

Audit 4

**Ermöglichung einer demokratisch basierten Musikwahl bei
Partys und Veranstaltungen**

Bogdan Krenz, Carlos Bystron



Der Inhalt

Wo wir hin wollten •

Wo wir gelandet sind •

Änderungen seit Audit 3 •

Reflexion •

Ein Rückblick - Wo wir hin wollten

Projektziel

Bei Veranstaltungen soll die Entscheidung über die gespielte Musik demokratisiert werden. Dem/Der Entscheidenden über die Musik (Host) muss dazu eine Möglichkeit geboten werden zu erfahren, welche Musik den meisten Veranstaltungsgästen (Guests) gefällt.

Strategische Ziele

- Veranstaltungsgäste müssen Musik teilen können, die Ihnen gefällt ohne viel Zeit für die Suche aufzubringen.
- Hosts müssen Musikvorschläge erhalten, die möglichst vielen Gästen gefallen
- Es muss ein lauffähiger Prototyp erstellt und getestet werden
- Veranstaltungsgäste sollten während der Veranstaltung die Qualität der gespielten Musik bewerten können.
- Hosts sollten während der Veranstaltung um live Feedback bitten und dieses einsehen können

Zu Anfang des Projektes, zum ersten Audit, wurde diese Zielhierarchie definiert. Ein kurzer Blick zurück soll helfen an die Zielsetzung zu erinnern und anhand dieser den Projekterfolg zu bewerten.

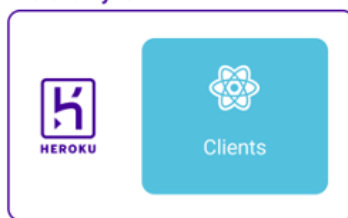
Vertikaler Prototyp - Wo wir gelandet sind



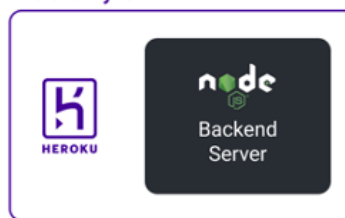
Client Anwendung <https://party-together.herokuapp.com>

Server Zugang <https://party-together-server.herokuapp.com/parties>

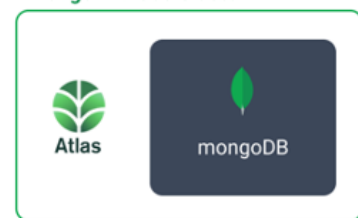
Heroku Dyno 1



Heroku Dyno 2



mongoDB Atlas Cluster 1



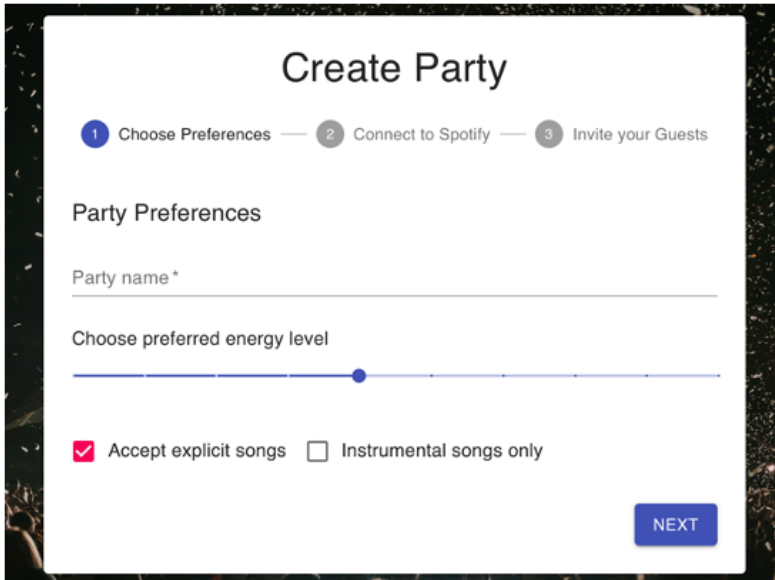
Änderungen seit dem 3. Audit

Anwendungslogik – Party Dashboard - Voting - Einladungen

- Anwendungslogik wurde implementiert, getestet und iteriert
- Partys können im Backend unterschieden werden und somit mehrere Veranstaltungen parallel laufen
- Der Host kann Gäste mit einem Link zur Party einladen, die Share-Button oder einen QR Code nutzen
- Vote-Funktion für Nutzer ohne Spotify wurde eingeführt
- Beim Erstellen der Party kann der Host Details über Party festlegen
- Party Dashboard wurde mit Echtzeitupdates ausgestattet um Votes sofort anzeigen zu können

Hier aufgelistet sind die Änderungen am vertikalen Prototypen, die seit dem dritten Audit umgesetzt wurden. Aufgrund des Feedbacks welches wir für Audit drei erhalten haben, haben wir uns im Anschluss konkrete Ziele für die Umsetzung des finalen Prototypens gesteckt. Diese konnten wir vollständig erreichen. Auf den folgenden Folien gehen wir konkret auf die wichtigsten Änderungen ein.

Anwendungslogik – Party Preferences



The screenshot shows a 'Create Party' form with three steps: 1. Choose Preferences, 2. Connect to Spotify, and 3. Invite your Guests. The first step is active. It includes a 'Party name' input field, a 'Choose preferred energy level' slider, and two checkboxes: 'Accept explicit songs' (checked) and 'Instrumental songs only' (unchecked). A 'NEXT' button is at the bottom right.

