Technology
Arts Sciences
TH Köln

Technische Hochschule Köln
Fakultät für Informatik und Ingenieurwissenschaften

BERICHT ZUM PRAXISPROJEKT

Plattform für die Spielersuche

Eine mit Javascript entwickelte Plattform

Vorgelegt an der TH Köln Campus Gummersbach
im Studiengang Wirtshaftsinformatik

ausgearbeitet von:
Timo Kalter 12345678
Carlo Menjivar 11117929

Erster Prüfer: Prof. Dr. Birgit Bertelsmeier
Zweiter Prüfer: <Name des 2. Prüfers>

Gummersbach, im <Monat der Abgabe>

Zusammenfassung

Platz für das deutsche Abstract...

Abstract

Platz für das englische Abstract...

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Problemstellung

Unsere Aufgabe ist es, eine Webplattform für Spieler des Videospiels "League of Legendsßu schaffen, auf der sich Spieler kennenlernen und zu einer gemeinsamen Partie verabreden können. Um eine möglichst gute Plattform zu bieten, sind verschiedene Schritte zu erledigen.

1.1 Aufmerksamkeit/Marketing

Im Rahmen der Projektarbeit steht Marketing nicht im Fokus. Wir beschreiben daher nur kurz, worauf zu achten ist, wenn das Produkt nach dem Praxisprojekt auf den Markt kommen soll.

Wie bei sozialen Plattformen und sozialen Netzwerken üblich, erhöht sich der Nutzen durch positive Netzwerkeffekte mit steigender Nutzeranzahl. Ein Ziel ist es daher, eine möglichst große Nutzerbasis aufzubauen, die unsere Webseite möglichst oft verwendet. Zuerst müssen wir Mögichkeiten finden, damit Personen von unserer Webseite überhaupt erfahren. Auf technischer Seite ist dies mit Suchmaschinenoptimierung möglich, Marketingtechnisch haben wir verschiedene Möglichkeiten der Werbung, unter anderem Bannerwerbung, Influencermarketing und "Kunden werben Kunden".

Sobald Personen unsere Webseite besuchen, müssen diese überzeugt werden, dass unser Produkt gut ist und dass sie ein Konto erstellen sollten. Es sollte daher auf einer ansprechenden Startseite das Produkt vorgestellt werden, auf dem erklärt wird, was das Ziel unserer Plattform ist und wie wir dieses Ziel erreichen wollen.

Je besser unser Produkt ist, desto wahrscheinlicher ist es, dass ein Nutzer durch intrinsische Motivation unsere Webseite verwendet. Es ist daher wichtig, ein Produkt mit möglichst hoher Qualität zu erstellen. Ein Nutzer, der von unserer Webseite begeistert ist, wird wahrscheinlicher die Webseite öfter verwenden und Freunden von der Webseite erzählen, was wiederum neue Nutzer generieren kann.

1.2 Das Nutzerkonto

Um die Webseite vollständig nutzen zu können, sollen Besucher der Seite ein Konto erstellen. Dazu sollen sie eine E-Mail angeben sowie ein Passwort und einen freigewählten Nutzernamen aussuchen. Auch sollen sie nach Kontoerstellung Nutzerdaten wie ein Profilbild, das Alter und einen Freitext angeben können. Die angegeben Daten sollen dem Nutzer dabei helfen, Kontakte auf der Plattform zu finden und können als Gesprächsthema dienen.

Auch ist geplant, dass das Nutzerkonto mit dem Riot-Konto verbunden werden kann. Dies soll es ermöglichen, dass Spielstatistiken, wie zum Beispiel die meist gespielten Cham-

pions, die Lieblingsposition und die Elo, angezeigt werden. Ein Nutzer, der auf Kontaktsuche ist, erhält damit mehr Informationen über den Spieler und kann somit besser abschätzen, ob er Kontakt aufnehmen will.

1.3 Kontaktsuche

Um mit anderen Nutzern in Kontakt treten zu können, soll es möglich sein, andere, zufällige Nutzerprofile anzuzeigen. Profile sollen einzeln hintereinander angezeigt werden; nachdem sich der Nutzer entschieden hat, ob er dem angezeigten Nutzer einen Like [GLOSSAR!] vergeben will, wird das nächste Profil angezeigt.

Sobald zwei Nutzer gegenseitig einen Like vergeben haben, sollen diese befreundet sein. Befreundete Nutzer sollen untereinander Nachrichten austauschen können.

1.4 Filter

Um die Qualität der angezeigten Profile zu erhöhen und eine möglichst gute Nutzererfahrung zu gewährleisten, soll es dem Nutzer möglich sein, seine Suche einzugrenzen. Durch Filter werden ihm nur die Nutzer angezeigt, die für ihn relevant sind.

1.5 Regeln und Richtlinien

Wir möchten mit unserer Plattform einen sicheren Ort bieten, auf dem die Spieler unabhängig ihrer Eigenschaften respektiert werden. Wir sind der Meinung, dass die meisten Nutzer in der Lage sind, respektvoll miteinander umzugehen. Dies können wir jedoch nicht gewährleisten.

Wir wollen uns daher das Recht vorbehalten, Nutzern, die gegen unsere Regeln und Richtlinien verstoßen, den Zugang zu unserer Plattform zu verwehren. Sollte sich ein Nutzer von einem anderen Nutzer beleidigt fühlen oder andersweitig der Meinung sein, dass dieser Nutzer gegen die Regeln verstößt, soll dieser Regelbrecher gemeldet werden können. Wir als Betreiber der Plattform sollen in der Lage sein, die Meldungen zu verarbeiten und angemessen mit den Regelbrechern zu verfahren.

Unabhängig davon soll es den Nutzern möglich sein, andere Nutzer zu blockieren. Blockierte Nutzer sollen dem aktiven Nutzer nicht mehr in der Spielersuche angezeigt werden, nicht mehr Nachrichten verschicken und aktive Chats sollen aufgelöst werden. Einen anderen Spieler zu blockieren heißt nicht zwangsläufig, dass dieser gegen Regeln oder Richtlinien verstößt, es kann auch sein, dass der Nutzer einfach den Kontakt abbrechen will. Sollte ein Nutzer jedoch gemeldet werden, soll er automatisch blockiert werden.

1.6 Internationalisierung

Die Webseite soll in Englisch angeboten werden, da dies eine der weit verbreitetsten Sprachen der Welt ist und damit viele Nutzer erreicht.

1.7 Einstellungen

Um den verschiedenen Vorlieben unterschiedlichster Nutzer gerecht zu werden, sollen diese in der Lage sein, verschiedene Optionen in den Einstellungen zu verwalten. Unter anderem sollen Nutzer in der Lage sein, auszusuchen, wie und ob sie über neue Nachrichten, Freundschaften und Neuigkeiten informiert werden.

1.8 Nachrichten

Sobald zwei Nutzer befreundet sind, können sie sich gegenseitig Nachrichten schreiben. Dazu soll auf einer dafür eingerichteten Seite in ein Eingabefeld der zu versendende Text eingegeben werden. Sobald die Nachricht abgeschickt wird, soll der andere Nutzer möglichst Zeitnah von dieser erfahren. Dies soll den glatten Ablauf von Chats ermöglichen.

Zeichen sollen nach dem UTF-8 Zeichensatz erlaubt sein.

1.9 Sicherheit

Die Sicherheit der Nutzerdaten ist von hoher Priorität. Passwörter müssen besonders geschützt und nach derzeitigem kryptologischen Stand der Technik gesichert werden, um den Zugriff von unautorisierten Angreifern zu unterbinden. Identifizierende Daten wie E-Mail-Adressen dürfen nicht öffentlich einsehbar sein.

Sollte in einem späteren Schritt die Plattform veröffentlicht werden, ist auf DSGVO-Konformität zu achten.

1.10 Dokumentation

"Technische Schulden"(en. "technical dept", schlechte Umsetzung von Software, die einen erheblichen Mehraufwand in der Zukunft bedeutet) sorgen in vielen Fällen für Probleme im Laufe des Produktlebenszyklus. Um die Wahrscheinlichkeit zu verringern, dass sich technische Schulden anhäufen und, soll eine gut leserliche Dokumentation zum Quellcode geschrieben werden, welche es allen Gruppenmitgliedern ermöglichen soll, den überblick zu halten, welcher Codeschnipsel welches Problem löst. Desweiteren sollen umfangreiche Tests durchgeführt werden, welche die Qualität des Codes gewährleisten sollen.

Um die Kommunikation zwischen Backend und Frontend zu gewährleisten, sollen Schnittstellen entsprechend dokumentiert werden. Es sollte einem Frontend-Entwickler

möglich sein, nur mit der Dokumentation auf die Schnittstelle zuzugreifen und die Informationen zu erhalten, die er benötigt, um entsprechende Daten im Frontend anzeigen zu können.

1.11 Konto löschen

Es wird Benutzer geben, die aus verschiedensten Gründen unsere Webseite nicht weiter verwenden wollen und ihr Konto löschen wollen. Unsere Aufgabe ist es, eine sichere Löschung des Benutzerkontos zu gewährleisten und persönliche Daten nach Datenschutzvorschriften aus der Datenbank zu entfernen.

Optional können wir zudem den Nutzer darum bitten, ein Formular auszufüllen, in dem dieser uns Rückmeldung geben kann, aus welchen Gründen er die Webseite nicht weiter verwenden will.

2 Einleitung

WORK IN PROGRESS...

2.1 Einführung in das Thema (Motivation, zentrale Begriffe etc.)

2.2 Hinführung zu den Ergebnissen

2.3 Ggf. Angabe des Schwerpunktes

2.4 Ggf. Einschränkungen darlegen

2.5 Problemstellung

2.6 Zielstellung der Arbeit

2.7 Fragestellung der Arbeit

2.8 Struktur der Arbeit

Dieser Projektbericht ist in folgenden Kapitel unterteilt: **Kapitel 2**

Kapitel 3

Kapitel 4? erläutert, welche JavaScript-Frameworks zu Beginn des Projekts berücksichtigt wurden. Er gibt einen Überblick über die Faktoren, die zu beachten sind, wenn man sich für React entscheidet, und zeigt schließlich, wie die Daten mit Hilfe der React JavaScript-Bibliothek und ihrer Hooks für den Endbenutzer visualisiert werden.

Kapitel 5? geht es um die Interaktion zwischen dem Client und dem Server.

Anhand von zwei Beispielen wird gezeigt, wie Lese- und Schreibabfragen mit Hilfe von GraphQL und ApolloClient durchgeführt wurden.

Kapitel 6? zeigt die Relevanz von Testfällen. Sie zeigt auch die Vorteile automatisierter und eingebetteter Testfällen in der Entwicklungsumgebung.

