daten zur Abfrage und Manipulation bereitstellt.

Die *Angular*- & *Ionic* App wird als HTML-Webanwendung implementiert und setzt mittels Angular das *Model-View-Controller Pattern* um. Die HTML-Dateien spiegeln die *View* wider, welche anhand von *TypeScript-Controllern* manipuliert wird.

4.2 Kommunikation mit API

Ein zentraler Aspekt der App ist die Kommunikation mit dem *Backend*. Aufgrund der Komplexität und Vielfalt der Daten und verfügbaren Pfade ist hier auf eine saubere und geordnete Abhandlung der Kommunikationsprozesse innerhalb der App zu achten. Das heißt, es gilt die Anzahl der Anfragen auf die API angemessen zu reduzieren. So sollten entsprechende Aktualisierungsintervalle für die Datenabfrage gewählt werden. Auch eine entsprechende Fehlerbehandlung der Datenabfrage soll in Betracht gezogen werden. Einen beispielhaften Auszug der verfügbaren Pfade der *PcPartApi* sieht man in Abbildung 2.



All Computer Parts List and view various computer parts		
GET	/pc-parts/all-parts	Retrieve all computer components
GET	/pc-parts/{id}	Get a computer part by id
DELETE	/pc-parts/{id}	Delete by id
PUT	/pc-parts/{type}	Add a new computer part
Comments & Ratings for Computer Configurations View and create comments and ratings for public computer configurations		
POST	/computers/computers/{computerID}/comment	Write a comment for a computer configuration
GET	/computers/{computerID}/comments	Get comments for a computer configuration
DELETE	/computers/{computerID}/comments/{commentID}	(moderator privileges needed) Delete a comment from a computer
Comments & Ratings for Computer Parts Write and look at ratings for computer parts		
POST	/pc-parts/{partId}/comment	Write and publish a comment for a computer part
GET	/pc-parts/{partId}/comments	Get comments for a computer part
DELETE	/pc-parts/{partId}/comments/{commentId}	[Elevated rights] Delete a comment from a computer part

Abbildung 2: Datenpfade der PcPartAPI (Auszug aus SwaggerUI)

5 Implementierung

Im Folgenden Kapitel wird nun die Implementierung der App erläutert.

5.1 Funktionalitäten

Beginnend wird dabei auf die Hauptfunktionalitäten der App eingegangen.

5.1.1 Navigation

Die Navigation der App erfolgt durch eine untere Navigationsleiste, welche stets sichtbar ist. Hier erhält man die Möglichkeit sich durch die drei Tabs der Anwendung zu bewegen. Dabei ist anzumerken, dass der Login-Tab rechts, bei erfolgreicher Anmeldung mit dem entsprechenden Nutzernamen ersetzt wird (Siehe Abbildung 3).



Abbildung 3: Untere Navigationsleiste

5.1.2 Login

Die Anmeldung erfolgt über zusätzlich erscheinendes Modal, welches beim Drücken auf den Login-Tab auf der Navigationsleiste erscheint. Dieses bietet die Möglichkeit, sich mit dem Nutzernamen sowie Passwort in der API anzumelden. Die Registrationsfunktion ist in diesem Modal auch enthalten (Siehe Abbildung 4).

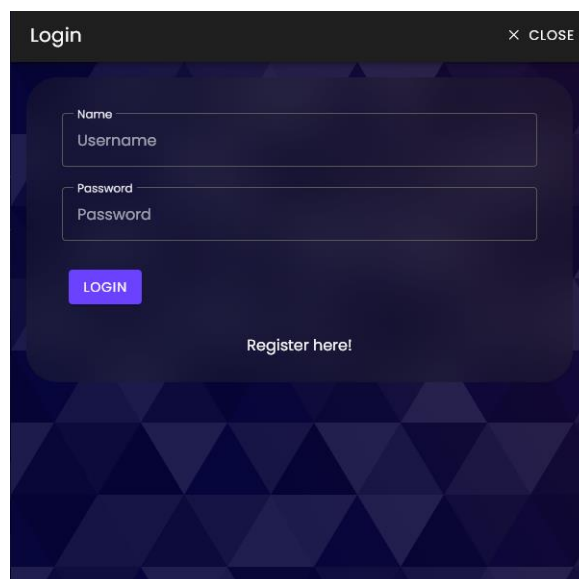


Abbildung 4: Login Modal

Das Formular in dieser Komponente erfolgt reaktiv, d. H. bei falschen Eingaben wird der Nutzer darüber informiert. Bei einer ungültigen Eingabe werden die fehlerhaften Felder rot markiert, dies ist in Abbildung 5 sichtbar.