Realisierung

- Beim Erstellen der Party wählt der Veranstalter mit dem Slider sein gewünschtes Energielevel für die Party
- Außerdem kann er angeben ob nur jugendfreie Inhalte gewünscht sind und ob lediglich Instrumentalmusik gespielt werden soll

Aufgrund des Feedbacks für Audit 3 lag der Schwerpunkt bei der Implementierung zuerst klar auf der Umsetzung der Anwendungslogik. War diese für Audit 3 lediglich in Pseudocode konzipiert so ist sie nun vollständig implementiert, getestet und mehrfach iteriert.

Die Grundidee zur Ermittlung passender Lieder hat sich dabei nicht geändert, die konkrete Umsetzung ist jedoch durch mehrere Iterationen angepasst. Auch die Schwerpunkte bei der Implementierung haben sich nach ersten Nutzertests verschoben.

Konkret heißt das folgendes:

Für Audit 3 schrieben wir: „Zuerst wird [...] ein gewünschtes Audioprofil der Party ermittelt“. Dies geschieht im fertigen Prototypen über die auf dieser Folie dargestellte Eingabemaske. Der Partyveranstalter kann hier über den Slider die gewünschte Stimmung für seine Party wählen. Eine niedrige Eingabe heißt dabei das eine ruhige Party mit leiser, akustischer Musik gewünscht ist. Je größer das Energielevel desto lauter und schneller wird die empfohlene Musik. Außerdem erhält der Host, wie in Audit 3 angedacht, die Möglichkeit nur instrumental Songs für seine Party zu

wünschen oder explizite Songs auszuschließen.

Anwendungslogik – Party Preferences

```
switch (party.preferredEnergy) {  
  case 1:  
    prefTempo      = [0, 100]  
    prefDanceability = [0, 1]  
    prefLoudness    = [-60, -5]  
    prefEnergy      = [0, 0.4]  
    prefAcousticness = [0.6, 1]  
    break;  
  case 10:  
    prefTempo      = [120, 400]  
    prefDanceability = [0.6, 1]  
    prefLoudness    = [-16, 0]  
    prefEnergy      = [0.5, 1]  
    prefAcousticness = [0, 0.3]  
    break;  
}
```

Realisierung

- Im Backend findet sich für jede der zehn Stufen des Sliders ein ein Case in einer Switch Anweisung
- Es werden passende Wertebereiche für Tempo, Lautstärke, Tanzbarkeit usw. festgelegt
- Durch Zuweisung von Energielevel zu Musik-Wertebereichen kann dem Host ein einfaches und schnelles Anlegen der Party ermöglicht werden

Im Backend findet im Folgenden eine Zuweisung von Energielevel zu Musik Wertebereichen statt. Im obigen Screenshot sehen wir die beiden Extremszenarien des niedrigsten sowie höchsten Energielevels. Betrachten wir diese kurz im Detail:

Um den Wunsch ruhiger Hintergrundmusik für ein gemütliches Zusammensitzen zu ermöglichen legen wir fest, dass Songs nicht schneller sein sollten als 100 Takte pro Minute. Die Energie des Songs sollte Werte unter 0,4 aufweisen, die *Acousticness* also *die* Wahrscheinlichkeit, dass ein Song rein akustische Elemente nutzt sollte bei über 60% liegen.

Schaut man sich im Vergleich dazu die Wertebereiche für ein wünscht es Energielevel von zehn an, so sieht man, dass beispielsweise das Tempo passender Songs bei mindestens 120 Tacken pro Minute liegen sollte. Auch die gewünschten Werte für Lautstärke und Energie sind, logischer weise, deutlich höher. Die Wahrscheinlichkeit, dass ein Song rein akustisch ist sollte hingegen mit unter 30% sehr gering ausfallen.

Selbstverständlich wurde im Prozess der Implementierung ebenfalls darüber nachgedacht alle Wertebereiche durch den Host festlegen zu lassen. Wir entschieden uns jedoch dagegen, um dem Partyveranstalter einen schnelleren Erstellungsprozess

der Party zu ermöglichen. Viele der Werte, die durch den Host hätten festgelegt werden müssen, sind nicht allgemein bekannt und bedürfen einer ausführlichen Erklärung. Für eine zukünftige Verbesserung unserer Anwendung könnte über eine Option zum eigenständigen wählen der Wertebereiche nachgedacht werden.

Diese sollte jedoch lediglich ergänzend, für besonders Interessierte Nutzer, angeboten werden.

Anwendungslogik - partyFit

```
// outlierScore get called for each partyValue to check if the songs value fit in the expected range
// all outlier scores are added and get deducted from one afterwards if the value doesn't exceed one
// if yes the party fit is 0, otherwise it's a percentage => highest for the best fit
const partyFit = outlierScore(song.tempo, prefTempo) * 0.01 + outlierScore(song.danceability, prefDanceability)
                + outlierScore(song.energy, prefEnergy) + outlierScore(song.loudness, prefLoudness) * 0.01
                + outlierScore(song.acousticness, prefAcousticness)

return partyFit <= 1 ? (1 - partyFit) * 100 : 0

// outlierScore checks if a value is inside a specified range
// if yes 0 is returned, if not it returns how far the value is outside the range

/**
 * @param {Float} songProfileValue the value of a song profile stat
 * @param {Array<Float>} preferredRange range the value should lay in
 */
function outlierScore(songProfileValue, preferredRange){
  if (preferredRange[0] <= songProfileValue && songProfileValue <= preferredRange[1]){
    return 0
  } else if (preferredRange[0] > songProfileValue) {
    return preferredRange[0] - songProfileValue
  } else {
    return songProfileValue - preferredRange[1]
  }
}
```

Ziel der Anwendungslogik ist es, wie bereits im Pseudocode für Audit 3 verdeutlicht, einen individuellen partyFit-Score (in Prozent) für jeden durch die Gäste hinzugefügten Song zu ermitteln.

