Songguesser

Software-Projekt / 113468b

Studierende

Lucas Goldner - @lg066 (113468)

Aykon Kücük - @ak298 (113468)

Yekta Leon Kücük - @ yk020 (113400)

Inhaltsverzeichnis

Einführung und Ziele

Aufgabenstellung

Song Guesser ist ein Multiplayer-Spiel, das bequem im Browser gespielt werden kann. Es besteht aus einem ansprechenden Frontend, welches in Next.js implementiert wurde, sowie einem robusten Backend, das auf dem Framework Nest.js basiert. Das Ziel des Spiels ist es, Songs auszuwählen, die andere Spieler anhand einer kurzen Vorschau erraten müssen. Für jede korrekte Antwort erhalten die Spieler Punkte, wobei die korrekte Benennung des Titels 100 Punkte und die korrekte Benennung des Interpreten 50 Punkte einbringt. Der Spieler mit den meisten Punkten am Ende des Spiels gewinnt.

Der Funktionsumfang des Minimum Viable Product kann wie folgt beschrieben werden:

- Die Möglichkeit für Benutzer, temporäre Konten zu erstellen
- Raumerstellung, damit andere Spieler beitreten können
- Eine Liste der verfügbaren Räume soll ansehbar sein
- Sobald sich alle Spieler im Raum befinden, kann der Raum-Ersteller das Spiel beginnen
- Innerhalb jeder Runde können die Spieler jeweils einen Song auswählen und die anderen Spieler müssen versuchen, diesen Song zu erraten
- Auch eine Chat-Funktion, die den Spielern ermöglicht, während des Spiels miteinander zu kommunizieren

Qualitätsziele

Qualitätsziel Motivation und Erläuterung

Modularität	Dadurch können Entwickler Code in wiederverwendbare Module aufteilen. Dies ermöglicht eine effiziente Code-Wiederverwendung, was insbesondere bei der Entwicklung großer Anwendungen Zeit und Mühe spart. Diese Modularität findet sich vor allem im Backend wieder, welches aus 4 Modulen besteht.
Funktionalität	Songguesser sollte den Spielern eine intuitiv bedienbare Benutzeroberfläche bieten, die es ihnen ermöglicht, Songs auszuwählen, die Vorschauen zu hören und ihre Vermutungen einzugeben. Das Spiel sollte auch in der Lage sein, das Spiel der Spieler zu verfolgen, Punkte zu sammeln und den Sieger des Spiels zu bestimmen.
Portabilität	Dadurch dass der Songguesser eine Webapps ist, ist es in der Lage, auf verschiedenen Plattformen ausgeführt zu werden, wie z.B. Desktop-Computern, Tablets oder Mobilgeräten
Skalierbarkeit	Das Spiel sollte in der Lage sein, sich an die Anzahl der Spieler anzupassen, die das Spiel spielen möchten. Das System sollte in der Lage sein, mit der Anzahl der gleichzeitigen Spieler umzugehen und sicherzustellen, dass die Leistung des Spiels nicht beeinträchtigt wird.

Randbedingungen

Technische Randbedingungen

Randbedingung Erläuterungen, Hintergrund

Anforderungen an Skalierbarkeit und Verfügbarkeit	Das System sollte in der Lage sein, sich an die Anzahl der Spieler anzupassen, die das Spiel spielen möchten. Es sollte auch sicherstellen, dass das Spiel immer verfügbar ist.
Technische Anforderungen	Das Spiel sollte auf verschiedenen Betriebssystemen, Webbrowsern und Geräten laufen, und mit einer angemessenen Bandbreite von Netzwerkverbindungen kompatibel sein. Es sollte auch sicherstellen, dass es mit den neuesten Web-Technologien kompatibel ist
Unabhängigkeit von MusikAPI	In Songguesser, wurde durch Abstrahierung darauf geachtet, dass sich die aktuelle API, welche für die Musik Querrys benutzt wird, leicht ersetzte werden kann. Somit ist Songguesser nicht gefährdet von zukünftigen API Änderungen, da es ersetzt werden kann

