HIVE STUDIO

A CASE STUDY BY

Edita Etube-Muchová

Nadiia Goncharenko

Kornelia Kopf

Content

Tables

Figures

1. Introduction and Goals

HiveStudio is an entertainment project to add, prioritise, play, and visually represent songs in playlists. The aim of this project is also to learn how to prepare the architecture concept on the hand of arc42.

The project result should be clear, easily understandable, and implementable for future developers. The main stakeholders of the project are user, developer, software architects and project leader.

1.1. Use-case diagram

The functions of HiveStudio as described in document “CaseStudy-HiveStudio_English” are summarized in this use-case diagram:

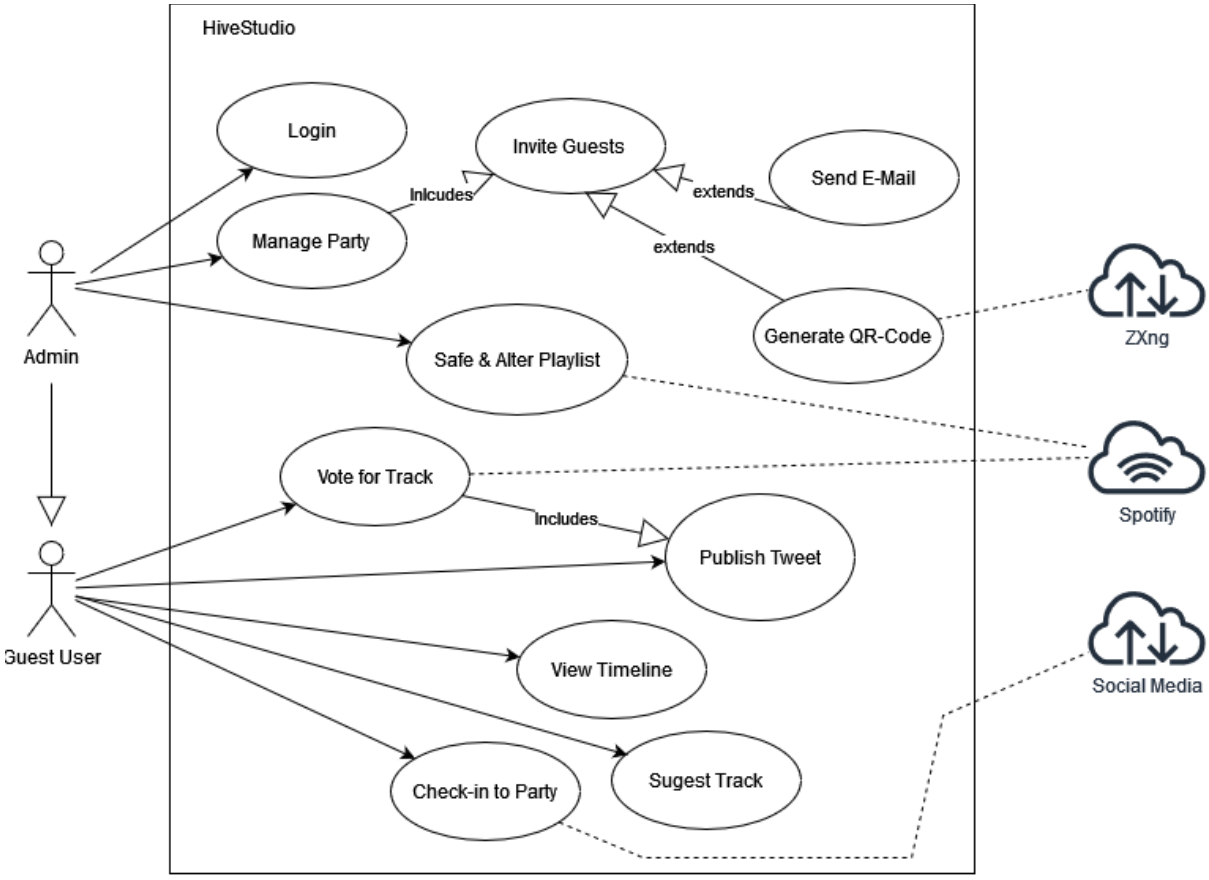


Figure 1: Use-Case Diagram

1.2. Requirements Overview

R1	Operating system-independent: The HiveStudio website should run on all operating systems Windows, Linux and Mac, as well as all types of modern browsers.
R2	Every user of HiveStudio can participate in creating a playlist, which contains music and videos.
R3	Server component of the system manages playlists and different types of users in a centralized way. Part of this HiveStudio.Server is also user authentication.
R4	There are two main types of clients, one manages playlist content, another one manages the visual representation and media management.
R5	Web service is responsible to export the functionality of Hive Studio.Server and provides the communication between the Hive.Studio.Server and its clients - HiveStudio.Playbox, HiveStudio.WebVote and HiveStudio.SimpleVote.
R6	HiveStudio.Playbox is used to manage user data, party data, invite users to parties, manage playlists and a party Twitter-like timeline, as well as play the item with the highest priority.
R7	HiveStudio.WebVote - with the help of this client, party guests can suggest new music tracks or videos and vote on the priority of the suggested media.
R8	HiveStudio.SimpleVote - the GUI client is necessary so the entire system can also be demonstrated without a web client. A subset of the functionality of the web client is to be implemented in this client.

Table 1: Requirements Overview

1.3. Quality Goals

#	Priority	Quality Goal	Important to	Scenario
QG1	1	**Interoperability**	software architects	Final system should be able to work with and communicate easily with other systems, which will be responsible for playing songs and videos like computers, TVs.
QG2	1	**Easy to use**	user	Final system should be intuitive, easy to use and administrate for users.
QG3	2	**Performance**	user	Software architecture built according to the provided specification with attention to high performance.
QG4	3	**Testable**	developer	Easy testing of all components of the software architecture.
QG5	2	**Correctness**	project leader	Software architecture built according to the provided specification.