Die Einleitung(DELETE AFTER CHECK WE COMPLETE ALL POINTS)

Die Einleitung umfasst folgende Elemente[a]:

- Einführung in das Thema (Motivation, zentrale Begriffe etc.)
- Hinführung zu den Ergebnissen
- Ggf. Angabe des Schwerpunktes
- Ggf. Einschränkungen darlegen
- Problemstellung
- Zielstellung der Arbeit
- Fragestellung der Arbeit
- Übersicht über die Kapitel geben:

[a] Vgl. u.a. [1], S. 5-6

3 Grundlagen

TODO: Carlo, to review:

3.1 Stack MERN

-Was macht dieser Stack so interessant?

3.2 Matching Plattformen

-Hier können wir die Logik hinter Swipe/Like erklären und wann wir es Nutzern ermöglichen, Nachrichten miteinander auszutauschen.

4 Datenbank

4.1 Anforderungen

Es wird eine moderne Plattform entwickelt, auf der sich Nutzer ähnlich Social Media Profile anderer Nutzer anschauen und miteinander Chats führen. Nutzer sollen in Echtzeit miteinander schreiben können und nahezu keine Wartezeit in Kauf nehmen müssen, um sich andere Profile anzeigen zu lassen. Die Datenbank muss nahezu immer erreichbar sein, da unsere Webseite ohne Datenbankanbindung nur beschränkt nutzbar ist. Es wird dabei versucht, Lesezugriffe zu optimieren Wenn Benutzer Änderungen an ihrem Profil vornehmen, kann eine geringe Wartezeit in Kauf genommen werden. Nutzer müssen bei Profilen nicht unbedingt die aktuelleste, gerade veränderte Profilbeschreibung sehen; es ist viel wichtiger, dass die Nutzer Nutzerprofile schnell sehen, all dass diese auf dem neuesten Stand sind. Webseiten wie unsere können "über Nachtßum Erfolg werden, es reicht ein großer Influencer, welcher seinen Followern von der Webseite berichtet und auf einen Schlag erreichen wir Nutzerzahlen, bei denen unsere Rechenleistung auf Grenzen stößt. Die Datenbank unserer Wahl sollte sich vor allem schnell, ohne viel Programmieraufwand, skalieren lassen, um die Zeit, in denen unsere Plattform ausgelastet ist, möglichst gering zu halten. Das Ziel ist, dass Nutzer ihr Verhalten nicht auf unsere Webseite anpassen müssen. Stattdessen soll die Webseite auf die Wünsche unserer Nutzer angepasst werden. Datenstrukturen sollen sich schnell und oft ändern können, um dies zu ermöglichen.

4.2 Wahl der Datenbank

In der näheren Auswahl der Datenbank standen die SQL-Datenbank PostgreSQL und MongoDB, eine NoSQL-Datenbank, zwei beliebte Datenbankmanagementsysteme mit unterschiedlichen Ansätzen. Wir werden uns im Folgenden die Gemeinsamkeiten und Unterschiede der Datenbanken anschauen und erklären, warum wir MongoDB als die für unser Projekt beste Lösung halten.

4.2.1 PostgreSQL

Michael Stonebreaker veröffentlichte 1974 in Berkeley unter BSD-Lizenz das RDBMS INGRES. Jahre später führte er das Projekt als Postgres (Post INGRES) weiter und fügte einen objektorientierten Ansatz hinzu. [12] Im Laufe der Jahre wurde PostgreSQL fortlaufend weiterentwickelt und hat viele neue Funktionen dazuerhalten. PostgreSQL beschreibt sich selbst als „das fortschrittlichste, quelloffene, relationale Datenbankmanagementsystem und habe sich in über 30 Jahren aktiver Entwicklung einen guten Ruf für Zuverlässlichkeit, Funktionsrobusheit und Leistung verdient“. [13] PostgreSQL soll auch in Zukunft unter einer quelloffenen Lizenz stehen, welche die kommerzielle Nutzung erlaubt. [14] PostgreSQL wurde exemplarisch als Option eines SQL-Systems verwendet, da

der Funktionsumfang vergleichbar mit anderen beliebten SQL-Systemen ist. [15]

4.2.2 MongoDB

Die 2007 neu gegründete Firma 10Gen, mittlerweile bekannt als MongoDB Inc., benötigte eine Datenbank, welche den Anforderungen ihrer quelloffenen Plattform-as-a-Service Cloud-Architektur gerecht werden würde. Das Team suchte nach einer Datenbank, die elastisch, skalierbar, einfach zu verwalten und für Entwickler und Anwender einfach zu benutzen ist. Unzufrieden mit den zu der Zeit auf dem Markt verfügbaren Datenbanksystemen wurde MongoDB, eine dokumentbasierte Datenbank entwickelt. Als das Team das Potenzial der Datenbank realisierte, wurde die Idee der Cloud-Plattform eingestellt und die Entwicklung von MongoDB gefördert.[19] Laut eigenen Aussagen ist „MongoDB [] eine universelle, dokumentbasierte, verteilte Datenbank für die moderne Anwendungsentwicklung und die Cloud, die in puncto Produktivität höchsten Ansprüchen gerecht wird“. [20]

4.2.3 MongoDB und PostgreSQL im Vergleich

Kriterien	PostgreSQL	MongoDB
Datenbanktyp	SQL, Objektrelational	NoSQL, Dokumentbasiert
Ranking nach DB-Engines [33]	577,50 Punkte, viertbeliebteste Datenbank, viertbeliebteste relationale Datenbank	496,50 Punkte, fünftbeliebteste Datenbank, beliebteste nicht-relationale Datenbank [34]
Architektur	Monolithisch	Dezentralisiert
Erscheinungsjahr	1989	2009
Datenschema	Ja	Selbstbeschreibende Dokumente
Referenzierung	Referenzierung per ID, erlaubt Foreign Keys	Referenzierung per ID oder eingebettetes Sub-Dokument
Datenstruktur	Tabellen bestehen aus Zeilen und Spalten	Kollektionen bestehen aus Dokumenten. Dokumente haben Felder. Dokumente der gleichen Kollektion müssen nicht die selben Felder besitzen.
Typisierung	Erlaubt verschiedene Datentypen und nutzerdefinierte Datentypen[16]	Erlaubt verschiedene Datentypen nach BSON Spezifikation [21]; ähnelt stark JSON
Horizontale Skalierung	Repliken nach Master-Slave-System [3]	Sharding und Replica Sets als Teil der Infrastruktur [22] [23]
Konsistenzmodell	ACID, optional eventuelle Konsistenz [2] [24]	ACID

[35] [36]

4.2.4 ACID

MongoDB ist seit Erschaffung auf Dokumentebene ACID-Konform und ab den Versionen 4.0 (Juni 2018 [25]) auf einem einzelnen Replik-Set bzw. ab Version 4.2 (August 2019 [25]) zwischen mehreren Replik-Sets bei multi-Dokument-Transaktionen ACID-Konform. [2] Dies erlaubt Älles-oder-NichtsTransaktionen, bei denen es kritisch ist, dass entweder alle Teile oder kein Teil der Transaktion ausgeführt wird und löst damit Probleme, die historisch nur mit SQL-Datenbanken lösbar waren.

4.2.5 Referenzierung und Denormalisierung

PostgreSQL erlaubt, andere Tabellen mit Foreign Keys (FK) gemäß SQL-Standard zu referenzieren. Der Wert eines FK muss korrekt sein, sollte der angegebene FK in der referenzierten Tabelle nicht existieren, wird ein Fehler geworfen. Auch ist es möglich, entsprechende Regeln zu definieren, was passieren soll, wenn der referenzierte Datensatz beispielsweise gelöscht wird. MongoDB ermöglicht auch, andere Dokumente per ID zu referenzieren, bietet jedoch keine direkte Option, zu definieren, was beim Löschen des referenzieren Datensatzes passieren soll. Es ist weiterhin möglich, Trigger beim Löschen des referenzierten Datensatzes feuern zu lassen, dies hat im Vergleich zum Foreign Key Constraint allerdings eine höhere Komplexität. Stattdessen setzt MongoDB darauf, alle relevanten Informationen soweit möglich in einem Dokument einzubetten.

Eingebettete Dokumente erlauben schnelleren Lesezugriff, da sich alle relevanten Informationen in einem Dokument befinden und - anders als in SQL - nicht über mehrere Tabellen mit im Laufe des Produktlebenszyklus immer komplexeren JOIN-Anweisungen zusammengefasst werden müssen. Dies vereinfacht das Schreiben von Abfragen und erhöht die Geschwindigkeit von Leseabfragen. Als Nachteil wird dabei in Kauf genommen, dass Daten über verschiedene Dokumente dupliziert werden und Schreibzugriffe dadurch langsamer werden - schließlich müssen Daten in verschiedenen Dokumenten erstellt oder aktualisiert werden. Jedes Mal, wenn das Profil eines Nutzers in der Kontaktsuche angezeigt werden soll oder wenn die Freundesliste oder ein Chat geöffnet wird, muss das Profil dieses Nutzers abgefragt werden. Das kann teilweise für hunderte Anfragen pro Profil am Tag sorgen und belastet entsprechend unsere Datenbank. Den Projektanforderungen nach ist außerdem wichtig, dass Nutzer schnell Profile angezeigt bekommen. Schreibanfragen, bei denen ein neues Profil erstellt wird oder ein Nutzer sein eigenes Profil aktualisiert, werden weitaus seltener vorkommen als Leseanfragen. Auch sind wir der Meinung, dass Nutzer eine längere Wartezeit bei Profilaktualisierungen eher hinnehmen, als bei Lesezugriffen zu warten. Entsprechend sind wir der Meinung, dass für unser Projekt die Vorteile eingebetteter Dokumente dessen Nachteile überwiegen.

4.2.6 Skalierbarkeit

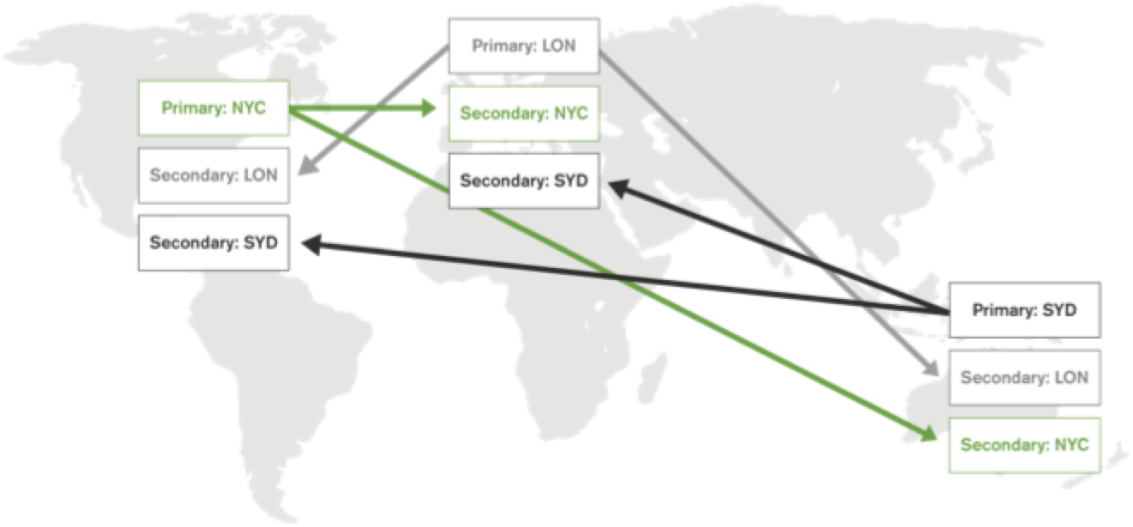
Sowohl PostgreSQL als auch MongoDB lassen sich vertikal skalieren - mit mehr Ressourcen läuft die Datenbank schneller. Bei horizontaler Skalierung verfolgen die beiden Datenbanken verschiedene Ansätze.