Organisatorische Randbedingungen

Randbedingung Erläuterungen, Hintergrund

Team	Das Team besteht aus drei Studierenden, welches von einem Professor betreut wird.
Zeitplan	Die Entwicklung beginnt am 6. Oktober 2022 und endet am 25. Februar 2023. Ende Januar soll ein erster funktionsfähiger Prototyp zur Medianight fertig sein. Die Entwicklung schließt mit der Version 0.2.0 ab.
Vorgehensmodell	Die Entwicklung findet iterativ und inkrementell im zweiwöchigem Zyklus statt, der durch eine Besprechung abgeschlossen/begonnen wird. Vor Beginn der Entwicklung wird ein Minimum Viable Product (MVP) definiert und es werden die Anforderungen erfasst.
Entwicklungswerkzeuge	Als Entwicklungsplattform wird Github mit der Versionsverwaltung Git eingesetzt. Github wird primär zur Verwaltung des Quellcodes, für die Pull Requests und das Issue Tracking verwendet. In GitHub Projects werden die Aufgaben für jede Woche unterteilt in Karten, welche bei PullRequests das gelinkte Issue schließen. Für das Testen wurde hauptsächlich Chrome und Safari verwendet.

Konventionen

Randbedingung Erläuterungen, Hintergrund

Dokumentation	Zur Dokumentation der Architektur kommt das deutschsprachige arc42-Template zum Einsatz.
Clean Code	Es werden uns vorgestellte und bekannte Design Patterns sowie Best Practices umgesetzt, um einen sauberen, wart- und erweiterbaren Quellcode zu produzieren.
Sprache	Der Quellcode inklusive Kommentare wird vollständig auf Englisch gehalten. Commit-Nachrichten und Merge-Requests werden auf Englisch geschrieben, wohingegen die Dokumentation auf Deutsch gehalten wird.
Benennung von Issues	Die Benennung von Issues erfolgt nach Kategorie (kategorie:feature), da nicht nur Codeaufgaben für die Spielentwicklung relevant sind. Z.b: Design:UI for Startmenu

Kontextabgrenzung

Schnittstelle	Bedeutung	Technischer Kontext
Benutzer	Der Benutzer kann an das Spiel Inputs senden, welche Auswirkungen auf das Spiel haben.	Menschlicher Benutzer
NextJs	NextJs als Framework für das Frontend	Framework
NestJs	NestJs als Framework für das Backend	Framework
Socket.io	Einfache Implementierung von Websockets, welche für realtime Datenübertragung genutzt wird	Bibliothek
Jest	Für Tests	Bibliothek

Fachlicher Kontext

Name	Zuständigkeitsbereich
Lucas Goldner	· Projektorganisation · Konzeptentwicklung (Definierend Models, DTOs) · Backendentwicklung · Frontendentwicklung · Supervisor

Aykon Kücük	· Backendentwicklung · Spotify Integration
Yekta Leon Kücük	· Frontendentwicklung · Logoerstellung · Vektorgrafiken erstellt · Design

Lösungsstrategie

Ein kurzer Überblick über die grundlegenden Entscheidungen und Lösungsansätze:

Entwurfs- und Architekturmuster

Für das Backend haben wir uns entschieden mit verschiedenen Modulen zu arbeiten. So haben wir jede bedeutende Funktionalität in ein Modul definiert.

Dabei sind am Ende 4 Module entstanden: Das User-Modul, welches den Ganzen Lebenszyklus eines Users verwaltet. Das Room-Modul, welches die Verwaltung der Räume übernimmt. Das Spotify-Modul, welches die Anbindung an die Spotify API sowie die Ausführung deren Funktionen übernimmt. Und zu Schluss das Game-Modul, welches eigentlichen Vorgänge zum Spielen des Spiels definiert.