Table 2: Quality Goals

1.4. Stakeholders

Role/Name	Expectations
User	Music fans who want to add, prioritize and play songs in playlists. The user is also able to manage the visual representation of the playlist.
Developer	developing HiveStudio website and application on the base of this arc42
Software architects (*Edita Etube Muchova, Nadiia Goncharenko, Kornelia Kopf*)	responsibly built and prepare arc42 according to the architecture concepts and technologies taught within SWA course
Project leader (Marvin Kosmider)	project overview and supervision

Table 3: Stakeholders

2.Architecture Constraints

2.1. Organizational Constraints

#	Constraint	Background and / or motivation
OC1	Configuration and version control / management	Private GoogleDoc
OC2	Architecture documentation	Structure-based on the English arc42-Template.
OC3	Language	English
OC4	Time schedule	Start in October and final results until 19.01. (presentation Date)
OC5	Published under an Open Source license	The source, including documentation, should be published as Open Source
OC6	Team	Edita Etube Muchova, Nadiia Goncharenko, Kornelia Kopf

Table 4: Organizational Constraints

2.2. Technical Constraints

#	Constraint	Background and / or motivation
TC1	Operating system-independent	The HiveStudio website should run on all operating systems Windows, Linux and Mac, as well as all types of modern browsers
TC2	Export of functionality	Entire functionality and interaction with the server is exported via HiveStudio.WebService
TC3	Authentication	Authentication of all components runs through HiveStudio.Server
TC4	Data management	Managing of all data should run via HiveStudio.PlayBox
TC5	Third party media platforms and services must be available “under an compatible open source license “	The interested developer or architect should be able to check out the sources, compile and run the application without problems compiling dependencies. All external dependencies should be available via the package manager of the App or at least through an installer.
TC6	Memory friendly	Memory can be limited due to deployment to cloud based host
TC7	Implementation in Java	Development with Java SE version 16. The engine should also run in newer Java versions, when available.

Table 5: Technical Constraints

3. System Scope and Context

This part explains the environment and context of HiveStudio.

3.1. Business Context

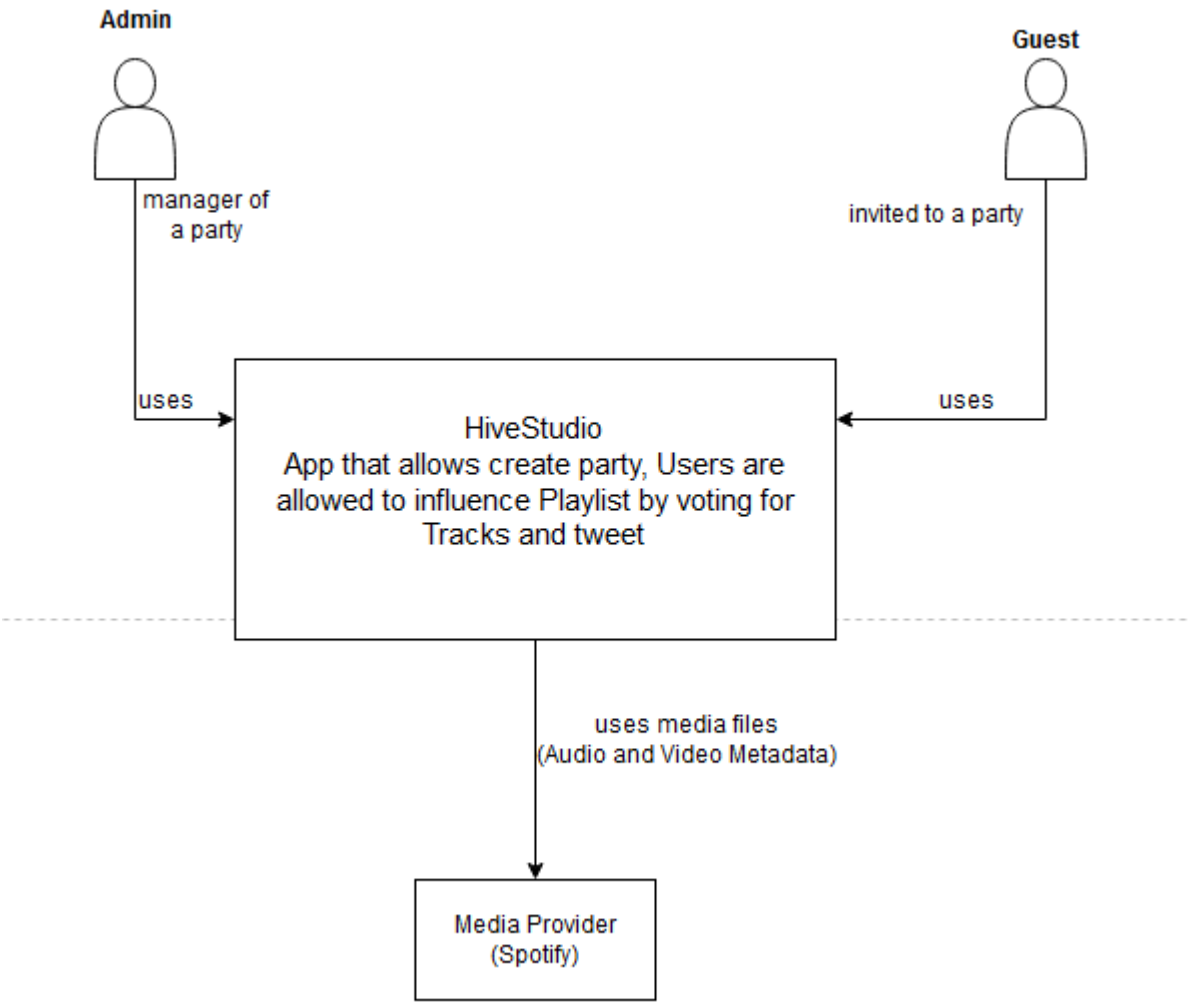


Figure 2: Business Context

User/ System	Description
Admin	Via PlayBox Admin can: - manage party/parties - manage playlist - manage timeline - invite guests to a party (via QR Code or Email)
Guest	Via WebVote Guest can: - join the party via QR Code or Email with a access code - view the playlist - suggests tracks/videos - vote for tracks/ videos - view timeline with posted messages - post short messages to the timeline
Media Provider (Spotify)	HiveStudio and its Server component are able to fetch tracks, videos and metadata from Spotify

