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) Carlo: sind folgende Schritte zu erledigen:

- Konzeption und Entwicklung eines Datenbanks
- Erstellung eines Canvan-Dashboards mit einem Projektmanagementstool für die Verwaltung der Aufgaben
- Entwicklung einer Schnittstelle zur Programmierung von Anwendungen (API)
- Entwicklung einer Benutzerüberfläche
- Einrichtung eines Repository für die Quellcodeverwaltung

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 Champions, 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

Die Plattform sollte einer sichere Ort sein, 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.

Es wird 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.

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.

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. Um eine potenziell große Anzahl an Nutzern in der Zukunft des Projektes verwalten zu können, sollte es angemessene Skalierungsmöglichkeiten geben.

Im Folgenden gehen wir näher auf die Ziele der Datenbank ein und vergleichen die NoSQL Datenbank MongoDB mit PostgreSQL, welche repräsentativ für SQL basierte ORDBMS steht.

4.1.1 Lese- und Schreibgeschwindigkeit

Um eine möglichst gute Benutzererfahrung zu gewährleisten, wird versucht, die Wartezeit beim Laden der Webseite zu verringern.
Eine Möglichkeit dafür ist eine optimistische Benutzeroberfläche. Bei dieser wird davon ausgegangen, dass der angefragte Schreibzugriff erfolgreich ist; dem Nutzer wird bereits visuell der Erfolgsfall angezeigt. Sollte der Schreibzugriff fehlschlagen, wird der Status der Benutzeroberfläche zurückgesetzt und der Nutzer durch eine Fehlermeldung informiert. Im Gegensatz zu einer realistischen Benutzeroberfläche, welche sich erst aktualisiert, wenn die Datenbank antwortet und der Schreibzugriff genehmigt wurde, erhält der Benutzer sofort eine Rückmeldung und muss nicht auf eine Antwort unseres Servers warten.
Anders als Schreibanfragen, welche mit einer Bestätigung oder Ablehung der Anfrage antworten, fordern Leseanfragen Daten an. Wartezeiten bei Leseanfragen können daher nicht maskiert werden.
Um in einem späteren Entwicklungsschritt die Reduzierung der Wartezeiten durch eine optimistische Benutzeroberfläche zu ermöglichen, werden daher langsamere Schreibgeschwindigkeiten in Kauf genommen, wenn sich mit dieser Entscheidung die Lesegeschwindigkeit erhöht.

SQL-Datenbanken wie PostgreSQL liegen meist in einer Normalform vor, um das Aktualisieren und Einfügen von Daten zu beschleunigen und die Konsistenz und Integrität gemäß ACID zu wahren. Neben diesen Vorteilen entstehen jedoch Nachteile, die sich negativ auf die Lesegeschwindigkeit auswirken. Dadurch, dass Daten nicht dupliziert werden, müssen bei Abfragen oft mehrere Tabellen zusammengeführt werden, wodurch Abfra-

gen komplexer werden und die Effizienz von Indexen abnimmt. Beides wirkt sich negativ auf die Lesegeschwindigkeit aus. Um die Lesegeschwindigkeit zu erhöhen kann Denormalisierung verwendet werden, dies benötigt jedoch eine umfassende Umstrukturierung der bestehenden Daten, kann zu Anomalien in Datensätzen führen und benötigt weitere Schritte, um die Veränderung von Daten auf andere Tabellen zu übertragen. Wir halten dies für einen hohen Aufwand, der mit Risiko und hohen Kosten verbunden ist.

MongoDB speichert Dokumente im JSON-Format und ermöglicht es, mehrere Werte für einen Schlüssel in Form eines Arrays zu hinterlegen. Auch ist es möglich, Dokumente in andere Dokumente einzubetten. Dies beschleunigt Leseabfragen, da die angeforderten Daten meist bereits in einem Dokument vorliegen, allerdings verringert sich die Schreibgeschwindigkeit, da Daten meist redundant in mehreren eingebetteten Dokumenten vorliegen und bei einer Änderung an mehreren Stellen überschrieben werden müssen. Anders als PostgreSQL ist MongoDB auf Grundlage dieser Art der Denormalisierung entworfen und ändert bei einem Schreibzugriff automatisch alle Instanzen des gleichen Dokumentes als Teil einer atomaren (Alles-oder-Nichts-)Transaktion. Auch Multi-Dokument-Transaktionen über verschiedene Shards und Replikatgruppen sind optional atomar. [**?**]

4.1.2 Flexible Datenstrukturen

Im Rahmen des MVPs stehen die Projektanforderungen fest, allerdings soll im späteren Verlauf des Projektes auf die Wünsche der Nutzer geachtet werden und entsprechende Anpassungen an der Webseite getätigt werden. Datenstrukturen werden verworfen und angepasst, wenn sich diese nicht als nützlich erweisen. Speziell in der Anfangsphase eines Teilprojektes erlauben flexible Datenstrukturen eine Lösung zu entwickeln, welche mit wenig Zeitaufwand ein akzeptables Ergebnis liefert. Sollte sich das Teilprojekt als erfolgreich erweisen, kann in einem späteren Entwicklungsschritt die Lösung inkrementell perfektioniert werden. Eine Datenbank, die sich flexibel verändern lässt, ermöglicht es, schneller Anpassungen durchzuführen und verkürzt damit die Entwicklungszeit.

Tabellen in PostgreSQL können mit DDL-Befehlen wie ALTER TABLE verändert werden. Spalten können hinzugefügt, entfernt oder verändert werden, durch bestehende Constraints wird das Entfernen oder Ändern spezieller Spalten jedoch in einigen Fällen von der Datenbank verhindert, um die Integrität der Daten zu gewährleisten. Das Ändern des Datentyps einer Spalte ist in der Regel nur möglich, wenn die Datentypen zueinander kompatibel sind (zum Beispiel Integer zu Long) oder die Spalte für alle Datensätze leer ist. Erfahrungsgemäß bereitet das Ändern von bestehenden Spalten oft Probleme oder ist zumindest umständlich, vor allem, wenn andere Tabellen auf die zu ändernde Tabelle referenzieren.

MongoDB speichert Dokumente in Kollektionen, Dokumente der gleichen Kollektion müssen nicht die gleiche Struktur aufweisen, da das Dokument dessen Struktur nach JSON-Spezifikation (durch die Angabe der Schlüssel-Wert-Paare) selbst beinhaltet. Dementsprechend können Dokumente mit neuen, anderen oder fehlenden Attributen direkt zu bestehenden Kollektionen hinzugefügt werden. Dies erlaubt es, bei der Weiterentwicklung von Kollektionen direkt die neuen Dokumente in bestehende Kollektionen einzufügen, ohne bestehende Dokumente verändern zu müssen. Es wird Zeit gespart und im Falle eines Fehlers lässt sich die Transaktion zurückrollen. Es sollte Wert auf die Abwärtskompatibilität gelegt werden, damit bestehende Schnittstellen ohne Veränderung weiter funktionieren.

