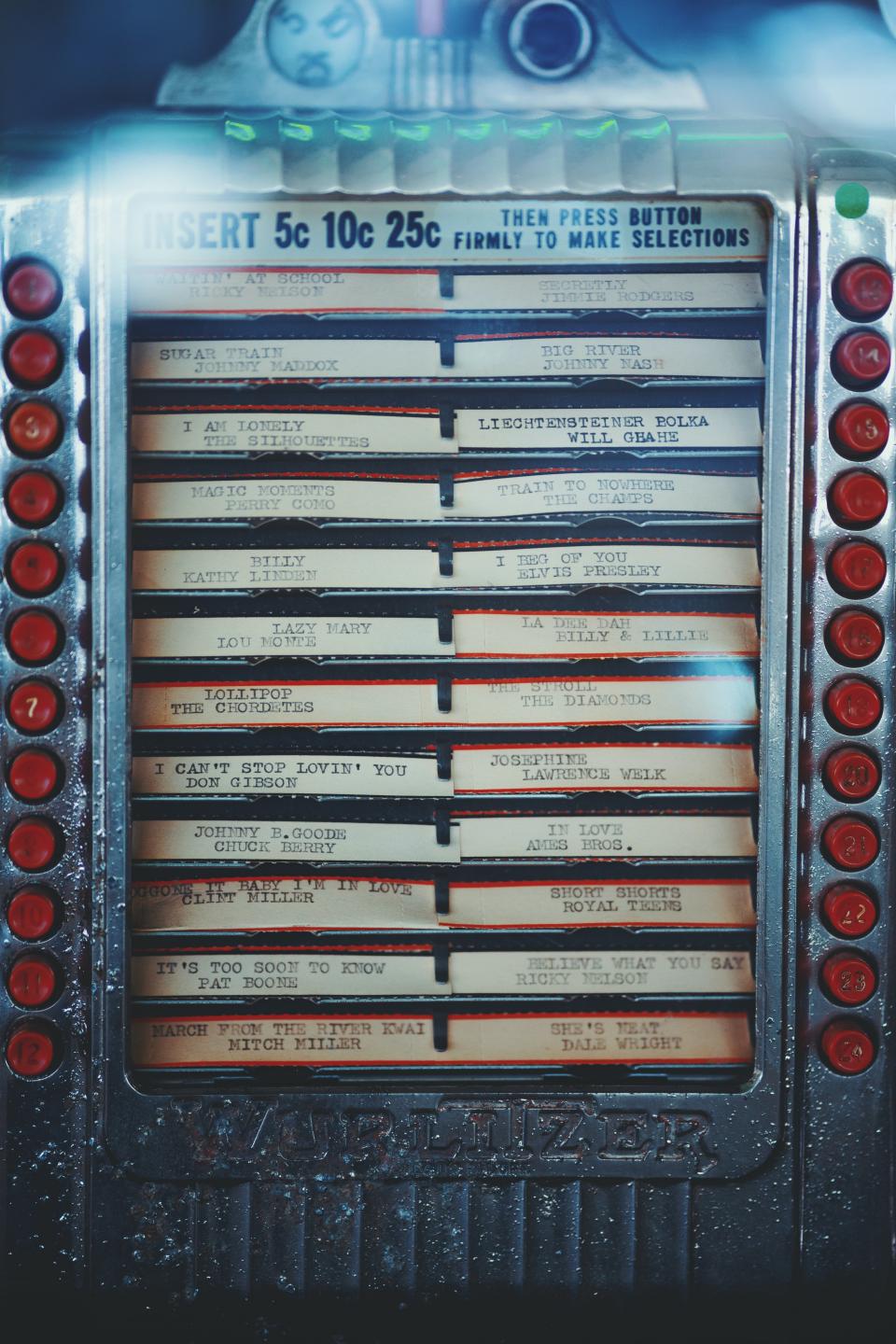


# **Audit 2**

**Ermöglichung einer demokratisch basierten Musikwahl bei  
Partys und Veranstaltungen**  
**Bogdan Krenz, Carlos Bystron**