Table 6: Business Context Supplement

3.2. Technical Context

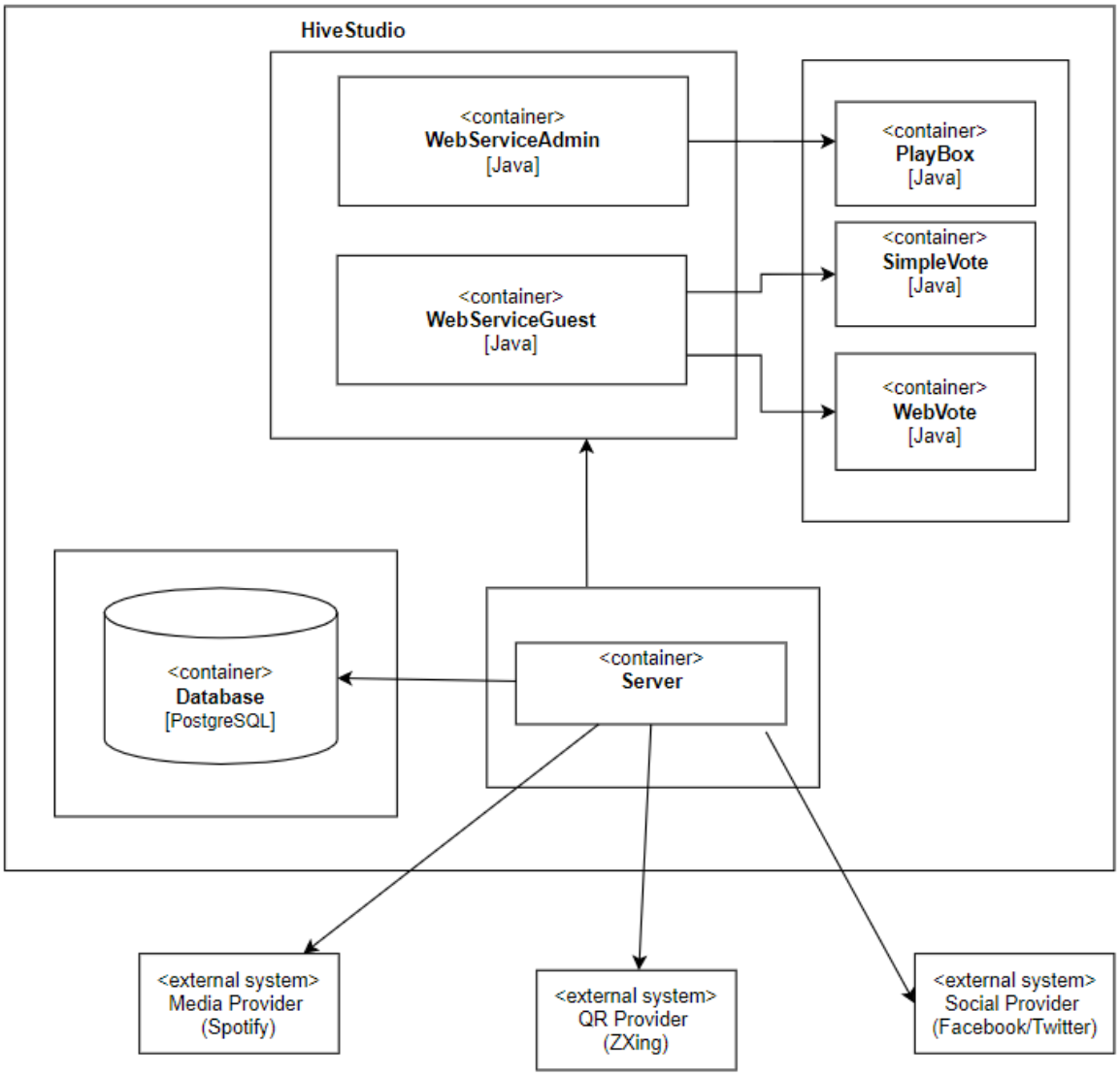


Figure 3: Technical Context

User/ System	Description
PlayBox	This client allows the user group administrator to manage the user master data, create parties and invite users to parties. PlayBox allows administrators to manage parties, tracks, timeline and interact with users.
SimpleVote	Basic client that supports only login, voting, suggesting and sending tweets for demonstration purpose without using WebClient at an early stage. SimpleVote is a simple Client that allows Login, suggesting a track, sending tweets and viewing a playlist.
WebVote	With the help of the web interface, users get an overview of all parties. You can get to the playlist of a party by checking in using a QR code or party code. In the playlist, users can vote for individual titles or invited users can suggest new titles. Of course, you can also listen to or look at the music or video titles directly. Voting processes or user comments are displayed in the party twitter timeline. With HiveStudio.WebVote, particular attention should be paid to ease of use and a visually appealing implementation. WebVote allows users viewing parties, tracks , timeline and interact with them.
WebServiceAdmin	Provides the functionality for Administrator interactions with the system via RESTful Interface. Exporting the required functionality for the SimpleVote and WebVote in the form of a web service, including all the Administrator funktionality.
WebServiceGuest	Provides the functionality for Guest interactions with the system via RESTful Interface. Exporting the required functionality for the PlayBox in the form of a web service, not including all the Administrator functionality.
Server	Main component of the system. Server handles data, user and party administration. Server also fetches data from external MediaProvider (Spotify) and SocialMedia Provider. This component is responsible for all funktionality needed by its clients.
Database	MySQL Database. Stores user information, hashed authentication credentials, access logs, parties, etc.

Table 7: Technical Context Supplement

4. Solution Strategy

Solution strategies are built on the basis of quality goals. They also reflect the experience and education level of the software architects, which led to the main goal - make it as simple as possible. Solutions strategies are explained in following table:

Quality goal	Solution approach
QG1 **Interoperability**	Implementing interface for video and music applications. The first version is planned to implement the Spotify API (application programming interface).
QG2 **Easy to use**	Start UX testing at the early stage, already at the stage of finishing SimpleVote. According to the results of the tests, the programming of WebVote will be modified.
QG3 **Performance**	Regular review of performance goals - provide voting (in 2 seconds), appear Tweets (in 3 seconds)
QG4 **Testable**	Creating test classes in Java.
QG5 **Correctness**	Programmed in Java SE, version 16.