4.1.3 Skalierbarkeit

Mit jedem neuen Nutzer unserer Plattform steigt die Diversität und somit die Chance, dass sich zwei Nutzer finden, welche zusammen passen und sich anfreunden. Je mehr Teilnehmer unserem Netzwerk angehören, desto höher ist die Anzahl der potenziellen Kommunikationspartner und somit der Nutzen und Wert der Plattform. Wir sprechen von einem positiven direkten Netzwerkeffekt. Je größer der Nutzen der Plattform, desto eher werden Webseitenbesucher weiteren potenziellen Besuchern von der Webseite erzählen, wodurch die Plattform weiter wächst und ihren Nutzen weiter ausbaut. Dies kann zu exponentiellem Wachstum führen. Desweiteren können große Persönlichkeiten der sozialen Medien („Influencer“) mit einer einzigen Bemerkung tausende Personen davon überzeugen, einen Blick auf unsere Webseite zu werfen.
Eine kurzfristige, rapide Vergrößerung der Nutzerbasis und exponentielles Wachstum stellen Datenbanken vor eine Herausforderung, die sich mit Skalierung lösen lässt. Auch sollen die Kapazitäten der Datenbank flexibel verringert werden können, um in Zeiten, in denen die Datenbank nicht ausgelastet ist, finanzielle Mittel zu sparen. Zur Skalierung wird eine Kombination aus vertikaler Skalierung (leistungsfähigere Hardware) und horizontaler Skalierung (mehr Geräte nebeneinander betrieben) gewählt. Vertikale Skalierung stößt auf Limitierungen, da der Preis von leistungsfähiger Hardware exponentiell skaliert. Weitere Skalierung ist dann nicht mehr rentabel. Bei horizontaler Skalierung müssen die einzelnen Knoten miteinander kommunizieren, die für die Kommunikation benötigten Ressourcen steigen mit jedem weiteren Knoten. Aus diesem Grund limitieren manche Datenbanken die Maximalanzahl an Knoten in einer Replikatgruppe. [20]
Desweiteren kann Datenbanksharding, eine Art der Datenbankpartitionierung, betrieben werden, bei dem eine Datenbank in mehrere Splitter bzw. Scherben aufgeteilt wird, welche jeweils einen Teil der Daten verwalten. Jeder dieser Splitter bildet wiederum eine eigene Replikatgruppe mit Primärknoten, Sekundärknoten und optional weiteren Knoten für Backups, Reportingtools und weitere. Durch diese Verfahren können Datenmengen verarbeitet werden, welche die Kapazität einer einzelnen Replikatgruppe übertreffen würde.

[29]

Vertikale Skalierung

Sowohl PostgreSQL als auch MongoDB lassen sich mit leistungsfähiger Hardware vertikal skalieren. Zwischen den beiden Datenbanken gibt es dahingehend keine nennenswerten Unterschiede, die Datenbanken schneiden in diesem Punkt ähnlich ab.

Horizontale Skalierung

Beide Datenbanken erlauben das Erstellen von Replikatgruppen mit einem Primärknoten, welcher für Lese- und Schreibzugriffe zur Verfügung steht und Sekundär- bzw. Standbyknoten, die je nach Einstellung entweder nur als Ausfallsicherheit dienen oder für Lesezugriffe zur Verfügung stehen. MongoDB unterstützt horizontale Skalierung nativ, während für PostgreSQL weitere Pakete benötigt werden. [25] Das Aufsetzen von MongoDB gestaltet sich entsprechend einfacher.

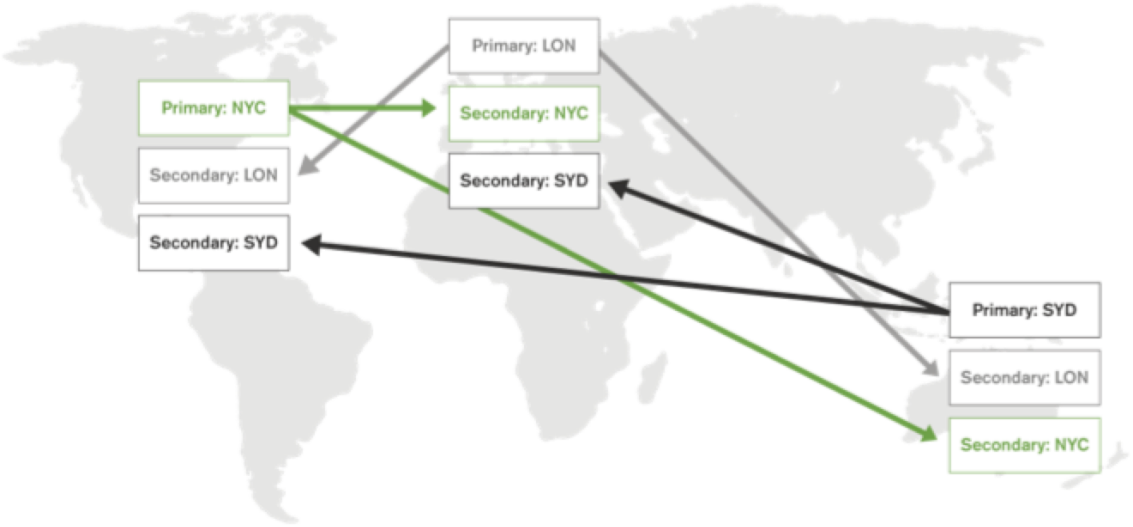
Sharding

Sharding stellt PostgreSQL vor eine Herausforderung, da komplexe JOIN-Anweisungen meist Daten von verschiedenen Shards erfragen, welche untereinander kommunizieren müssen. Diese Abfragen benötigen unverhältnismäßig viele Ressourcen und machen Performanzgewinne des Shardings zunichte. Es ist folglich schwer, Sharding mit den Tabellenstrukturen und der Normalisierung von SQL-Datenbanken zu vereinen. Daher ist darauf zu achten, Shards so zu konzipieren, dass Abfragen, welche Informationen von verschiedenen Shards erfragen, so selten wie möglich vorkommen.

MongoDB setzt auf Dokumente, welche in den meisten Fällen nicht auf Referenzen zu anderen Dokumenten angewiesen sind. JOIN-artige Anweisungen werden eher selten verwendet, Dokumente besitzen oft eine flachere Datenstruktur als die Pendants der SQL-Tabellen. Infolgedessen entstehen weniger komplexe Abfragen, es müssen seltener Dokumente verschiedener Shards kombiniert werden, die Datenbank wird weniger beansprucht. Sharding in MongoDB ist mit niedrigen Kosten verbunden und benötigt selten eine exakt konzipierte Struktur; es ist verhältnismäßig einfach, mit Sharding eine MongoDB Datenbank zu skalieren.
Zusätzlich erlaubt Sharding bei international angelegten Projekten durch geschickte Wahl der Position der Datenbankknoten, die Latenzzeit bei Abfragen zu minimieren (siehe Schaubild).

Fazit

Beide Datenbanken liefern ausgereifte Methoden, um mit immer größer werdenden Datenmengen zurecht zu kommen. Für kleine bis mittlere Projekte sollten PostgreSQLs Skalierungsmöglichkeiten ausreichen, während sich große Projekte durch bessere Möglichkei-



[27]

Abbildung 1: Regionale Shards mit Replikatgruppen. Sekundärknoten befinden sich für niedrigere Latenzzeiten bei Lesezugriffen in anderen Regionen

ten des Shardings in MongoDB noch weiter skalieren lassen. Auch scheint sich MongoDB mit weniger Arbeitsaufwand horizontal skalieren zu lassen, während bei PostgreSQL zuerst Anpassungen getätigt werden müssen. Damit ist MongoDB für dieses Projekt leicht präferiert.

