System über Ernährungsbildung - Audit 3

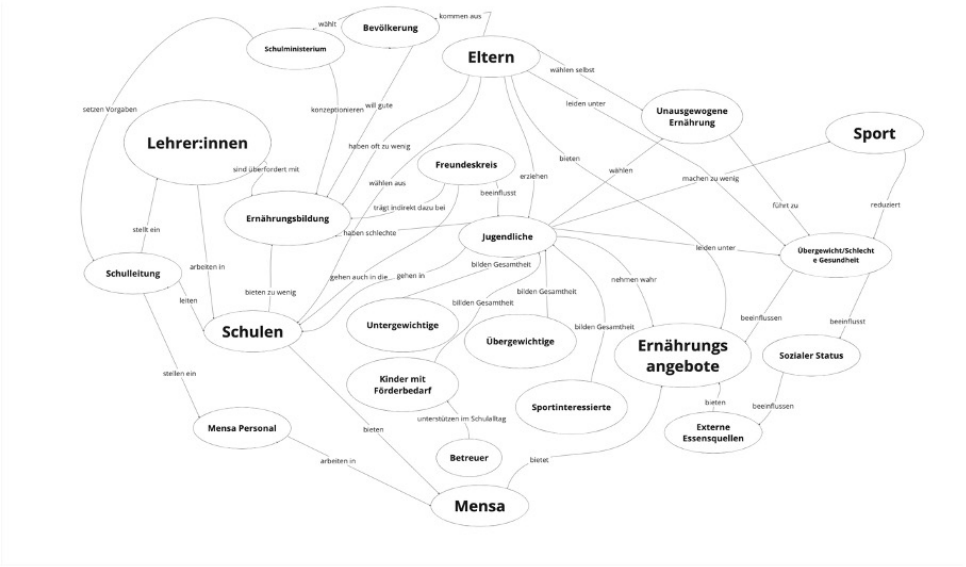
Ernährungsbildung für Jugendliche in der Schule sowie privat.

Gruppe: Sven, Prerit, Caro

Github: https://github.com/Svenjatron/EPWS2324NienhausSinghJungjohann

Iterierte Artefakte

Domänenmodell

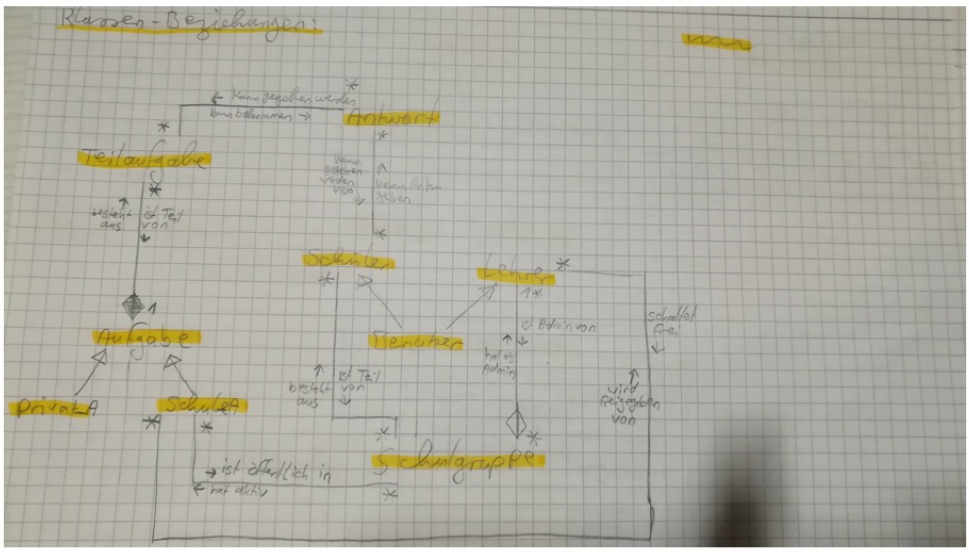


Github Link - https://github.com/Svenjatron/EPWS2324NienhausSinghJungjohann/blob/main/README.md#dom%C3%A4nenmodell-iteriert-3

Kinder mit Förderbedarf und Betreuer hinzugefügt.