PostgreSQL nutzt ein Master-Slave-System bzw. Primary-Standby-System mit Lastverteilung.[3] Die Standby-Knoten sind Kopien des Primärknotens und können Lesezugriffe verarbeiten. Schreibzugriffe werden nur vom Primärknoten angenommen. Sollte der Primärknoten versagen, bietet PostgreSQL keine automatische Lösung dieses Problems, ohne Drittsoftware

muss manuell ein neuer Primärknoten gewählt werden. Es gibt nur einen Primärknoten, für Multi-Master-Systeme wird Drittsoftware benötigt. Selbst mit synchronen Repliken dauert es einen Moment, bis die Standby-Knoten die Aktualisierungen der Datenbank übernommen haben, dies kann dazu führen, dass Abfragen mit veralteten Datensätzen beantwortet werden.[18] Consistency nach CAP-Theorem ist für PostgreSQLs Master-Slave-Systeme somit nicht mehr einhundertprozentig gegeben. Durch Sharding lässt sich die Datenbank in einzelne Knoten aufteilen, die jeweils nur einen Teil der Daten beinhalten. Dies erlaubt es, die Last und benötigte Speicherkapazität pro Knoten weiter zu verringern. Es ist möglich, mehrere Shards vom gleichen Knoten verwalten zu lassen; dies erleichtert die Skalierung, da die Anzahl der Shards flexibel an die technischen Ressourcen des verwaltenden Knoten angepasst werden kann.

MongoDB benutzt Replik-Sets, welche ähnlich wie das vorgestellte Master-Slave-System von PostgreSQL funktionieren. Der Primärknoten ist der einzige Knoten mit Schreibzugriff und repliziert die Änderungen auf die Sekundärknoten. Alleridngs hat MongoDB ein automatisches System, welches einen Ausfall des Primärknotens abfängt. Alle Knoten teilen den anderen Knoten ihren „Herzschlag“ mit, in dem sie sich gegenseitig anpingen. Sollte der Herzschlag des Primärknotens ausfallen, wählen die Sekundärknoten unter ihnen einen neuen Primärknoten aus, welcher dann die Aufgaben des früheren Primärknotens übernimmt. Es ist möglich, Knoten mit mehr Rechenleistung eine höhere Priorität zuzuweisen, um die Wahrscheinlichkeit zu erhöhen, dass dieser Knoten der nächste Primärknoten wird. Auch ist es möglich, zu verhindern, dass spezielle Knoten Primärknoten werden, dies ist ratsam für Knoten, die schlechtere Hardware besitzen oder eine höhere Latenz aufweisen, weil sie beispielsweise in einer entfernten Region stehen. Den Knoten können zudem verschiedene Rollen zugewiesen werden, versteckte Knoten können für Datenbankauswertungen verwendet werden, während verzögerte Knoten einen historischen Schnappschuss der Datenbank speichern und Aktualisierungen der Datenbank mit einer Verzögerung ausführen und somit als Backup dienen können. [27][28][29][30] MongoDB erlaubt zudem Systeme mit mehreren Primärknoten, in den meisten Fällen sind Optionen wie Sharding jedoch besser geeignet. [31] Shards wiederum können als eigene Replica-Sets eingesetzt werden. Dies erlaubt es, für verschiedene Regionen eigene Shards der Datenbank einzurichten, welche dann wiederum in einem eigenen Replica-Set gesteuert werden.

Jedes Jahr fallen größere Mengen an Daten an. Durch diese Entwicklung ist die Möglichkeit, horizontal skalieren zu können, immer wichtiger geworden. Zur Anfangszeit von PostgreSQL hat es für die meisten Projekte gereicht, vertikal zu skalieren, erst über die Jahre wurden Techniken zur horizontalen Skalierung entwickelt. Für Techniken wie der automatischen Wahl eines neuen Primärknotens oder Multi-Master-Systemen benötigt es Drittsoftware. In MongoDB ist horizontale Skalierung seit jeher wichtige Priorität und ist als solches stark in die Datenbankinfrastruktur eingebettet. Wir sind der Meinung, dass



[31]

Abbildung 1: Aktiv-Aktiv-Architektur mit Datenbank-Sharding

die von MongoDB verwendeten Lösungen zur horizontalen Skalierung ausgereifter sind und gleichzeitig weniger fachliches Wissen benötigen und damit entsprechend schneller und einfacher durchführbar sind.

4.2.7 Flexibilität

Wir haben die Erfahrung gemacht, dass vor allem in der Anfangsphase von Projekten Datenbankstrukturen oft angepasst und verworfen werden. MongoDB nutzt Kollektionen, dessen Dokumente nicht die exakt gleiche Strutur aufweisen müssen, anders als dies bei einer ORDBMS-Tabelle wie bei PostgreSQL der Fall ist. Diese Flexibilität ermöglicht es, die Datenbank schnell auf sich wechselnde Projektziele anpassen zu können und erhöht somit die Geschwindigkeit, mit der an agilen Projekten gearbeitet werden kann. Dabei ist darauf zu achten, dass die Datenbankänderungen nicht zu mehr Problemen führen, als sie lösen. Sollte dieses Risiko beachtet werden, eignet sich MongoDB für unser Projekt in diesem Punkt besser als PostgreSQL.

4.2.8 Das CAP-Theorem

Brewers CAP-Theorem sagt aus, dass es in verteilten Systemen wie Datenbanken nicht möglich ist, gleichzeitig Consistency (Konsistenz), Availability (Verfügbarkeit) und Partition Tolerance (Partitionstoleranz) zu gewährleisten. Man muss sich für zwei entscheiden. Dies ist eine starke Simplifizierung, die den heutigen Datenbanken in der Form nicht gerecht wird und zwölf Jahre später von Brewer angepasst wurde. Viele aktuelle Datenbanken erlauben es, in unterschiedlichen Konfigurationen verschiedene Ziele zu erreichen

und auch wenn weiterhin nur zwei der drei Ziele vollständig erfüllt werden können, ist das CAP-Theorem flexibler als ursprünglich angenommen

Wir haben am Fallbeispiel von PostgreSQL zeigen können, dass im Multiknoten-System ein Teil der Konsistenz aufgegeben werden kann, um kürzere Antwortzeiten und Partitionstoleranz (Standby-Knoten dürfen ausfallen) zu gewährleisten. Die Einführung eines Multi-Master-Systems würde eine Erhöhung der Partitionstoleranz (beliebige Knoten dürfen ausfallen) zu Kosten von Konsistenz (gleichzeitige Schreibzugriffe auf verschiedene Knoten) erwirken und somit das System Richtung AP-Datenbank verschieben. Synchrone Replizierung auf Standby-Knoten erhöht die Konsistenz, erhöht aber gleichzeitig die benötigte Zeit einer Operation und verringert somit die Verfügbarkeit, schiebt also somit den Fokus Richtung CP-Datenbank. Diese Funktionen sind in anderen Datenbanken in ähnlicherweise zu finden und erlauben aktuellen Datenbanken, fein auf die Anforderungen des Benutzers angepasst zu werden.

4.2.9 Erfahrung

Das Entwicklerteam hat in der Vergangenheit bereits Erfahrung in MongoDB sammeln können und ist gut mit der Datenbank zurecht gekommen. Die Entwickler wissen, wie die Datenbank funktioniert und auf was zu achten ist. Dies verringert das Risiko und spart Zeit, da die Technologie bereits bekannt ist.

4.2.10 Dateiformat

MongoDB speichert Daten im „binary JSON“-Format. JSON, die „JavaScript Object Notation “, ist „ein schlankes Datenaustauschformat, welches für Menschen einfach zu lesen und für Maschinen einfach zu parsen [] ist“ [38]. JSON als semistrukturiertes Dateiformat eignet sich gut für Schnittstellendaten. Zudem wird im gewählten MERN-Techstack ausschließlich JavaScript verwendet - das JavaScript native Dateiformat JSON ist daher ohne Umwandlungen direkt verwendbar und der Umgang für unsere Entwickler bereits bekannt. Dies verringert die Gefahr möglicher Komplikationen und spart Lern- und Programmieraufwand.

4.2.11 Beliebtheit

Eine beliebte Datenbank zu wählen hat einige Vorteile. Beliebte Datenbanken werden meist aktiv weiterentwickelt, haben oft mehr Tools von Drittanbietern welche die Arbeit erleichern und haben meist eine aktive Programmierergemeinschaft, welche bei kleineren Problemen schnell aushelfen kann. Bei der Expandierung des Projekts gibt es zudem eine größere Auswahl an qualifiziertem Fachpersonal. Nach dem Ranking von DB-Engines schneiden PostgreSQL und MongoDB ähnlich ab. PostgreSQL ist mit 577,50 Punkten die viertbeliebteste ORDBMS und insgesamt die viertbeliebteste Datenbank. MongoDB ist

mit 496,50 Punkten die beliebteste Dokumentdatenbank und damit insgesamt die fünftbeliebteste Datenbank. Platz Eins bis Vier werden von ORDBMS-Datenbanken ein. [36] In der Entwicklerumfrage 2021 von StackOverflow kommt man auf ähnliche Ergebnisse. Von den 49.537 Befragten gaben 36,1% an, PostgreSQL zu nutzen, während 26,4% MongoDB nutzen. PostgreSQL landet damit auf Platz 2, MongoDB auf Platz 4. [37] Beide Datenbanken erfreuen sich großer Beliebtheit und werden allen Anschein nach noch etliche Jahre verwendet werden. Im NoSQL-Markt ist MongoDB am beliebtesten, während es im SQL-Markt weitere andere beliebte Datenbanken gibt. In diesem Aspekt hat keine Datenbank Vorteile gegenüber der Anderen.

4.2.12 Fazit

Wir haben nach einer Datenbank gesucht, welche bei Leseanfragen schnell antwortet, schnelle Änderungen der Datenstrukturen zulässt, eine hohe Erreichbarkeit aufweist und sich schnell und einfach skalieren lässt. Dafür nehmen wir langsamere Schreibanfragen und Inkonsistenz bei Leseanfragen in Kauf. Wir konnten zeigen, dass sowohl PostgreSQL als auch MongoDB ausgereifte Lösungen sind, sich MongoDB als Gesamtpaket für unsere speziellen Anforderungen besser eignet.

