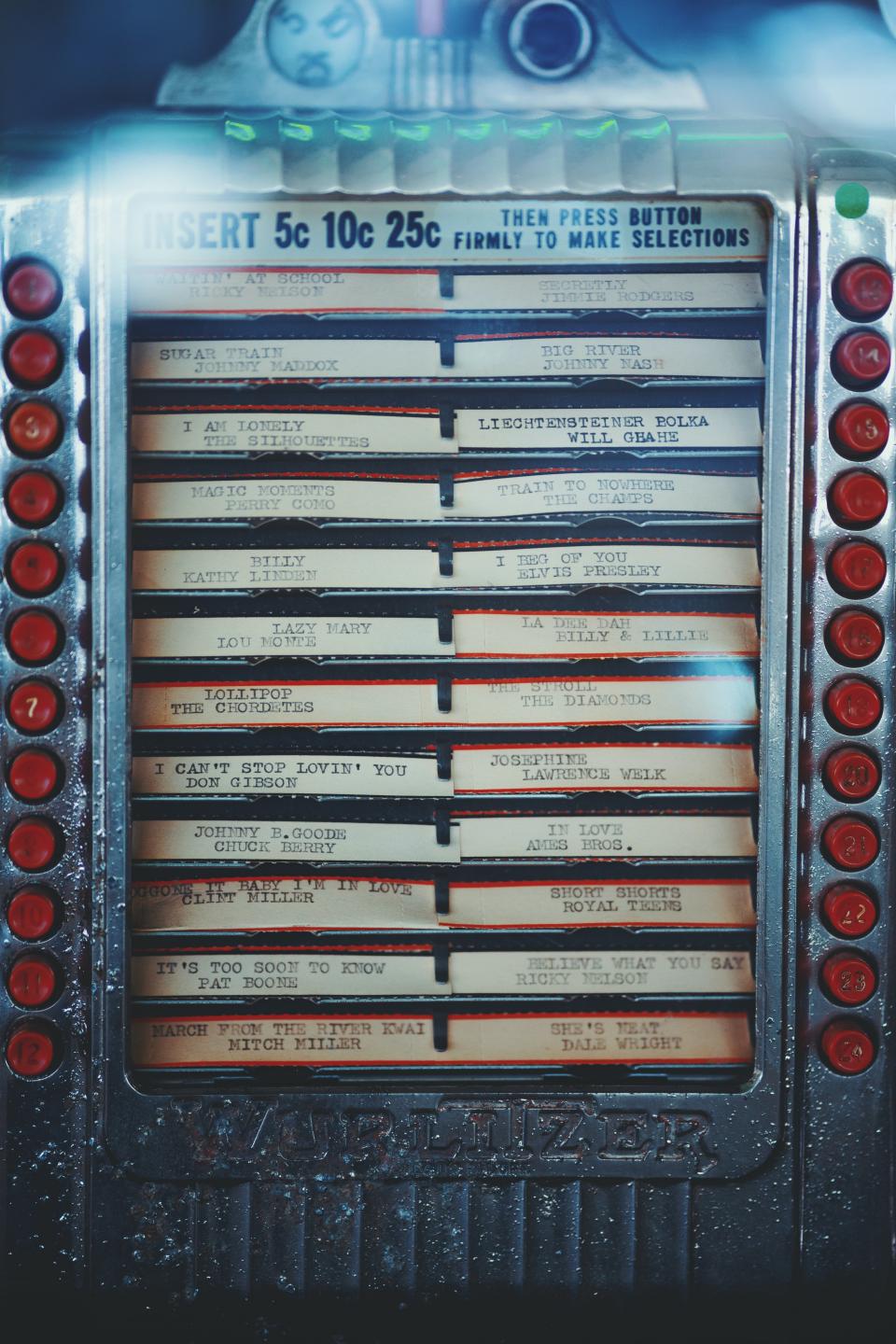


# **Audit 4**

**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

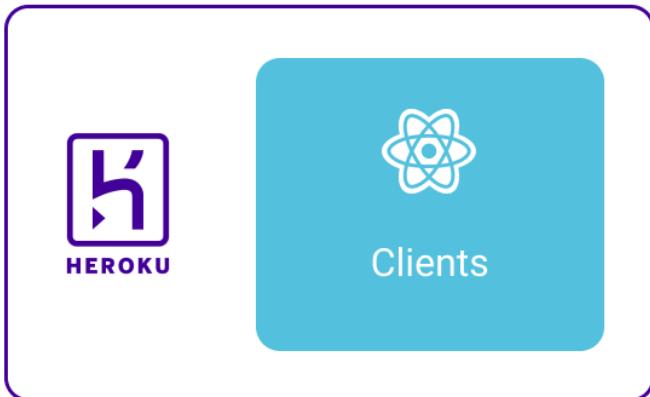
# Vertikaler Prototyp - Wo wir gelandet sind