Github Link - https://github.com/Svenjatron/EPWS2324NienhausSinghJungjohann/blob/main/README.md#dom%C3%A4nenmodell-iteriert-3

Klassendiagramm (Beziehungen)



Github Link -https://github.com/Svenjatron/EPWS2324NienhausSinghJungjohann/blob/main/Artefakte/Klassendiagramm%20Beziehungen.jpg

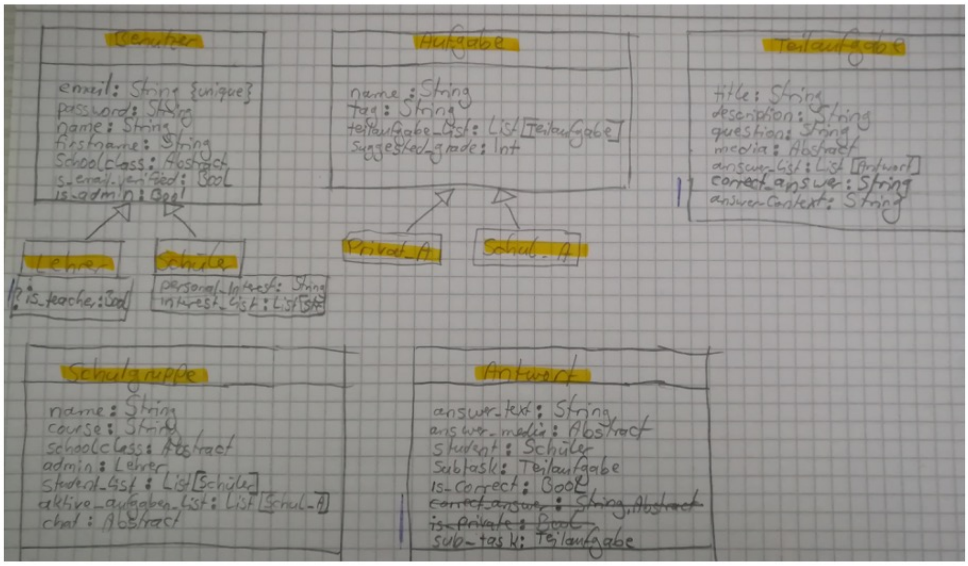
Klassendiagramm (Beziehungen)

https://github.com/Svenjatron/EPWS2324NienhausSinghJungjohann/blob/c33488d1bc373091267023005dddc292c0713944/Artefakte/Klassendiagramm%20Beziehungen.jpg

Das Klassendiagramm stellt die Beziehung zwischen den Aufgaben und den Benutzern dar. Für eine bessere Übersicht wurden die Beziehungen und die Klassenstrukturen in zwei Diagramme aufgeteilt. Die Darstellung der Klassen befindet sich auf der nächsten Seite.

Aufgaben werden in Privat-Aufgaben und Schul-Aufgaben unterteilt (die genaue Variante, diese im Code auseinanderzuhalten, wird noch erarbeitet.). Jede Aufgabe muss aus mindestens einer Teilaufgabe bestehen, sonst kann die Aufgabe an sich logischerweise nicht existieren. ("Aufgabe - Teilaufgabe" ist das Äquivalent zu "Arbeitsblatt - Teilaufgaben") Zu jeder Teilaufgabe können Schüler(Benutzer) eine Antwort geben. Die Benutzer werden in Schüler und Lehrer aufgeteilt (die genaue Variante, diese im Code auseinanderzuhalten, wird noch erarbeitet.). Schülker können wie bereits erwähnt, Teilaufgaben beantworten bzw Aufgaben allgemein beantworten, sowie Teil einer Schulgruppe sein. Lehrer Können Schulgruppen erstellen , sowie Schulaufgaben freischalten, sodass Schüler in der Schulgruppe diese bearbeiten können. Jede Schulgruppe kann nur mit einem Admin, also einem Lehrer, existieren. Sonst würde das Konzept des betreuenden Lehrers so keinen Sinn machen. (Zum Verständnis: Privataufgaben werden den Schülern per Algorithmus vorgeschlagen, was jedoch nicht im Diagramm abgebildet werden kann.)

