

Virtuelle Jukebox

Projekthandbuch

Projektnummer: PIE.XX.XXX

Version: 1.0

Datum: 21.11 2019

David Böhm-Vrana

Mathias Dittrich

Tobias Egger

Paul Götzinger

Sophia Nunner

Änderungsverzeichnis

|  |  |  |  |
| --- | --- | --- | --- |
| Versions | Datum | Änderung | Ersteller |
| 1.0 | 21.11.2019 | Erste Version | D. Böhm-Vrana, M. Dittrich, T. Egger, P. Götzinger, S. Nunner |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Inhalt

[1 Anforderungsdokument 4](#_Toc25263128)

[2 Use Cases 5](#_Toc25263129)

[3 Projektpläne 6](#_Toc25263130)

[3.1 Projektorganisation 6](#_Toc25263131)

[3.2 Projektstrukturplan 7](#_Toc25263132)

[3.3 Spezifikation der Arbeitspakete 8](#_Toc25263133)

[3.4 Meilensteinplan 11](#_Toc25263134)

[3.5 Personaleinsatzplan 12](#_Toc25263135)

[3.6 Personalkosten 13](#_Toc25263136)

[3.7 Projektkosten 14](#_Toc25263137)

[3.8 Risikoanalyse 15](#_Toc25263138)

# Anforderungsdokument

Das Anforderungsdokument beschreibt ausführlich die Anforderungen und ist als externes Dokument (Anforderungsdokument.docx) angelegt. Für genauere Informationen zu den Anforderungen siehe entsprechendes Dokument.

# Use Cases

| Erstmalige Verbindung zum Server | |
| --- | --- |
| Use Case ID | UC000 |
| Precondition | Hostname / Ip sowie Port des Servers sind bekannt und Client und Server befinden sich im selben Netzwerk. Es ist noch kein Token für die REST Kommunikation bekannt. |
| Description | 1. Leere http Post Request mit den relevanten Servedaten wird abgesetzt. 2. Warten auf entsprechende Rückmeldung 3. Überprüfung des Statuscodes sowie der retournierten Daten 4. Bei Fehlerhafter Kommunikation wird entsprechendes Errorhandling ausgeführt 5. Bei erfolgreicher Kommunikation werden Daten ausgewertet und der übergebene Token gespeichert, damit dieser für zukünftige Kommunikationen verwendet werden kann. |
| Postcondition | Bei Fehlkommunikation wurde entsprechendes Errorhandling getriggert. Bei erfolgreicher Verbindung wurde Token für künftige Server Kommunikation gespeichert. |

| Anzeige vom aktuellen Track | |
| --- | --- |
| Use Case ID | UC001 |
| Precondition | Hostname / Ip sowie Port des Servers sind bekannt, Client und Server befinden sich im selben Netzwerk und Token für Serverkommunikation wurde übermittelt und gespeichert. Track wird von einem anderen User oder Admin Client gestartet. Client hat bereits eine Long Polling Get Request auf den aktuellen Track abgesetzt und wartet auf die Server Antwort. |
| Description | 1. Nach dem Start übermittelt der Server die entsprechend definierten Daten über den Long Polling Get Request. 2. Warten auf entsprechende Rückmeldung 3. Überprüfung des Statuscodes sowie der gesendeten Daten 4. Bei Fehlerhafter Kommunikation wird entsprechendes Errorhandling ausgeführt 5. Bei erfolgreicher Kommunikation werden Daten ausgewertet und der Datenspeicherungs- /Verwaltungsebene übergeben 6. Absetzen eines neuen Long Polling Get Requests um erneute Änderungen zu erfassen 7. Datenspeicherungs- /Verwaltungsebene: Entsprechende Speicherung der übergebenen Daten und weitere Übergabe zum UI Layer 8. UI Layer: Entsprechende graphische Aufbereitung der erhaltenen Daten im aktuellen Menü |
| Postcondition | Bei Fehlkommunikation wurde entsprechendes Errorhandling getriggert. Bei erfolgreicher Verbindung wurden Daten gespeichert und entsprechend im User Menü graphisch dargestellt. Weiter wurde ein neuer Long Polling Get Request abgesetzt. |

| Anzeige der aktuellen Playlist | |
| --- | --- |
| Use Case ID | UC002 |
| Precondition | Hostname / Ip sowie Port des Servers sind bekannt, Client und Server befinden sich im selben Netzwerk und Token für Serverkommunikation wurde übermittelt und gespeichert. Playlist wurde geändert und ist somit auf dem Client nicht mehr aktuell. Client hat bereits eine Long Polling Get Request auf die aktuelle Playlist abgesetzt und wartet auf die Server Antwort. |
| Description | 1. Nachdem sich die Playlist geändert hat übermittelt der Server die entsprechend definierten Daten über den Long Polling Get Request. 2. Warten auf entsprechende Rückmeldung 3. Überprüfung des Statuscodes sowie der gesendeten Daten 4. Bei Fehlerhafter Kommunikation wird entsprechendes Errorhandling ausgeführt 5. Bei erfolgreicher Kommunikation werden Daten ausgewertet und der Datenspeicherungs- /Verwaltungsebene übergeben 6. Absetzen eines neuen Long Polling Get Requests um erneute Änderungen zu erfassen 7. Datenspeicherungs- /Verwaltungsebene: Entsprechende Speicherung der übergebenen Daten und weitere Übergabe zum UI Layer 8. UI Layer: Entsprechende graphische Aufbereitung der erhaltenen Daten im aktuellen Menü |
| Postcondition | Bei Fehlkommunikation wurde entsprechendes Errorhandling getriggert. Bei erfolgreicher Verbindung sind Daten gespeichert und entsprechend im User Menü graphisch dargestellt. Weiter wurde ein neuer Long Polling Get Request abgesetzt. |
| Anzeige der aktuellen Playlist | |
| Use Case ID | UC003 |
| Precondition | Hostname / Ip sowie Port des Servers sind bekannt, Client und Server befinden sich im selben Netzwerk und Token für Serverkommunikation wurde übermittelt und gespeichert. Aktuelle Playlist ist am Client vorhanden und wird angezeigt. |
| Description | 1. User setzt in der GUI einen entsprechenden Upvote eines Tracks ab 2. Übermittlung des Upvotes von der UI Ebene zu der Datenspeicherungs- /Verwaltungsebene 3. Datenspeicherungs- /Verwaltungsebene: Entsprechende Speicherung des Upvotes und weitere Übergabe zum Network Layer 4. Network Layer: Absetzen eines entsprechenden http Put Requests entsprechendem dem definierten Interface 5. Auf Server Antwort warten 6. Überprüfung des Statuscodes und eventueller Daten (z.B.: neue Upvotes des Tracks) der Server Antwort 7. Rückmeldung der Daten an die Datenspeicherungs- /Verwaltungsebene 8. Datenspeicherungs- /Verwaltungsebene: Entsprechende Speicherung des Upvotes und weitere Übergabe zum UI Layer 9. UI Layer: Entsprechende graphische Aufbereitung der erhaltenen neuen Upvotes im aktuellen Menü und entsprechendes Feedback an den User |
| Postcondition | Bei Fehlkommunikation wurde entsprechendes Errorhandling getriggert. Bei erfolgreicher Verbindung wurde Token für künftige Server Kommunikation gespeichert. |