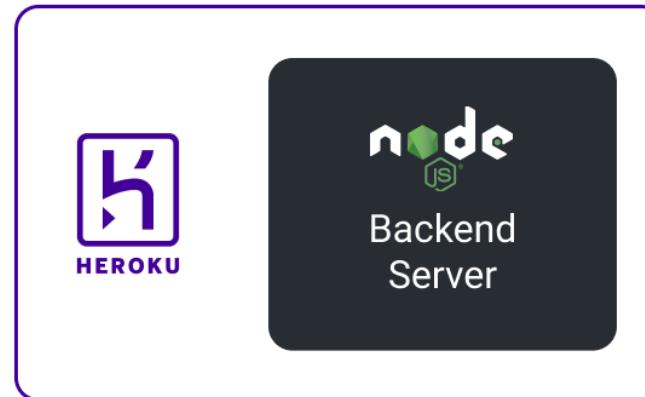
Client Anwendung <https://party-together.herokuapp.com>

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

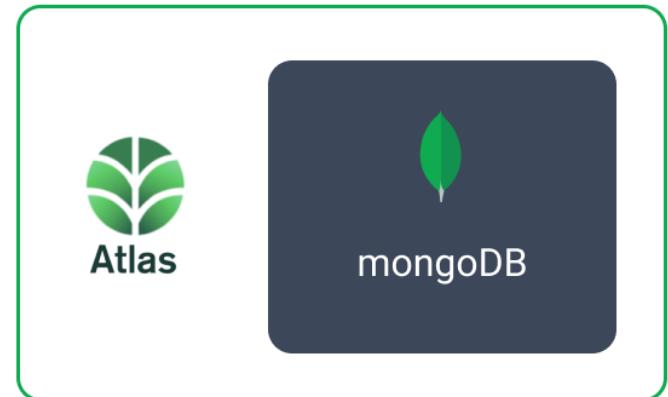
Heroku Dyno 1



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

# Anwendungslogik – Party Preferences

The screenshot shows a mobile application interface titled "Create Party". At the top, there is a navigation bar with three steps: "1 Choose Preferences", "2 Connect to Spotify", and "3 Invite your Guests". The current step is "Choose Preferences".

The main section is titled "Party Preferences". It includes a field for "Party name\*" with a placeholder "Party name\*". Below it is a slider titled "Choose preferred energy level" with a blue dot indicating the current level.

At the bottom, there are two checkboxes: one checked with the text "Accept explicit songs" and another unchecked with the text "Instrumental songs only". A large blue "NEXT" button is located at the bottom right.

## Realisierung

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

# 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 Energilevel zu Musik-Wertebereichen kann dem Host ein einfaches und schnelles Anlegen der Party ermöglicht werden

# Anwendungslogik - partyFit

```
// outlierScore get called for each partyValue to check if the songs value fit in the expeckted 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]
    }
}
```