# Der Inhalt

- Erfordernisse •
- Anforderungen •
- User Journeys •
- Projektrisiken •
- Proof of Concepts •
- Projektplan •

# Erfordernisse

- Der Guest braucht Zugang zu seinen Playlists
- Der Guest braucht eine Möglichkeit mit dem Host zu kommunizieren ohne dass der Host mit jedem Guest in direkten Kontakt kommt
- Der Host braucht eine Möglichkeit um Feedback von den Guests einzuholen
- Der Host bestimmt die gespielte Musik, hat jedoch die Möglichkeit Musik auszuwählen, die den Guests auch gefallen wird
- Der Guest will nicht den ganzen Abend am Handy verbringen, um Musik für die Veranstaltung auszusuchen
- Der Guest und der Host brauchen eine Möglichkeit miteinander zu kommunizieren ohne dabei den Flow der Veranstaltung zu unterbrechen

# Anforderungen

- Das System soll eine Kommunikation zwischen dem Host und den Guests bereitstellen, ohne dass sich die gesamte Party vor den DJ versammelt
- Das System soll eine Feedback Feature beinhalten, mit Hilfe dessen der Host eine Rückmeldung von den Guests erhält
- Das System soll dem Host auf Anfrage die passenden Lieder in einer Liste bereitstellen
- Das System soll den User authentisieren, um die Einzigartigkeit des Users zu gewährleisten, um keine Manipulation der Playlists durchführen zu können
- Das System soll zwischen verschiedenen Veranstaltungen unterscheiden können
- Die Guests müssen in der Lage sein ihre passende Veranstaltung ohne großen Aufwand zu suchen und beitreten
- Dem Guest soll die Möglichkeit geboten werden auf Events des Host auf Wunsch zu subscriben
- Das System soll nur Daten nutzen, die für die Umsetzung der Ziele des Projekts relevant sind um das Recht auf Datenschutz des Users nicht zu verletzen

# Vier Arten von Usern

**Guest1** nutzt die volle Anwendungslogik

**Guest2** nutzt nur das Teilen von Musik

**Guest3** nutzt nur das Feedback System

**Host** nutzt das System für das Feedback von den Usern und den Musikwunsch der Guests zu erfahren

## User Journey #1 (Guest #1)

1. Der Guest ist auf der Homepage, meldet sich mit seinem Spotify Account an.
2. Dem Guest wird Vorgeschlagen auf zukünftige Events zuzugreifen.
3. Der Guest bestätigt die Teilnahme an Events und Teilt im nächsten Screen seine Musik mit.
4. Dem Guest wird die Liste angezeigt die an das System geschickt werden kann.
5. Das System bearbeitet die Liste und schickt sie weiter an den Host.
6. Der Guest hat die Möglichkeit auf Events zu reagieren um zu signalisieren, dass ihm die Musik gefällt / nicht gefällt.

## **User Journey #2 (Guest #2)**

1. Der Guest ist auf der Homepage, meldet sich mit seinem Spotify Account an.
2. Dem Guest wird Vorgeschlagen auf zukünftige Events zuzugreifen.
3. Der Guest lehnt die Teilnahme an Events ab und Teilt im nächsten Screen seine Musik mit.
4. Dem Guest wird die Liste angezeigt die an das System geschickt werden kann.
5. Das System bearbeitet die Liste und schickt sie weiter an den Host.

## **User Journey #3 (Guest #3)**

1. Der Guest erstellt einen Acc. um sich bei dem Dienst anzumelden.
2. Der Guest sucht seine Veranstaltung und tritt dieser bei.
3. Dem Guest wird Vorgeschlagen auf zukünftige Events zuzugreifen.
4. Der Guest bestätigt die Teilnahme an Events und Teilt im nächsten Screen seine Musik mit.
5. Der Guest hat die Möglichkeit auf Events zu reagieren um zu signalisieren, dass ihm die Musik gefällt / nicht gefällt.