Table 8: Solution Approaches

5. Building Block View

5.1. Level 0

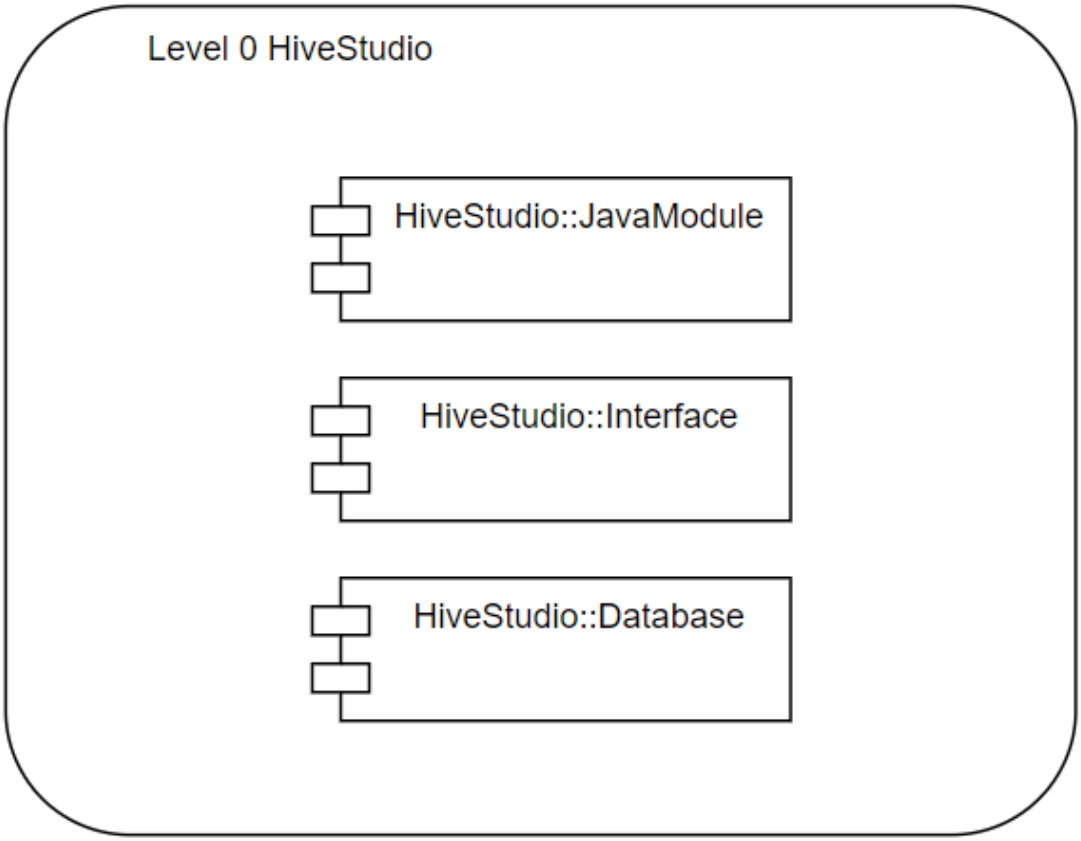


Figure 4: Level 0

Blackbox	Description
HiveStudio::Java Modules	Backend application runs on the supported server. Application provides all the required functions
HiveStudio::Interfaces	Interface for all needed APIs (Application Programming Interface) like Spotify API, social media API.
HiveStudio:Database	Stores all relevant data.

Table 9: Annex to Level 0

5.2. Level 1:

The functions of HiveStudio components as described in document “CaseStudy-HiveStudio_English” are summarized in this diagram:

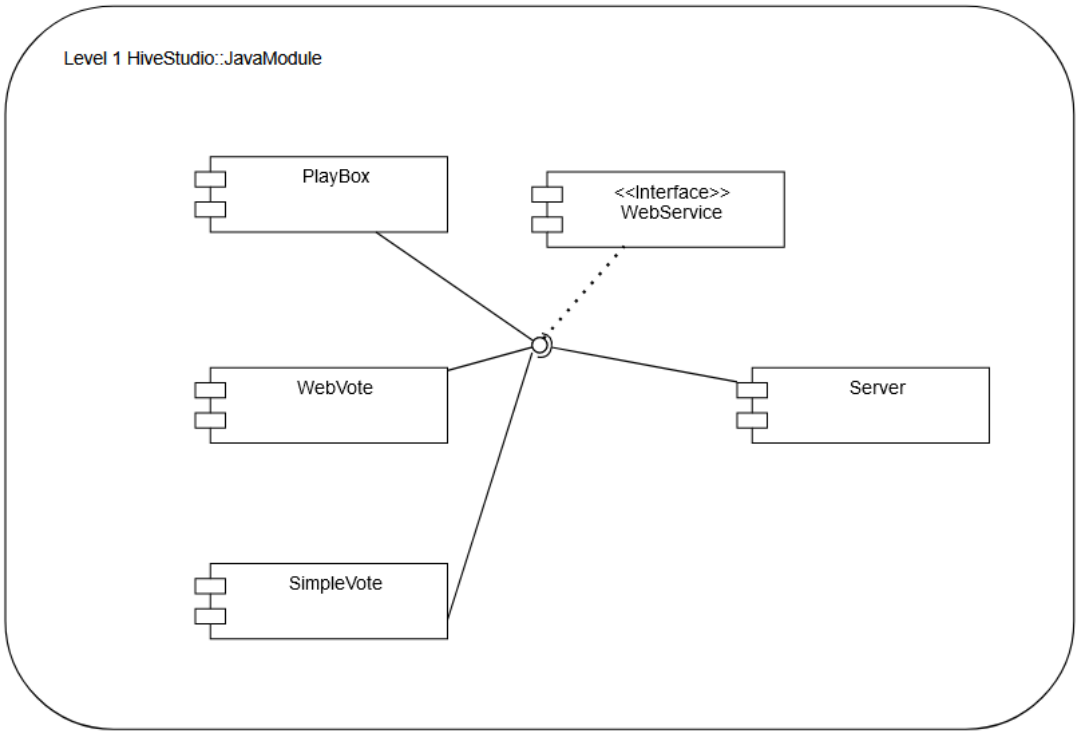


Figure 5: Level 1 - HiveStudio::JavaModule

5.3. Level 2

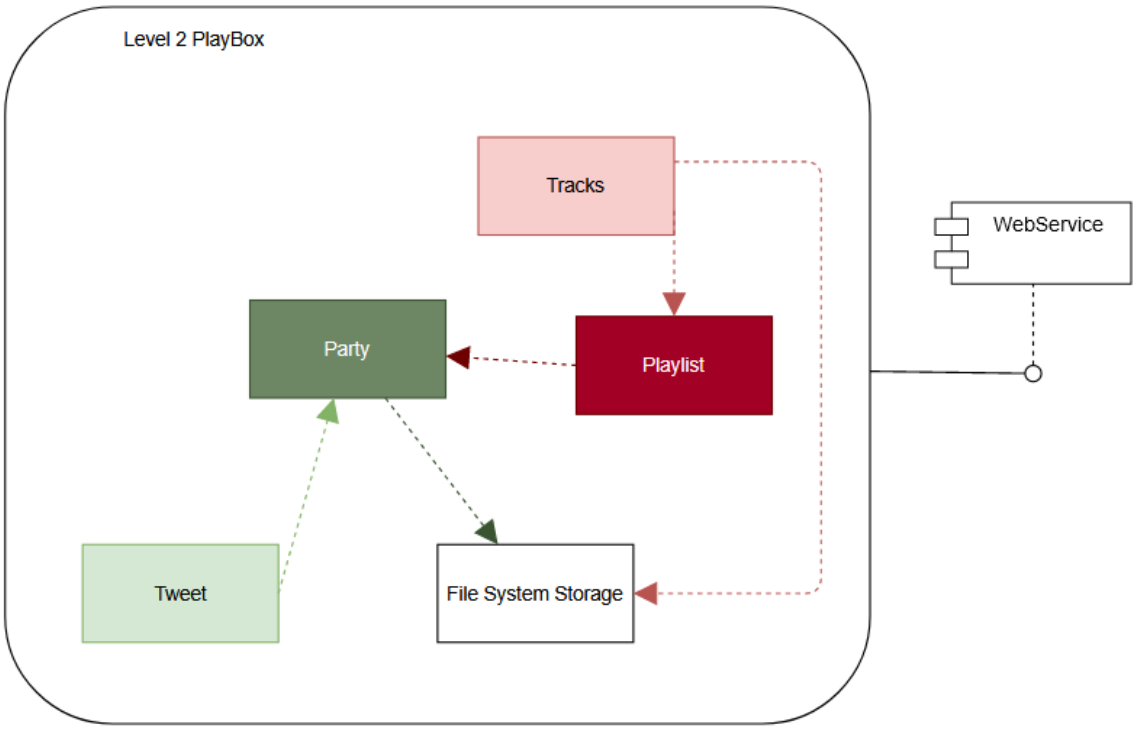


Figure 6: Level 2 - HiveStudio.PlayBox

BLACK BOX	DESCRIPTION
Tweet	get information either from vote (like) or user (comment) and sends it to the party
Tracks	get information from vote (liked video/song) and stored the information to File System Storage, sends it to the Playlist
Party	Party gets the information from Tweet about all party likes or comments and about actual playlist. PlayBox stores the current Playlist and tweets into the File System Storage. Administrator accesses the Party to manage it. PlayBox plays songs/videos from Music API.
Playlist	Playlist holds the current list of most liked tracks. As soon as the class Tracks gets a new vote, the list of tracks is rearranged.
File System Storage	In File System Storage are Tracks and Parties stored.

Table 10: Annex to Level 2

5.4. Level 3

This white box on level 3 explains the black box “Tweet”.

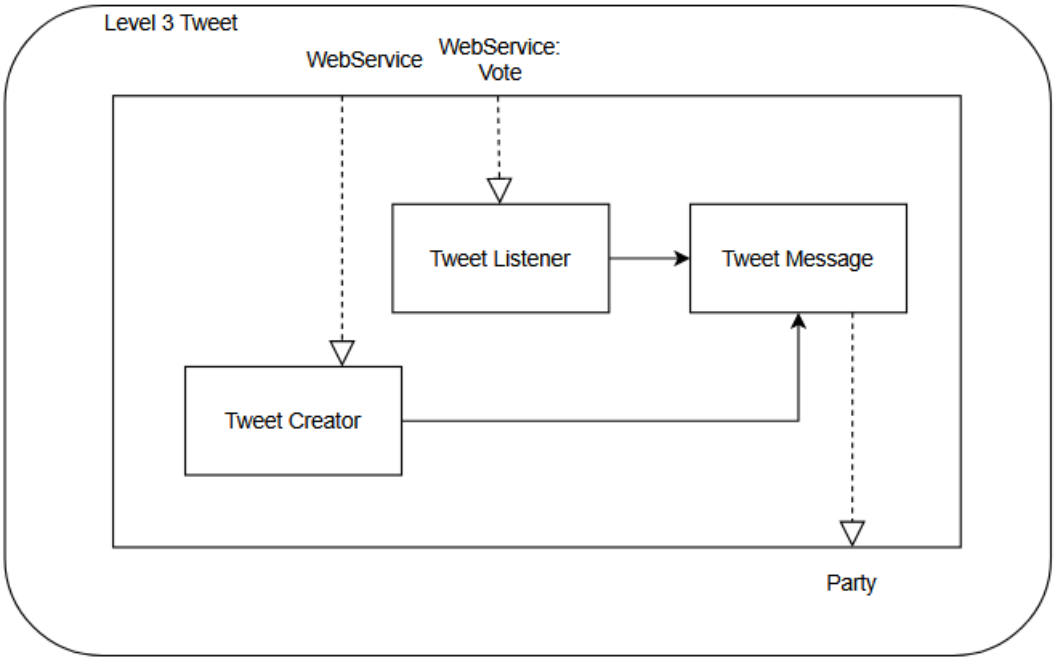


Figure 7: Level 3 - Tweet

BLACK BOX	DESCRIPTION
Tweet Creator	contains all methods for writing and publishing an individual tweet
Tweet Listener	is triggered when a new vote is cast for a title
Tweet Message	contains methods to create and publish an automatic tweet

Table 11: Annex to Level 3

6. Runtime View

We have picked two examples to demonstrate the runtime view.