Bis zu dem hier dargestellten Codesegment kommen lediglich die Lieder, die bereits die vorausgehende Prüfung nach Ausschlusskriterien überstanden haben. Bei diesem wird beispielsweise geprüft, ob es sich bei dem Lied nicht um gesprochene Inhalte, wie beispielsweise Titel aus einem Hörbuch, handelt. Bei den Ausschlusskriterien wird ebenfalls die Prüfung jugendfreier Inhalte, sofern vom Host gewünscht, vorgenommen. Sollte ein Song bei der Prüfung nach Ausschluss Kriterien durchfallen, so wird ein partyFit von -1 zurückgegeben.

Für die Songs, die eine Prüfung der Ausschlusskriterien bestehen, wird der partyFit im obigen Codesegment berechnet. Die Funktion outlierScore nimmt dabei jeweils einen Musikwert des Liedes, welches geprüft wird, sowie den vorher im Switch Case festgelegten, gewünschten Wertebereich für die jeweilige Party. Liegt der Wert des Liedes dabei im Wertebereich wird 0 zurückgegeben, liegt er außerhalb erfolgt die Abweichung als Rückgabe.

Die Abweichungen werden daraufhin addiert und im Anschluss von 1 abgezogen. Die Werte für Tempo und Lautstärke werden dabei mit 0,01 multipliziert da ihre Wertebereiche nicht von 0-1, sondern in größeren Skalen angegeben werden. Gibt es für einen Song also keine Abweichungen, da er allen Kriterien perfekt entspricht, wird ein partyFit von 100% zurückgegeben.

Echtzeit Updates

```
io.on("connection", (socket) => {  
  
  socket.on("host", (partyID) => {  
    console.log("New host at Party", partyID, "connected")  
    socket.join(partyID)  
    getRoomAndEmit(io, partyID)  
  })  
  
  socket.on("guest", (partyID) => {  
    console.log("New guest at Party", partyID, "connected")  
    getRoomAndEmit(io, partyID)  
  })  
  
  socket.on("satisfactionVote", (partyID, vote) => {  
    getRoomAndEmit(io, partyID, vote)  
  })  
  
  socket.on("disconnect", () => {  
    console.log("Client disconnected")  
  });  
})
```

Realisierung

- Echtzeit Updates wurden mit Hilfe von Socket.io umgesetzt
- Für Host werden in Räume eingeteilt, die mit Hilfe der partyID unterschieden werden können
- Echtzeit Updates werden beim aktualisieren der Playlist und bei der Zufriedenheitsbefragung genutzt

Eine der größten Neuerungen ist die Einführung von echtzeit Updates für das Dashboard. Während erster Nutzertests fiel auf, dass eine zeitliche Kopplung der Aktualisierung des Party Dashboards und dem beitreten neuer Gäste einen großen Mehrwert für unsere Anwendung bietet.

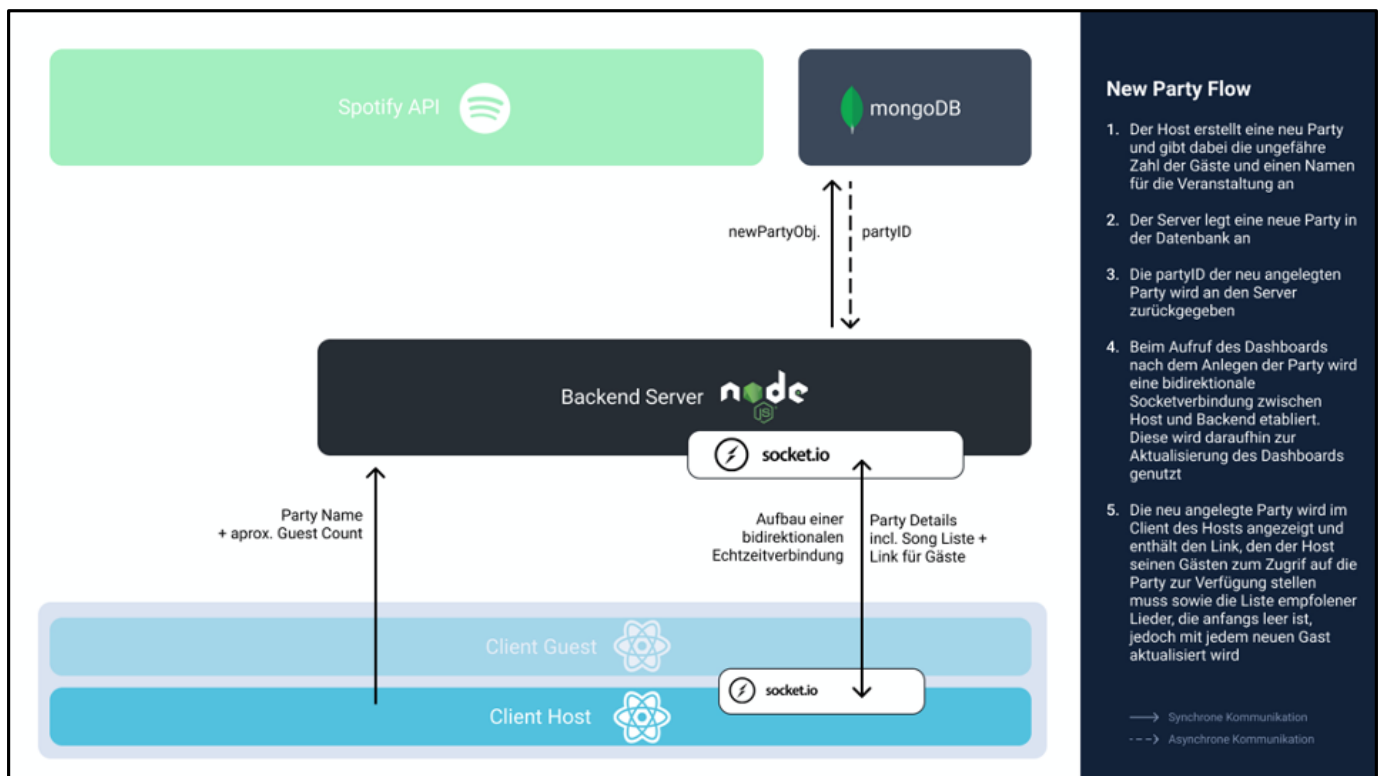
Die Alternativen, ein eigenständiges Refreshen durch den Dashboard Nutzer oder ein konstantes Refreshen mithilfe eines Timers stellten sich als wenig praktikabel, beziehungsweise unnötig beanspruchend für das Backend heraus. Des Weiteren ist durch das echtzeit Update Feature die Möglichkeit geschaffen live Feedback durch die Zuhörenden zu erhalten.