## **User Journey #4 (Host)**

1. Der Host meldet sich mit seinem Spotify Account an.
2. Der Host erstellt eine Veranstaltung.
3. Der Host erhält im laufe der Veranstaltung eine vom System erstellte Liste von Liedern, die den HörerInnen gefallen
4. Der Host startet ein Event, um von den Gästen Feedback zu bekommen, ob die gespielte Musik den Erwartungen entspricht.

# Die Projektrisiken

Architekturell	Kommunikation	NutzerInnen
<ul style="list-style-type: none"><li>• Starke Abhängigkeit von Drittanbietern (Projektidee steht und fällt mit APIs)<ul style="list-style-type: none"><li>– Während der Entwicklung können neue Restriktionen für die Nutzung der API eingeführt werden</li><li>– Datenstrukturen und Zugriffe können geändert werden</li><li>– Die Anbieter können die Kosten für die Nutzung frei bestimmen</li></ul></li><li>• Datenunabhängige Struktur kann nicht gewährleistet werden</li><li>• Integrität der Datenbank kann nicht sichergestellt werden</li></ul>	<ul style="list-style-type: none"><li>• Datenbankabfragen könnten bei Veranstaltungen mit vielen Gästen zu lange dauern</li><li>• Während des Aktualisierens der Liste vorgeschlagener Lieder (wenn ein Lied gespielt wurde oder neue Vorschläge integriert werden) ist diese zu lange nicht sichtbar und der Host weiß nicht welche Musik gespielt werden soll</li><li>• Die Authentifizierung beim Streaming Anbieter ist nicht wie geplant mit einem redirect zur App des Anbieters möglich</li></ul>	<ul style="list-style-type: none"><li>• NutzerInnen könnten Datenschutzbedenken haben und ihre Streamingdaten nicht teilen wollen</li><li>• Login Prozess könnte zu lange dauern und NutzerInnen vergraulen</li><li>• Keine Übereinstimmungen bei den Musikwünschen der Zuhörenden</li><li>• NutzerInnen wissen Benutzername und Passwort ihres Streaming Anbieters nicht</li></ul>

# Die Projektrisiken - 2

Technisch	Organisation	Kompetenzorientiert
<ul style="list-style-type: none"><li>• Bottlenecks bei Server u. Datenbank</li><li>• Persistenz -&gt; Daten können bei Systemabsturz verloren gehen (besonders gespielte Lieder u. Präferenzen der Zuhörenden)</li><li>• Das identifizieren verschiedener Versionen des gleichen Songs kann scheitern -&gt; Song wird mehrmals gespielt bzw. nicht obwohl er vielen gefällt</li><li>• Die API des Streaming Anbieters ist instabil und zwischenzeitlich nicht erreichbar</li><li>• Der Push Notification Service funktioniert anders als erwartet, kann Notifications nicht zustellen bzw. den User nicht benachrichtigen</li><li>• Die benötigten Ressourcen werden durch die API nicht zur Verfügung gestellt</li><li>• Die Anwendungslogik zum finden passender Lieder kommt in angemessener Zeit zu keinem Ergebnis</li></ul>	<ul style="list-style-type: none"><li>• Es besteht Gefahr, dass lediglich Musikvorschläge generiert werden, die für eine Party unbrauchbar sind (Musik zu langsam, zu traurig, etc.)</li><li>• Die Anwendungslogik schafft es nicht eine faire Verteilung der gespielten Musik zu ermöglichen (Musik die nur einem/wenigen NutzerInnen gefällt)</li></ul>	<ul style="list-style-type: none"><li>• Fehlende Erfahrung bei der Umsetzung großer Projekte insbesondere Zeitmanagement</li><li>• Fehlende Erfahrung mit dem versenden von Push Notifications</li></ul>