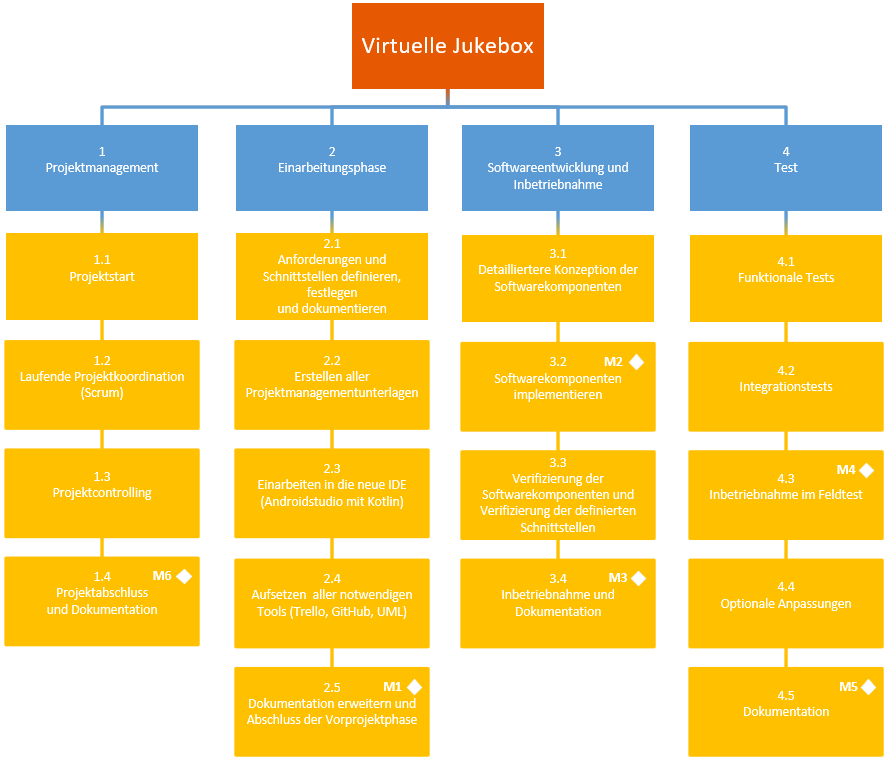
**TODO: Lied suchen Use Case fehlt noch**

# Projektpläne

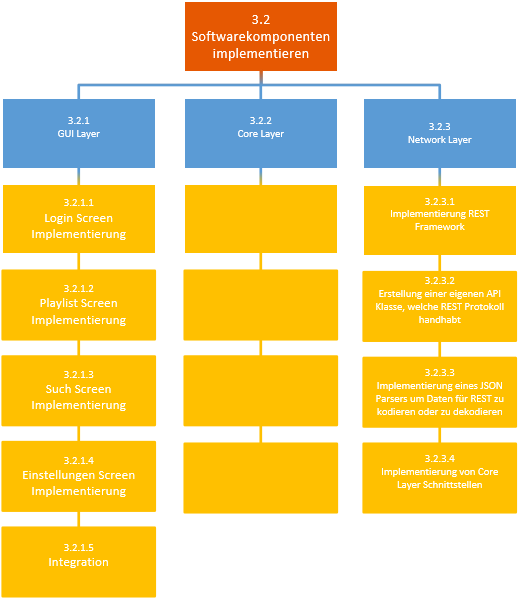
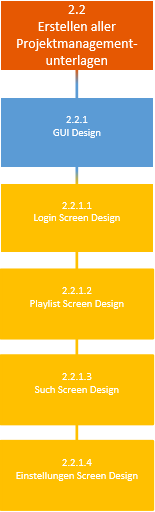
## Projektorganisation

| Projektrolle | Aufgabenbereiche | Name |
| --- | --- | --- |
| Product Owner | Produktvision und Wirtschaftlichkeit, Schnittstelle zum Kunden, Legt Produkteigenschaften und das Ziel fest (Product Backlog und Prioritäten-Festlegung), | P. Götzinger |
| Scrum Master | Stellt sicher, dass sein Team die Theorie, Praktiken und Regeln von Scrum einhält. Organisiert Meetings (Moderator), überwacht und optimiert die Zusammenarbeit des Teams. | - |
| Entwickler | Schreiben den Code und liefern am Ende eines Sprints ein fertiges Inkrement aus. Organisieren und managen Ihre Arbeit selbst. | D. Böhm-Vrana, M. Dittrich, T. Egger, S. Nunner |
|  |  |  |