Zur Umsetzung der Echtzeit Updates wurden Sockets genutzt. Nach einer ausführlichen Recherche entschieden wir uns für die Nutzung des Anbieters Socket.io, eine API die bidirektionale Echtzeitkommunikation und ereignisbasierte Kommunikation unterstützt.

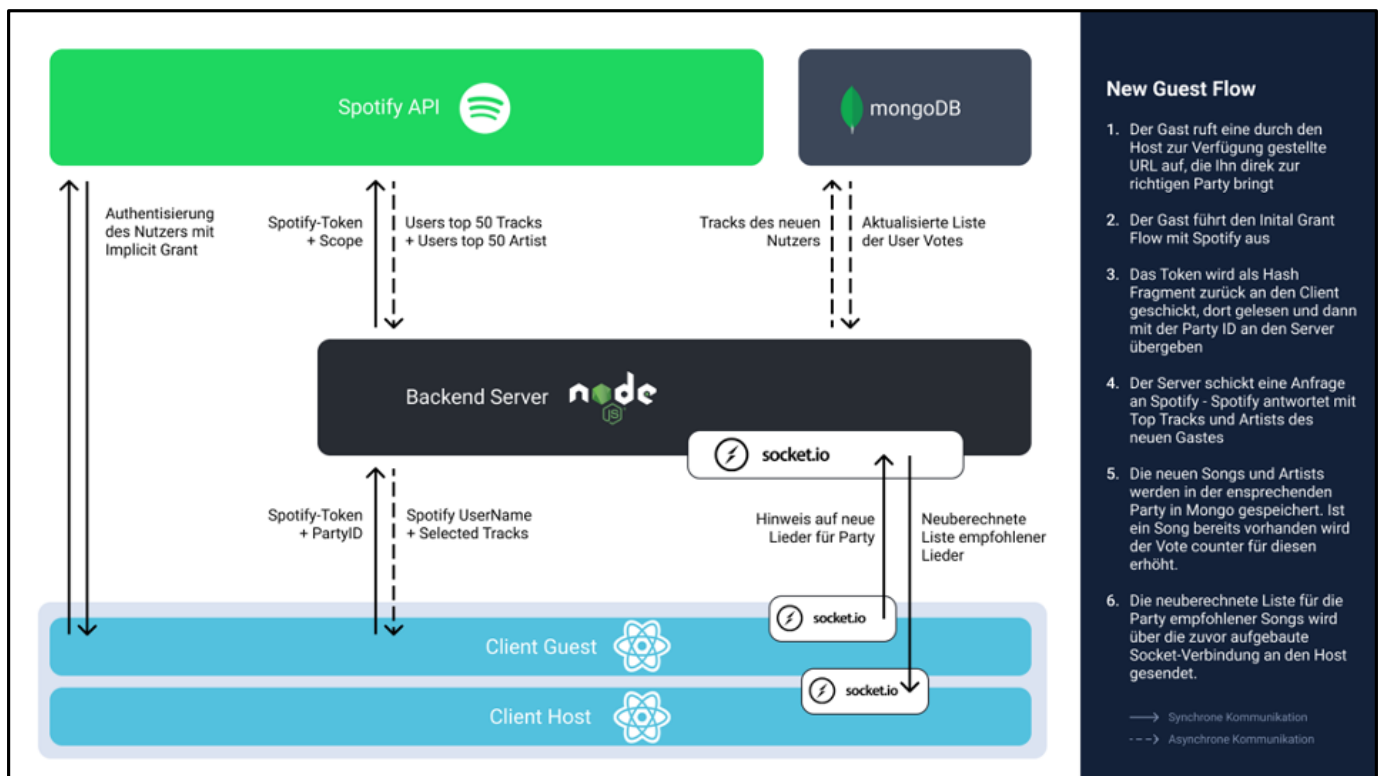
Ein großer Vorteil von Socket.io für unsere Anwendung lag in der Nutzbarkeit von Räumen. Für unseren Anwendungszweck wurde jeder Party ein Raum zugewiesen, welcher mithilfe der Party ID identifiziert werden kann. Die Clients der Host treten

diesen Räumen bei, die Clients der Guests senden ihre Informationen daraufhin diese Räume. Somit können Broadcast Nachrichten an alle Hosts vermieden und dennoch kann eine Nutzung mehrerer Dashboards ermöglicht werden.