# PoC 1 - User Authentication

## Beschreibung

Im bestmöglichen Fall soll eine Authentifizierung der NutzerInnen über eine Weiterleitung zur App des Anbieters auf dem Smartphone erfolgen. Dies soll durch die Nutzung von Deeplinks ermöglicht werden. Dort kann dann der Nutzung zugestimmt und die zu teilenden Daten gewählt werden. Dies ermöglicht ein Anmelden ohne die Eingabe eines Passwortes durch die NutzerInnen. Es ist davon auszugehen, dass der überwiegende Teil der NutzerInnen die App ihres jeweiligen Streaming Dienstes auf dem Smartphone installiert hat. Für die NutzerInnen, die die App nicht installiert haben soll eine Authentifizierung mit Email und Passwort ermöglicht werden.

## Abgedeckte Projektrisiken

- Die Authentifizierung beim Streaming Anbieter ist nicht wie geplant mit einem redirect zur App des Anbieters möglich
- Login Prozess könnte zu lange dauern und NutzerInnen vergraulen
- NutzerInnen wissen Benutzername und Passwort ihres Streaming Anbieters nicht
- Die API des Streaming Anbieters ist instabil und zwischenzeitlich nicht erreichbar

## Exit-Kriterien

- Eine erfolgreiche Authentifizierung hat stattgefunden
- Die Anmeldung durch redirect ist möglich

## Fail-Kriterien

- Eine erfolgreiche Authentifizierung via redirect ist nicht möglich

## Fallbacks

- Die Anmeldung ist lediglich über Email und Passwort möglich

# PoC 2 - DB testing

## Beschreibung

Eine passendes Datenbanksystem wird gewählt ein Schema entwickelt und die DB mit ersten Testdaten gefüllt. Dabei soll insbesondere geprüft werden, ob gewünschte Zugriffsgeschwindigkeiten erreicht werden können, die Persistenz sowie Integrität gewährleistet ist und die Datenbank auch sehr große traffic, welche bei Großveranstaltungen auftreten könnte, verarbeiten kann.

## Abgedeckte Projektrisiken

- Datenbankabfragen könnten bei Veranstaltungen mit vielen Gästen zu lange Dauern
- Datenunabhängige Struktur kann nicht gewährleistet werden
- Integrität der Datenbank kann nicht sichergestellt werden
- Persistenz -> Daten können bei Systemabsturz verloren gehen (besonders gespielte Lieder u. Präferenzen der Zuhörenden)

## Exit-Kriterien

- Datenbank bildet gewünschtes Schema ab
- Datenbank kann gewünschte Zugriffsgeschwindigkeiten erreichen
- Datenbank hält hoher traffic stand

## Fail-Kriterien

- Benötigte Daten können nicht rechtzeitig bzw. gar nicht abgerufen werden
- Datenbank bricht bei großer Traffic zusammen

## Fallbacks

- Wechsel des Datenbanksystems
- Anpassen des Schemas

# PoC 3 - API und DB testing

## Beschreibung

Bei diesem PoC soll sichergestellt werden, dass die benötigten Informationen von der API abgefragt und anonymisiert in unserer systemeigenen DB gespeichert werden können. Insbesondere soll bei diesem PoC die durchschnittlich übermittelte Anzahl von Liedern sowie die Zugriffsgeschwindigkeit der API ermittelt werden. Des Weiteren sollen die request Parameter für das künftige System ermittelt und getestet werden.