## Allgemeiner Projektstrukturplan



## Detaillierter Subprojektstrukturplan





## Spezifikation der allgemeinen Arbeitspakete

|  |
| --- |
| 1 - Projektmanagement |
| 1.1 - Projektstart |
| **AP-Inhalt:** Terminlich fixierter, dokumentierter Startermin des Projektes. |
| **AP-Nicht-Inhalte:**  Arbeit am Projektinhalt selbst. |
| **AP-Ergebnisse:** Die Arbeit am Projekt wird ab jetzt entsprechend dem PSP durchgeführt. |
| **AP-Ressourcen:** Bürokratische Vorarbeiten abgeschlossen. |
| 1.2 – Projektkoordination (Scrum) |
| **AP-Inhalt:** Organisatorische Kontrolle des Projekts. Agiler Entwicklungsansatz unter Berücksichtigung von Scrum Methoden |
| **AP-Nicht-Inhalte:** Arbeit am Projektinhalt selbst. |
| **AP-Ergebnisse:** Rechtzeitige Vervollständigung des Projektes. |
| 1.3 - Projektcontrolling |
| **AP-Inhalt:** Stetige Kontrolle des Projektablaufes. |
| **AP-Nicht-Inhalte:** Arbeit am Projektinhalt selbst. |
| **AP-Ergebnisse:** Rechtzeitige Vervollständigung des Projektes. |
| 1.4 – Projektabschluss und Dokumentation |
| **AP-Inhalt:** Der Gesamtumfang des Projektes ist abgeschlossen und wird dokumentiert. |
| **AP-Nicht-Inhalte:** Abänderung und Arbeit am Projektinhalt selbst. |
| **AP-Ergebnisse:** Anforderungsgerechte Vervollständigung und Dokumentation des Projektes. |
| **AP-Ressourcen:** Dokumentation der einzelnen Projektphasen sowie das vollständige Produkt |
|  |
| 2 - Einarbeitungsphase |

|  |
| --- |
| 2.1 – Anforderungen und Schnittstellen definieren, festlegen und dokumentieren |
| **AP-Inhalt:** Vor- und Nachteile Client-Server Kommunikationstechniken, Information und Evaluierung externer Tools, Ausarbeitung einer internen Klassenstruktur (Design der Software). |
| **AP-Nicht-Inhalte:** Aufsetzen der externen Tools. Start von Software-Implementierungen. |
| **AP-Ergebnisse:** Auskunft über die zu verwendende Client-server Kommunikations-technologie, Erstellen einer internen Klassenstruktur nach der Entwicklung erfolgen kann, Information über externe Toolchains. |
| **AP-Ressourcen:** Client-Server Kommunikation (REST vs. MQTT), Interne Klassenstruktur, Externe Toolchains. |

|  |
| --- |
| 2.2 – Erstellen aller Projektmanagementunterlagen |
| **AP-Inhalt:** Analyse der mathematischen Modelle zur Beschreibung der Nachführungsrouten. |
| **AP-Nicht-Inhalte:** Entwicklung eigener Algorithmen und Software. |
| **AP-Ergebnisse:** Projektmanagementunterlagen sind erstellt und liefern Rahmenbedingungen für die Implementierung |
| **AP-Ressourcen:** Projektstrukturplan, Arbeitspakete definieren, Zeit- und Kostenschätzung, Risikoanalyse, Anforderungsdokument |
|  |
| 2.3 – Einarbeiten in die neue DIE (Androidstudio mit Kotlin) |
| **AP-Inhalt:** Erstes kennenlernen der IDE und Kotlin. Erstellen des Hello World Projekts welches als Basis für alle weiteren Entwicklungen gilt. |
| **AP-Nicht-Inhalte:** Entwicklung von projektrelevanter Software |
| **AP-Ergebnisse:** Hello World Projekt liegt vor und Basis Funktionen der IDE wurden verstanden. |
| **AP-Ressourcen:** Hello World Androidstudio Projekt liegt vor |
| 2.4 – Aufsetzen der notwendigen Tools (Trello, GitHub, UML) |
| **AP-Inhalt:** GitHub Repo wird angelegt und Hello World Projekt ist mit .gitignore hinzugefügt. Trello mit Scrum Plugin wurde aufgesetzt und entsprechend mit Tasks befüllt. |
| **AP-Nicht-Inhalte:** Entwicklung von projektrelevanter Software |
| **AP-Ergebnisse:** Teammitglieder sind mit Tools vertraut und haben Zugriff auf diese. |
| 2.5 – Dokumentation erweitern und Abschluss der Vorprojektphase |
| **AP-Inhalt:** Erweitern bestehender Dokumentationen, Gegencheck der Schnittstellen und Anforderungsdokument mit anderen Projektteams. |
| **AP-Nicht-Inhalte:** Entwicklung von projektrelevanter Software |
| **AP-Ergebnisse:** Anforderungen, Schnittstellen und Softwaredesign stehen fest und Vorprojektphase ist abgeschlossen |
|  |
| **3 - Softwareentwicklung und Inbetriebnahme** |
| 3.1 – Detaillierte Konzeption der Softwarekomponenten |
| **AP-Inhalt:** Softwaredesign wird verfeinert und innerhalb des Teams abgeglichen. |
| **AP-Nicht-Inhalte:** Ausgehende und Eingehende Schnittstellen ändern. |
| **AP-Ergebnisse:** Softwaredesign wird verfeinert, interne Schnittstellen genau definiert. |
| 3.2 – Softwarekomponenten implementieren |
| **AP-Inhalt:** Implementierung der Softwarekomponenten. Testen der einzelnen Softwarekomponenten. |
| **AP-Nicht-Inhalte:** Integrationstests mit anderen Softwarekomponenten. |
| **AP-Ergebnisse:** Die benötigten Softwarekomponenten sind implementiert und einzeln getestet. |
| 3.3 – Verifizierung der Softwarekomponenten und Verifizierung der definierten Schnittstellen |
| **AP-Inhalt:** Verifikation der fixierten Softwarekomponenten auf Funktion und Einhaltung der definierten Schnittstellen. |
| **AP-Nicht-Inhalte:** Ausführen von Teamübergreifenden Integrationstests |
| **AP-Ergebnisse:** Die vollständige Funktion der Softwarekomponenten ist verifiziert. |
| 3.4 - Inbetriebnahme und Dokumentation |
| **AP-Inhalt:** Inbetriebnahme der Hardware um Softwareimplementierungen zu ermöglichen. |
| **AP-Nicht-Inhalte:** Implementierung von Software. |
| **AP-Ergebnisse:** Die Software ist funktionsfähig. |
| **AP-Ressourcen:** Softwarekomponenten. |
|  |
| **4 - Test** |
| 4.1 - Funktionale Tests |
| **AP-Inhalt:** Test einzelner funktionaler Anforderungen. |
| **AP-Nicht-Inhalte:** Anpassung der Software. |
| **AP-Ergebnisse:** Die Funktion des Produktes ist getestet und bestätigt. |
| **AP-Ressourcen:** Vollständiges funktional getestetes Produkt. |
| 4.2 - Integrationstests |
| **AP-Inhalt:** Test funktionaler Anforderungen in Verbindung mit anderen Projektteams. |
| **AP-Nicht-Inhalte:** Anpassung der Software. |
| **AP-Ergebnisse:** Die Funktion des Produktes ist getestet und bestätigt. |
| **AP-Ressourcen:** Vollständiges funktionale getestetes Produkt mit erfolgreicher Serverkommunikation. |
| 4.3 - Inbetriebnahme im Feldtest |
| **AP-Inhalt:** Inbetriebnahme und umfassende Tests des Produktes. |
| **AP-Nicht-Inhalte:** Anpassung der Software. |
| **AP-Ergebnisse:** Die grundsätzliche Funktion des Produktes ist getestet und bestätigt. |
| **AP-Ressourcen:** Vollständiges getestetes Produkt. |
| 4.4 - Optionale Anpassungen |
| **AP-Inhalt:** Durchführung notwendiger Anpassungen und begleitende Tests. |
| **AP-Nicht-Inhalte:** Grundlegende Neuentwicklungen. |
| **AP-Ergebnisse:** Die fehlerfreie Funktion des Produktes ist getestet und bestätigt. |
| **AP-Ressourcen:** Funktionale- und Integrations-Testergebnisse und vollständiges Produkt. |
| 4.5 – Dokumentation (gilt ebenfalls für 2.5, 3.4) |
| **AP-Inhalt:** Vollständige Dokumentation der angeschlossenen Projektphase. |
| **AP-Nicht-Inhalte:** Arbeit am Inhalt der Projektphase. |
| **AP-Ergebnisse:** Vollständige Dokumentation der angeschlossenen Projektphase. |
| **AP-Ressourcen:** Alle Inhalte der Projekthase wurden abgeschlossen. |