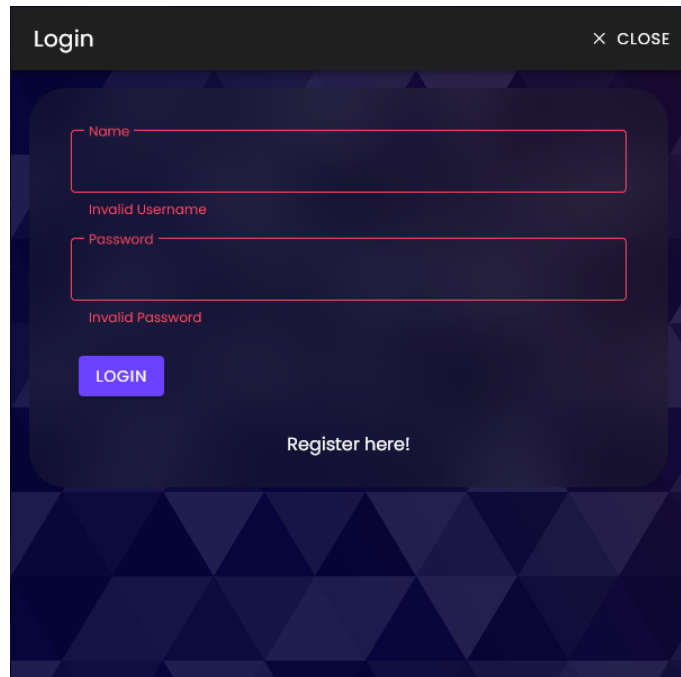
A dark-themed login modal window with a title bar 'Login' and a close button 'X CLOSE'. The modal contains two input fields: 'Name' and 'Password'. Both fields have red borders and red error messages below them: 'Invalid Username' for the Name field and 'Invalid Password' for the Password field. Below the fields is a blue 'LOGIN' button and a link 'Register here!'.

Abbildung 5: Login Modal bei ungültiger Eingabe

Ist der Login erfolgreich getätigt, so wird auf ein weiteres Fenster verwiesen, welches den Nutzer begrüßt und eine kurze Information bezüglich der Nutzerrechte angibt (siehe Abbildung 6).

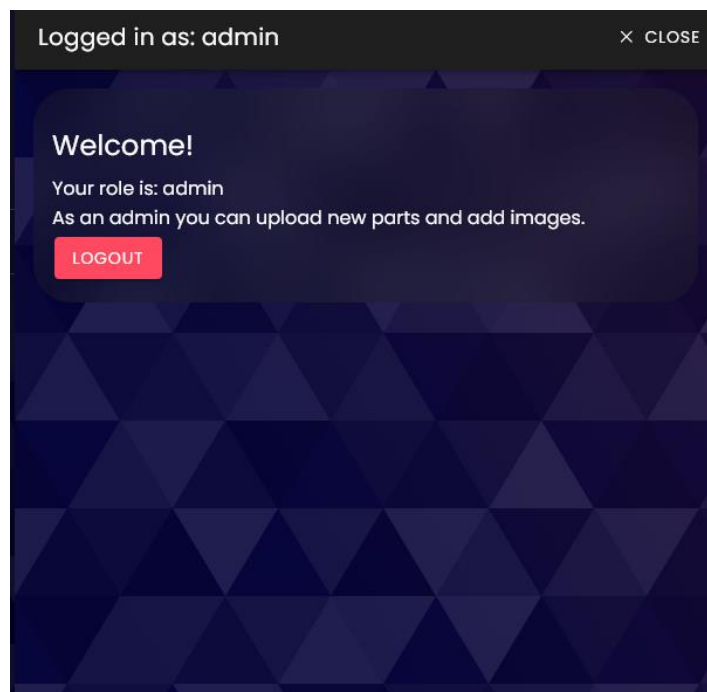
A dark-themed modal window with a title bar 'Logged in as: admin' and a close button 'X CLOSE'. The modal contains a 'Welcome!' message, the text 'Your role is: admin', and a description of admin privileges: 'As an admin you can upload new parts and add images.' Below this text is a red 'LOGOUT' button.