## Abgedeckte Projektrisiken

- Die benötigten Ressourcen werden durch die API nicht zur Verfügung gestellt
- Persistenz -> Daten können bei Systemabsturz verloren gehen (besonders gespielte Lieder u. Präferenzen der Zuhörenden)
- Die API des Streaming Anbieters ist instabil und zwischenzeitlich nicht erreichbar

## Exit-Kriterien

- Ressourcen können in angemessener Zeit von API abgerufen werden
- Die API liefert die benötigten Daten

## Fail-Kriterien

- Benötigte Daten können nicht rechtzeitig bzw. gar nicht abgerufen werden

## Fallbacks

- Wechsel der API

# PoC 4 - Anwendungslogik und Systemarchitektur

## Beschreibung

Ein erster vertikaler Prototyp soll die Umsetzbarkeit des gesamten angedachten Systems und insbesondere die der Anwendungslogik validieren. Aus den Musikbibliotheken mindestens zweier verbundener Clients sollen Vorschläge generiert werden über deren Qualität dann von den NutzerInnen abgestimmt werden soll. Hierbei ist das Abstimmungsverfahren noch nicht in der Anwendung implementiert, es geht lediglich um eine iterative Optimierung der Anwendungslogik um die Vorschläge zu verbessern.

## Abgedeckte Projektrisiken

- Die Anwendungslogik zum finden passender Lieder kommt in angemessener Zeit zu keinem Ergebnis
- Keine Übereinstimmungen bei den Musikwünschen der Zuhörenden
- Es besteht Gefahr, dass lediglich Musikvorschläge generiert werden, die für eine Party unbrauchbar sind (Musik zu langsam, zu traurig, etc.)
- Die Anwendungslogik schafft es nicht eine faire Verteilung der gespielten Musik zu ermöglichen (Musik die nur einem/wenigen NutzerInnen gefällt)
- Die benötigten Ressourcen werden durch die API nicht zur Verfügung gestellt
- Während des Aktualisierens der Liste vorgeschlagener Lieder (wenn ein Lied gespielt wurde oder neue Vorschläge integriert werden) ist diese zu lange nicht sichtbar und der Host weiß nicht welche Musik gespielt werden soll

## Exit-Kriterien

- Es werden Vorschläge generiert die allen verbundenen NutzerInnen gefallen (Ein geeignetes Bewertungsverfahren ist zu entwickeln)
- Die Vorschläge werden in angemessener Zeit aktualisiert wenn sich ein neuer Client verbindet

## Fail-Kriterien

- Es werden Vorschläge generiert die nicht allen verbundenen NutzerInnen gefallen (Ein geeignetes Bewertungsverfahren ist entwickeln)

## Fallbacks

- Iteration und Verbesserung der Anwendungslogik

# PoC 5 - Push Notifications

## Beschreibung

Eine simple Anwendung des Notification Services zum Versenden von Push Benachrichtigungen soll die Nutzung veranschaulichen und uns helfen Möglichkeiten und Grenzen beim Versenden von Benachrichtigungen besser einschätzen zu können.

## Abgedeckte Projektrisiken

- Der Push Notification Service funktioniert anders anders als erwartet, kann Notifications nicht zustellen bzw. den User nicht benachrichtigen
- Fehlende Erfahrung mit dem versenden von Push Notifications