Für die UI haben wir uns am Anfang auf eine Farbpalette beschränkt, um eine einheitlich aussehende grafische Benutzeroberfläche zu kreieren. Dabei haben wir uns als Hauptfarbe für unser Giftgrün entschieden, zusammen mit weißen und schwarzen Elementen.

Zu Beginn des Website-Entwurfs haben wir uns auf bestimmte Design-Aspekte geeinigt, wie die Farbpalette und das Logo. Anschließend haben wir einige Skizzen auf Papier angefertigt und diese dann in Wireframes auf dem Computer umgesetzt. Diese Wireframes dienten uns als Grundlage für das Layout der Website. Schließlich haben wir die Website mit CSS gestaltet.

Technologieentscheidungen

Das Frontend der Website wurde mithilfe von dem Next.js Framework aufgebaut und das Backend mithilfe vom Nest.js Framework.

Beim Frontend wurde mit create-next-app eine neue Next.js-Anwendung erstellt. Die einzelnen Seiten wurden in TypeScript-Code geschrieben und anschließend mit CSS gestaltet. Die Next.js Applikation hat uns ermöglicht schnell eine Basis aufzubauen und auch während der Entwicklung durch das Hot-Refresh System eine effiziente Frontend-Entwicklung bereitet.

Unsere Musik-Funktionen werden mithilfe der Spotify-API realisiert.

Zur Erstellung des Logos und anderer Grafiken wurde Adobe Illustrator genutzt.

Hauptsächlich genutzte Programmiersprachen sind TypeScript, HTML und CSS.

Organisatorische Entscheidungen zur Erreichung der wichtigsten Qualitätsanforderungen

Das Projekt wurde zu Beginn auf GitHub erstellt. Anschließend hat jedes Teammitglied einen Fork der main repository erstellt, an welchen weitergearbeitet wurde. Damit konnten alle Teammitglieder parallel entwickeln und anschließend in das main repository mergen.

Die Verwendung des project boards hat uns erlaubt einen guten Überblick unserer Aufgaben zu behalten. In diesem wurden immer wieder die Aufgaben jedes Teammitglieds notiert.

Als zweite Plattform wurde Discord genutzt. Dadurch wurde uns eine schnelle Kommunikation ermöglicht, falls Fragen oder Redebedarf über ein Thema entstanden ist. Auch hier haben wir wöchentlich unsere Ziele festgehalten.

Über Discord haben wir dann auch ein wöchentliches Meeting ausgemacht in dem wir uns über den aktuellen Stand unserer Arbeit, aufgekommene Fragen und neue Aufgaben ausgetauscht haben.

Querschnittliche Konzepte

Erweiterbarkeit

Mit Abgabe dieses Projekts haben wir eine Basis für unser Songguesser Spiel aufgebaut. Damit kann man das Grundkonzept des Spiels, Songs mit seinen Freunden zu erraten, nutzen. Somit haben wir unser gesetztes MVP erfüllt. Die Erweiterbarkeit ist uns beim Entwickeln aber immer ein wichtiger und zu beachtender Punkt gewesen.

Durch unsere Modularität ist der Code leicht nachvollziehbar und strukturiert aufgebaut. Das Spiel kann dadurch um viele weitere Spielmodi erweitert werden. Man könnte beispielsweise zwischen verschiedenen Musikgenres wählen und gemeinsam mit Freunden schauen, wer mehr zufällig ausgewählte Songs erraten kann.

Es wäre auch möglich eine Verknüpfung mit dem Spotify Konto des Users zu implementieren. Dadurch könnte man als Einzelspieler Spiel versuchen aus seinen eigenen Playlists oder aus seinen Top Artists Lieder zu erraten.

Es bestehen viele Möglichkeiten weitere Modis hinzuzufügen oder Optimierungen vorzunehmen.