4.3 Schemata

Folgende Kollektionen wurden verwendet, um die Daten optimal zu verwalten.

4.3.1 Nutzer

Wenn ein Nutzer einen Account erstellt, gibt dieser dessen eMail, den gewünschten Benutzernamen und das gewünschte Passwort an. Durch Indexe wird die Einzigartigkeit von eMail und Benutzername geprüft, das Passwort wird aus Sicherheitsgründen in einer separaten Kollektion gespeichert. Nach der Kontoerstellung kann der Nutzer das Geburtsdatum, die gesprochenen Sprachen, das Geschlecht, die Spielposition und einen Freitext angeben und ein Bild hochladen, welches als Avatarbild dient. Felder wie der normalisierte Name, das Alter und die Rolle werden automatisch generiert. Im Laufe der Nutzung unserer Plattform wird der Nutzer andere Spieler als Freunde hinzufügen - diese werden in einer Freundesliste gespeichert. Auch steht es dem Nutzer frei, Andere zu blockieren - in diesem Fall wird die NutzerID des Blockierten auf die Blockliste hinzugefügt. Privatsphäre und damit die Sicherheit der eigenen Daten ist uns wichtig. Wir wollen daher die Anzahl an Daten, die ein Nutzer von sich preisgeben müssen, so gering wie möglich halten. Die Email-Adresse, das Geburtsdatum und Freundes- und Blockliste sind für an-

dere Nutzer nicht einsehbar, bis auf den Benutzernamen muss eine Person keine Daten öffentlich angeben.

4.3.2 Sprache

Damit Nutzer ihre Sprache wählen können, bieten wir die Wahl zwischen 187 Sprachen nach ISO 639-1 Norm an.[41]

Feld	Beschreibung	Beispiel
id	Alpha-2 Code der Sprache	en, de, fr
name	Englische Schreibweise der Sprache	English, German, French
nativer Name	native Schreibweise der Sprache	English, Deutsch, français

Normale Objekt-IDs von MongoDB enthalten einen Zeitstempel und einen inkrementellen Zähler.[32] Diese Daten halten wir bei Sprachen, also öffentlichen Stammdaten, die sich über einen langen Zeitraum nicht verändern werden, nicht sinnvoll. Stattdessen haben wir den Alpha-2-Code der Sprache als ID gewählt, der auf die Sprache schließen lässt. Nutzerdokumente referenzieren die Sprache per ID, dies erlaubt es uns, direkt im Nutzerdokument anhand der Sprach-ID zu erkennen, welche Sprachen der Nutzer spricht. Sprachen können sowohl anhand der englischen Schreibweise als auch der nativen Schreibweise gefunden werden. Dies erleichtert Nutzern auch nicht-englischsprachigen Nutzern, ihre Sprache auswählen zu können.

4.3.3 Passwort

Feld	Beschreibung
id	Standardmäßige MongoId
password	Bcrypt Hash bestehend aus Versionsnummer, Komplexität, Salt und Hash
NutzerID	MongoId des zugehörigen Nutzers

[35] [36]

Das Passwort wird nicht im Nutzerdokument gespeichert. Würde das Passwort im Nutzerdokument gespeichert werden, könnte es schon durch kleine Programmierfehler dazu kommen, dass normale Nutzer das gehashte Passwort anderer Nutzer ermitteln könnten. Um dieses Sicherheitsproblem direkt zu eliminieren, wird daher für jedes Passwort ein eigenes, vom Nutzerdokument isoliertes Dokument verwendet. Das Passwort wird durch bcrypt, einem Blowfish-basiertem Hashalgorithmus, auf der Datenbank als Hash mit Salt gespeichert. Die Komplexität des Hashes ist durch uns wählbar. Bei der Wahl der Komplexität ist die Sicherheit gegen Rechengeschwindigkeit abzuwägen; eine höhere Komplexität erhöht die benötigte Zeit pro Versuch eines Angreifers, das Passwort zu knacken, erhöht gleichzeitig aber auch die Zeit, die unser Server benötigt, um ein neues Passwort zu generieren oder den Anmeldeversuch eines ehrlichen Nutzers zu bestätigen. Eine zu hohe

Komplexität kann daher unseren Server stark verlangsamen und macht Angriffszenarien per (D)DOS gefährlicher, da gezielte Anmeldeversuche viel Last auf dem Server erzeugen. Um die Wahrscheinlichkeit von Glückstreffern bei Angriffen zu verringern, verlangen wir außerdem, dass das Passwort mindestens aus 8 Zeichen, darunter 1 Großbuchstabe, 1 Kleinbuchstabe und 1 Ziffer erstellt wird. Für mehr Varianz in den Passwörtern sind zudem einige Sonderzeichen erlaubt. Durch gewählte Restriktionen besteht eine 1:1-Relation zwischen Passwörtern und Nutzerkonten.

4.3.4 Like

Feld	Beschreibung
id	Standardmäßige MongoId
Sender	NutzerID der Person, die den Like/Dislike versendet.
Empfänger	NutzerID der Person, die den Like/Dislike empfängt.
Status	Gibt an, ob es sich um einen Like oder Dislike handelt.

[35] [36]

Der Name "Like"für diese Datenstruktur ist irreführend, da in der Kollektion sowohl Likes als auch Dislikes (angegeben durch den Status) gespeichert werden. Immer wenn ein Nutzer bei der Kontaktsuche angibt, ob er mit einer Person Kontakt aufnehmen oder diesen vermeiden will, wird ein Dokument angelegt. Wenn der Nutzer in Kontakt treten will, wird zusätzlich geprüft, ob bereits ein Datensatz existiert, bei dem Sender und Empfänger vertauscht sind - ob sich also die Nutzer gegenseitig einen Like gegeben haben. Sollte das der Fall sein, werden beide Datensätze gelöscht und die Nutzer zur Freundesliste des jeweils anderen hinzugefügt. in dem Fall können die Nutzer miteinander kommunizieren.

4.3.5 Chat

Feld	Beschreibung
id	Standardmäßige MongoId
Teilnehmer	Array von NutzerIDs, die dem Chat beiwohnen. Aktuell maximal 2.
Nachrichten	Array von Nachrichten, die die Nutzer untereinander ausgetauscht haben.

[35] [36]

Wenn zwei Nutzer sich befreunden, wird ein Chat zwischen diesen erstellt. Wenn ein Nutzer die Freundschaft beendet, verlässt dieser Nutzer gleichzeitig den Chat. Mit der gewählten Struktur sind auch Gruppenchats ohne Änderung der Datenbank möglich, für die Projektanforderungen steht diese Funktion jedoch nicht im Fokus. Um Ressourcen zu sparen, wird bei der Standard-Datenbankabfrage nur die neueste Nachricht geladen. Diese Abfrage dient für Vorschaubilder des Chats in der Kontaktliste. Außerdem ist es möglich, durch Seitennummerierung (Pagination) je Abfrage 20 Nachrichten zu erfragen. Diese Methoden verringern die Serverlast, meist reicht es Nutzern, die letzten 20 Nachrichten

eines Chats zu lesen. Nachrichten sind eingebettete Dokumente eines Chats. Sie weisen folgende Datenstruktur auf:

Nachricht

Feld	Beschreibung
id	Standardmäßige MongoId
Inhalt	Text der Nachricht
Autor	NutzerID des Verfassers der Nachricht

[35] [36]

4.4 Database-as-a-Service

Um eine Datenband selbst zu betreiben fehlen uns Kapazitäten und eine Infrastruktur. Wir sind daher auf eine Database-as-a-Service-Lösung (DBaaS) angewiesen. MongoDB Inc. bietet mit MongoDB Atlas eine Database-as-a-Service Lösung an, die Flexibel auf die Größe und Auslastung des Projektes angepasst werden kann. Dazu gibt es verschiedene Datenbankstufen, die mit höheren Kosten mehr Rechenleistung und weitere Funktionen erhält. Zwischen den Stufen kann flexibel gewechselt werden, um den aktuellen Anforderungen gerecht zu werden. Kostenpflichtige Stufen bieten die Möglichkeit an, Backups einzurichten, ab Stufe M10 stehen Tools zur Verfügung, die Metriken in Echtzeit anzeigen, automatisch archivieren, Empfehlungen zur Leistungsoptimierung erstellen und langsame Datenbankabfragen zur Diagnose anzeigen. In der Entwicklungsphase haben wir uns für die kostenlose Stufe entschieden, da die Funktionen und Leistung dafür ausreichen. Sollte das Produkt auf den Markt gehen, werden wir die kostengünstigste Stufe wählen, um die Option von Backups zu erhalten. Wenn das Projekt erfolgreich ist und wir viele Nutzer anziehen, wird flexibel, abhängig von benötigter Leistung, eine höhere Stufe gewählt. Sowohl die Produktions-, als auch die Entwicklungsumgebung werden als eigene Datenbanken von MongoDB Atlas gehosted. Dies verringert das Risiko von Code, der auf der lokalen Maschine funktioniert, aber auf der Produktionsumgebung Fehler wirft. Durch die Nutzung gleicher Werkzeuge und gleicher Technologie wird die Werkzeuglücke verringert und dementsprechend die Dev-Prod-Vergleichbarkeit erhöht. [39]

Diese lassen sich schnell einrichten, fallen selten aus und werden automatisch mit Updates versorgt. Es stehen Reportingtools zur Verfügung, die das Auswerten der Datenbank erleichern. DBaaS spart viel administrativen Aufwand und verringert damit die Kosten.

5 Avatarbilder

Statt Bilddateien für Avatare direkt auf der Datenbank zu speichern, speichern wir nur URIs zu den Bildern auf unserer Datenbank. Die Bilder selbst werden auf AWS S3 gehosted, einem Speichersystem, welches für BLOB-Dateien optimiert ist. Dies nimmt der

Datenbank Last ab und erhöht die Anfragegeschwindigkeit bei Lese- und Schreibzugriffen des Avatarbildes. Der S3-Speicher wurde so eingerichtet, dass das Backend Zugriffsrechte zum erstellen und löschen von Dateien hat. Es wurde dabei mit Minimalprinzip vorgegangen, damit ein möglicher Angriff auf unser System die geringsten Schäden im Erfolgsfall verursacht, hat unser Backend nur die nötigsten Zugriffsberechtigungen auf S3. Die Datenbank wird mithilfe von GraphQL angesprochen, für S3 hat sich diese Lösung jedoch nicht angeboten. Für das Hochladen von Profilbildern wurde eine weitere Route im Backend erstellt, bei der mit den npm-Packeten Multer und Multer-S3 kontrolliert wird, ob es sich bei der vom Nutzer hochgeladenen Datei um eine Bilddatei handelt und ob diese eine bestimmte Bildgröße nicht übersteigt.