## Exit-Kriterien

- Benachrichtigungen können erfolgreich zugestellt werden

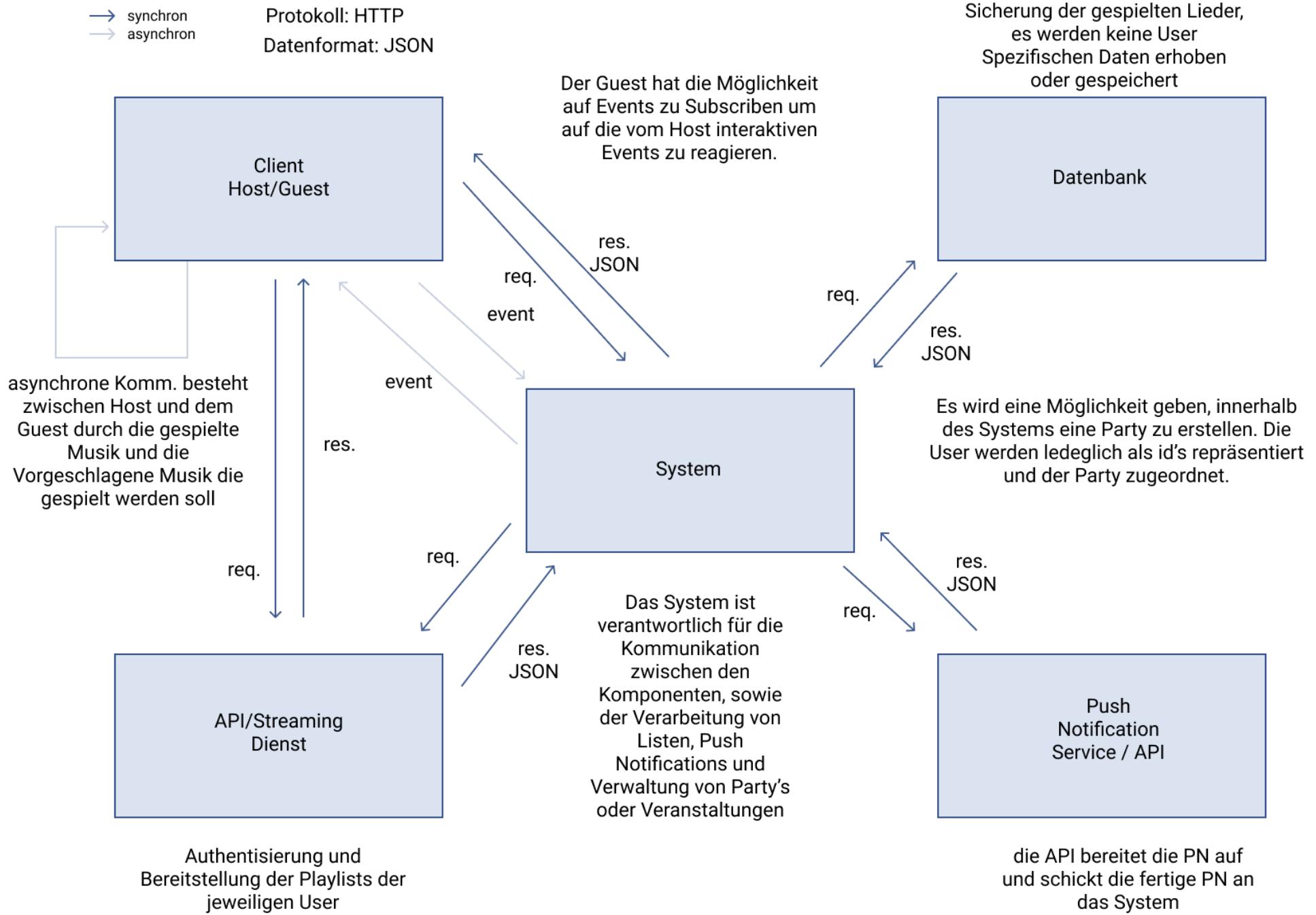
## Fail-Kriterien

- Benachrichtigungen kommen nicht an
- Benachrichtigungen sind zeitlich stark verzögert
- Benachrichtigungen werden nicht sichtbar (aufwecken des Bildschirms, ggf. Vibration oder Ton) zugestellt

## Fallbacks

- Wählen eines anderen PSN Services

# Das Architekturdiagramm 2.0



## Interface Design 0 Std. (50)

 Erstellen eines Moodboards 0 Std. (5)

 Wahl der Designziele 0 Std. (5)

 Wireframes 0 Std. (20)

 Styleguide 0 Std. (10)

## Realisierung der PoCs 0 Std. (85)

 PoC 1 - User Authentication 0 Std. (10)

 PoC 2 - DB testing 0 Std. (15)

 PoC 3 - API und DB testing 0 Std. (20)

 PoC 4 - Anw.logik u. Systemarchitektur 0 Std. (35)

 PoC 5 - Push Notifications 0 Std. (5)

# Projektplan – Work planned

*you are* ♪  
*What you listen to*