In der Funktion *getRoomAndEmit*, welche im obigen Screenshot zu sehen ist, wird unterschieden ob diese mit oder ohne den Parameter *vote* aufgerufen wird. Wenn die Funktion mit dem Parameter *Vote* aufgerufen wird, sendet sie das Ergebnis der Zufriedenheitsbefragung eines Gastes an den Host. Dort wird dieses Abstimmungsergebnis dann in Relationen zu den Ergebnissen anderer Gäste gesetzt und im Dashboard angezeigt. Wird die Funktion lediglich mit dem Parameter *partyID* aufgerufen, wird die aktualisierte Playlist gesendet.

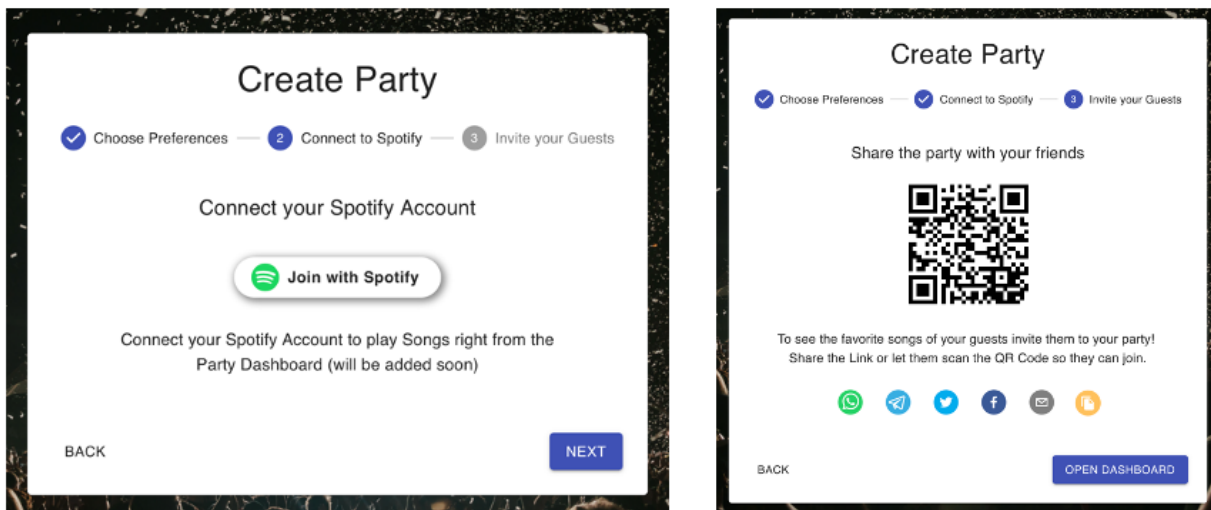


Die beiden wichtigsten User Flows unsere Anwendung sind im Vergleich zu Audit drei größtenteils gleich geblieben. Lediglich aufgrund der Einbindung von Socket.io ergeben sich einige Unterschiede. Somit stellt der Client des Hosts nun beim ersten Aufruf des Party Dashboards eine Verbindung zum Socket des Servers her. Über diese Verbindung werden daraufhin jegliche Aktualisierungen der Playlist sowie der Zufriedenheitsbefragung der Nutzer übertragen.



Tritt ein neuer Gast der Party bei so ist auch dieser Ablauf größtenteils unverändert. Die Authentisierung mit Spotify findet statt, die top Tracks des Nutzers werden von Spotify gefetched und an den Nutzer zurückgegeben. Neu ist, dass der Client des Nutzers darauf hin über eine Socket Verbindung den Client des Hosts über die Aktualisierung der Playlist informiert. Daraufhin werden über die Socket-Verbindung des Hosts die neuen Lieder ausgetauscht.

Redesign des Frontends – Party Erstellung



Im Vergleich zum letzten Audit wurde ebenfalls eine umfassende Änderung des Frontends vorgenommen. Erstmalig ist das Design vollständig responsive und die Screens des Host-Flows für eine Ansicht am Desktop optimiert. Der Guests-Flow wurde hingegen unter dem Paradigma Mobile-First erstellt. Zum Redesign wurden Material Design Komponenten genutzt.