4.1.4 Hochverfügbarkeit

Für den Anmeldevorgang, die Registrierung und alle Seiten unserer Plattform, die einen angemeldeten Nutzer voraussetzen, ist eine Datenbankanbindung zwingend erforderlich. Diese Teile machen den Großteil der Webseite aus, jede Minute, in der die Datenbank ausfällt, fällt unser Service nahezu komplett aus. Es bietet sich eine Datenbank mit Hochverfügbarkeit an, um das Risiko eines Ausfalls unserer Plattform so weit wie möglich zu verringern.
Um die Erreichbarkeit der Datenbank zu gewährleisten, bietet es sich an, Replikatgruppen zu erstellen. Eine Replikatgruppe besteht aus mehreren Datenbankprozessen, auch Knoten genannt, welche den gleichen Datensatz verwalten. Sollten durch Hardwarefehler einzelne Knoten ausfallen oder Softwarefehler Knoten dazu zwingen, neu gestartet zu werden, ist die Datenbank, wenn auch mit verringerter Leistung, weiterhin erreichbar. Die dadurch erreichte Fehlertoleranz schafft Sicherheit und verringert die Chance, dass die Datenbank komplett ausfällt drastisch.
Meist hat nur ein Prozess der Replikatgruppe Schreibrechte, um die Konsistenz bei gleichzeitigen Schreibzugriffen zu wahren. Neben diesem Primärknoten existieren oft mehrere Sekundärknoten, welche die Schreibvorgänge des Primärknotens kopieren und für Lesezu-

griffe zur Verfügung stehen. Es existieren andere Architekturen mit mehreren Primärknoten, die gleichzeitige Schreibzugriffe erlauben und besondere Herausforderungen bei der Datenkonsistenz stellen, diese werden hier jedoch nicht betrachtet.

PostgreSQL bietet verschiedene Lösungsansätze. Bei synchronen Lösungen muss ein Schreibauftrag von allen Knoten durchgeführt sein, um als bestätigt zu gelten. Asynchrone Lösungen erlauben einen Puffer zwischen dem Bestätigen eines Schreibauftrags und dessen Kopie auf andere Knoten, wodurch das Wechseln zu einem Backupknoten zu Datenverlust führen kann und Sekundärknoten bei Lesezugriffen ein leicht veraltetes Ergebnis liefern können. Dafür gewinnt eine asynchrone Lösung an Performanz, da sich die Wartezeit stark verringert; auch müssen die anderen Knoten seltener befragt und damit weniger belastet werden. Um die Knoten auf dem gleichen Stand zu halten und im Fall eines Ausfalls Daten wiederherstellen zu können, wird ein *Write-Ahead Log* (WAL) angelegt, welcher unmittelbar nach jedem Commit die Transaktion in einen Transaktionslog schreibt. Dieser wird dann von anderen Knoten ausgelesen und die Transaktion wird kopiert. [32]
Sollte der Primärknoten ausfallen, muss dies detektiert werden und ein Sekundärknoten mit so wenig Verzögerung wie möglich als neuer Primärknoten ausgewählt werden. PostgreSQL bietet keine automatische Ausfallsicherung, es muss daher eine Drittanbietersoftware verwendet werden oder ein eigenes Skript geschrieben werden. Um die Belastung der einzelnen Knoten möglichst gleich zu halten, sollte ein Prozess zur Lastverteilung existieren. Den Recherchen nach bietet PostgreSQL selbst keine native Lastverteilung, daher werden Lösungen wie die kostenlose Open-Source Software HAProxy (High Availability Proxy) verwendet.

Knoten in MongoDBs Replikatgruppen teilen sich gegenseitig durch Ping-Befehle ihren „Herzschlag“ mit, um zu ermitteln, ob ein Knoten ausgefallen ist. Sollte der Primärknoten „sterben“, wählen die „überlebenden“ Knoten in einer Abstimmung den nächsten Primärknoten, welcher die Schreibaufträge des vorigen Primärknotens übernimmt. Der Prozess einer Wahl sollte in der Regel nicht länger als 12 Sekunden dauern und passiert vollautomatisch. Knoten mit mehr Leistung kann eine höhere Priorität zugewiesen werden, um diesen als präferierten Primärknoten zu definieren. Auch eignet es sich, Knoten in anderen Regionen für die Rolle des Primärknoten als unwählbar zu definieren, da sich die Latenz ansonsten drastisch erhöhen würde. Ein Replikset kann aus maximimal 50 Knoten bestehen, wovon bis zu 7 zum Primärknoten wählbar sind. [20] [21] Diese Zahl ist vermutlich groß genug, um einen gleichzeitigen Ausfall aller Knoten aus technischer Sicht nahezu unmöglich zu gestalten, wenn sich die Knoten zusätzlich auf verschiedenen physischen Geräten befinden.
In seltenen Fällen kann es vorkommen, dass der Primärknoten ausfällt und vor seinem

Ausfall Schreibaufträge bestätigt, diese aber nicht an die Standbyknoten weiterleiten kann. Wenn der frühere Primärknoten der Replikatgruppe wieder beitritt, in diesem Fall als Sekundärknoten, unterscheidet sich dessen Schreibhistorie von der anderer Knoten. Die Historie des früheren Primärknoten wird zurückgerollt und die bestätigten Schreibaufträge sind verloren.
MongoDB versucht durch verschiedene Techniken, Rollbacks zu vermeiden und erlaubt es unter Verlust von Effizienz, mit Schreibbestätigungen erst zu antworten, wenn der Großteil der Replikatgruppe diese bestätigt hat. Sollte es trotzdem zu einem Rollback kommen, müssen einzelne Schreibaufträge oft manuell nach bestem Gewissen der Datenbankexperten wiederholt werden. [22]
Leseanfragen werden per Standardkonfiguration an den Primärknoten gestellt, um möglichst aktuelle Daten liefern zu können. Diese Präferenz lässt sich ändern, um den Primärknoten zu entlasten, auf speziell eingerichtete Knoten mit optimisierten Indexen zugreifen zu können, die Latenz zu verringern oder beim Ausfall des Primärknotens weiterhin Lesezugriffe zu ermöglichen. Auch ist eine Wahl zwischen asynchronen und synchronen Operationen möglich, dabei sind asynchrone Operationen aufgrund der höheren Performanz die Standardeinstellung.

MongoDB überzeugt im Punkt der Hochverfügbarkeit gegenüber PostgreSQL. Um PostgreSQL hochverfügbar zu machen, sind einige Anpassungen und Expertenwissen oder Drittanbietersoftware nötig. MongoDB scheint von der Architektur auf Hochverfügbarkeit ausgerichtet zu sein und liefert Funktionen für eine automatische Ausfallsicherung, welche das System nach kurzer Zeit ohne manuelle Eingriffe wieder voll funktionstüchtig machen. Wir glauben, dass mit MongoDB eine Datenbank eingerichtet werden kann, die mit wenig Aufwand stabil für sehr geringe Ausfallraten sorgen kann.

4.1.5 Dateiformat

MongoDB speichert Daten im JSON-Format. JSON, die „JavaScript Object Notation“, ist „ein schlankes Datenaustauschformat, welches für Menschen einfach zu lesen und für Maschinen einfach zu parsen [] ist“ [15]. JSON als semistrukturiertes Dateiformat eignet sich daher gut für Schnittstellendaten.
Desweiteren wird im gewählten MERN-Techstack ausschließlich JavaScript verwendet - das JavaScript native Dateiformat JSON ist daher ohne Umwandlungen direkt verwendbar und der Umgang für das Entwicklerteam bereits bekannt. Dies verringert die Gefahr möglicher Komplikationen und spart Lern- und Programmieraufwand. Diesen Vorteil besitzt PostgreSQL nicht, Abfrageergebnisse müssen für die Schnittstelle zuerst in JSON-Dateien geändert werden, was einen zusätzlichen Programmierschritt bedeutet.