Easy accessibility - Spotify Embed

Eines unserer Prinzipien bei der Entwicklung war von Anfang an die einfache Zugänglichkeit des Spiels. Angestrebt wurde ein Party-Game bei welches sehr schnell und großen Aufwand ohne viel Vorbereitung gestartet werden kann.

Daher haben wir uns dafür entschieden keine Accounterstellung sowie User Authentifizierung einzubauen. Das Einzige, was ein User angibt, ist ein Name, wodurch sein „Account“ erstellt wird. Dieser hält auch nur so lange wie das Spiel geht und sein Browser refreshed wird.

Um zu verhindern, dass jeder Spieler sich in sein Spotify Konto anmelden muss, haben wir mit Spotify Embed gearbeitet. Dabei handelt es sich um ein embedded Spotify Widget, bei der man einen kurzen Ausschnitt der Musik abspielen kann. Das ist genau passend für unser Spielprinzip, da nur wenige Sekunden des Songs benötigt werden. Zudem ist für das embedded Widget keine Anmeldung jedes Users nötig.

Socket.io

Die Verbindung von Frontend und Backend ist mit Sockets aufgebaut. Über diese Sockets werden von der Frontend Seite aus Nachrichten versandt. Jedes Modul unseres Backends hat ein Gateway. In diesem werden die Nachrichten abgefangen woraufhin entsprechend etwas im Backend getan wird. Genauso können vom Backend aus Nachrichten mit optionalen Anhängen über das Socket versendet werden. So haben wir ein einfaches System aufgebaut, mit welchem die Beiden Instanzen miteinander kommunizieren können und sich Pakete schicken können.

DTO - Datentransferobjekt

Wenn eine Nachricht über ein Socket versendet wird, kann ein Objekt als „Paket“ mitgeschickt werden. In vielen Fällen haben wir dafür DTOs

definiert. Diese Objekte beinhalten dann jede nötige Information, um z.B. etwas im Backend auszuführen, auf eine kommende Nachricht vom Frontend.