Klassendiagramm (Klassen)



Github Link - https://github.com/Svenjatron/EPWS2324NienhausSinghJungjohann/blob/main/Artefakte/Klassendiagramm%20Klassen%20-%20Iteriert.jpg

Klassendiagramm (Klassen)

https://github.com/Svenjatron/EPWS2324NienhausSinghJungjohann/blob/c33488d1bc373091267023005dddc292c0713944/Artefakte/Klassendiagramm%20Klassen%20-%20Iteriert.jpg

Das Klassendiagramm stellt die Klassen und ihre Struktur dar. Für eine bessere Übersicht wurden die Beziehungen und die Klassenstrukturen in zwei Diagramme aufgeteilt. Die Darstellung der Beziehungen befindet sich auf der vorherigen Seite.

Der Benutzer hat zwei Subklassen, Lehrer und Schüler. Wie diese Später im code genau unterschieden werden, steht noch nicht fest. Eine Unterscheidung muss stattfinden, da sie unterschiedliche Berechtigungen haben und nur Lehrer als Admin einer Schulgruppe gelten können. Schüler haben zudem persönliche Interessen (Interessnes-Tags) mit denen ihnen Privat-Aufgaben zugeteilt werden können. Benutzer haben allgemein Standart-Attribute wie "email" oder "password", welche zum Anmelden dienen.

Aufgabe hat zwei Subklassen, Privat_Aufgaben und Schul_Aufgaben. Die Unterscheidung muss stattfinden, da die Aufgaben in jeweils anderen Kontexten bearbeitet werden. Privataufgaben können mit Interessens-Tags zugeordnet werden. Schulaufgaben werden allen Schülern in einer Schulgruppe vom Lehrer/Admin zur Bearbeitung freigegeben. Jede Aufgabe besteht aus mehreren Teilaufgaben, welche in der "teilaufgaben_liste" referenziert werden.