4.1.6 Fazit

Kriterien	PostgreSQL	MongoDB
Lesegeschwindigkeit	Mittel	Hoch
Schreibgeschwindigkeit	Mittel	Langsam - Mittel
Datenstrukturen	Strenges Tabellenschema	Schemafrei durch selbstbeschreibende JSON-Dokumente
Skalierbarkeit	Meist vertikal, horizontal benötigt erweitertes Setup	Meist horizontal, unterstützt nativ Sharding
Hochverfügbarkeit	Replikatgruppen, Lastverteilung und automatische Ausfallsicherung meist über Drittsoftware	native Replikatgruppen und automatische Ausfallsicherung, Lastverteilung bei Wahl eines Sekundärknotens, automatische Wahl des nächsten Primärknotens bei Ausfall
Dateiformat	internes Format, wird bei Abfragen in lesbaren Tabellen ausgegeben	JSON

Es konnte gezeigt werden, dass beide Datenbanken eine ausgereifte Architektur besitzen und sich daher beide gut für Projekte eignen. Für dieses Projekt hat jedoch MongoDB einige Vorteile, die sich darauf zurückführen lassen, aus welchem Beweggrund MongoDB entstanden ist.
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.[26]
Aufgrund MongoDBs Historie sind automatische Ausfallsicherung, horizontale Skalierung und Sharding native Funktionen, die sich mit wenig Entwicklungsaufwand einstellen und skalieren lassen. Auf der Plattform ist es gut denkbar, dass auf einen komplexen Schreibzugriff dutzende, vielleicht hunderte Lesezugriffe kommen, die höhere Geschwindigkeit ist bemerkbar. Auch integriert sich das von MongoDB gewählte Dateiformat JSON gut mit der weiteren Architektur des verwendeten MERN-Techstacks. Aus den genannten Gründen fällt die Wahl für das Projekt auf MongoDB.

4.2 Schemata

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

Nutzer

Wenn ein Nutzer einen Account erstellt, gibt dieser seine 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 stellt einen hohen Stellenwert dar. Die Anzahl an Daten, die ein Nutzer von sich preisgeben müssen, soll so gering wie möglich halten werden. Die Email-Adresse, das Geburtsdatum und Freundes- und Blockliste sind für andere Nutzer nicht einsehbar; bis auf den Benutzernamen, welcher einen Fantasienamen darstellen kann, muss eine Person keine Daten öffentlich angeben.

Sprache

Damit Nutzer ihre gesprochenen Sprachen wählen können, bieten wir die Wahl zwischen 187 Sprachen nach ISO 639-1 Norm an.[14]

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

Standardmäßige Objekt-IDs von MongoDB enthalten einen Zeitstempel und einen inkrementellen Zähler.[28] Diese Daten sind bei Sprachen - öffentlichen Stammdaten, die sich über einen langen Zeitraum nicht verändern werden - nicht relevant. Stattdessen wurde der Alpha-2-Code der Sprache als ID gewählt, der in den meisten Fällen auf die Sprache schließen lässt. Nutzerdokumente referenzieren die Sprache per ID. Dadurch stellt es kein großes Problem dar, direkt im Nutzerdokument anhand der Sprach-ID zu erkennen, welche Sprachen der Nutzer spricht. Fehler können einfacher vermieden werden und Tests sind einfacher menschlich nach Richtigkeit kontrollierbar.

Sprachen können sowohl anhand der englischen Schreibweise als auch der nativen Schreibweise gefunden werden. Dies erleichtert auch nicht-englischsprachigen Nutzern, ihre Sprache auswählen zu können.

Passwort

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

Das Passwort wird nicht im Nutzerdokument gespeichert. Wenn das Passwort im Nutzerdokument gespeichert werden würde, könnten schon kleine Programmierfehler dazu führen, 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-basierten Hashalgorithmus, auf der Datenbank als Hash mit Salt gespeichert. Die Komplexität des Hashes ist frei 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 den Server stark verlangsamen und macht Angriffszenarien per (D)DOS ((distributed) denial of service, Überlastung des Servers durch übermäßigen Datenverkehr) gefährlicher, da gezielte Anmeldeversuche viel Last auf dem Server erzeugen. In Zukunft werden weitere Limitierungen auf Seiten des Backends erstellt, um wiederholte Anmeldeversuche zu bremsen.
Um die Wahrscheinlichkeit von Glückstreffern bei Angriffen zu verringern, wird verlangt, 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.

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.

Anmerkung: Der Name „Like“ für diese Datenstruktur kann irreführend sein, 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 und ein Dokument der Kollektion Chat erstellt. in dem Fall können die Nutzer miteinander kommunizieren. Dislikes sorgen dafür, dass ein Kontakt in Zukunft nicht mehr möglich ist.

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.

Sollten sich zwei Nutzer befreunden, wird zwischen diesen ein Chat 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, falls dies in Zukunft eine erwünschte Funktion sein sollte.
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 Pagination (Seitennummerierung) je Abfrage 20 Nachrichten zu erfragen. Diese Methoden verringern die Serverlast, da in vielen Fällen nicht mehr als die erste Seite der Abfrage relevant ist.

Nachricht

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

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

4.3 Database-as-a-Service

Um eine Datenbank selbst zu betreiben fehlt es dem Projekt an fachlichen Kapazitäten und einer Infrastruktur, welche physische Datenbankserver unterstützt. Es bietet sich daher eine Database-as-a-Service-Lösung (DBaaS) an.
MongoDB Inc. bietet mit MongoDB Atlas eine DBaaS 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 realen Auslastungen gerecht zu werden. Kostenpflichtige Stufen bieten die Möglichkeit an, Backups einzurichten, für höhere Kosten stehen Tools zur Verfügung, die Metriken in Echtzeit anzeigen, automatisch archivieren, Empfehlungen zur Leistungsoptimierung erstellen und langsame Datenbankabfragen zur Diagnose und Optimierung anzeigen. In der Entwicklungsphase wurde sich für die kostenlose Stufe entschieden, da die Funktionen und Leistung für die Entwicklungsumgebung ausreichen. Sollte das Produkt auf den Markt gehen, wird auf eine kostengünstigste Stufe gewechselt, um die Option von Backups zu erhalten. Sollte das Projekt erfolgreich sein und viele Nutzer anziehen, wird flexibel, abhängig von benötigter Leistung, eine teurere Stufe mit mehr Leistung 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. [1]

4.4 Avatarbilder

Statt Bilddateien für Avatare direkt auf der Datenbank zu speichern, was mit langen Wartezeiten auf die Datenbank einhergehen würde, werden nur die URIs zu den Bildern auf der Datenbank gespeichert. Für das Speichern der Binärdateien wurde sich für AWS S3 entschieden, 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. Beim Verteilen der Zugriffsrechte wurde nach Minimalprinzip vorgegangen, das Backend besitzt nur die Zugriffsrechte die es benötigt und keine weiteren. Ein möglicher Angriff verursacht dadurch weniger Schaden, als wenn das Backend alle Zugriffsrechte hätte.
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-Paketen 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.