## Spezifikation der detaillierten Subarbeitspakete

|  |
| --- |
| **2.2 Erstellen aller Projektmanagementunterlagen** |
| **2.2.1 – GUI Design** |
| 2.2.1.1 – Login Screen Design |
| **AP-Inhalt:** Konzeptionelles Design und Mockup des Login Screens. |
| **AP-Nicht-Inhalte:** Implementierungen, Design in Android Studio. |
| **AP-Ergebnisse:** Im entwickelten Design werden alle UI Elemente vorgesehen, die bei der Implementierung benötigt werden, um die gewünschte Funktionalität zu gewährleisten. Das erstellte Mockup kann als Design-Vorlage zur anschließenden Implementierung verwendet werden. |
| **AP-Ressourcen:** Mockup des Login Screens realisiert mittels Design-Tool „NinjaMock“. |
| 2.2.1.2 – Playlist Screen Design |
| **AP-Inhalt:** Konzeptionelles Design und Mockup des Playlist Screens. |
| **AP-Nicht-Inhalte:** Implementierungen, Design in Android Studio. |
| **AP-Ergebnisse:** Im entwickelten Design werden alle UI Elemente vorgesehen, die bei der Implementierung benötigt werden, um die gewünschte Funktionalität zu gewährleisten. Das erstellte Mockup kann als Design-Vorlage zur anschließenden Implementierung verwendet werden. |
| **AP-Ressourcen:** Mockup des Playlist Screens realisiert mittels Design-Tool „NinjaMock“. |
| 2.2.1.3 – Such Screen Design |
| **AP-Inhalt:** Konzeptionelles Design und Mockup des Such Screens. |
| **AP-Nicht-Inhalte:** Implementierungen, Design in Android Studio. |
| **AP-Ergebnisse:** Im entwickelten Design werden alle UI Elemente vorgesehen, die bei der Implementierung benötigt werden, um die gewünschte Funktionalität zu gewährleisten. Das erstellte Mockup kann als Design-Vorlage zur anschließenden Implementierung verwendet werden. |
| **AP-Ressourcen:** Mockup des Such Screens realisiert mittels Design-Tool „NinjaMock“. |
| 2.2.1.4 –Einstellungen Screen Design |
| **AP-Inhalt:** Konzeptionelles Design und Mockup des Einstellungen Screens. |
| **AP-Nicht-Inhalte:** Implementierungen, Design in Android Studio. |
| **AP-Ergebnisse:** Im entwickelten Design werden alle UI Elemente vorgesehen, die bei der Implementierung benötigt werden, um die gewünschte Funktionalität zu gewährleisten. Das erstellte Mockup kann als Design-Vorlage zur anschließenden Implementierung verwendet werden. |
| **AP-Ressourcen:** Mockup des Einstellungen Screens realisiert mittels Design-Tool „NinjaMock“. |