Diese bieten diverse Vorteile:

- Sie sind vollständig getestet und den Nutzern durch diverse Web Anwendungen sowie die Nutzung auf Android Smartphones vertraut.
- Accessibility Features (Barrierefreiheit) sind nativ vorhanden
- Eine Anpassung des Designs über globale Variablen wie Abstände und Farben ist einfach möglich

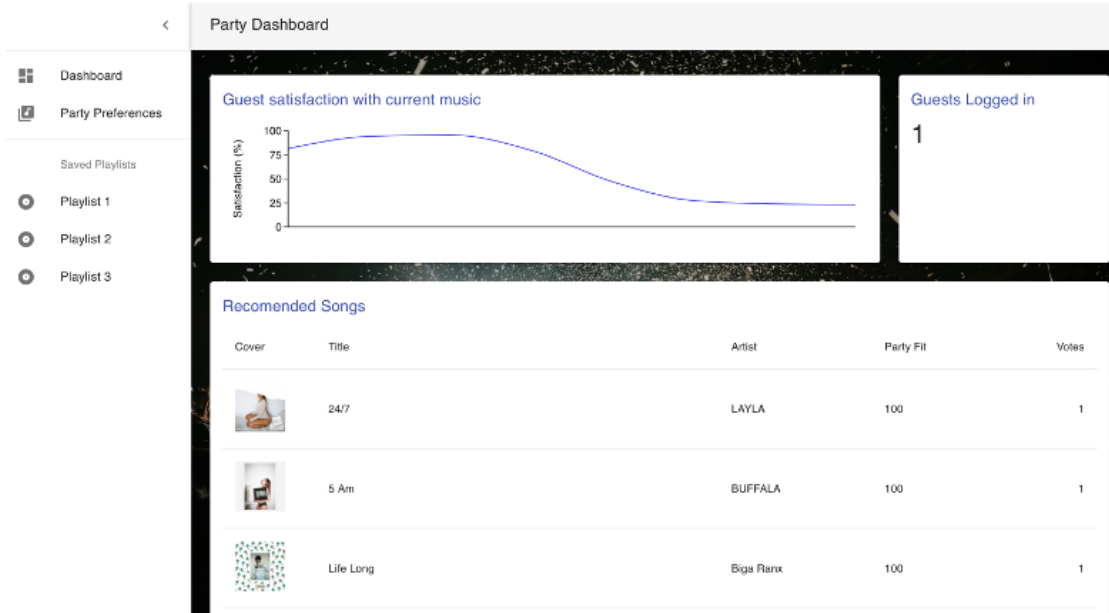
Im Folgenden werden wir nun auf die einzelnen Screens eingehen. Die Party-Preference Maske des Host-Flows wurde bereits auf Folie 6 behandelt weshalb wir diese hier auslassen.

Im zweiten Schritt der Party Erstellung wird idealerweise die Connection zu Spotify für den Host aufgebaut, um die empfohlenen Lieder direkt über Spotify abzuspielen

oder oder als Playlist abzuspeichern. Dieser Schritt wurde jedoch aufgrund von Zeitmangel und anderweitigen Prioritäten ausgelassen, da sie keine kritische Anforderung des MVP darstellt.

Im letzten Schritt wird dem Host die Möglichkeit gegeben die Party mit seinen Guests zu teilen. Sei es über Social-Media-Kanäle, Messenger Dienste, per Mail, QR-Code oder Link.

Redesign des Frontends – Party Dashboard



Der nächste Schritt führt den Host zum Dashboard, dem Herzen unserer Anwendung, wo ihm die Playlist, der User Count und anderer Informationen zur Verfügung gestellt werden.

Die Daten werden Live geladen und jede der Kacheln einzeln gerendert. Ändert sich beispielsweise die Playlist müssen der User Count und der Graf welcher die Zufriedenheit der Gäste mit der gerade gespielten Musik zeigt nicht aktualisiert werden.

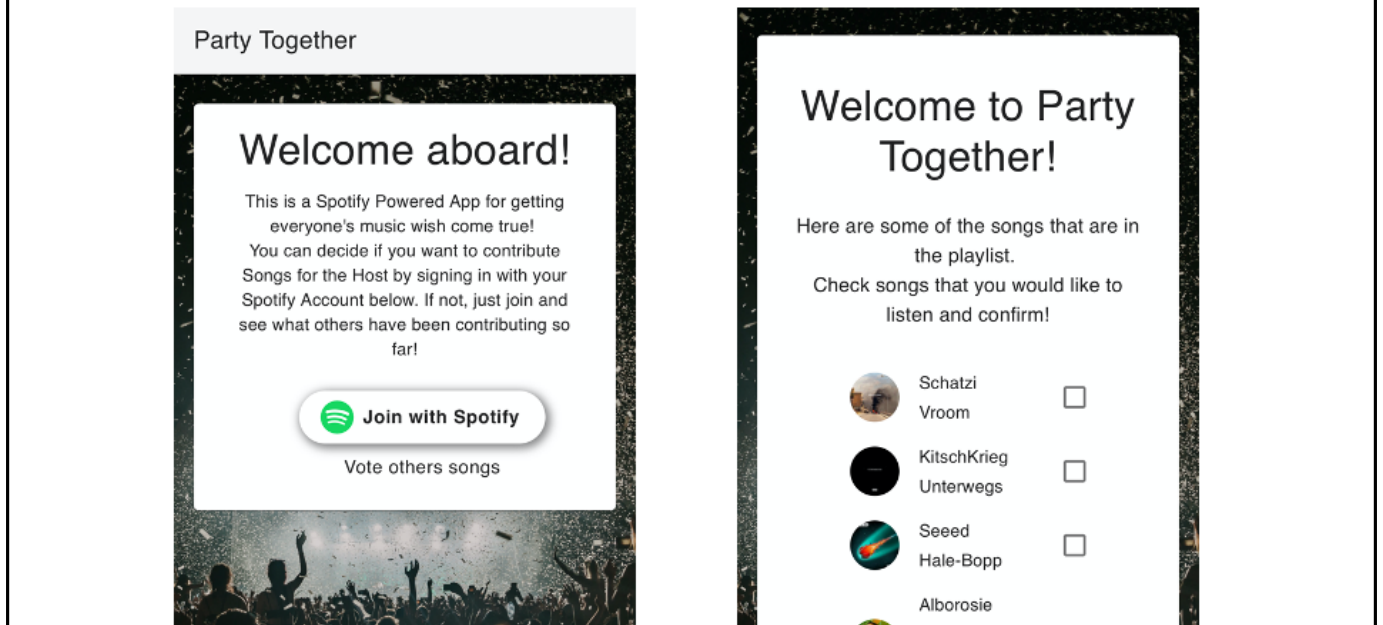
Dieser Screen soll eine Vorstellung darüber geben, wie das Dashboard potenziell mit vollen Funktionalitäten, die es nicht alle in unseren Prototypen geschafft haben, aussehen könnte.