4.5 Fazit

Mit den gewählten Kollektionen und der Wahl von speziellen Hosts sind wir in der Lage, Nutzern eine Datenbank anzubieten, die eine hohe Erreichbarkeit aufweist, sich leicht an die Auslastung skalieren lässt, personenbezogene Daten geheim hält und ein Maß an Datensicherheit bietet, welches der Größe des Projektes entspricht. Flexible Datenstrukturen erlauben schnelle Anpassungen des Projektes. Die Verwendung eines bekannten Dateiformats, welches sich gut für Schnittstellen eignet, sorgt für Zeitersparnisse in der Entwicklung des Frontends und der Datenbankschnittstelle und reduziert somit den Aufwand. Außerdem werden durch eingebettete Dokumente, Denormalisierung und Seitennummerierung Optimierungen durchgeführt, die als Ziel haben, möglichst schnell Antworten auf Leseanfragen zu bieten.

5 Frontend

In diesem Kapitel werden Angular und React für die Entwicklung des Frontends bewertet.
Beide Technologien haben unter anderem die Vorteile, gut dokumentiert zu sein und große Gemeinschaften zu haben. Dies begünstigt den Lernprozess und verschleunigt mögliche Problembehebungen.

Sie haben die Unterstützung von Unternehmen wie Facebook und Google. Sie sind komponentenbasiert, was ihre Testbarkeit erleichtert.
Es wurden die oben genannten Technologien ausgewählt, weil diese an den ersten Stellen in diversen Ranglisten auftauchen[41].

Folgende Kriterien wurden berücksichtigt:

- Der Schwierigkeitsgrad, um die Technologie zu lernen. Die Lernkurve ist wichtig, weil das Entwicklungsteam begrenzte Zeit für die Entwicklung des Projekts hat.
- Die Akzeptanz bei den Entwicklern und die Anzahl von heruntergeladenen NPM-Paketen sind relevant, weil vermehrte Nutzung dazu führen kann, dass Probleme schneller identifiziert und gelöst werden.[18]
- Persönliche Präferenz und Vorkenntnisse des Entwicklungsteams.

5.1 Popularität und Statistiken

Die Informationen unten geben Hinweise auf den Nutzen und die Beliebtheit der Open-Source-Projekten React und Angular. Diese Indikatoren sind wichtig, denn je mehr Menschen an einem Open-Source-Projekt beteiligt sind, desto mehr Menschen tragen dazu bei, Fragen zu beantworten und Probleme[1] zu lösen [18].

Developer Survey 2021 - StackOverFlow

StackOverFlow ist die größte Gemeinschaft von Softwareentwicklern, wo Wissen und Fragen zur Softwareentwicklung ausgetauscht wird.
Seit 2011 führt StackOverFlow eine jährliche Umfrage zur Softwareentwicklung durch.

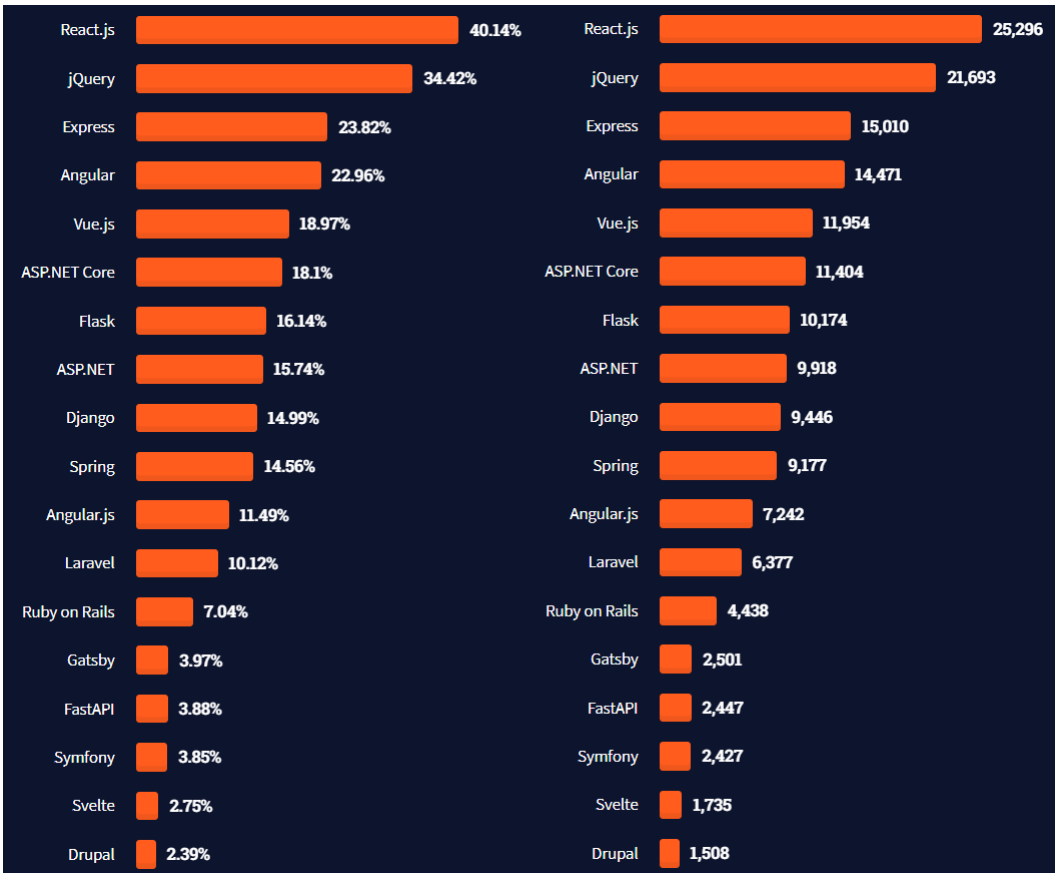


Abbildung 2: Which web frameworks and libraries have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the framework and want to continue to do so, please check both boxes in that row.) [41]

[1] Code issues

Github

GitHub ist die Plattform für die Versionskontrolle von Source-Code, wo mehr als 238 Millionen Repositories[2] verwaltet werden[11].

Die Anzahl der Repositories, in denen Code für Angular und, React gespeichert wird, gibt einen Hinweis auf die Beliebtheit dieses Frameworks.

Github Statistiken		
	Angular/core	React
Anzahl von Repositories	2,011,663	7,868,546
Stars insgesamt [3]	77.2k	170k

[9, 10]

NPM Paketen

NPM verwaltet die Abhängigkeiten eines Angular oder React Projekts.

Die Anzahl der heruntergeladenen NPM-Pakete gibt einen Hinweis auf die Nutzung der Technologie.

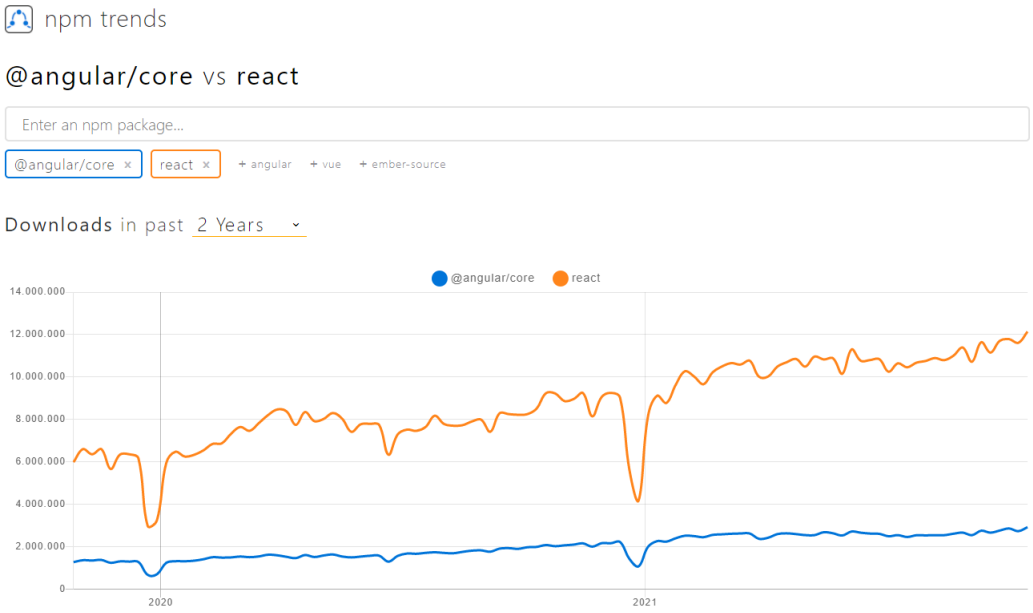


Abbildung Tabelle 5.1: heruntergeladenen NPM-Pakete @angular/core vs react [31]