Abbildung 6: Login Modal bei erfolgreicher Anmeldung

5.1.3 Dashboard

Das Dashboard bildet das Hauptfeature der App. Hier hat man die Möglichkeit, sich alle Komponenten gefiltert anzeigen zu lassen. Eine Darstellung sieht man in Abbildung 7. Hier können alle Komponenten aus der API nach dem Typen sortiert werden. Auch ist eine Sortierung nach Hersteller- oder Komponentename möglich.

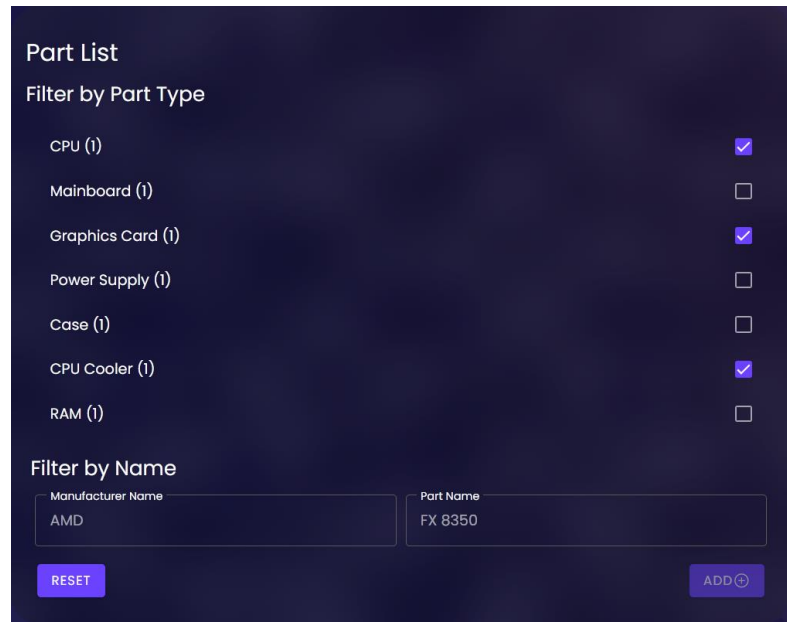


Abbildung 7: Filter-Komponente

Die Darstellung der einzelnen Komponenten erfolgt als *Card*. Diese bieten einen schnellen Überblick über die Informationen der Komponente. Zum einen besteht hier auch die Möglichkeit, ein Bild für diese hochzuladen und festzusetzen. Auch die Kommentare und Bewertungen lassen sich hier hinzufügen (Abbildung 8).



Abbildung 8: Ansicht einer Komponente

Da nun auch neue Komponenten hinzugefügt werden sollen, lässt sich mit dem Drücken des *Add*-Buttons, ein Modal öffnen, welches die entsprechenden Formulare zum Hochladen beinhaltet (Abbildung 9).