Über das Menü links, dass sich auch einklappen lässt, soll es später möglich sein die Partypräferenzen während der laufenden Veranstaltung zu verändern. Des Weiteren sollen Playlists aus der aktuellen angezeigten Liste erstellt und gespeichert werden können.

In einer späteren Iteration wäre über die Möglichkeit eines Kontos für die Party Veranstalter nachzudenken. Dieses würde den Vorteil bieten, dass der Host das

Dashboard schließen und es später über seinen Account erneut erreichen kann.
Außerdem könnte so der Zugriffs auf das Dashboard durch Dritte verhindert werden.

Redesign des Frontends – Guest Flow



Der User Flow für die Party Gäste wurde ebenfalls überarbeitet.

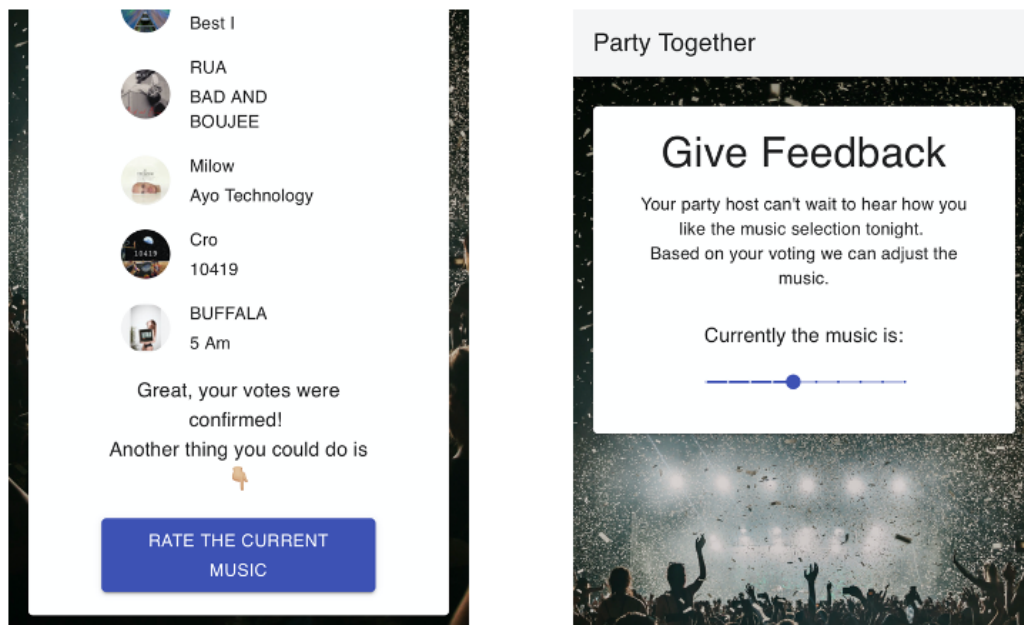
Wenn der Guest auf die URL aus den Socials oder anderen Kanälen zugreift, wird er auf den Screen weitergeleitet, der ihn in der App Willkommen heißt. Wie in den Use Cases beschrieben wird hier zwischen einem Guest mit Spotify Account und einem Guest ohne Spotify Account unterschieden.

Der Guest hat die Möglichkeit seine Lieblingslieder auf die Liste des Hosts zu setzen indem er sich mit seinem Spotify Account einloggt. Den Rest erledigt der Server im Hintergrund. Wenn der Guest auf den „Join with Spotify“ Button klickt, wird er zu Spotify weitergeleitet und es wird nach seinem Einverständnis gefragt. Wird diese gegeben, landet der Guest auf einem Screen der ihm zehn seiner Lieblingssongs anzeigt, um zu bestätigen, dass die Verbindung erfolgreich war.

Der Screenshot rechts zeigt die Seite welche angezeigt wird wenn man die Anwendung ohne Spotify Account nutzt. Hier werden den Nutzern die 30 Lieder angezeigt, die momentan die Playlist anführen. Die Guests können dann Songs aussuchen die ihnen gefallen und diese markieren. Weiter unter, am Ende der Liste befindet sich ein „Confirm“ Button, um die Auswahl zu bestätigen und das Ergebnis

an den Server zu schicken. Im Server wird dann die Liste mit den Votes bearbeitet und dem Host wird live die aktualisierte Liste angezeigt. So wird der Nutzern, auch ohne Spotify Account, die Möglichkeit geboten aktiv auf die Liste der gespielten Lieder Einfluss zu nehmen.

Redesign des Frontends – Guest Flow



Ebenfalls neu implementiert wurde die Möglichkeit über die persönliche Zufriedenheit mit der aktuellen Musik abzustimmen. Diese Funktion wurde bereits als strategisches Ziel in der Zielhierarchie formuliert. Zu Ende der Implementationsphase musste die Entscheidung getroffen werden ob diese Live Abstimmungsfunktion oder die Funktion Lieder direkt aus der Anwendung abzuspielen umgesetzt wird.