# Echtzeit Updates

```
io.on("connection", (socket) => {

  socket.on("host", (partyID) => {
    console.log("New host at Party", partyID, "connected")
    socket.join(partyID)
    getRoomAndEmit(io, partyID)
  })

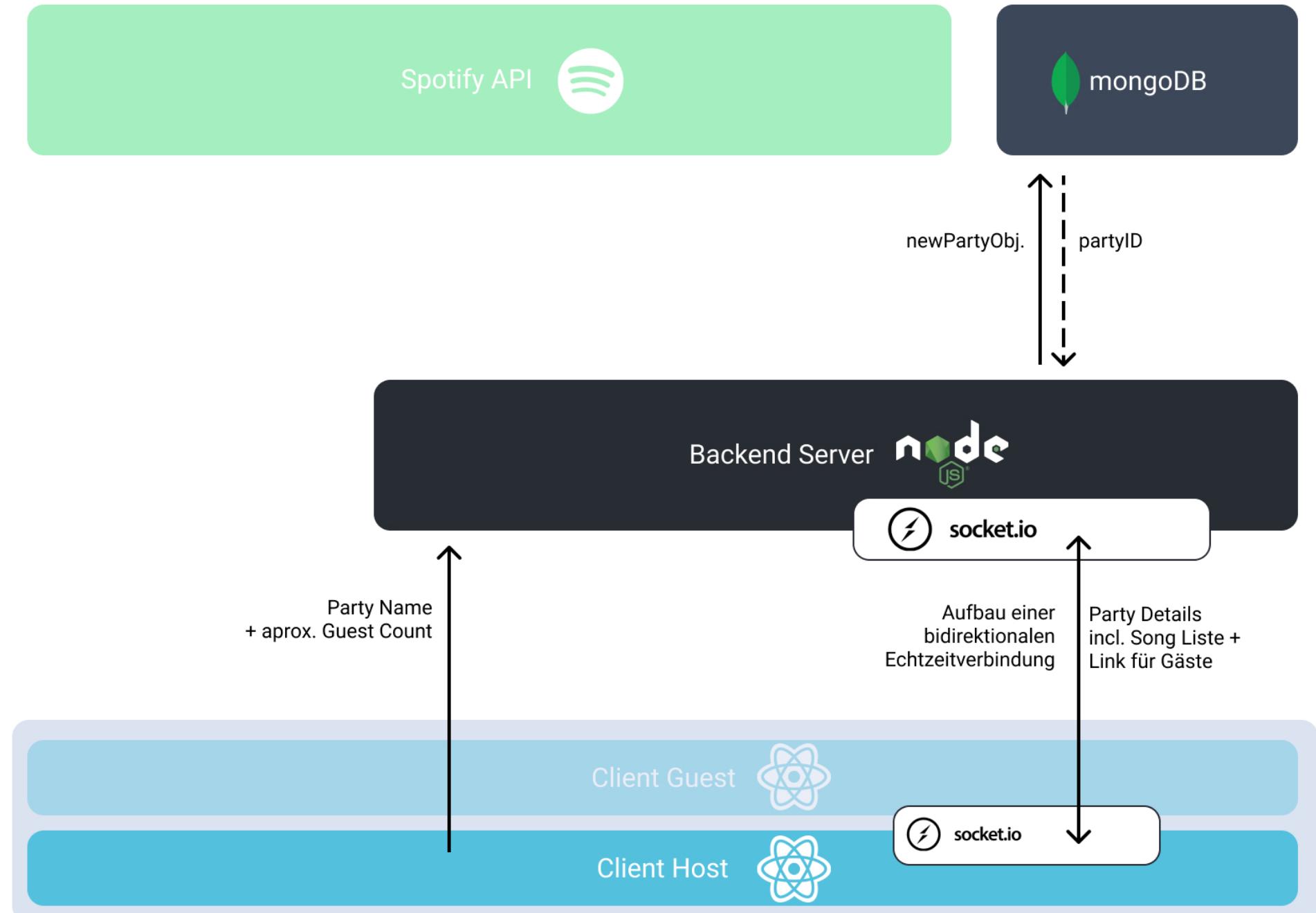
  socket.on("guest", (partyID) => {
    console.log("New guest at Party", partyID, "connected")
    getRoomAndEmit(io, partyID)
  })

  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

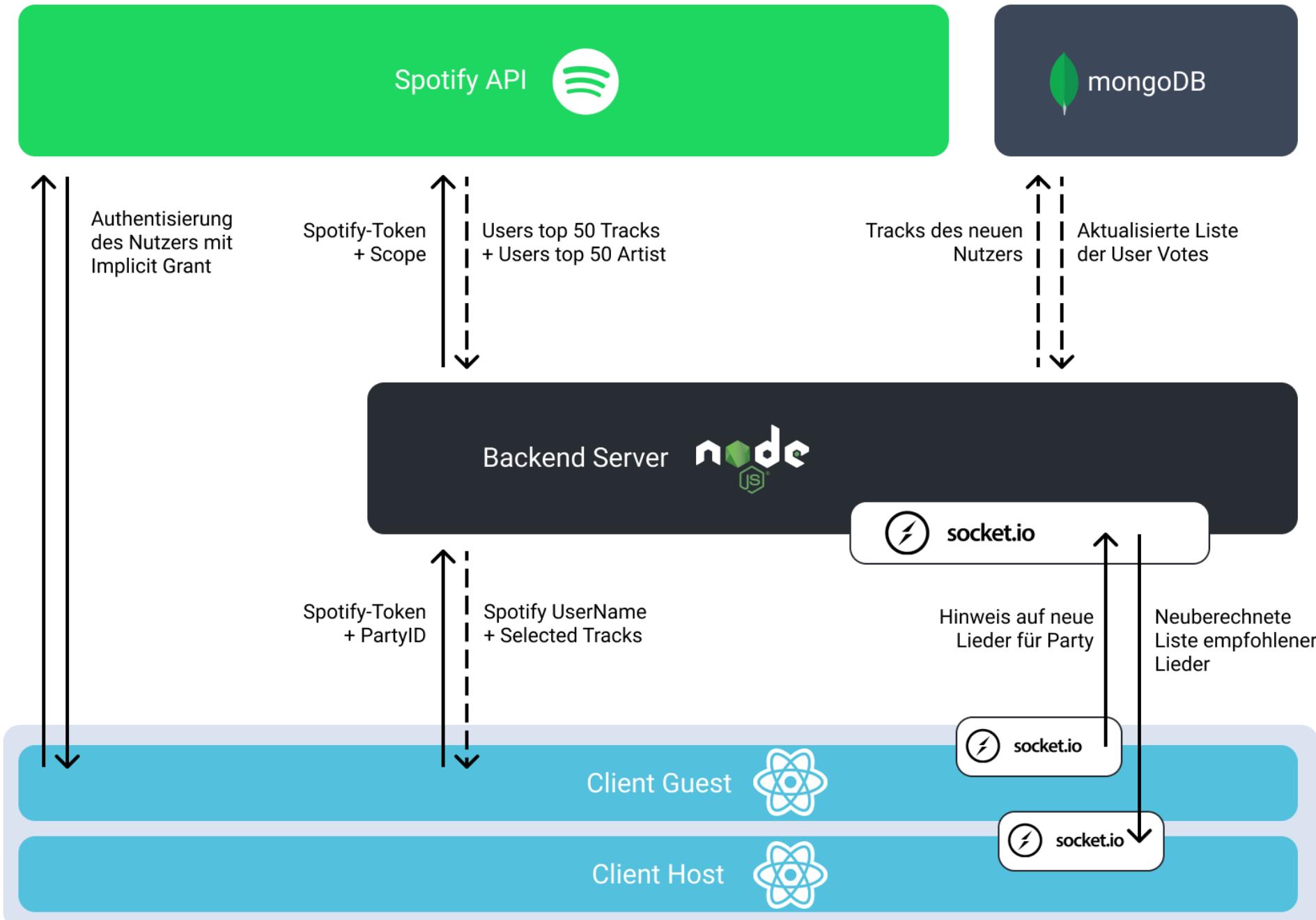


## New Party Flow

- Der Host erstellt eine neu Party und gibt dabei die ungefähre Zahl der Gäste und einen Namen für die Veranstaltung an
- Der Server legt eine neue Party in der Datenbank an
- Die partyID der neu angelegten Party wird an den Server zurückgegeben