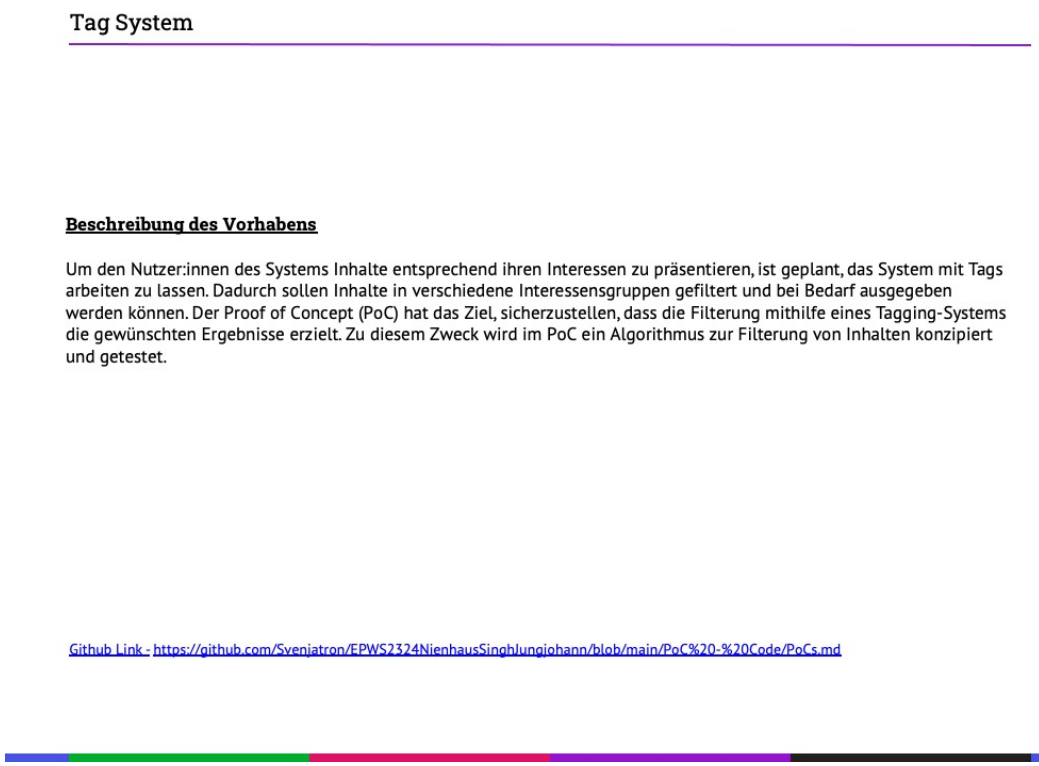
Teilaufgabe enthalten allgemeine Informationen wie "title" oder "description", sowie die "question". Beim Coden ist aufgefallen, dass es Sinn macht, hier in Teilaufgaben-Typen zu unterscheiden (als Subklassen). Das konnte aus zeitlichen Gründen hier noch nicht iteriert werden, wird aber in Zukunft noch geupdated. Neden der "question" gibt es eine Liste mit allen gegebenen Antworten von Schülern. Diese Struktur-Entscheidung steht noch nicht fest. Denn es macht in unseren Augen aktuell keinen Sinn, alle jemals gegebenen Aufgaben jedes Schülers in der Teilaufgabe zu speichern. Dies war auch die Kritik ein einem Open Space. Dort hieß es, dass man das noch während des Codens lösen kann. Aktuell besteht die Idee, in der Schüler-Klasse

eine Referenz auf alle vom Schüler gegebenen Antworten zu hinterlegen. Diese Idee steht jedoch ebenfalls noch nicht fest und wird in Zukunft überarbeitet, damit zusammenhängende Antworten, Schüler und Teilaufgaben sinvoll miteinander verknüpft werden! Zusätzlich stehen gegebenenfalls eine "correct_answer" bereit, sowie "answer_context", um dem Schüler eine Hilfestellung zum Thema zu geben.

Die Klasse Schulgruppe besteht aus allgemeinen Infos wie "name", "course" (also der Kursbezeichnung wie z.B "Deutsch"), "schoolclass" (also Schulklasse z.B "7a"), der "admin" (also der Lehrer), sowie eine "student_list" mit allen Schülern in der Gruppe. Die Liste "aktive_aufgabe_list" beinhaltet alle aktuell bearbeitbaren Aufgaben, welche die Schüler einsehen und bearbeiten können. Der Lehrer (admin) muss diese Vorher freigegeben haben. Zudem besteht die Möglichkeit, an einem "chat" Teilzunehmen, welcher als Kritik aus einem Open Space hier aufgeführt wurde. Damit sollen auch Schüler, welche nicht vor Ort in der Schule sind, die Möglichkeit haben, an einer Diskussion zu den Aufgaben Teilnehmen zu können. Die genaue Umsetzung steht derzeit noch nicht fest. Die Klasse Antwort beinhaltet "answer_text" und "answer_media" als Möglichkeit für den Schüler, eine Antwort auf eine Frage zu geben. Diese Attribute stehen noch nicht fest!

Beim Coden wurde für jeden Teilaufgaben-Typ jeweils auch ein Antwort-Typ al sSubklasse zu "Antwort" erstellt, in der man an den jeweiligen Aufgaben-Typ angepasst Antworten kann. Zum Beispiel muss man bei einer Multiple Choice-Aufgabe keinen Antworttext schreiben, sondern mehrere Antworten ankreuzen. Dieses Konzept steht noch nicht fest und wurde somit in diesem Klassendiagramm nicht dargestellt. "schüler" und "subtask" sind eine Referenz auf den Schüler, welcher die Antwort gegeben hat, sowie zu welcher Teilaufgabe. "is_correct" stellt gegebenenfalls dar, ob die Antwort richtig oder falsch ist.