|  |
| --- |
| **3.2 – Softwarekomponenten implementieren** |
| **3.2.1 – GUI Layer** |
| 3.2.1.1 – Login Screen Implementierung |
| **AP-Inhalt:** Implementierung der Eingabe des Benutzernamen und Auswahl des Servers zur Verbindung. Senden des Benutzernamen an Server. Informationsabfrage vom Server. Jeweils Aufruf der vom Server zur Verfügung gestellten Funktionen. |
| **AP-Nicht-Inhalte:** Authentifizierung, Implementierung und Gewährleistung einer funktionierenden Serverkommunikation. |
| **AP-Ergebnisse:** Benutzeroberfläche ermöglicht die Eingabe des Benutzernamen und der Serverauswahl. |
| **AP-Ressourcen:** Sourcecode und Layout File für Login Screen. |
| 3.2.1.2 – Playlist Screen Implementierung |
| **AP-Inhalt:** Implementierung der Anzeige des aktuell abgespielten Titels. Anzeige des Voting-Ergebnisses. Fortschrittsanzeige. Anzeige der Lieder in der Warteschlange mit Reihung nach Votes. Möglichkeit zur Abstimmung für alle Titel in der Warteschlange. Jeweils Aufruf der vom Server zur Verfügung gestellten Funktionen. |
| **AP-Nicht-Inhalte:** Authentifizierung, Implementierung und Gewährleistung einer funktionierenden Serverkommunikation. |
| **AP-Ergebnisse:** Benutzeroberfläche ermöglicht die Anzeige des aktuellen Titels und der restlichen Titel in der Warteschlange. |
| **AP-Ressourcen:** Sourcecode und Layout File für Playlist Screen. |
| 3.2.1.3 – Such Screen Implementierung |
| **AP-Inhalt:** Implementierung der Eingabe von Suchanfragen in der Suchleiste. Anzeige der Suchergebnisse nach Kommunikation mit Server. Möglichkeit des Hinzufügens eines Suchergebnisses zur Playlist**.** |
| **AP-Nicht-Inhalte:** Implementierung der serverseitigen Suchfunktion. |
| **AP-Ergebnisse:** Benutzeroberfläche ermöglicht die Eingabe von Suchanfragen und Hinzufügen von Titeln zur Playlist. |
| **AP-Ressourcen:** Sourcecode und Layout File für Such Screen. |
| 3.2.1.4 –Einstellungen Screen Implementierung |
| **AP-Inhalt:** Implementierung der Auswahl eines verfügbaren Servers. Trennen der Verbindung zum aktuellen Server. |
| **AP-Nicht-Inhalte:** Erneute Eingabe des Benutzernamens bei Verbindung mit anderem Server. |
| **AP-Ergebnisse:** Serverwechsel möglich. Trennen vom Server möglich. |
| **AP-Ressourcen:** Sourcecode und Layout File für Einstellungen Screen. |
| 3.2.1.5 – Integration |
| **AP-Inhalt:** Zusammenfügen aller Screens. Navigation zwischen den Screens. Implementierung der Button-Leiste in der Fußzeile. |
| **AP-Nicht-Inhalte:** Implementierung der einzelnen Screens. |
| **AP-Ergebnisse:** Funktionierende Navigation zwischen den Screens und eventuelle Informationsweitergabe zwischen den Screens. |

|  |
| --- |
| **3.2.2 – Core Layer** |
| 3.2.2.1 – |
| **AP-Inhalt:** |
| **AP-Nicht-Inhalte:** |
| **AP-Ergebnisse:** |
| **AP-Ressourcen:** |
| 3.2.2.2 – |
| **AP-Inhalt:** |
| **AP-Nicht-Inhalte:** |
| **AP-Ergebnisse:** |
| **AP-Ressourcen:** |
| 3.2.2.3 – |
| **AP-Inhalt:** |
| **AP-Nicht-Inhalte:** |
| **AP-Ergebnisse:** |
| **AP-Ressourcen:** |
| 3.2.2.4 – |
| **AP-Inhalt:** |
| **AP-Nicht-Inhalte:** |
| **AP-Ergebnisse:** |
| **AP-Ressourcen:** |
| |  | | --- | | **3.2.3 – Network Layer** | | 3.2.3.1 – Implementierung REST Framework | | **AP-Inhalt:** Suche und entsprechende Implementierung eines geeigneten REST Frameworks in Android Studio mit Kotlin. | | **AP-Nicht-Inhalte:** Authentifizierung, Implementierung und Gewährleistung einer funktionierenden Serverkommunikation. | | **AP-Ergebnisse:** Framework wurde integriert und kann von einer Klasse verwendet werden. | | **AP-Ressourcen:** REST Plugin installiert und im Projekt referenziert. | | 3.2.3.2 – Erstellung einer eigenen API Klasse, welche REST Protokoll handhabt | | **AP-Inhalt:** Implementierung einer eigenen Klasse welche das zuvor hinzugefügte REST Framework verwenden kann. Die Klasse bietet alle notwendigen Methoden für die Serverkommunikation an. Daten müssen JSON formatiert übergeben werden. | | **AP-Nicht-Inhalte:** Auswertung von Daten. Kodierung / Dekodierung von Daten. | | **AP-Ergebnisse:** API Klasse ermöglicht Serverkommunikation über bereitgestellte Methoden. | | **AP-Ressourcen:** Sourcecode für eine eigene abgekapselte Klasse | | 3.2.3.3 – Implementierung eines JSON Parsers um Daten für REST zu kodieren oder zu dekodieren | | **AP-Inhalt:** Implementierung einer weiteren Klasse welche die Daten für die Serverkommunikation handhabt. Die Daten müssen entsprechend dem JSON Format kodiert oder dekodiert werden. Dient als Bindeglied zwischen Network und Core Layer. | | **AP-Nicht-Inhalte:** Serverkommunikationsbezogene Daten/Attribute. | | **AP-Ergebnisse:** Ermöglicht Kodierung oder Dekodierung durch entsprechende Methoden. | | **AP-Ressourcen:** Sourcecode | | 3.2.3.4 –Implementierung von Core Layer Schnittstellen | | **AP-Inhalt:** Implementierung der notwendigen Core Layer Schnittstellen um Core Layer über geänderte Daten zu informieren und diese bereitzustellen. Sowie um Daten vom Core Layer an den Server zu senden und eine entsprechende Rückmeldung an den Core Layer zu liefern. | | **AP-Nicht-Inhalte:** Core Layer Algorithmik. | | **AP-Ergebnisse:** Schnittstellen für Core Layer stehen bereit. | | **AP-Ressourcen:** Sourcecode | |

|  |
| --- |
| **4.1 – Funktionale Tests** |
| **4.1.1 – GUI Layer** |
| 4.1.1.1 – Login Screen Test |
| **AP-Inhalt:** Test einzelner funktionaler Anforderungen. |
| **AP-Nicht-Inhalte:** Anpassung der Software. |
| **AP-Ergebnisse:** Die Funktion des Login Screens ist getestet und bestätigt. |
| **AP-Ressourcen:** Vollständig funktional getesteter Login Screen mit erfolgreicher Serverkommunikation. |
| 4.1.1.2 – Playlist Screen Test |
| **AP-Inhalt:** Test einzelner funktionaler Anforderungen. |
| **AP-Nicht-Inhalte:** Anpassung der Software. |
| **AP-Ergebnisse:** Die Funktion des Playlist Screens ist getestet und bestätigt. |
| **AP-Ressourcen:** Vollständig funktional getesteter Playlist Screen mit erfolgreicher Serverkommunikation. |
| 4.1.1.3 – Such Screen Test |
| **AP-Inhalt:** Test einzelner funktionaler Anforderungen. |
| **AP-Nicht-Inhalte:** Anpassung der Software. |
| **AP-Ergebnisse:** Die Funktion des Produktes ist getestet und bestätigt. |
| **AP-Ressourcen:** Vollständig funktional getesteter Such Screen mit erfolgreicher Serverkommunikation. |
| 4.1.1.4 –Einstellungen Screen Test |
| **AP-Inhalt:** Test einzelner funktionaler Anforderungen. |
| **AP-Nicht-Inhalte:** Anpassung der Software. |
| **AP-Ergebnisse:** Die Funktion des Produktes ist getestet und bestätigt. |
| **AP-Ressourcen:** Vollständig funktional getesteter Einstellungen Screen mit erfolgreicher Serverkommunikation. |