6.1. <Runtime Scenario “New vote”-Tweet >

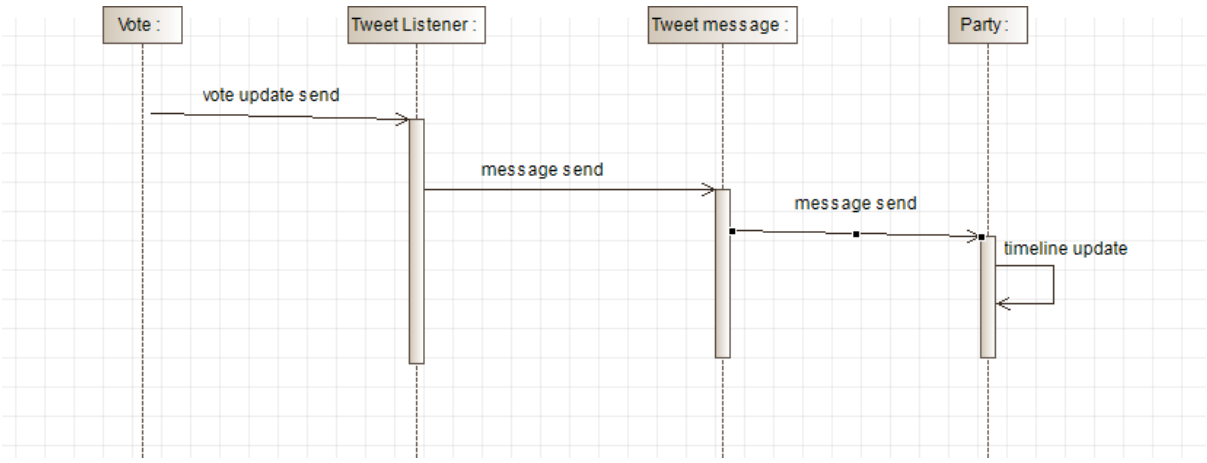


Figure 8: Runtime Scenario "New vote"-Tweet

6.2. <Runtime Scenario Created Tweet >

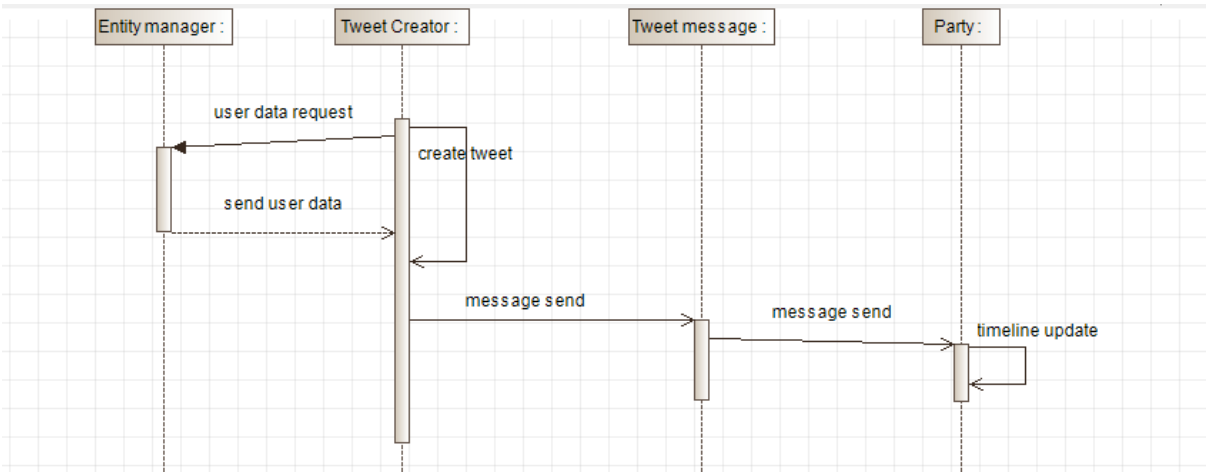


Figure 9: Runtime Scenario Created Tweet

7. Deployment View

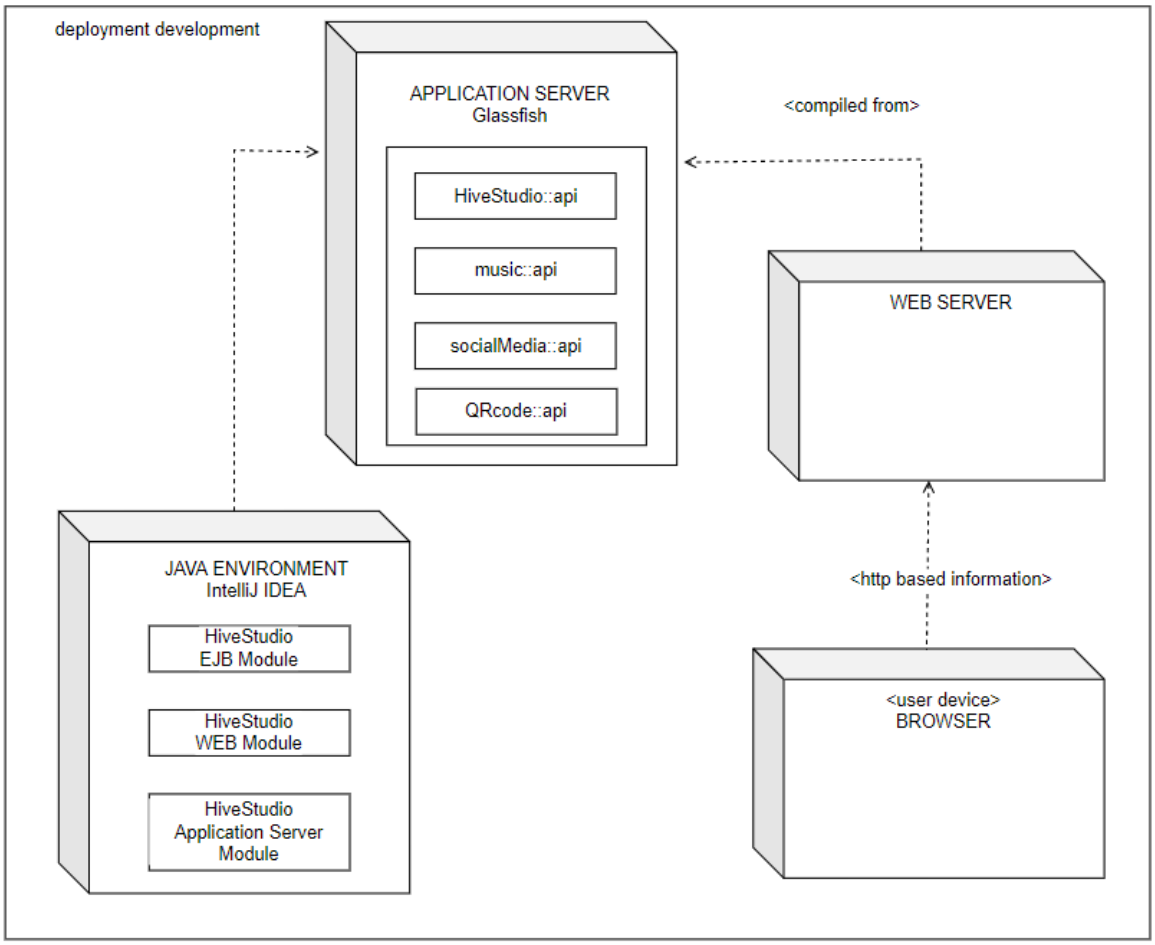


Figure 10: Deployment View

Node / Artifact	Description
Java Environment	*HiveStudio* development takes place here with all Java Modules and Components. Modules are built as .jar and .war files.
Application Server	Server that hosts the application. They communicate with Java Environment through interfaces.
Web Server	Request from client server is stored here and forwarded through Web Server
Browser	A client browser to access the application. All major browsers (Chrome, Firefox, Safari, IE / Edge) should work.

Table 12: Annex to Deployment View

8. Cross-cutting Concepts

8.1. Entity Relationship Diagram

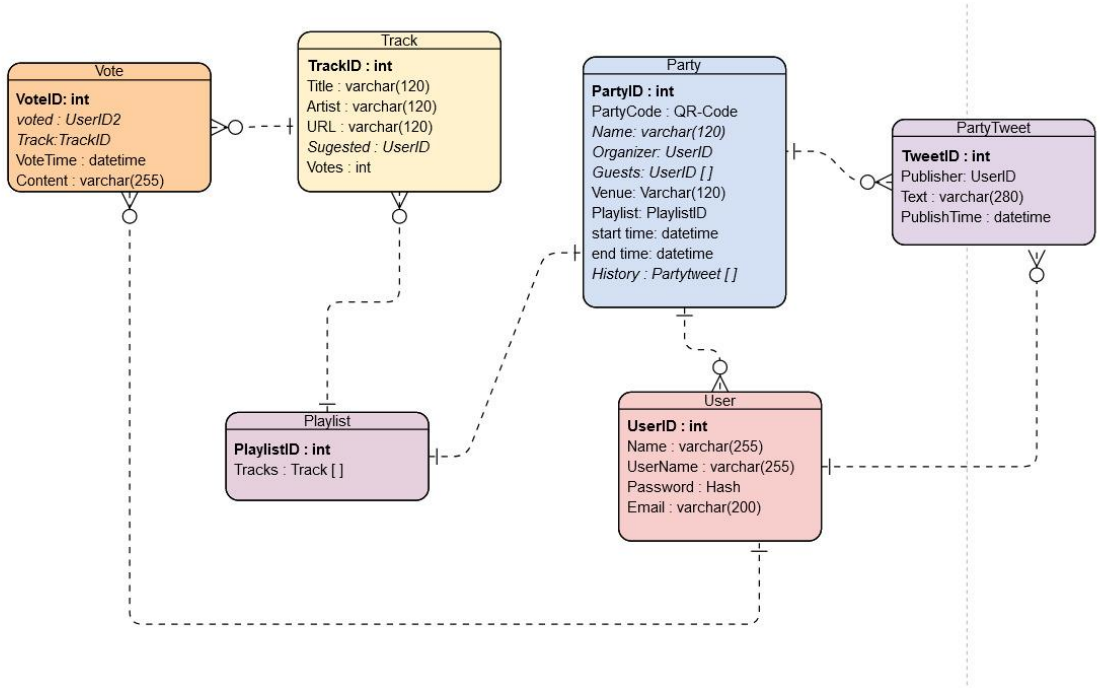


Figure 11: ER-Diagram

Name	Description
User	For each user, the access data and e-mail address are to be saved. A user can be assigned one of the following two roles: • Administrator: Administrators can use the HiveStudio.PlayBox. Administrators authenticate themselves with their username and password. • Guest: Guests are authorized to perform all functions offered by the HiveStudio clients. Guests authenticate themselves with a party code, which only allows access to a specific party.
Party	This entity is used to manage the master data of a party: name, venue, start and end time, party code, etc.

Playlist	The playlist is the collection of all tracks suggested for a party. The playlist is assigned to a party.