6 Fazit

Mit den gewählten Kollektionen und der Wahl von speziellen Hosts sind wir in der Lage, Nutzern eine Datenbank zu geben, die eine hohe Erreichbarkeit aufweist, leicht zu skalieren ist, personenbezogene Daten geheim hält und ein Maß an Datensicherheit bietet, welches der größe des Projektes entspricht. Durch eingebettete Dokumente, Denormalisierung und Seitennummerierung wurden Optimierungen durchgeführt, die als Ziel haben, möglichst schnell Antworten auf Leseanfragen zu bieten.

7 Benutzeroberfläche

Nachdem die Projektanforderungen definiert waren, wurden verschiedene Optionen für die Entwicklung der Benutzeroberfläche bewertet.

Folgende JavaScript Frameworks wurden berücksichtigt: Angular, Vue und React.

Es war möglich das Projekt auch mit Angular durchgeführt werden können, mit dem Unterschied, dass die Entwicklung mit Angular mehr Zeit und Lernaufwand gekostet hätte, außerdem ist das Projekt nicht komplex genug, um das Angular-Ökosystem zu benötigen.

7.1 Überlegungen bei der Verwendung von React

Die Entscheidung zwischen Vue und React wurde unter anderem wegen des Unternehmens hinter React getroffen, zwar Facebook. Auch für persönliche Vorlieben.

7.1.1 Vorteile

Deklarativ

Mit React ist es leicht, interaktive Benutzeroberflächen zu erstellen. Man kann einfache Ansichten für jeden Zustand der Anwendung. React aktualisiert und rendert effizient genau die richtigen Komponenten, wenn sich deren Daten ändern. Durch deklarative Ansichten wird der Code vorhersehbarer und einfacher zu debuggen.

Komponentenbasiert

Gekapselte Komponenten, die ihren eigenen Zustand verwalten. Da die Komponentenlogik in JavaScript geschrieben wird, kann man problemlos umfangreiche Daten durch die Anwendung leiten und den Zustand aus dem DOM heraushalten.

Große Entwickler-Community

React besteht aus rund 56.162 professionellen Entwicklern auf der ganzen Welt.

Laut einer StackOverFlow Umfrage hat React.js im Jahr 2021 jQuery als das am häufigsten verwendete Web-Framework überholt. [1]

[1] Vgl. [6]

7.1.2 Nachteile

Bei der Auswahl des Frameworks wollten wir so unvoreingenommen wie möglich sein, daher listen wir einige Aspekte auf, die bei Projekten mit React zu beachten sind.

JSX

Während dies für einige Entwickler ein Nachteil sein könnte, ist es wichtig zu beachten, dass JSX auch seine Vorteile hat und hilft, den Code vor Injektionen zu schützen.

Ein hohes Entwicklungstempo

Entwickler, die das Entwicklungstempo als Nachteil sehen, würden argumentieren, dass sie die Arbeit mit React ständig neu erlernen müssen und es schwierig ist, damit Schritt zu halten.

Es ist wichtig festzustellen, dass neue Entwicklungen des Frameworks verbessern und dazu beitragen, dass er ein höheres Leistungsniveau erreicht.

Eine zu leichte Dokumentation

Aufgrund der rasanten Entwicklung ist die Dokumentation in Bezug auf die neuesten Aktualisierungen und Änderungen oft spärlich. [2]

7.2 React Hooks

Beginnend mit 16.8.0, enthält React eine stabile Implementierung von React Hooks. Ab dieser Version wird empfohlen, keine Klassen mehr für die Erstellung von Komponenten zu verwenden.

7.2.1 useState

Der Hook useState gibt uns die Möglichkeit, den Zustand unserer Anwendung zu verwalten. Sie besteht aus mindestens einen Wert und einer Funktion, die die besagte Variable aktualisiert. Der Wert bei der Definition kann ein Zahl, ein String, ein Array oder sogar ein Objekt sein. Darüber hinaus kann bei der Definition von useState ein Anfangswert festgelegt werden.

```
const [wert, definiereWert] = useState(Anfangswert);
```

In unserem Projekt haben wir useState verwendet, um alle damit verbundenen Informationen zu manipulieren und dem Benutzer anzuzeigen.

```
const [profile, setProfile] = useState(state)
```

[2] Vgl. [4]

Die Variable state enthält die Daten des angemeldeten Benutzers. Diese Daten sind die Antwort auf die an den Server gesendete Anfrage. Hier sind einige Beispieldaten für die Variable state.

```
name: "gilo2754"
aboutMe: "I like video code video games too"
avatar:
    "https://lol-friendfinder-dev.s3.eu-central-1.amazonaws.com/avatars/a028fd00-e767-
dateOfBirth: "1990-06-01T00:00:00.000Z"
gender: "male"
ingameRole: (2) ["Fill", "Bot"]
languages: (2) ["bm", "de"]
friends:(5) ["60eb61b33a2481451c1cd7ac", "60eb61b33a2481451c1cd7b0",
    "60eb61b33a2481451c1cd7b1", "60eb61b33a2481451c1cd7b4",
    "60eb61b33a2481451c1cd7b7"]
blocked: (1) ["60eea5eed7e23b5ef08f68af"]
```

In späteren Kapiteln wird näher erläutert, wie diese Informationen, die wir durch einen einzigen Aufruf an den Server erhalten, in den Chat- und Benutzerkomponenten genutzt werden können.

Wie man sieht, gibt es innerhalb des states nicht nur einen Wert, sondern mehrere. Es handelt sich eigentlich um ein Objekt. Objekte sind dasselbe wie Variablen in JavaScript, der einzige Unterschied ist, dass ein Objekt mehrere Werte in Form von Eigenschaften und Methoden enthält. Objekte können andere Objekte, Strings, Arrays, Zahlen und so weiter enthalten.

7.2.2 useEffect

useEffect ermöglicht es uns, verschiedene Arten von Effekten zu erzeugen, nachdem eine Komponente gerendert wurde.

Der Hook useEffect entspricht einer Kombination aus componentDidMount, componentDidUpdate und componentWillUnmount.

In unserem Projekt manipulieren wir die Anzeige der Daten abhängig der Änderungen des lokalen Zustands.
Im Zustand „profile“ werden die Änderungen, die der Benutzer an seinen Daten vornimmt, bevor sie in der Datenbank gespeichert werden.
Der globale Zustand „state“ ist in der Komponente App enthalten. In diesem wird die Antwort von der Datenbankenabfrage gespeichert.
Wir definieren den Zustand als Abhängigkeit von useEffect, so dass er bei jeder Änderung in der Benutzeroberfläche aktualisiert wird.

```
useEffect(() => {
     setProfile(state)
  }, [state])
```

Es ist möglich, mehrere useEffects in unserem Code zu haben und mit ihrer jeweiligen Abhängigkeiten separat definieren. Wenn wir wollen, dass der Code in unserem useEffect nur einmal nach dem ersten Rendering ausgeführt wird, definieren wir ein leeres Array als Abhängigkeit.

7.2.3 useContext

useContext hat es uns ermöglicht, Benutzerdaten auf einfache Weise in den Komponenten zu teilen, in denen diese Daten benötigt werden. In frühen Versionen der Anwendung wurde für den gleichen Zweck props verwendet. Unserer Erfahrung nach ist useContext eine elegantere und sauberere Lösung für da gleiche Ziel.

useContext Provider

```
<MyContext.Provider value={/* irgendein Wert */}>
```

Jedes Context-Objekt kommt mit einer Provider React Komponente, welche konsumierenden Komponenten erlaubt, die Veränderungen von Context zu abonnieren. [3]

[3] Vgl. u.a. [**?**]

In unserem Projekt haben wir den Provider in der Komponente App folgendermaße definiert.

```
import { React, useState, useEffect, createContext } from "react"

export const GlobalContext = createContext()
export default function App() {
    const [token, setToken] = useState()
    const [state, setState] = useState(null)

    ...

    return (
        <GlobalContext.Provider value={{ token, setToken, state, setState,
            refetch }}>
          <>
            <MyNavbar />

            <Switch>
              <Route exact path="/" component={Home} />
              <Route path="/login" component={() => <Login />} />
              <Route path="/signup" component={SignUp} />
              <Route exact path="/users" component={() => <Users />} />
              <Route exact path="/profile" component={() => <Profile />} />
              <Route exact path="/chat" component={() => <Chat />} />
              <Route exact path="/ChatMessage" component={() =>
                  <ChatMessage />} />

              <Route component={NotFound} />
            </Switch>
          </>
        </GlobalContext.Provider>
      )
    }
//Code-Auszug in frontend/src/App.js
```

Mit dem obigen Code sind wir bereit, die Werte innerhalb von value in jeder Komponente innerhalb des Contexts zu verwenden.

In unserem Fall können die Werte token und state gelesen werden. Darüber hinaus sind die Funktionen setToken, setState und refetch ebenfalls verfügbar.

useContext Consumer

Nachfolgend zeigen wir, wie GlobalContext in der Komponente Chat.js verwendet wurde, um die Freunde des angemeldeten Benutzers zu ermitteln.

Die Kennungen der Freunde waren notwendig, um die Kommunikation zu ermöglichen und die zwischen den Freunden ausgetauschten Nachrichten zu laden.

```
        import { useContext, useEffect, useState, React } from "react"
        import { GlobalContext } from "../App"
        //mithilfe von Destrukturierende Zuweisung wurden neue Variablen und
            Funktionen definiert
        const { token, state, setState, refetch } = useContext(GlobalContext)

        ...

        return !token ? (

        ...

            <div className="user-many">
            {state?.friends &&
              state?.friends?.map((item, index) => {
                return (

//Die Komponente FriendList gibt den Namen und den Avatar des Benutzers auf
    der Grundlage der eingegebenen ID zurueck.
                  <FriendList
                    setUserID={setUserID}
                    setUserNameChat={setUserNameChat}
                    setChatAvatar={setChatAvatar}
                    userId={item.user}
                    searchUser={searchUser}
                    setSearchUser={setSearchUser}
                    key={index + 1}
                  />
                )
              })}
          </div>
        )
      ...
//Code-Auszug in frontend/src/Chat.js
```

7.2.4 Alternative zu useContext

Bei einer früheren Version der kürzlich erläuterten Implementierung haben wir „props“ verwendet. Dadurch haben wir die gemeinsame Nutzung von Daten zwischen Komponenten ermöglicht. Wir beschlossen, diese Idee zu ändern, da der Code schwieriger zu pflegen war.

8 Serveranfragen

8.1 GraphQL

GraphQL ist eine Abfragesprache und Server-Laufzeitumgebung für APIs. Ihre Aufgabe ist es, genau die Daten zu liefern, die anfordert werden, und nicht mehr.
Mit GraphQL sind APIs schnell, flexibel und einfach für Entwickler.

Laut dem 2020 State of the API Report von Postman.com steht GrapQL an fünfter Stelle der spannendsten Technologien für 2021. Vgl. u.a. [10]

Im Hinblick auf die Art und Weise, wie Abfragen an den Server mithilfe von GraphQL behandelt werden können, sind folgende Aspekte zu beachten.