|  |
| --- |
| **4.1.2 – Core Layer** |
| 4.1.2.1 – |
| **AP-Inhalt:** |
| **AP-Nicht-Inhalte:** |
| **AP-Ergebnisse:** |
| **AP-Ressourcen:** |
| 4.1.2.2 – |
| **AP-Inhalt:** |
| **AP-Nicht-Inhalte:** |
| **AP-Ergebnisse:** |
| **AP-Ressourcen:** |
| 4.1.2.3 – |
| **AP-Inhalt:** |
| **AP-Nicht-Inhalte:** |
| **AP-Ergebnisse:** |
| **AP-Ressourcen:** |
| 4.1.2.4 – |
| **AP-Inhalt:** |
| **AP-Nicht-Inhalte:** |
| **AP-Ergebnisse:** |
| **AP-Ressourcen:** |
| |  | | --- | | **4.1.3 – Network Layer** | | 4.1.3.1 – REST Verbindungs- und Authentifizierungstests | | **AP-Inhalt:** Test des REST Frameworks auf grundsätzliche Funktion und Verwendbarkeit. | | **AP-Nicht-Inhalte:** Anpassung der Software. | | **AP-Ergebnisse:** Die Grundsätzliche Funktion des REST Frameworks ist getestet und es konnte eine Verbindung mit dem Server hergestellt werden. | | **AP-Ressourcen:** Funktional getestetes REST Framework. | | 4.1.3.2 – API Klassen Tests | | **AP-Inhalt:** Alle Methoden der API Klasse werden auf ihre Funktion überprüft. Hierzu soll auch das Error-Handling überprüft werden. | | **AP-Nicht-Inhalte:** Anpassung der Software. | | **AP-Ergebnisse:** Die Funktion der API Klasse ist getestet und bestätigt. Serverkommunikation und Authentifizierung war erfolgreich. | | **AP-Ressourcen:** Vollständig funktional getestete API Klasse. | | 4.1.3.3 – Daten Integritätstests | | **AP-Inhalt:** Überprüfung des JSON Parsers. Daten werden über bereitgestellte Methoden kodiert und dekodiert. Daten müssen nach durchlauf der Tests unverändert vorliegen und dürfen in keiner Weise verändert worden sein. Server versteht die kodierten Daten. | | **AP-Nicht-Inhalte:** Anpassung der Software. | | **AP-Ergebnisse:** Die Funktion der Klasse ist getestet und bestätigt. | | **AP-Ressourcen:** Vollständig funktional getestete Parser Klasse. | | 4.1.3.4 –Interface Tests | | **AP-Inhalt:** Test aller Methoden die dem Core Layer zur Verfügung gestellt werden. | | **AP-Nicht-Inhalte:** Anpassung der Software. | | **AP-Ergebnisse:** Die Interfaces wurden getestet und ihre Funktion bestätigt. | |

## Meilensteinplan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PSP-Code | Meilenstein | Basistermin | Aktueller Termin | Ist-Termin |
| 2.5 | M1 – Dokumentation erweitern und Abschluss der Vorprojektphase | 27.11.2019 | - | - |
| 3.2 | M2 – Grundlegende Implementierung der Softwarekomponenten | 09.12.2019 | - | - |
| 3.4 | M3 – Inbetriebnahme und Abschluss des ersten vollen Testdurchlaufs. | 11.12.2019 | - | - |
| 4.3 | M4 – Abschluss des Feldtests | 12.12.2019 | - | - |
| 4.5 | M5 – Anpassungen sind abgeschlossen und finaler Testdurchlauf wurde abgeschlossen und dokumentiert | 04.12.2019 | - | - |
| 1.4 | M6 – Projektabschluss | 19.12.2019 | - | - |