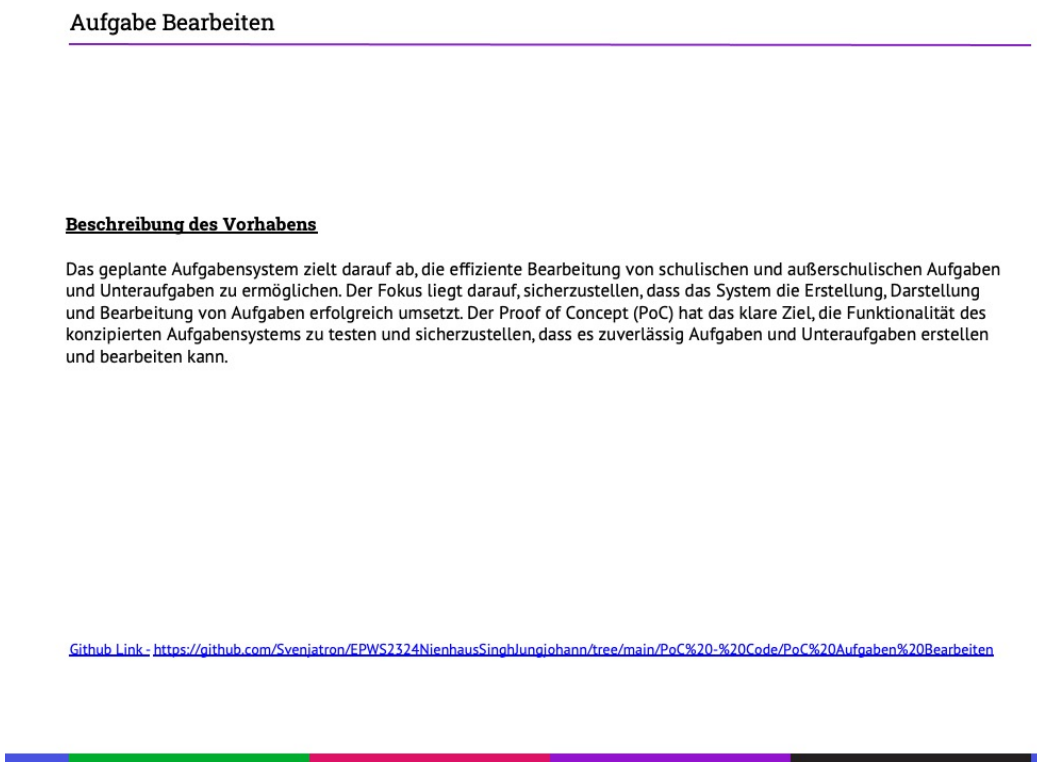
Durchgeführte POCs



Um den Nutzer:innen des Systems Inhalte entsprechend ihren Interessen zu präsentieren, ist geplant, das System mit Tags arbeiten zu lassen. Dadurch sollen Inhalte in verschiedene Interessensgruppen gefiltert und bei Bedarf ausgegeben werden können. Der Proof of Concept (PoC) hatte das Ziel, sicherzustellen, dass die Filterung mithilfe eines Tagging-Systems die gewünschten Ergebnisse erzielt. Zu diesem Zweck wurde im PoC ein Algorithmus zur Filterung von Inhalten konzipiert und getestet. Der durchgeführte PoC ist erfolgreich abgeschlossen worden.