Vorteile

- GraphQL-Aufrufe werden in einem einzigen Round Trip gehandhabt. Wir bekommen genau die Daten, die angefragt haben (kein Over-Fetching).
- Stark definierte Datentypen verringern das Risiko einer Fehlkommunikation zwischen Client und Server.
- GraphQL ist introspektiv. So können wir eine Liste der verfügbaren Datentypen anfordern. Dies ist ideal für automatisch erstellte Dokumente.
- Eine Anwendungs-API kann sich mit GraphQL weiterentwickeln, ohne dass bestehende Anfragen beeinträchtigt werden.
- GraphQL schreibt keine spezifische Anwendungsarchitektur vor. Es kann auf einer vorhandenen REST-API installiert und mit aktuellen API-Management-Tools verwendet werden.
- Als Alternative zu REST ermöglicht GraphQL Entwicklern die Erstellung von Abfragen zur Extraktion von Daten aus mehreren Quellen mit einer einzigen API-Abfrage.

Nachteile

- Für Entwickler, die sich bereits mit REST-APIs auskennen, bedeutet GraphQL weiteren Lernaufwand.
- Mit GraphQL verschiebt sich die Funktionalität von Datenabfragen zur Serverseite, was zusätzliche Komplexität für Serverentwickler bedeutet.

4

[4] Vgl. u.a. [9]

8.1.1 GraphQL Playground

Mit GraphQL Playground haben wir die Möglichkeit, alle Abfragen und Mutationen zu testen. Wir erhalten Zugriff auf relevante Informationen wie verfügbare Felder und deren Datentyp. Diese Informationen werden aktualisiert, wenn der Servercode geändert wurde. Dadurch wurde eine aktuelle API-Dokumentation gewährleistet. Für unser Projekt war es sehr praktisch und hat die Kommunikation als Entwickler effizienter gemacht.

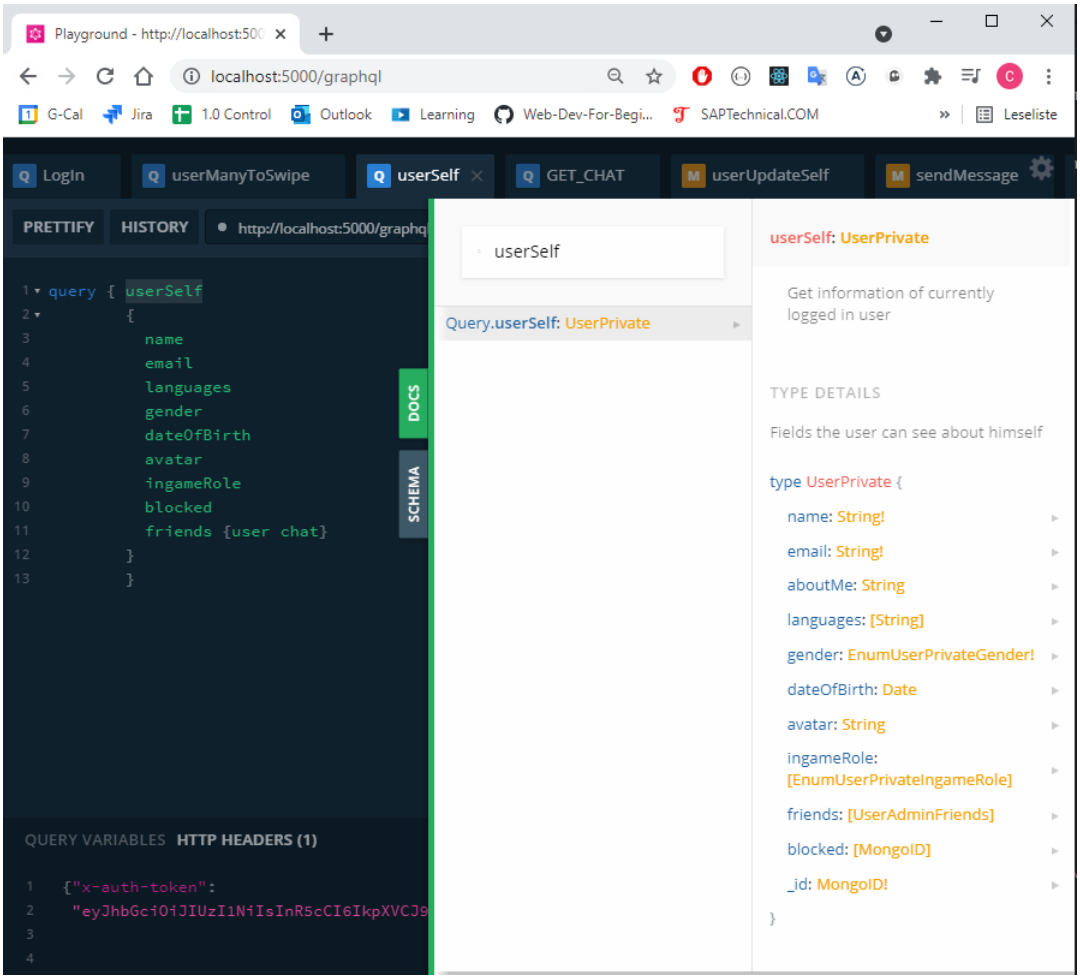


Abbildung Unterunterabschnitt 8.1.1: Genau derselbe Code wird für die Abfrage von Benutzerinformationen später verwendet.

8.2 Implementierung der Serveranfragen mit GraphQL

Alle Abfragen und Mutationen wurden in einem separaten Ordner gesammelt. Damit soll eine saubere Struktur des Codes gewährleistet werden. Diese wurden für die spätere Verwendung in den React-Komponenten exportiert.

Mithilfe der Hooks useQuery bzw. useMutation von Apollo Client wurden die Lese- und Schreibabfragen durchgeführt.

Apollo Client

Warum haben wir uns für Apollo entschieden? Was ist ApolloClient? TODO

8.2.1 Leseabfrage

Nachdem eine Abfrage exportiert wurde, ist sie bereit, in einer React-Komponente importiert und angewendet zu werden.

```
import { GET_MY_INFO } from "./GraphQL/Queries"
import { useQuery } from "@apollo/client"

const { loading, error, data } = useQuery(
GET_MY_INFO,
ContextHeader(token),
)
//Code-Auszug in frontend/src/App.js
```

Die Konstante ContextHeader enthält das Token in der Struktur, die erforderlich ist, um die Abfrage nur dann stellen zu können, wenn der Benutzer dazu berechtigt ist.
Sollte das Token einen undefinierten Wert, null oder ungültig enthalten, wird der Server ein Fehler zurückgegeben.
Der useQuery Hook liefert ein Ergebnisobjekt, welches eine der folgenden Optionen zurückgibt.

loading:
Ein boolescher Wert, der angibt, ob die Abfrage in Bearbeitung ist. Wenn loading wahr ist, ist die Anfrage noch nicht abgeschlossen. Typischerweise kann diese Information verwendet werden, um einen Lade-Spinner anzuzeigen.

error:
Ein Laufzeitfehler mit den Eigenschaften von GraphQL Errors und network Error. Dieses enthält Informationen darüber, was bei der Abfrage fehlgeschlagen ist.

date:
Ein Objekt, das das Ergebnis der GraphQL-Abfrage enthält.
Es enthält die tatsächlichen Daten vom Server.

8.2.2 Mutationen

8.2.3 Subscriptions

TODO after receiving feedback about the rest.

8.3 Axios

Zusätzlich zu den GraphQL-Abfragen wurde eine Post-Anfrage mit Axios bereitgestellt. Mit dieser war es möglich, Bilder auf die S3 Speicher von AWS hochzuladen.

> Axios ist ein Promise-basierter HTTP-Client für node.js und den browser. Auf der Server-Seite wir das modul http verwendet, während im Browser XML-HttpRequests (ajax) ausgeführt werden.
>
> [5]

```
function disableBtn() {
    const uploadBtn = document.getElementById("uploadBtn")
    uploadBtn.disabled = true
    uploadBtn.style.background = "#000000"
  }

  function fileUploadHandler() {
    errored ?
     disableBtn()
     /*
    Tritt ein Fehler auf, wird die Abfrage nicht gesendet und der Knopf
        deaktiviert
    Ein Fehler tritt auf, wenn die Datei zu gross ist oder oder wenn sie
        ein ungueltiges Format hat
    */
    :
    console.log("uploading pic...", file?.name)
    const fd = new FormData()
    /* avatar ist der Name der Datei, die hochgeladen wird
      file ist der zu sendende Wert */
    fd.append("avatar", file)

    axios
      .post(urlAvatar, fd, {
        headers: {
          "x-auth-token": TOKEN,
        },
      })
      .then((res) => {
        setState((state) => ({ ...state, avatar: res?.data?.location }))
        //In location finden wir die URL fuer das gerade hochgeladende Bild
```

[5] Vgl. u.a. [11]

```
        })
    }
```

Das Format der hochzuladenden Dateien wurde limitiert, damit nur zulässige Dateien an den Server gesendet werden.

```
const admittedImageFormats = ["png", "jpg", "jpeg"]
```

Die Größe der hochzuladenden Datei wurde um 1 MB abgegrenzt. Durch die Eigenschaft „size“ der ausgewählten Datei konnten wir auf die Größe der Datei zugreifen.

```
const imageSize = e.target.files[0].size
```

9 Qualitätssicherung

Software Testing

Obwohl unser Projekt relativ klein ist, wollten wir, wo immer möglich, automatisierte Testfälle einbeziehen.

Für das Frontend haben wir End-to-End Testfälle mit Cypress geschrieben.

So können wir in Sekundenschnelle feststellen, ob etwas in unserer Anwendung defekt ist.

Ein mögliches Szenario ist, dass eine Komponente, die in anderen Komponenten verwendet wird, geändert wurde, ohne zu berücksichtigen, dass sie in einer anderen Komponente für einen anderen Zweck verwendet wurde.

Das ist der Fall bei der Komponente AvatarImage. Dies ist eine Funktionskomponente, die 3 Parameter erhält: Größe des Bildes, Bild-URL und Benutzername.

Zu Beginn des Projekts wurde nicht daran gedacht, die Größe des Bildes über einen Parameter dieser Funktion zu steuern. Im Laufe des Projekts wurde uns klar, dass wir die Logik in diesem Element wiederverwenden konnten.

Innerhalb der Komponente wird geprüft, ob eine URL existiert, und wenn ja, wird das mit dem Link verbundene Bild gezeichnet. Falls es keine URL angegeben wurde, werden die ersten beiden Buchstaben des Benutzernamens verwendet, um ein Standardsymbol zu erzeugen.