[2] Ablage wo Quellkode gespeichert wird

Google Trends

Laut Google Trends war React zwischen 01.11.2020 und 26.10.2021 das am häufigsten konsultierte Frontend Technologie in Deutschland.
Diese Metrik ist ein weiterer Aspekt für die Beliebtheit von Framework.

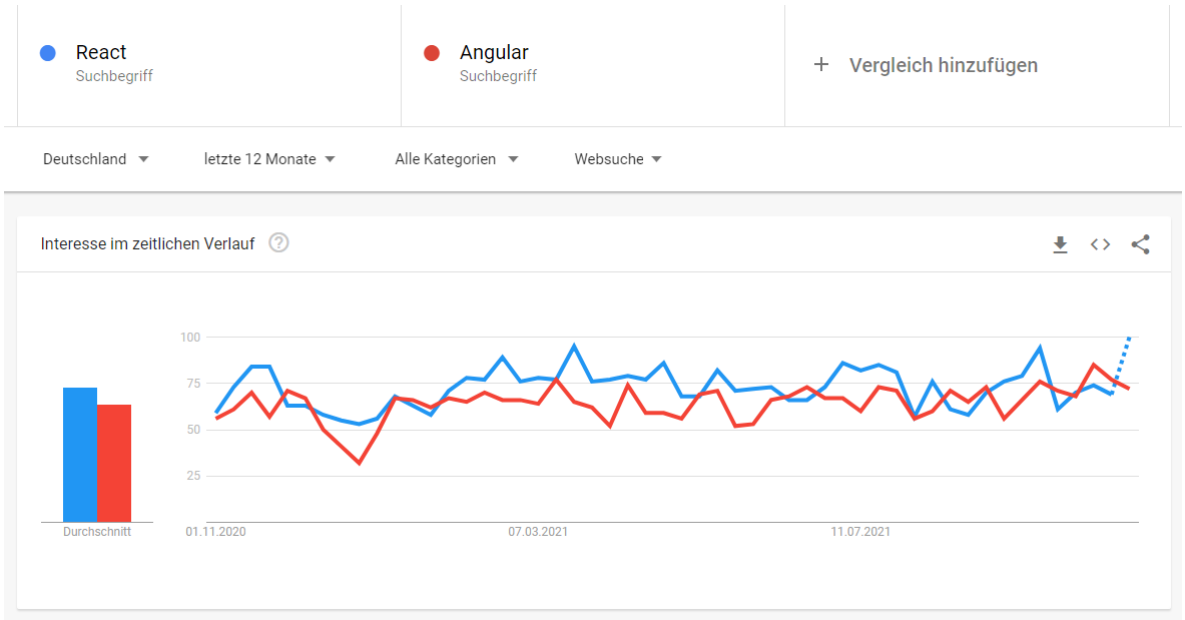


Abbildung Tabelle 5.1: Google Trends Angular vs React [12]

Jobangebote

Als angehende Informatiker sind berüfliche Perspektiven relevant, deswegen wurde den Arbeitsmarkt in der Entscheidungsfindung berücksichtigt.

Anzahl der Jobangebote bei LinkedIn in Deutschland:
Angular: 7.657 Ergebnisse[16]
React: 11.523 Ergebnisse[17]

AB HIER ÜBERARBEITEN:

5.2 Framework vs Bibliothek

In dieser Arbeit stehen ein Framework und eine Bibliothek zur Auswahl.
Beide Technologien wurden für die Entwiklung von SPAs[4] konzipiert.

Eine Bibliothek wie React bietet mehr Kontroll darauf, wie der Code entwickelt werden kann und erlaubt die Applikation mit anderer Bibliotheken zu erweitern. Zum Beispiel das State-Management mit Redux[39] verwendet wird.

Andererseits gibt das Framework von Angular genauer vor, wie der Code entwickelt werden muss und beinhaltet.

[4]Single-page application [30]

5.3 Angular

Angular ist das Framework von Google für Mobil- und Webanwendungen.

Angular hat eine steile Lernkurve im Vergleich zu React.[6], S.25.

Zu beachten ist, dass Angular nicht JavaScript als Hauptprogrammiersprache verwendet, sondern TypeScript. Dies bedeutet zusätzlichen Aufwand für das Erlernen des Frameworks. Dieses Framework ist besonders geeignet für fortgeschrittene, umfangreiche Projekte. QUELLE?

5.4 React

Bei der endgültigen Entscheidung wurden React aufgrund ihrer geringen Lernkurve, ihrer guten State Management, ihre große Popularität und ihrer Einfachheit berücksichtigt.

Die Entscheidung fiel für React aufgrund der Vorkenntnisse des Entwicklerteams, der Anzahl an Jobangeboten und der Unterstützung durch ein Unternehmen wie Facebook.

5.4.1 Vorteile

Deklarativ

Mit React ist es möglich, interaktive Benutzeroberflächen, Ansichten für jeden Zustand der Anwendung zu erstellen. Durch deklarative Ansichten wird der Code vorhersehbarer und einfacher zu debuggen.

Komponentenbasiert

React ist auf gekapselte Komponenten basiert, die ihren eigenen Zustand verwalten.

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. [41]

5.4.2 Nachteile

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.[35]

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.[34]

5.5 React Hooks

Beginnend mit 16.8.0, enthält React eine stabile Implementierung von React Hooks.

Ab dieser Version wird React empfohlen, keine Klassen mehr für die Erstellung von Komponenten zu verwenden. Mit Klassen geschriebene Komponenten werden weiterhin unterstützt und müssen nicht neu geschrieben werden. [38]

useState Der Hook useState bietet die Möglichkeit an, den Zustand einer Anwendung zu verwalten. Sie besteht aus mindestens einen Wert (value) und einer Funktion (updateValue), die den Wert aktualisiert.

In unserem Projekt wurde useState verwendet, um alle Benutzerdaten zu manipulieren und dem Benutzer anzuzeigen.

Diese Benutzerdaten sind die Antwort auf die an den Server gesendete Abfrage. Ein Beispiel von einer Server-Antwort befindet sich im Anhang 1. Eine genauere Funktionsweise der Serverabfragen folgt in Kapitel Serverabfragen.

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

Unter anderem ist mit useEffect folgenden möglich:

-Aktualisieren des DOM,
-Abrufen und Konsumieren von Daten von einer Server-Schnittstelle,
-Einrichten eines Abonnements, usw.

Bei diesem Projekt wird die Anzeige der Daten abhängig der Änderungen des lokalen Zustands manipuliert.