Track	This entity is used to manage the metadata of music titles or videos: title, author/artist, URL, etc. A track clearly belongs to a playlist. For each track it is also necessary to save who suggested it. The media file itself is not stored permanently in the database or locally. Make sure to use an online media platform like YouTube, Vimeo, etc... If runtime efficiency requires it, media files can be temporarily stored (downloaded and deleted after usage).
Vote	This entity stores which user voted for which track and when.
Party tweet	A party tweet is a short message (280 signs) that is assigned to a user and a party inside the system (no integration to the actual twitter.com platform).

Table 13: Entities to ER-Diagram

8.2. Safety

No part of the system has life endangering aspects.

8.3. JavaScript and CSS optimization

JavaScript and CSS dependencies are managed through Maven dependencies in the form of webjars wherever possible.

8.4. User Interface

The default user interface for *HiveStudio* which is packaged within the final artifact is a Single Page Application written in JavaScript using *AngularJS* together with a very default *Bootstrap* template.

8.5. Exception/Error Handling

Errors handling inconsistent data (regarding the data models constraint) as well as failures to validation are mapped to HTTP errors. Those errors are handled by the

frontend's controller code. Technical errors (hardware, database etc.) are not handled and may lead to application failure or lost data.

8.6. Internationalization

HiveStudio's supported language is English.

8.7. Usability

Ensuring the Quality Goals "Easy to use", "Readable" and "Attractiveness": To finish a milestone while developing, it is mandatory to prove that the usability was successfully tested.

8.8. Testable

The project contains Unit tests in the standard location of a Maven project. At the time of writing those tests cover > 95% of the code written. Tests must be executed during build and should not be skipped.

8.9. Build-Management

The application can be built with Maven without external dependencies outside Maven.