In der aktuellen Version des Codes wird diese Komponente in vier anderen Komponenten wieder verwendet. Wenn das Projekt weiter wachsen würde, würde auch die Möglichkeit von Fehlern im Code zunehmen. Und es wäre schwieriger, sie zu finden.

Ohne automatisierte Tests müssten Sie normalerweise manuell prüfen, ob die übergeordneten Komponenten noch wie erwartet funktionieren.

```
//in constants.js sind Standard-Testdaten
import * as c from "../constants.js"

const qtyLanguages = 101
const qtyGenders = 8

describe("Check if all profile elements have been rendered", () => {
  it("Login with existing account", () => {
    cy.typeLogin(c.email, c.password)
  })

  it("Username was rendered", () => {
    cy.get('#username').should("be.visible")
  })
```

```
  it("Gender selection was rendered", () => {
    cy.get('#dropdown-gender').should(
      "be.visible")
  })

//Der folgende Test prueft, ob eine Mindestanzahl von Elementen angezeigt wird
it(`Gender options is not empty and >= to ${qtyGenders}`, () => {
  cy.get('#dropdown-gender').click()
  cy.get(`.dropdown-menu > :nth-child(${qtyGenders})`).should(
      "be.visible")
  //close the dropdown
  cy.get('#dropdown-gender').click()

  })
...
})
```

9.1 Die Testfälle für unser Projekt

Nachstehend einer Überblick über die Testfälle bei Cypress.

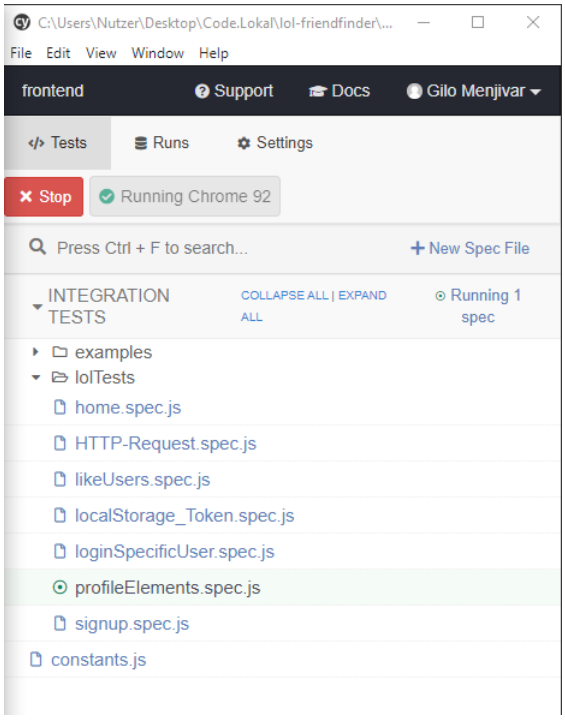


Abbildung Abschnitt 9.1: Testfälle in Cypress

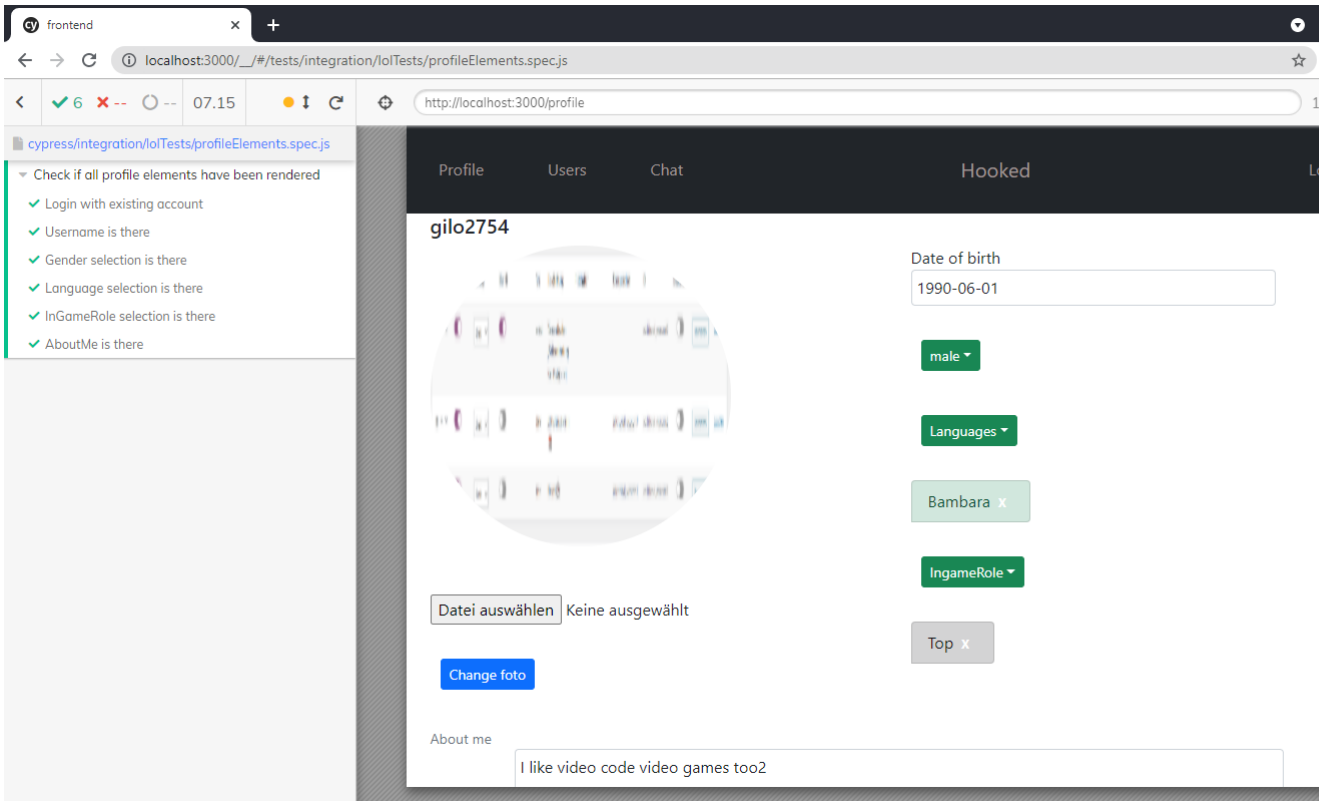


Abbildung Abschnitt 9.1: Grafische Darstellung der verschiedenen Tests in Cypress

home.spec.js

Prüfen Sie, ob die Startseite „Home“ gerendert wurde.

likeUsers.spec.js

Mit diesem Testfall wird überprüft, ob nach der Vergabe von einem „Like“ oder einem „Dislike“ ein anderer Nutzer angezeigt wird.

Testfall localStorage Token

Hier wird der Wert des JSON-Web-Tokens zu verschiedenen Zeitpunkten überprüft. Die Erwartung ist, dass das Token null ist, wenn der Benutzer nicht angemeldet ist. Dieses Token wird in localStorage gespeichert. Es besteht auch die Möglichkeit, dass das Token abgelaufen ist, wodurch alle Anfragen an den Server, die eine Authentifizierung erfordern, unmöglich werden.

profileElements.spec.js

Der Testfall prüft, ob die Elemente und Komponenten der Komponente Profil gerendert wurden und sichtbar sind. Diese Elemente sind userName, Gender und AboutMe. Die Komponenten sind Language und InGameRole. Außerdem wird überprüft, ob Komponenten, ein Element „Dropdown“ enthalten, eine Mindestanzahl von Elementen enthalten, die angezeigt werden müssen.

signUp.spec.js

Dieser Testfall erstellt einen neuen Benutzer mit einer zufälligen E-Mail und einem zufälligen Benutzernamen.

Commands bei Cypress

In den Testfällen gibt es Aktionen, die sich immer wieder wiederholen, zum Beispiel die Anmeldung eines Nutzers. Zu diesem Zweck wurde in Cypress ein wiederverwendbarer Befehl definiert.

```
Cypress.Commands.add("typeLogin", (email, password) => {
    cy.visit("/login")

    cy.get("[id=email-input]").type(email).should("have.value", email)

    cy.get("[id=password-input]").type(password).should("have.value",
        password)

    cy.get("[id=btn-submit]").click()
})
```

Mit anderen Worten, es handelt sich um eine Funktion, die zwei Parameter, E-Mail und Passwort, erhält. Beide Parameter werden in die entsprechenden Eingabefelder eingetragen, abschließend wird die Eingabetaste gedrückt. Diese Funktion kann in jedem anderen Testfall aufgerufen werden, ohne dass sie importiert werden muss.

10 Glossar

Hooks:

...

Framework:

...

JSX:
Es heißt JSX und ist eine Syntaxerweiterung für JavaScript. Wir empfehlen, sie mit React zu verwenden, um zu beschreiben, wie die Benutzeroberfläche aussehen soll. JSX erinnert vielleicht an eine Template-Sprache, aber es verfügt über die volle Leistungsfähigkeit von JavaScript. JSX erzeugt React-Ëlemente"

Over-Fetching:
Empfang von überschüssigen Daten durch eine Abfrage.

Web Token:
JSON-Web-Tokens sind eine dem Industriestandard RFC 7519 entsprechende Methode zur sicheren Darstellung von Forderungen zwischen zwei Parteien.

undefined:
Eine Variable, der kein Wert zugewiesen wurde oder die überhaupt nicht deklariert wurde (nicht deklariert, existiert nicht), ist undefiniert. Eine Methode oder Anweisung gibt auch undefiniert zurück, wenn der ausgewerteten Variablen kein Wert zugewiesen wurde. Eine Funktion gibt undefiniert zurück, wenn kein Wert zurückgegeben wurde.

FormData:
Die FormData-Schnittstelle bietet eine einfache Möglichkeit, eine Reihe von Schlüssel/Wert-Paaren zu erstellen, die die Felder eines Formulars und ihre Werte darstellen und mit der XMLHttpRequest.send()-Methode einfach gesendet werden können.

componentDidMount:
componentDidMount() wird unmittelbar nachdem eine Komponente (Einfügen in den Baum) montiert. Die Initialisierung, die DOM-Knoten erfordert, sollte hier erfolgen. Wenn Daten von einem Endpunkt geladen werden müssen, ist dies ein guter Ort, um die Netzwerkanfrage zu instanziieren. Diese Methode ist ein guter Ort, um Abonnements einzurichten. Wenn das der Fall ist, sollte es nicht vergessen werden, sich in componentWillUnmount() abzumelden.

componentDidUpdate:
componentDidUpdate() wird unmittelbar nach der Aktualisierung aufgerufen. Diese Methode wird beim ersten Rendering nicht aufgerufen.

componentWillUnmount:
componentWillUnmount() wird aufgerufen, unmittelbar bevor eine Komponente demontiert und zerstört wird. Man sollte in dieser Methode alle notwendigen Bereinigungen durchgeführen, wie z. B. das Ungültigmachen von Zeitgebern, das Abbrechen von Netzwerkanforderungen oder das Aufräumen von Abonnements, die in componentDidMount() erstellt wurden.