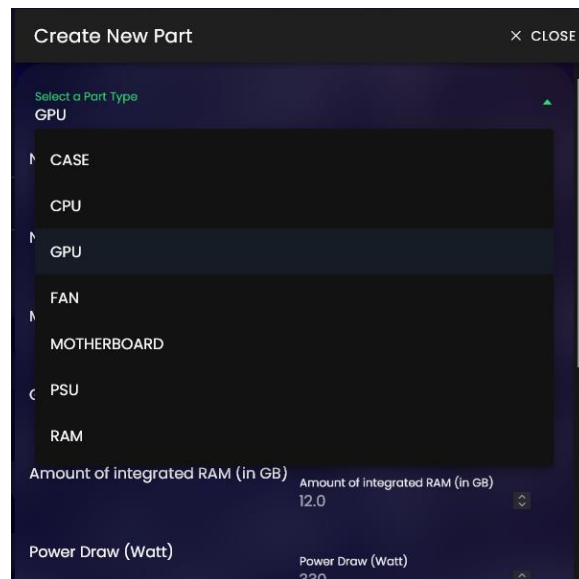


Abbildung 9: Formular zum Hinzufügen einer neuen Komponente

5.1.4 Konfiguration Erstellen

Sind nun genügend Komponenten in der API vorhanden, so können diese zu einer Konfiguration zusammengestellt werden (Abbildung 10). Hier kann zum einen, eine neue Konfiguration mit einem Namen angelegt werden. Diese kann dann ausgewählt werden und man erhält die Möglichkeit, die in einem Computer notwendigen Komponenten auszuwählen und für die Konfiguration festzusetzen. Hier lässt sich auch die Sichtbarkeit der Konfiguration festlegen, wenn diese auf Öffentlich gesetzt wird, so können andere Nutzer diese Konfiguration auch einsehen sowie Kommentieren.

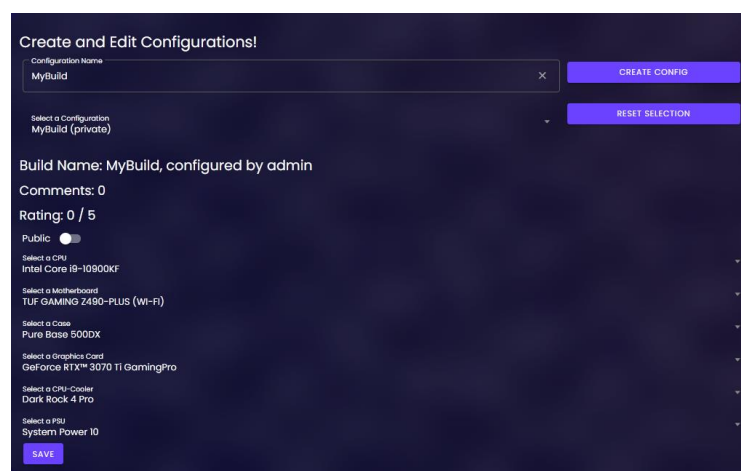


Abbildung 10: PC-Konfigurator

Ist eine solche Konfiguration festgelegt und angemessen zusammengebaut, so wird ein Bestätigungs-Toast eingeblendet. Eine Bestätigung als *Toast* dieser Art lässt sich auch in vielen anderen Formularen und Aktionen der App wiederfinden.

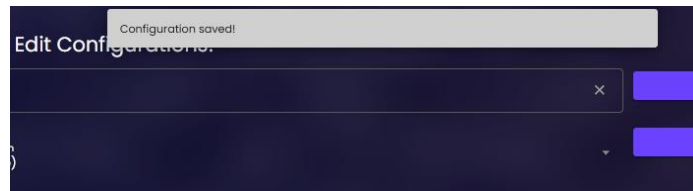


Abbildung 11: Bestätigungs-*Toast*

5.1.5 Öffentliche Konfigurationen

Im Tab der öffentlichen Konfigurationen werden nun alle öffentlich eingestellten Konfigurationen als *Card* angezeigt. Auch hier erhält der Nutzer die Möglichkeit, diese zu kommentieren sowie zu bewerten.

5.2 Technische Besonderheiten

Abschließend in der Implementation wird auf besondere technische Aspekte sowie Problemstellungen eingegangen.

5.2.1 API Service

Da die Kommunikation mit der API ein zentraler Aspekt der App ist, ist es von großer Bedeutung die Datenabhandlung angemessen und sicher umzusetzen. Dazu wurden entsprechende Datentypen sowie zentral verfügbare Services innerhalb der Angular-Anwendung angelegt. Dieser interne API-Service kümmert sich um die entsprechenden GET, POST, DELETE & UPDATE http-Anfragen und sorgt für eine reibungslose Kommunikation. Ein wichtiger und für die nutzerfreundlichkeit kritischer Faktor ist die Asynchronität solcher Kommunikationsprozesse. Da hier mit Latenz zu rechnen ist, gibt der API-Service *Observable*-Objekte zurück, welche mittels eines *subscribe* abgehört werden können. Dies ermöglicht im Betrieb zeitlich korrekte Anpassungen der Oberfläche, um dem Nutzer ein Feedback der Anfragen zu geben. Ein Beispiel wären die Toast-Bestätigungsnachrichten, welche beim Erstellen von neuen Komponenten oder Konfigurationen eingeblendet werden.