8.10. Usability

Ensuring the Quality Goals "Easy to use", "Readable" and "Attractiveness": To finish a milestone while developing, it is mandatory to prove that the usability was successfully tested.

9. Design Decisions

9.1. Using temporary local file storage for video and track data

Problem
HiveStudio needs optionally to store "large" objects for temporary use during the party: Mediafiles with its metadata.

Considered Alternatives:

- use of cloud storage (for example S3)
- use of local file system

Decision
Local file system was selected as it is expected to have enough storage. If HiveStudio should be runnable in cloud-based setup, one has to create an abstraction over the local file system currently used.

9.2. Using of program languages

Problem
The used programming language should be:

- easy to program
- functional, able to cover all requirements.

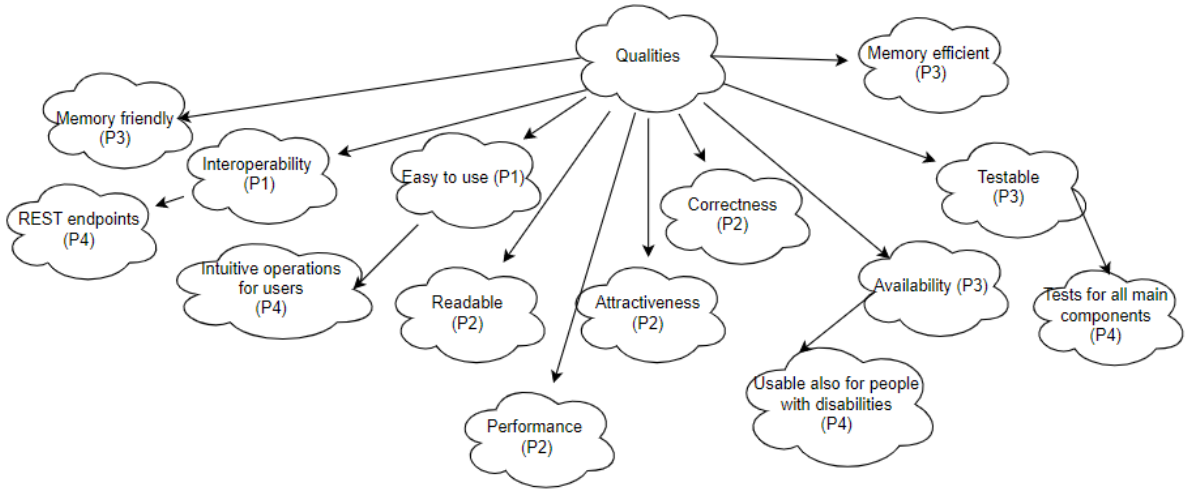
Considered Alternatives

- C++
- Python
- Java

Decision
Java was chosen because of its functionality and widespread use. A wide range of interfaces and off-the-shelf solutions is also a plus.

10. Quality Requirements

10.1. Quality Tree



10.2. Priority Table:

Priority 1 (P1)	Priority 2 (P2)	Priority 3 (P3)	Priority 4 (P4)
Interoperability			REST endpoints
Easy to use			Intuitive operations for users
	Readable		
	Performance		
	Attractiveness		
	Correctness		
	Performance		
		Availability	Usable also for people with

			disabilities
		Testable	Tests for all main components
		Memory friendly	
		Memory efficient	

Table 14: Priority Table

10.3. Quality Scenarios

Scenario	Description
Testable	Provide test files in Java.
Intuitive operations for users	Apply user friendly concept according to 10 Usability Heuristics for User Interface Design
Tests for all main components	Coverage of 90 % code with tests in the development phase.
Memory efficient	For a party of 50 guests 200 MB should be available.
Memory friendly	If Media were saved it should be deleted as soon as it was played in the playlist. Cash should be cleaned automatically every 2 hours during a party.
Performance	Voting should be provided in 2 seconds.
Performance	Tweets should appear in max. 3 seconds in a timeline after posting / voting for a track.
Readable	Defined operations should be easy to understand, follow and execute.
Interoperability	Check if different systems, devices, applications or products are able to connect and communicate in a coordinated way, without effort of the end user. Provide Interoperability Testing with at least 20 test cases to

	prove: 1. Data access 2. Data transmission
REST endpoints	Provide at 2 test cases for each REST Endpoint java class.
Easy to use	1. No broken links 2. Website search included for “lost” user 3. Use of short sentences - easier understandable for people with lower skill of language or with reading/understanding disabilities
Attractiveness	Ensure responsive design
Correctness	Coverage of at least 95 % of provided project specification.
Availability	Control if following features are part of the project, if not then adjust: 1. text alternatives for non-text content and time-based media are available 2. functionality is available from the keyboard 3. site’s appearance and behaviour is predictable 4. maximising compatibility with current and future hardware and software (including assistive technologies)
Usable also for people with disabilities	Control the website with an accessibility checker (webaccessibility.com, e.g.). When the score is less than 90%, improve accessibility.

Table 15: Quality Scenarios

11. Risks and Technical Debts

To minimise the risks and achieve the certainty, early proof of concept is planned. Though there can come to following risks:

Risk	Solution
Damage of MySQL database	Regular backups
Low software quality due to lack of testing	At least one test environment should be made available to the team. In addition, the team should have direct access to the test server.
Server crash	Prepare BackUp Servers with regular updates and tests
Insufficient memory	Non-monolite structures are used. Prepare for adding new Servers if needed.
Insufficient memory	Preparing a support plan for the use of cloud storage and preparing the legal side
Slow data processing speed	Regular tests for encrypted connections and their optimization
Slow data processing speed	Checking on enough servers in System

Table 16: Technical Risks

12. Glossary

Term	Definition
spa	Single Page Application: Application which interacts with the user by dynamically rewriting the current web page with new data from the server, instead of the default method of a web browser loading entire new pages.
API	Application Programming Interface: This is a type of software interface, connecting computers or computer programs, offering a service to other pieces of software.
REST	Representational state transfer
REST endpoint	A REST endpoint is a resource located on a server, which can be accessed with a RESTful URL.
WAR file	***W**eb **AR**chive*: the standard container file format for packaging Java EE applications as a single, deployable unit which can be deployed on Java EE-compliant application servers like Tomcat, JBoss, Glassfish.
EJB	Enterprise Java Bean: used to develop scalable, robust and secured enterprise applications in java.

Table 17: Glossary