Im Bereich Profil werden die Änderungen erstmal zwischengespeichert, bevor sie in der Datenbank gespeichert werden.
Die Abfrage zur Aktualisierung der Daten wird in dem Moment an den Server gesendet, in dem der Benutzer auf die Taste „Save/Speichern“ drückt.
Zu diesem Zeitpunkt wird die Benutzeroberfläche durch useEffect aktualisiert, da sich eine der Abhängigkeitsvariablen geändert hat.

Es ist möglich, mehrere useEffects zu einstellen und mit ihrer jeweiligen Abhängigkeiten separat definieren. Wenn es erfordelich ist, dass der Code innerhalb dem useEffect nur einmal nach dem ersten Rendering(deutsches WORT) ausgeführt wird, muss ein leeres Array als Abhängigkeit definiert werden.

useContext useContext ermöglicht Daten innerhalb bestimmte Komponenten zu teilen.

In diesem Projekt war useContext eingesetzt, um die Benutzerdaten und den Authentifizierungstoken über die Komponenten „Profile“, „Chat“, „Users“ (to swipe) zu teilen. In frühen Versionen des Projekts wurde für den gleichen Zweck props verwendet. [36] UseContext bietet eine weitere Lösung, um Daten zwischen Komponenten auszutauschen.[37]

Im Anhang 5 und 6 befidet sich der Code mit dem den Authentifizierungstoken und die Benutzerdaten global mithilfe von useContext verwaltet wird.

5.6 Fazit

Es ist anzumerken, dass die Entwicklung des Projekts mit jedem der beiden Frameworks möglich war. ...

6 Serverabfragen

6.1 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. [33]

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 Abfragen 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.

5

[5] Vgl. u.a. [40]

6.1.1 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.

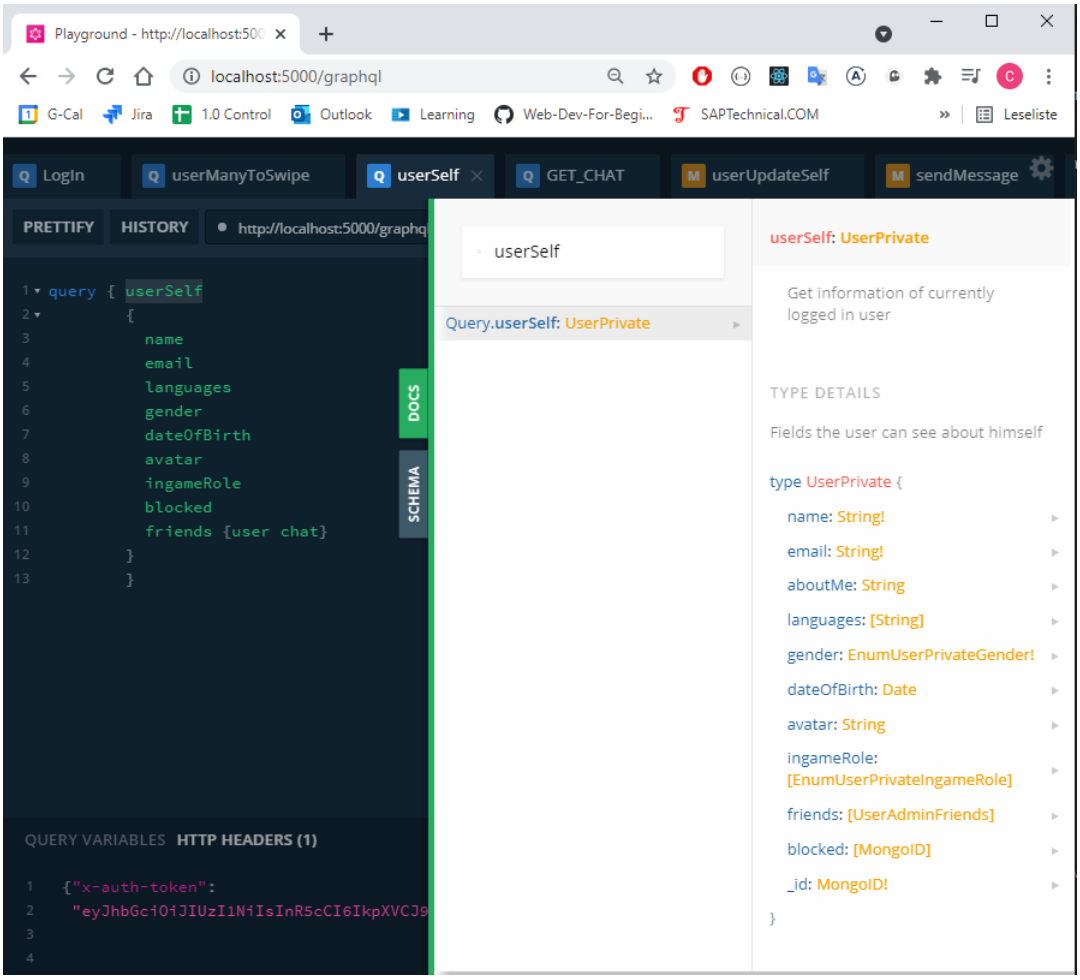


Abbildung Unterunterabschnitt 6.1.1: Genau derselbe Code wird für die Abfrage von Benutzerinformationen später verwendet.

6.2 Implementierung der Serverabfragen 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

6.2.1 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.

6.2.2 Mutationen

6.2.3 Subscriptions

TODO after receiving feedback about the rest.

6.3 Axios

Zusätzlich zu den GraphQL-Abfragen wurde eine Post-Anfrage mit Axios bereitgestellt. Mit dieser war es möglich, Bilder auf eine S3 Speichereinheit von Amazon Web Services hochzuladen.
Ein Auszug aus dem zu diesem Zweck verwendeten Code findet sich in Anhang 4.

Das Format der hochzuladenden Dateien wurde auf .png, .jpg und .jpeg beschränkt, damit nur zulässige Dateien an den Server gesendet werden.
Die Größe der hochzuladenden Datei wurde um 1 MB abgegrenzt.

7 Qualitätssicherung

Software Testing

Obwohl das Projekt relativ klein ist, wurde die Wichtigkeit von automatisierten Tests nicht unterschätzt.
Für das Frontend wurden End-to-End Testfälle mit Cypress geschrieben. Auf diese Weise ist es möglich in Sekundenschnelle festzustellen, ob etwas in unserer Anwendung defekt ist.
Ein Szenario, in dem dies hilfreich ist, ist, wenn eine Unterkomponente in anderen Komponenten verwendet wird.
Durch die Änderung der Unterkomponente kann sich diese in einer unerwünschten Weise verhalten.
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. Fehler zu finden, wäre in dem Fall aufwändiger.
Ohne automatisierte Tests, ist manuelles Testing nötig.
Im Anhang 2 befindet sich ein Code-Auszug eines Testfälles End-to-End.

7.1 Die Testfälle für unser Projekt

Nachstehend einer Überblick über die Testfälle bei Cypress.