Destrukturierende Zuweisung:
Die destrukturierende Zuweisung ermöglicht es, Daten aus Arrays oder Objekten zu extrahieren, und zwar mit Hilfe einer Syntax, die der Konstruktion von Array- und Objekt-Literalen nachempfunden ist. https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

Akronyme:
AWS
Amazon Web Services
DOM
Document Object Model
API
Application Programming Interface
AJAX
Asynchronous JavaScript And XML
AWS
Amazon Web Services
DOM
Document Object Model
API
Application Programming Interface
AJAX
Asynchronous JavaScript And XML

ORDBMS	ObjektRelationales DatenBankMangagementSystem
BSD-Lizenz	eine Open-Source (quelloffene) Lizenz
ACID	Atomicity (Atomarität), Consistency (Konsistenz), Isolation, Durability (Beständigkeit). Diese Eigenschaften sind bei Transaktionen in Datenbankmanagementsystemen häufig erwünscht
BASE	Basically Available (meist erreichbar), Soft state (ohne festen Zustand), Eventual consistency (eventuell konsistent). Gegenteil von ACID (Säure/Base), meist in NoSQL-Datenbanken erwünschte Eigenschaften
BLOB	Binary Large Object; Große binäre Dateien, unter anderem Bild- und Audiodateien.
URI	Unified Resource Identifier; Webstandard, mit dem Ressourcen im Internet identifiziert werden.

11 Zusammenfassung und Ausblick

TEXT MUSS KONTROLIERT WERDEN...
Es hat sich gezeigt, dass moderne Entwicklungswerkzeuge für JavaScript auch ohne umfassende Kenntnisse der Softwareentwicklung zugänglich sind.

Das Ziel, eine echte Plattform zu schaffen, wurde im Zeitraum von Mai 2021 bis Mitte August 2021 erreicht.

Das heißt, ein Team von zwei Studenten mit grundlegenden Programmierkenntnissen war in der Lage, eine funktionelle Plattform zu schaffen, die die Registrierung, die Anmeldung der Benutzer, die Verwaltung von persönlichen Daten, die Interaktion mit anderen Benutzern auf der Grundlage ihrer Präferenzen umfasst und einen auf Textnachrichten basierenden Kommunikationskanal.

12 Quellenverzeichnis

13 Quellenverzeichnis

13.1 Literatur

[1] Stickel-Wolf, Christine; Wolf, Joachim (2011): Wissenschaftliches Lernen und Lerntechniken. Erfolgreich studieren—gewusst wie!. Wiesbaden: Gabler.

[2] vgl. MongoDB (2019). Multidocument ACID Transactions (S.4/24). URL: https://webassets.mongodb.com/finalmongodb_multidocument_acid_trnasactions.pdf [30.09.2021]

[3] Truică; Boicea; Rădulescu (2013):Asynchronous Replication in Microsoft SQL Server, PostgreSQL and MySQL (S.51, Z.5f.). URL: https://www.researchgate.net/profile/Ciprian-Octavian-Truica/publication/264416935_Asynchronous_Replication_in_Microsoft_SQL_Server_PostgreSQL_and_MySQL/links/53dbe6160cf216e4210c0375/Asynchronous-Replication-in-Microsoft-SQL-Server-PostgreSQL-and-MySQL.pdf [30.09.2021]

13.2 Internetquellen

[1] Bertelsmeier, Birgit (o. J.): Tipps zum Schreiben einer Abschlussarbeit. Fachhochschule Köln-Campus Gummersbach, Institut für Informatik. http://lwibs01.gm.fh-koeln.de/blogs/bertelsmeier/files/2008/05/abschlussarbeitsbetreuung.pdf (29.10.2013).

[2] Halfmann, Marion; Rühmann, Hans (2008): Merkblatt zur Anfertigung von Projekt-, Bachelor-, Master- und Diplomarbeiten der Fakultät 10. Fachhochschule Köln-Campus Gummersbach.http://www.f10.fh-koeln.de/imperia/md/content/pdfs/studium/tipps/anleitungda270108.pdf (29.10.2013).

[3] Offizielle Vue-Website: Vergleich zwischen Vue, React und Angular. https://vuejs.org/v2/guide/comparison.html#Preact-and-Other-React-Like-Libraries (unbekannte Veröffentlichung).

[4] Offizielle React-Website: React Hooks. https://reactjs.org/docs/hooks-faq.html#which-versions-of-react-include-hooks

[5] Offizielle Website Apollo für React. https://www.apollographql.com/docs/react/

[6] StackOverFlow: Developer Survey 2021. https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks

[7] Elad Elrom: React and Libraries. https://link.springer.com/content/pdf/10.1007%2F978-1-4842-6696-0.pdf

[8] Stoyan Stefanov: Durchstarten mit React. https://content-select.com/media/moz_viewer/5d5fc360-478c-4038-ac17-246bb0dd2d03/language:de

[9] Red Hat: Was ist GraphQL? https://www.redhat.com/de/topics/api/what-is-graphql

[10] Postman: 2020 State of the API Report https://www.postman.com/state-of-api/the-future-of-apis/#the-future-of-apis

[11] Offizielle Website Axios https://axios-http.com/

[12] vgl. Britannica: Michael Stonebraker | Biography, MIT, Facts, & Turing Award, URL: https://www.britannica.com/biography/Michael-Stonebraker#ref1245757 [30.09.2021]

[13] PostgreSQL: The world's most advanced open source database, URL: https://www.postgresql.org/ [30.09.2021]

[14] vgl. PostgreSQL: License, URL: https://www.postgresql.org/about/licence/ [30.09.2021]

[15] vgl. DB Engines: Microsoft SQL Server vs. MySQL vs. Oracle vs. PostgreSQL Comparison, URL: https://db-engines.com/en/system/PostgreSQL%3BMicrosoft+SQL+Server%3BMySQL%3BOracle [30.09.2021]

[16] vgl. PostgreSQL: Datentyen, URL: https://www.postgresql.org/docs/9.5/datatype.html [30.09.2021]

[17] vgl. Google Cloud: How to set up PostgreSQL for high availability and replication with Hot Standby, URL:https://cloud.google.com/community/tutorials/setting-up-postgres-hot-standby [30.09.2021]

[18] vgl. PostgreSQL: Multimaster - PostgreSQL wiki, URL:https://wiki.postgresql.org/wiki/Multimaster [30.09.2021]

[19] vgl. The Register: MongoDB daddy: My baby beats Google BigTable, URL: https://www.theregister.com/2011/05/25/the_once_and_future_mongodb/ [30.09.2021]

[20] MongoDB: Die beliebteste Datenbank für moderne Apps, URL: https://www.mongodb.com/de-de [30.09.2021]

[21] vgl. MongoDB: BSON Types — MongoDB Manual, URL: https://docs.mongodb.com/manual/reference/bson-types/ [30.09.2021]

[22] vgl. MongoDB: Sharding, URL: https://docs.mongodb.com/manual/sharding/ [30.09.2021]

[23] vgl. MongoDB: Replizierung, URL: https://docs.mongodb.com/manual/replication/ [30.09.2021]

[24] vgl. MongoDB: Häufige Fragen | MongoDB: Wie sorgt MongoDB für Konsistenz, URL: https://docs.mongodb.com/manual/core/read-isolation-consistency-recency/ [30.09.2021]

[25] vgl. MongoDB: Software Lifecycle Schedules, URL: https://www.mongodb.com/support-policy/lifecycles [30.09.2021]

[26] vgl. MongoDB: How to Scale MongoDB: What is vertical scaling in MongoDB?, URL: https://www.mongodb.com/basics/scaling [30.09.2021]

[27] vgl. MongoDB: Replication — MongoDB Manual, URL: https://docs.mongodb.com/manual/replication/ [30.09.2021]

[28] vgl. MongoDB: Replica Set Elections — MongoDB Manual, URL: https://docs.mongodb.com/manual/core/replica-set-elections/ [30.09.2021]

[29] vgl. MongoDB: Hidden Replica Set Members — MongoDB Manual, URL: https://docs.mongodb.com/manual/core/replica-set-hidden-member/#std-label-replica-set-hidden-members [30.09.2021]

[30] vgl. MongoDB: Delayed Replica Set Members — MongoDB Manual, URL: https://docs.mongodb.com/manual/core/replica-set-delayed-member [30.09.2021]

[31] vgl. MongoDB: Active-Active Application Architectures with MongoDB, URL: https://www.mongodb.com/developer/article/active-active-application-architectures/ [30.09.2021]

[32] vgl. MongoDB: ObjectId — MongoDB Manual, URL: https://docs.mongodb.com/manual/reference/method/ObjectId/ [30.09.2021]

[33] vgl. DB Engines: Berechnungssmethode <sic!> der Wertungen im DB-Engines Rankinghttps://db-engines.com/de/ranking_definition [30.09.2021]

[34] vgl. DB-Engines: Ranking, URL: https://db-engines.com/de/ranking [30.09.2021]

[35] vgl. MongoDB: MongoDB vs PostgreSQL, URL: https://www.mongodb.com/compare/mongodb-postgresql [30.09.2021]

[36] vgl. DB-Engines: MongoDB vs. PostgreSQL Vergleich, URL: https://db-engines.com/de/system/MongoDB%3BPostgreSQL [30.09.2021]

[37] vgl. Stack Overflow: Stack Overflow Developer Survey 2021, URL: https://insights.stackoverflow.com/survey/2020#technology-databases-all-respondents4 [30.09.2021]

[38] JSON.org: Einführung in JSON, URL: https://www.json.org/json-de.html [30.09.2021]

[39] vgl. 12Factor.net: The Twelve-Factor-App: X.Dev-Prod-Vergleichbarkeit, URL: https://12factor.net/de/ [30.09.2021]

[40] vgl. Microsoft Azure: Was ist Datenbanksharding?, URL: https://azure.microsoft.com/de-de/overview/what-is-database-sharding/ [30.09.2021]

[41] vgl. ISO: ISO 639-1:2002 - Codes for the representation of names of languages — Part 1: Alpha-2 code, URL: https://www.iso.org/standard/22109.html [30.09.2021]

Erklärung über die selbständige Abfassung der Arbeit

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht.
Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

_____

(Ort, Datum, Unterschrift)

Hinweise zur obigen *Erklärung*

- Bitte verwenden Sie nur die Erklärung, die Ihnen Ihr **Prüfungsservice** vorgibt. Ansonsten könnte es passieren, dass Ihre Abschlussarbeit nicht angenommen wird. Fragen Sie im Zweifelsfalle bei Ihrem Prüfungsservice nach.
- Sie müssen **alle abzugebende Exemplare** Ihrer Abschlussarbeit unterzeichnen. Sonst wird die Abschlussarbeit nicht akzeptiert.
- Ein **Verstoß** gegen die unterzeichnete *Erklärung* kann u. a. die Aberkennung Ihres akademischen Titels zur Folge haben.