## Personaleinsatzplan

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PSP-Code | AP-Bezeichnung | Dittrich | Egger | Götzinger | Nunner | Böhm-Vrana | **Summe Personenstunden** |
| **1** | **Projektmanagement** |  |  |  |  |  | **43,5** |
| 1.1 | Projektstart | 0,5 | 0,5 | 0,5 | 0,5 | 0,5 | **2,5** |
| 1.2 | Projektkoordination (Scrum) | 4,0 | 4,0 | 4,0 | 4,0 | 4,0 | **20,0** |
| 1.3 | Projektcontrolling | 1,5 | 1,5 | 3,0 | 1,0 | 1,0 | **8,0** |
| 1.4 | Projektabschluss und Dokumentation | 2,0 | 3,0 | 2,0 | 3,0 | 3,0 | **13,0** |
| **2** | **Einarbeitungsphase** |  |  |  |  |  | **29,0** |
| 2.1 | Anforderungen und Schnittstellen definieren, festlegen und dokumentieren | 2,0 | 2,0 | 2,0 | 2,0 | 2,0 | **10,0** |
| 2.2 | Erstellen aller Projektmanagementunterlagen | 2,0 | 1,0 | 2,0 | 1,0 | 1,0 | **7,0** |
| 2.3 | Einarbeiten in die neue IDE (Androidstudio mit Kotlin) | 1,0 | 1,0 | 1,0 | 0,0 | 0,0 | **3,0** |
| 2.4 | Aufsetzen aller notwendigen Tools (Trello, GitHub, UML) | 0,5 | 0,5 | 1,0 | 0,5 | 0,5 | **3,0** |
| 2.5 | Dokumentation erweitern und Abschluss der Vorprojektphase | 2,0 | 1,0 | 1,0 | 1,0 | 1,0 | **6,0** |
| **3** | **Softwareentwicklung und Inbetriebnahme** |  |  |  |  |  | **73,5** |
| 3.1 | Detaillierte Konzeption und Inbetriebnahme | 3,0 | 3,0 | 3,0 | 3,0 | 3,0 | **15,0** |
| 3.2 | Softwarekomponenten implementieren | 7,0 | 6,5 | 5,5 | 8,5 | 8,5 | **36,0** |
| 3.3 | Verifizierung der Softwarekomponenten und Verifizierung der definierten Schnittstellen | 3,0 | 3,5 | 2,5 | 3,0 | 3,0 | **15,0** |
| 3.4 | Inbetriebnahme und Dokumentation | 1,5 | 1,5 | 1,5 | 1,5 | 1,5 | **7,5** |
| **4** | **Test** |  |  |  |  |  | **46,5** |
| 4.1 | Funktionale Tests | 2,5 | 1,5 | 2,5 | 3,5 | 3,5 | **13,5** |
| 4.2 | Integrationstests | 1,5 | 2,5 | 1,5 | 0,5 | 0,5 | **6,5** |
| 4.3 | Inbetriebnahme im Feldtest | 1,5 | 1,5 | 1,5 | 1,5 | 1,5 | **7,5** |
| 4.4 | Optionale Anpassungen | 2,0 | 3,0 | 3,0 | 3,0 | 3,0 | **14,0** |
| 4.5 | Dokumentation | 1,0 | 1,0 | 1,0 | 1,0 | 1,0 | **5,0** |
|  |  |  |  |  |  |  | **192,5** |

## Personalkosten

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PSP-Code | AP-Bezeichnung | Dittrich | Egger | Götzinger | Nunner | Böhm-Vrana | **Kosten** |
|  | Kosten je Personenstunde | 100 | 100 | 100 | 100 | 100 |  |
| **1** | **Projektmanagement** |  |  |  |  |  | **4.350,0** |
| 1.1 | Projektstart | 50,0 | 50,0 | 50,0 | 50,0 | 50,0 | **250,0** |
| 1.2 | Projektkoordination (Scrum) | 400,0 | 400,0 | 400,0 | 400,0 | 400,0 | **2.000,0** |
| 1.3 | Projektcontrolling | 150,0 | 150,0 | 300,0 | 100,0 | 100,0 | **800,0** |
| 1.4 | Projektabschluss und Dokumentation | 200,0 | 300,0 | 200,0 | 300,0 | 300,0 | **1.300,0** |
| **2** | **Einarbeitungsphase** |  |  |  |  |  | **2.900,0** |
| 2.1 | Anforderungen und Schnittstellen definieren, festlegen und dokumentieren | 200,0 | 200,0 | 200,0 | 200,0 | 200,0 | **1.000,0** |
| 2.2 | Erstellen aller Projektmanagementunterlagen | 200,0 | 100,0 | 200,0 | 100,0 | 100,0 | **700,0** |
| 2.3 | Einarbeiten in die neue IDE (Androidstudio mit Kotlin) | 100,0 | 100,0 | 100,0 | 0,0 | 0,0 | **300,0** |
| 2.4 | Aufsetzen aller notwendigen Tools (Trello, GitHub, UML) | 50,0 | 50,0 | 100,0 | 50,0 | 50,0 | **300,0** |
| 2.5 | Dokumentation erweitern und Abschluss der Vorprojektphase | 200,0 | 100,0 | 100,0 | 100,0 | 100,0 | **600,0** |
| **3** | **Softwareentwicklung und Inbetriebnahme** |  |  |  |  |  | **7.350,0** |
| 3.1 | Detaillierte Konzeption und Inbetriebnahme | 300,0 | 300,0 | 300,0 | 300,0 | 300,0 | **1.500,0** |
| 3.2 | Softwarekomponenten implementieren | 700,0 | 650,0 | 550,0 | 850,0 | 850,0 | **3.600,0** |
| 3.3 | Verifizierung der Softwarekomponentenund Verifizierung der definierten Schnittstellen | 300,0 | 350,0 | 250,0 | 300,0 | 300,0 | **1.500,0** |
| 3.4 | Inbetriebnahme und Dokumentation | 150,0 | 150,0 | 150,0 | 150,0 | 150,0 | **750,0** |
| **4** | **Test** |  |  |  |  |  | **4.650,0** |
| 4.1 | Funktionale Tests | 250,0 | 150,0 | 250,0 | 350,0 | 350,0 | **1.350,0** |
| 4.2 | Integrationstests | 150,0 | 250,0 | 150,0 | 50,0 | 50,0 | **650,0** |
| 4.3 | Inbetriebnahme im Feldtest | 150,0 | 150,0 | 150,0 | 150,0 | 150,0 | **750,0** |
| 4.4 | Optionale Anpassungen | 200,0 | 300,0 | 300,0 | 300,0 | 300,0 | **1.400,0** |
| 4.5 | Dokumentation | 100,0 | 100,0 | 100,0 | 100,0 | 100,0 | **500,0** |
|  |  |  |  |  |  |  | **19.250,0** |