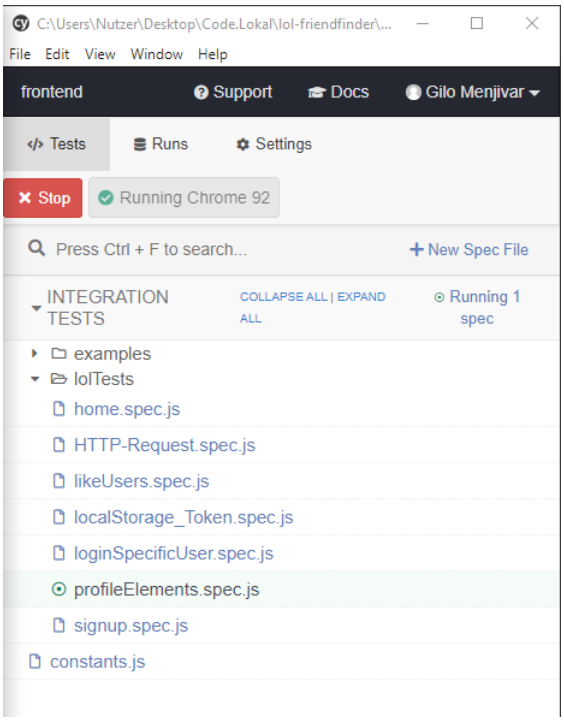


Abbildung Abschnitt 7.1: Testfälle in Cypress

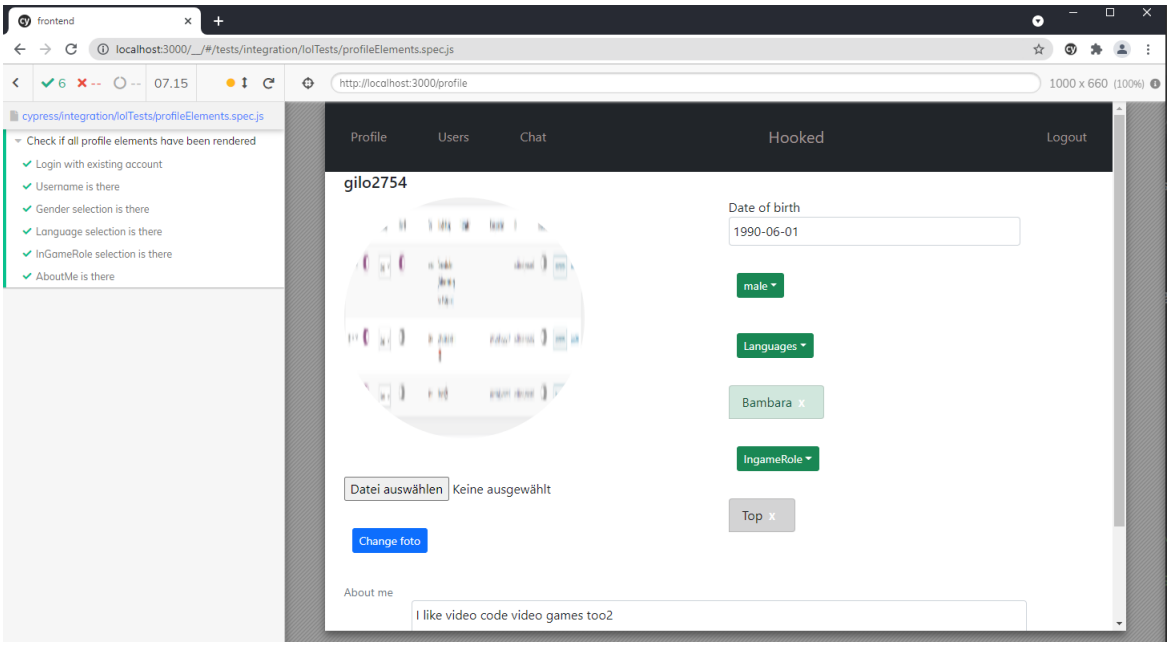


Abbildung Abschnitt 7.1: Grafische Darstellung der verschiedenen Tests in Cypress

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 Abfragen 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.
Diese setzen sich aus nativen Cypress-Befehlen zusammen. Es handelt sich praktisch um benutzerdefinierte Funktionen, die häufig bei Tests verwendet werden.
Ein Beispiel ist im Anhang 3 zu finden.

8 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.

9 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.

10 Quellenverzeichnis

11 Quellenverzeichnis

11.1 Literatur

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

11.2 Internetquellen

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

[2] Offizielle Website Apollo für React. https://www.apollographql.com/docs/react/ (Abgerufen am 15.10.2021)

[3] Axios Dokumentation https://axios-http.com/docs/intro (Abgerufen am 20.10.2021)

[4] 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).

[5] Elad Elrom: React and Libraries. https://link.springer.com/content/pdf/10.1007%2F978-1-4842-6696-0.pdf, S. 25

[6] Entscheidungshilfe für die Webentwicklung anhand des Vergleichs von drei führenden JavaScript Frameworks: Angular, React and Vue.js https://reposit.haw-hamburg.de/bitstream/20.500.12738/8417/1/BA_Wohlgethan_2176410.pdf

[7] Github Open Issues and Stars · angular/angular https://github.com/angular/angular/issues (Abgerufen am 24.10.2021)

[8] Github Open Issues and Stars · React https://github.com/facebook/react/issues (Abgerufen am 26.10.2021)

[9] Github Repositories · facebook/react https://github.com/facebook/react/network/dependents (Abgerufen am 24.10.2021)

[10] Github Repositories · angular/angular https://github.com/angular/angular/network/dependents?package_id=UGFja2FnZS00NTE2NDYyMzQ%3D (Abgerufen am 24.10.2021)

[11] Total Github Users/Issues/Repositories https://github.com/search (Abgerufen am 29.10.2021)

[12] Google Trends React-Angular 1.11.2020-26.10.2021 https://trends.google.com/trends/explore?geo=DE&q=React,Angular (Abgerufen am 29.10.2021)

[13] 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).

[14] 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]

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

[16] Jobsangebote Angular https://www.linkedin.com/jobs/angular-jobs/(Abgerufen am 27.10.2021)

[17] Jobsangebote React https://www.linkedin.com/jobs/react-jobs/ (Abgerufen am 27.10.2021)

[18] Building leadership in Open Source Community https://www.linuxfoundation.org/tools/building-leadership-in-an-open-source-community/ (Abgerufen am 24.10.2021)

[19] https://docs.mongodb.com/manual/core/replica-set-elections/#std-label-replica-set-elections

[20] https://docs.mongodb.com/manual/core/replica-set-elections/#non-voting-members

[21] https://docs.mongodb.com/manual/reference/replica-configuration/#mongodb-rsconf-rsconf.members-n-.priority

[22] https://docs.mongodb.com/manual/core/replica-set-rollbacks/

[23] https://docs.mongodb.com/manual/core/read-preference/

[24] https://docs.mongodb.com/realm/mongodb/specify-cluster-read-preference/

[25] PostgreSQL vs. MongoDB Scalability https://www.openlogic.com/blog/postgresql-vs-mongodb

[26] 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]

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

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

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

[30] Mozilla Glosary: Single-page-applicartion https://developer.mozilla.org/en-US/docs/Glossary/SPA (Abgerufen am 29.10.2021)

[31] NPM Downloads @angular/core vs react https://www.npmtrends.com/react-vs-@angular/core(Abgerufen am 29.10.2021)

[32] https://www.postgresql.org/docs/current/wal-intro.html

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

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

[35] Das React Framework – von Anwendungen, Funktionen und den Vor- & Nachteilen der Komponenten. https://kruschecompany.com/de/react-framework/

[36] React Components and Props https://reactjs.org/docs/components-and-props.html

[37] React Context vs Props https://medium.com/geekculture/props-drilling-v-s-context-api-which-one-is-the-best-75c503d21a65

[38] React Hooks F.A.Q. https://reactjs.org/docs/hooks-faq.html#do-i-need-to-rewrite-all-my-class-components

[39] React Redux. https://redux.js.org/

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

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

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

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.