Ein Blick in die anfänglich aufgestellte Zielhierarchie sowie eine Rückbesinnung auf die zentrale Aufgabe der Anwendung veranlasste uns das live Feedback umzusetzen. Diese Funktion ist insbesondere für große Veranstaltungen sehr nützlich, da es dem DJ nicht möglich ist auf andere Art und Weise zu erfahren wie die Gäste seine gespielte Musik finden. Die technische Umsetzung erfolgt ebenfalls über Socket.io. Es handelt sich um eine direkte Kommunikation von Client zu Client, der Server dient lediglich als Gateway. Die Abstimmungsergebnisse werden nicht persistent gespeichert, da sie lediglich kurzzeitige Relevanz besitzen.

Dinge, die wir gerne noch ändern würden

Neue Strategische Ziele

- Musikwiedergabe vom Dashboard ermöglichen
- Benachrichtigungen implementieren
- Den Musikwahl Algorithmus verbessern und eine eigene explorative Suche hinzufügen
- Die Routes absichern um mehrfaches Abstimmen zu unterbinden
- Die Anwendung für kleiner Partys vollständig automatisieren („DJ-less“-Mode)

Obwohl wir, insbesondere in den letzten Wochen, mit Hochdruck an dieser Anwendung gearbeitet haben ist es uns leider trotzdem nicht möglich gewesen alle Ideen und Vorstellungen vollständig zu implementieren.

Zuerst ist da, ganz klar, die Musikwiedergabe direkt vom Dashboard. Dies ist ein Feature, bei welchem wir davon ausgehen, dass es hauptsächlich bei kleineren Veranstaltungen genutzt würde, da DJs meist professionelle Software nutzen. Für eine Wohngemeinschaft oder bei Haus Partys könnte sich diese Funktion jedoch als äußerst nützlich erweisen, da so ein einmaliges Starten der Playlist ohne eine zuständige Person für die Musik möglich wäre.

Eines des Features welches wir als Optional angesetzt hatten, sind die Push Notifications, um Gäste beispielsweise während der Veranstaltung an das Abstimmen über ihre aktuelle Zufriedenheit mit der gespielten Musik zu erinnern. Diese Funktion sehen wir nach wie vor kontrovers, denn im Vordergrund des Projektes stand durchweg die Regel, dass die Gäste so wenig Zeit wie möglich an ihren Geräten verbringen und sich der Veranstaltung widmen sollten.

Die kritischste Schwäche des Prototyps sind zur Zeit die ungeschützten Routes. Das

hat zur Folge, dass man wenn man den Browser refresht, wieder auf die Liste der Lieder voten und somit das Ergebnis stark beeinträchtigen kann. Dieses Problem ist in nächsten Iterationen priorisiert zu lösen.

Ständig zu verbessern ist sicherlich der Musikwahl Algorithmus. Auch wenn dieser bereits im vertikalen Prototypen gute Ergebnisse liefert, wäre beispielsweise eine explorative Suche anhand einiger weiterer Eingabedaten des Party Veranstalters (bspw. zu Genre und Referenzinterpreten) nützlich, um zu vermeiden, dass anfangs keine Musik Vorschläge gemacht werden können.

Eine weitere interessante Idee und konsequente Fortsetzung des erstgenannten Punktes wäre eine vollständige Automatisierung der Anwendung in einem Modus der für Veranstaltungen ohne DJ vorgesehen ist. Bei diesem könnten unter anderem die Abstimmungsergebnisse der Zufriedenheits Befragung dazu genutzt werden automatisch die Playlist anzupassen.

Auch wenn wenn man diese Liste mit Sicherheit noch deutlich fortsetzen könnte lässt sich ebenfalls festhalten, dass wir von den anfänglich formulierten strategischen Zielen für das Entwicklungsprojekt alle erfolgreich umsetzen konnten. Somit sind wir mit dem Ziel Erreichungsgrad zum jetzigen Zeitpunkt absolut zufrieden.

Hinsichtlich des Prozesses lässt sich sagen, dass ein früherer Beginn mit der Implementierung sicherlich dazu beigetragen hätte das Projekt zum jetzigen Zeitpunkt in einem noch fortschrittlicheren Stadium vorstellen zu können. Insbesondere in der Vorweihnachtszeit sind die Projektaktivitäten teilweise etwas zu kurz gekommen. Darüber hinaus sind wir mit der eigenen Vorgehensweise weitesgehend zufrieden. Insbesondere wenn man in Betracht zieht, dass die Zusammenarbeit ausschließlich digital und ohne jegliche persönliche Treffen ablief kann man von einem reibungslosen Verlauf sprechen.



Unter diesem Motto haben wir mit Hochdruck an der Umsetzung einer Anwendung gearbeitet, die dazu beitragen kann, dass in Zukunft mehr Menschen an dem Auswahlprozess für die Musik auf Veranstaltungen beteiligt sein können. Auch wenn es sicherlich diverse Punkte gibt in denen unsere Anwendung noch verbessert werden kann und muss sind wir mit dem erreichten Ergebnis zufrieden. Bereits im jetzigen Stadium der App sehen wir einen klaren Mehrwert für Partys und Veranstaltungen und werden daran arbeiten diesen auch nach Abschluss des Entwicklungsprojekts noch weiter zu steigern.