Eine genauere textuelle Beschreibung des / der PoCs inklusive Exit- und Fail-Kriterien, Fallbacks finden sich hier:
https://github.com/Svenjatron/EPWS2324NienhausSinghJungjohann/blob/main/PoC%20-%20Code/PoCs.md

Der Code zu dem PoC findet sich hier:
https://github.com/Svenjatron/EPWS2324NienhausSinghJungjohann/blob/main/PoC%20-%20Code/PoC%20Tag-System/Poc_Tag_System.kt



Dieser PoC soll darstellen, wie eine Aufgabe von einem Schüler bearbeitet werden kann.
Als Referenz wird ein Arbeitsblatt des BLE Medienservice genutzt (https://www.ble-medienservice.de/1642-2-verkaufstricks-im-supermarkt-mit-mir-nicht.html)
Dabei wird beispielhaft eine Aufgabe mit einer Teilaufgabe erstellt und diese Ausgegeben.

Die angelegten Klassen wurden auf Basis des Klassendiagramms bereits erstellt und dienen als Basis (Z. 9-160).
Deshalb werden in diesem Beispiel nicht alle verwendet, da sie dazu nicht zwingend notwendig sind.
Es sei gesagt, dass einige Attribute und Klassen zusätzlich zum Klassendiagramm hinzugefügt wurden, da es während dem Coden mehr Sinn gemacht hat.
Wie bereits im Abschnitt des Klassendiagramms erwähnt, ist noch nicht klar, wie die zusammenhängenden Teilaufggaben, Schüler, und Antworten sinnvoll miteinander verknüpft werden.
Daher besteht eine Referenz auf alle jemals gegebenen Antworten eines Schülers in der Klasse "Schueler" "gegebene_Antworten" (Z. 39),
sowie eine Referenz in der klasse "Teilaufgabe" auf alle jemals von jedem Schüler gegebenen Antworten (Z. 76 "answerList) (was in unseren Augen wenig Sinn ergibt).
Diese Referenzierungen haben beide ihre Nachteile und müssen noch überarbeitet werden!
Man kann von Schüler -> gegebene_Antworten(Z. 39) -> Antwort(Z. 130) -> subtask (Z. 132) nicht auf die jeweilige "Aufgabe" referenzieren zu der die referenzierte Teilaufgabe(Z. 132) gehört.
Zudem macht es wenig Sinn, zu einer Teilaufgabe alle jemals gegebenen Antworten aller Schüler zu speichern (Z. 76 "answerList"). Dies erfüllt aktuell keinen Zweck.
Eine sinnvolle Lösung wird in Zukunft noch erarbeitet und ergibt sich wahrscheinlich beim weiteren Coding
Teilaufgabe hat (anders als im Klassendiagramm) Subklassen für einen Aufgabentyp bekommen ("Teilaufgabe_MC" steht z.B für "Teilaufgabe Multiple Choice" (Z. 87)).
Antwort hat (anders als im Klassendiagramm) ebenfalls Subklassen passend zu den

Aufgabentypen bekommen, da auf jeden Typ anders geantwortet wird ("Antwort_MC" steht für "Antwort_Multiple Choice" (Z. 142)).

Neben den Klassen gibt es noch die allgemeine Funktion "aufgabeAusführen()" (Z. 198), welche Teilaufgaben einer Aufgabe ausführt, und dem Schüler somit die Möglichkeit gibt, zu jeder Teilaufgabe eine Antwort zu schreiben.
Der Funktion wird eine Aufgabe (welche bearbeitet werden soll), sowie der bearbeitende Schüler übergeben.
Dann werden alle Teilaufgaben iteriert (Z. 204) und jeweils pro Teilaufgabe die Funktionen "display_task()"(Z.96) und "answer_task()" (Z. 109) der Klasse "Teilaufgabe_MC" aufgerufen.
Diese sorgen für die Anzeige der Aufgabe, sowie die Möglichkeit der Bearbeitung/Beantwortung dieser.

In der Main werden alle nötigen Objekte erstellt (Z. 173-184), und die Aufgabe durch "aufgabeAusführen() ausgeführt (Z. 186).
Danach werden die Antworten zur Überprüfung ausgegeben.
Es sei gesagt, dass es bei dieser Aufgabe keine "korrekte" Antwort gibt.

Ausblick Audit 4

Geplante Artefakte

- Wireframes
- Funktionaler Prototyp
- Tagliste
- Poster
- Fazit