- Beim Aufruf des Dashboards nach dem Anlegen der Party wird eine bidirektionale Socketverbindung zwischen Host und Backend etabliert. Diese wird daraufhin zur Aktualisierung des Dashboards genutzt
- Die neu angelegte Party wird im Client des Hosts angezeigt und enthält den Link, den der Host seinen Gästen zum Zugriff auf die Party zur Verfügung stellen muss sowie die Liste empfolener Lieder, die anfangs leer ist, jedoch mit jedem neuen Gast aktualisiert wird

→ Sychrone Kommunikation

→ Asynchrone Kommunikation



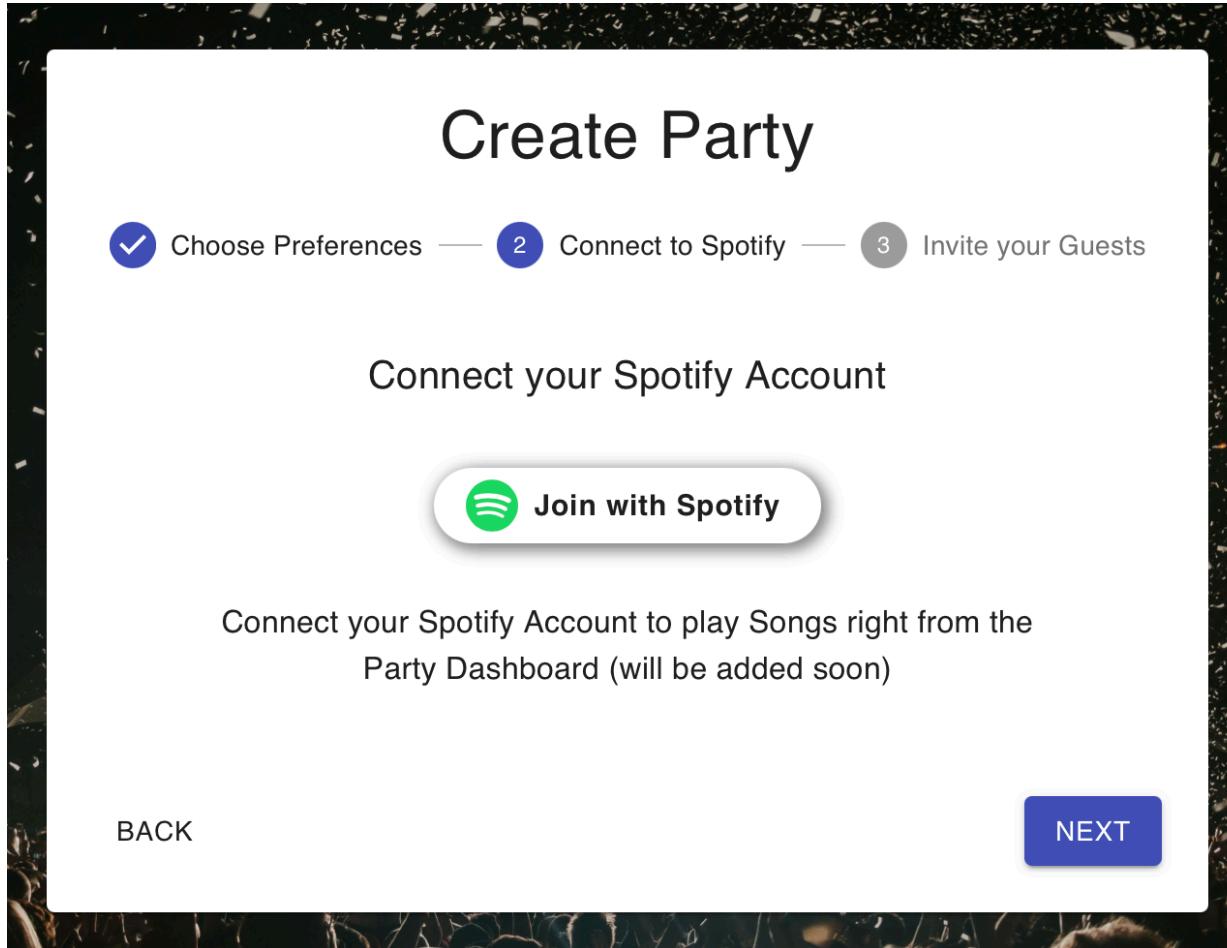
## New Guest Flow

- Der Gast ruft eine durch den Host zur Verfügung gestellte URL auf, die ihn direkt zur richtigen Party bringt
- Der Gast führt den Initial Grant Flow mit Spotify aus