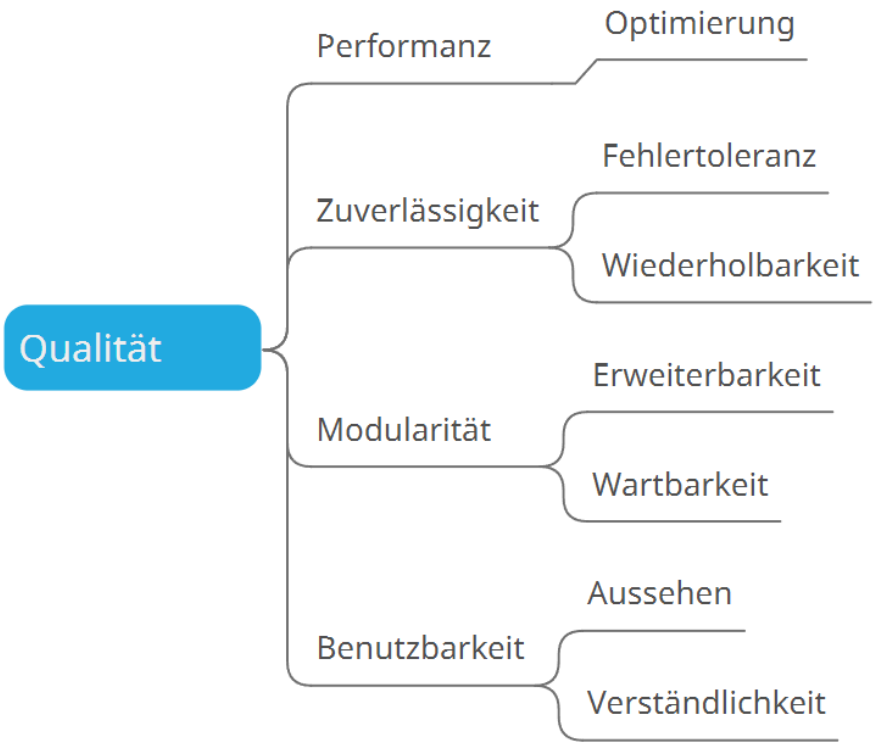
Entwurfsentscheidungen

Frameworks und Bibliotheken

Zweck	Getroffene Entscheidung	Alternativen
Framework	**Next.js**: Immer mehr genutztes Framework für populäre Applikationen. Zero Config Prinzip, welches die Applikation immer automatisch baut für eine angenehme Entwicklung. Stabile Sicherheit.	**React**: Sehr populäres Framework mit viel Support. Gute Dokumentationen und einfach zu lernen. Weniger interaktiv.
Music API	**Spotify**: Sehr weit bekannte Musikplattform. Sehr große Auswahl an Songs. Developer-freundliche API. Große Community an Developern => Troubleshooting wird vereinfacht. Ausführliche und gute Dokumentationen.	**Deezer**: Keine API-Authentifizierung. Eine sehr große Auswahl an Songs. Etwas weniger bekannte API. **Amazon Music API**: Wenig Funktionen ohne Amazon Devloper Account oder Authentifizierung.

Qualitätsanforderungen

Qualitätsbaum



Qualitätsszenarien

Qualitätsziel	Szenario
Modularität - Erweiterbarkeit	Ein Entwickler will neue Funktionen hinzufügen und eventuell einen neuen Spielmodus erstellen. Durch die Modularität der Hauptfunktionen ist das nachvollziehen der schon vorhandenen Prozesse sehr leicht. Ein Entwickler kann schnell die Struktur der Applikation verstehen und ein Modul mit neuen Funktionen erweitern oder ein neues Modul erstellen.
Benutzbarkeit - Verständlichkeit	Eine Gruppe aus Freunden möchte eine Runde gemeinsam spielen. Dadurch, dass das Ganze eine Webapplikation ist, kann man einfach über viele verschiedene Plattformen auf das Spiel zugreifen. Durch die intuitiv gestaltete Benutzeroberfläche kann sehr schnell ein Raum erstellt werden, ohne dass Accounts erstellt werden müssen, woraufhin die anderen Spieler

	dem Spiel beitreten können. Der Aufwand ein gemeinsames Spiel zu erstellen ist minimal.
Modularität - Wartbarkeit	Ein Entwickler stößt auf einen Fehler und möchte diesen beheben. Viele Funktionen werden in der Konsole gelogged, sodass ein Entwickler schnell nachvollziehen kann, welche Funktionen durchgeführt und welche eventuell fehlgeschlagen sind. Durch die Modularität ist es auch sehr einfach einzelne Prozesse unabhängig voneinander zu betrachten und zu Debuggen.

Risiken und technische Schulden

Risiko	Eventualfallplanung	Risikominderung
Aufwand der Implementierung Da dies für einige Teammitglieder die erste größere Entwicklung an einer Webapplikation ist, kommen zu den technischen Hürden auch ein zu erreichender Lernprozess hinzu. Durch die geringe Anzahl an Teammitgliedern ist eine komplett fehlerfreie Implementation in der gegebenen Zeit nicht gewährleistet.	Herunterbrechen der Funktionalität auf das grundlegende Spiel ohne viele andere optionale Funktionen. Auf der Basis dieser Demo kann, falls realistisch, dann ein größeres und stabileres Spiel aufgebaut werden mit mehr Funktionen.	Der Arbeitsaufwand wird durch das Festlegen eines MVP reduziert, erreichbare und motivierende Ziele gesetzt, und ein funktionelles Endprodukt, das die grundlegende Idee des Spieles vermittelt, produziert. Trotz der Verkleinerung des Scopes kann das Spiel auch nach der Projektarbeit erweitert werden.