## Projektkosten

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PSP-Code** | **AP-Bezeichnung** | **Personalkosten** | **Fremdkosten** | **Sachkosten** | **Investitionen** | **Risikominimierung** | **Summe** |
| **1** | **Projektmanagement** | **4.350,0** | **0,0** | **0,0** | **0,0** | **0,0** | **4.350,0** |
| 1.1 | Projektstart | 250,0 |  |  |  |  | **250,0** |
| 1.2 | Projektkoordination (Scrum) | 2.000,0 |  |  |  |  | **2.000,0** |
| 1.3 | Projektcontrolling | 800,0 |  |  |  |  | **800,0** |
| 1.4 | Projektabschluss und Dokumentation | 1.300,0 |  |  |  |  | **1.300,0** |
| **2** | **Einarbeitungsphase** | **2.900,0** | **0,0** | **0,0** | **0,0** | **0,0** | **2.900,0** |
| 2.1 | Anforderungen und Schnittstellen definieren, festlegen und dokumentieren | 1.000,0 |  |  |  |  | **1.000,0** |
| 2.2 | Erstellen aller Projektmanagementunterlagen | 700,0 |  |  |  |  |  |
| 2.3 | Einarbeiten in die neue IDE (Androidstudio mit Kotlin) | 300,0 |  |  |  |  |  |
| 2.4 | Aufsetzen aller notwendigen Tools (Trello, GitHub, UML) | 300,0 |  |  |  |  | **300,0** |
| 2.5 | Dokumentation erweitern und Abschluss der Vorprojektphase | 600,0 |  |  |  |  | **600,0** |
| **3** | **Softwareentwicklung und Inbetriebnahme** | **7.350,0** | **0,0** | **0,0** | **0,0** | **0,0** | **7.350,0** |
| 3.1 | Detaillierte Konzeption und Inbetriebnahme | 1.500,0 |  |  |  |  | **1.500,0** |
| 3.2 | Softwarekomponenten implementieren | 3.600,0 |  |  |  |  | **3.600,0** |
| 3.3 | Verifizierung der Softwarekomponenten und Verifizierung der definierten Schnittstellen | 1.500,0 |  |  |  |  | **1.500,0** |
| 3.4 | Inbetriebnahme und Dokumentation | 750,0 |  |  |  |  | **750,0** |
| **4** | **Test** | **4.650,0** | **0,0** | **0,0** | **0,0** | **0,0** | **4.650,0** |
| 4.1 | Funktionale Tests | 1.350,0 |  |  |  |  | **1.350,0** |
| 4.2 | Integrationstests | 650,0 |  |  |  |  | **650,0** |
| 4.3 | Inbetriebnahme im Feldtest | 750,0 |  |  |  |  | **750,0** |
| 4.4 | Optionale Anpassungen | 1.400,0 |  |  |  |  | **1.400,0** |
| 4.5 | Dokumentation | 500,0 |  |  |  |  | **500,0** |
|  |  |  |  |  |  |  | **19.250,0** |

## Risikoanalyse

| PSP-Code | Risiko-Beschreibung/Ursache | Risiko-kosten | Eintritts-wahr-schein-lichkeit | Risiko-Budget | Ver-zöge-rung | Präventive und korrektive Maßnahmen | Risiko-minimie-rungs-kosten |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | Definitionen der Schnittstelle unzureichend für die Anforderungen. Neue Definitionen lassen sich schwer in bestehendes System integrieren. | 600 | 40% | 240 | 1W | Laufende Synchronisation mit anderen Projektteams. | - |
| 3 | Erstellung und Implementierung (Zeitaufwändig, komplex, da kein Vorwissen in der Entwicklung Mobiler Applikationen besteht) | 600 | 25% | 150 | 2W | Termingerechter Beginn. Geeignete Auswahl an externen Libraries um die Implementierung zu erleichtern. | - |
| 4 | Test (Optionale Anpassungen, Fehler in der Software) | 1200 | 10% | 120 | 1W | Projektcontrolling, Tests einzelner Projektphasen. | - |

# Applikations-Mockup

Es wurde bereits ein erstes Mockup für die zu erstellenden Applikation generiert um einen ersten Eindruck zu vermittelt. Weiter lässt sich dadurch ebenfalls ein Eindruck über die Funktionalität sowie der Gestaltung gewinnen:

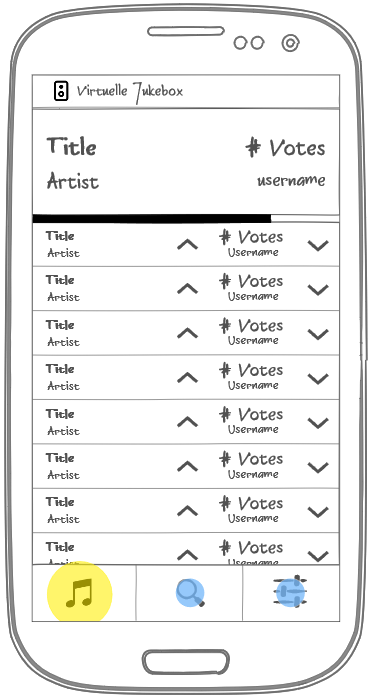


Abbildung 2 - Playlist Screen

Abbildung 1 - Login Screen

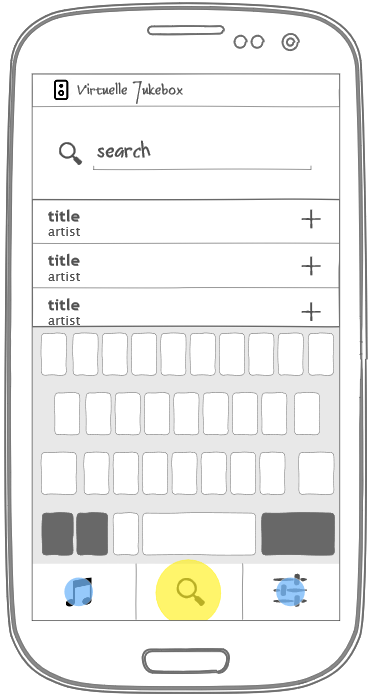
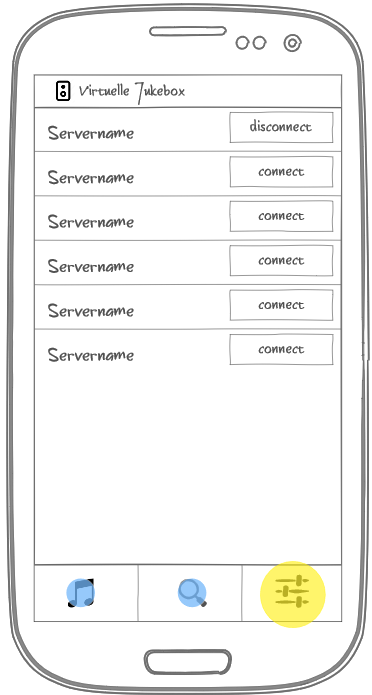


Abbildung 3 - Search Screen

Abbildung 4 - Settings Screen

Erstellt wurde dieses Mockup mittels „NinjaMock“ und kann unter folgendem Link eingesehen werden:

<https://ninjamock.com/s/LTMWMFx>