5.2.2 Reactive Forms

Da die API zahlreiche und verschiedene Datentypen sowie Komponenten hergibt, wird ein Konzept überlegt, wie die Formularerstellung für diese vereinfacht werden kann. Angular bietet Reactive Forms, d. H. HTML-Formulare, welche programmiertechnisch manipuliert werden

können. So können Validationen oder gar die Formularerstellung mittels einiger Parameter automatisiert werden. Durch diesen Ansatz können die vom Nutzer anzufordernden Daten für z. B. eine Komponentenerstellung einfach und schnell umgesetzt werden. Dazu wurde ein eigener Datentyp für Formularfragen erstellt (FormQuestion). Im Programm kann diese dann parametrisiert werden und als Liste an eine Angular-Komponente übergeben werden, welche diese dann als Formular darstellt. So lassen sich die detailreichen Formulare aus Abbildung 9 relativ einfach Erstellen und wiederverwenden.

```
// Add fields according to specific part types
switch (this.partType) {
  /*****
   * CPU
   *****/
  case 'CPU':
    questions.push(
      new FormQuestion<any>({
        value: '',
        key: 'cpuCores',
        label: 'CPU Core Count',
        placeholder: '16',
        controlType: 'textbox',
        type: 'number',
        validators: [Validators.required, Validators.min(1)],
      })
    );
    questions.push(
      new FormQuestion<any>({
        value: '',
        key: 'cpuThreads',
        label: 'CPU Thread Count',
        placeholder: '32',
        controlType: 'textbox',
        type: 'number',
        validators: [Validators.required, Validators.min(1)],
      })
    );
};
```

Abbildung 12: Programmatische Erstellung von Formularen mit Reactive Forms

5.3 Installation & Inbetriebnahme

Für die Inbetriebnahme des Backends ist die entsprechende Readme.md in dessen Ordner zu folgen. Es ist wichtig, dass Docker auf dem System installiert und ausgeführt ist. Diese kann dann mittels `<./mvnw.cmd quarkus:dev>` gestartet werden.

Für die Angular/Ionic App ist es wichtig, diese mittel folgender Befehle zu starten:

`<npm install>`

`<ng serve>`

Die Anwendung wird dann unter dem Port 4200 zur Verfügung stehen.

6 Fazit

Abschließend lässt sich sagen, dass in Bezug auf die Motivation und Zielsetzung des hier beschriebenen Projektes alle Anforderungen erfolgreich erfüllt und umgesetzt wurden. Es konnte ein angemessen PC-Konfiguration App entwickelt werden, welche die Hauptfunktionalitäten der API nutzerfreundlich umsetzen kann. Dank der von Angular bereitgestellten Reactive Forms und Hilfsklassen, konnte eine effektive Methode entwickelt werden, welche die große Menge an Formulardaten und Feldern relativ einfach und wiederverwendbar umsetzen, kann. Dies erleichtert die Kommunikation mit der API und sichert das Erstellen von neuen Komponenten. Auch das Konfigurationsfeature konnte erfolgreich implementiert werden. Der Nutzer hat die Möglichkeit, aus den verfügbaren Komponenten, eine individuelle Konfiguration zu erstellen.

Abbildungsverzeichnis

Abbildung 1: Öffentliche PC-Konfiguration (https://pcpartpicker.com/b/bH7XsY)	2
Abbildung 2: Datenpfade der PcPartAPI (Auszug aus SwaggerUI)	6
Abbildung 3: Untere Navigationsleiste	7
Abbildung 4: Login Modal	7
Abbildung 5: Login Modal bei ungültiger Eingabe	8
Abbildung 6: Login Modal bei erfolgreicher Anmeldung	8
Abbildung 7: Filter-Komponente.....	9
Abbildung 8: Ansicht einer Komponente	9
Abbildung 9: Formular zum Hinzufügen einer neuen Komponente.....	10
Abbildung 10: PC-Konfigurator	10
Abbildung 11: Bestätigungs- <i>Toast</i>	11
Abbildung 12: Programmatische Erstellung von Formularen mit Reactive Forms	12

Erklärung

Hiermit versichere ich, dass ich diese Hausarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben.

Datum: 28.09.2023

.....