- Das Token wird als Hash Fragment zurück an den Client geschickt, dort gelesen und dann mit der Party ID an den Server übergeben
- Der Server schickt eine Anfrage an Spotify - Spotify antwortet mit Top Tracks und Artists des neuen Gastes
- Die neuen Songs und Artists werden in der entsprechenden Party in Mongo gespeichert. Ist ein Song bereits vorhanden wird der Vote counter für diesen erhöht.
- Die neuberechnete Liste für die Party empfohlener Songs wird über die zuvor aufgebaute Socket-Verbindung an den Host gesendet.

→ Synchrone Kommunikation

→ Asynchrone Kommunikation

# Redesign des Frontends – Party Erstellung



# Redesign des Frontends – Party Dashboard

< Party Dashboard

- Dashboard
- Party Preferences
- Saved Playlists
- Playlist 1
- Playlist 2
- Playlist 3

Guest satisfaction with current music

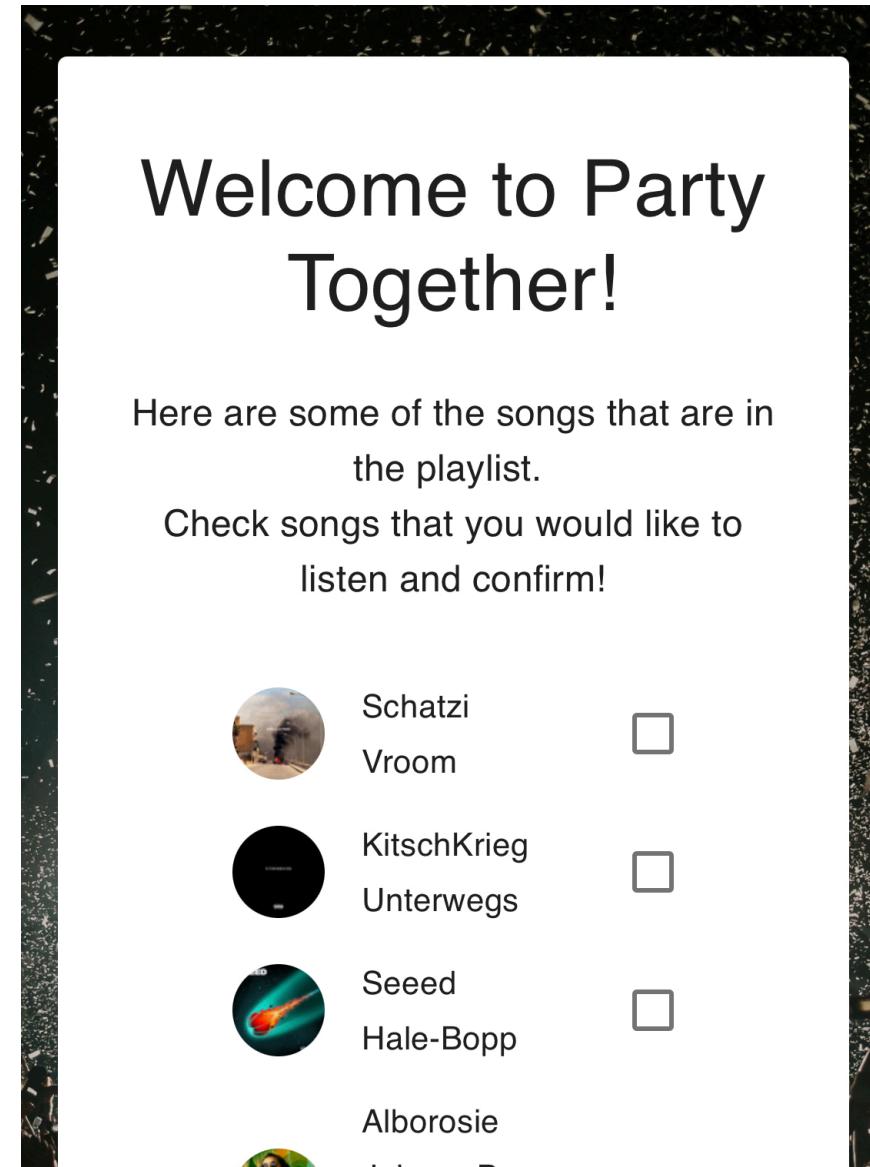
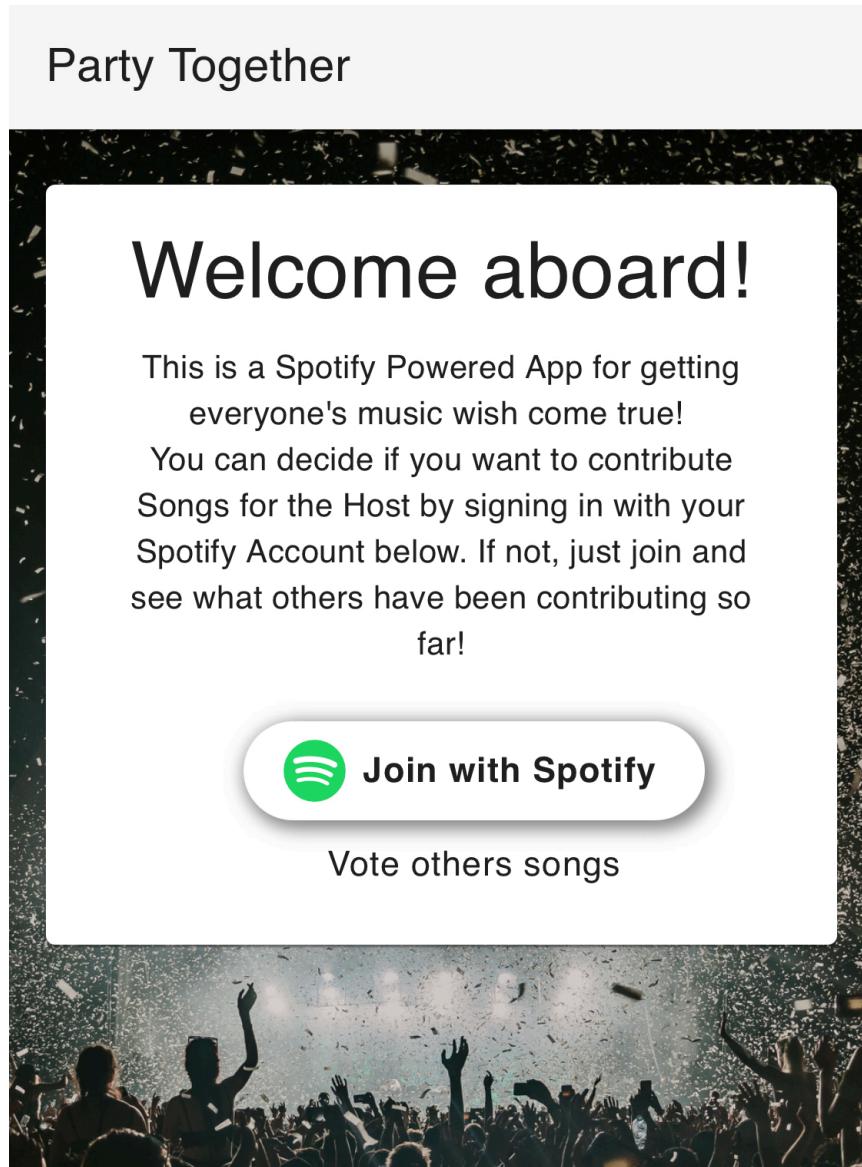
A line chart titled "Guest satisfaction with current music" showing satisfaction percentage over time. The y-axis is labeled "Satisfaction (%)" and ranges from 0 to 100. The x-axis represents time. The line starts at approximately 80%, rises to a peak of about 98% at the midpoint, and then gradually declines to around 25% by the end.

Guests Logged in  
1

Recomended Songs

Cover	Title	Artist	Party Fit	Votes
	24/7	LAYLA	100	1
	5 Am	BUFFALA	100	1
	Life Long	Biga Ranx	100	1

# Redesign des Frontends – Guest Flow



# Redesign des Frontends – Guest Flow

The interface features a vertical list of music tracks with small circular profile pictures next to each track name. The tracks listed are:

- Best I
- RUA
- BAD AND BOUJEE
- Milow
- Ayo Technology
- Cro
- 10419
- BUFFALA
- 5 Am

Below the list, a message says "Great, your votes were confirmed!" followed by "Another thing you could do is" and a downward-pointing arrow icon.

**RATE THE CURRENT MUSIC**

The interface has a header "Party Together" and a main title "Give Feedback". It includes a message from the host: "Your party host can't wait to hear how you like the music selection tonight. Based on your voting we can adjust the music." Below this, it says "Currently the music is:" followed by a horizontal slider with a blue dot in the middle. The background of the slide shows a blurred image of a crowded party with confetti.

# 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)

*you are* ♪  
*What you listen to*