HiveStudio - The next-gen party entertainment platform!

No more boring parties where only the host's taste in music is played: with *HiveStudio* every guest can easily and interactively influence the playlist of the evening!

Via the *HiveStudio* website, visitors can regularly cast their vote for a music track or a video. The music track or the video with the most votes will be played. If there is nothing suitable, new titles can also be proposed - and the voting starts all over! The *HiveStudio.PlayBox* is responsible for the playback, which as a highlight of every party provides additional atmosphere via projector or television.

The software system should be implemented using the architecture concepts and technologies taught within the SWA course as far as possible.

Functional Requirements

In this Case Study the architecture of a multimedia software system called *HiveStudio* should be developed. Every user of *HiveStudio* can participate in creating a playlist for a party or event, which contains music and videos. The software system consists of a server component, where playlists are managed in a centralized way and different kinds of clients. With one of these clients, users can manage the playlist content. Another client manages the visual representation and the media management (play, pause, skip, shuffle, …)

- *HiveStudio.Server* is the main component for authoring users, parties and playlists. In addition, the server component manages user authentication.
- *HiveStudio.WebService* exports the whole functionality of *HiveStudio.Server* into a webservice. Via this, webservice clients (*HiveStudio.PlayBox* & *HiveStudio.WebVote*/ *HiveStudio.SimpleVote*) can communicate with the server.
- *HiveStudio.PlayBox* is used to manage user data, party data, invite users to parties, manage playlists and a party Twitter-like timeline, as well a play the item with the highest priority.
- *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.
- *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.

a) Datamodel:

The main attributes of the most important entities in the system are defined below. The requirements specification can result in additional entities and further properties of these listed entities, which must be supplemented accordingly.

- (a.1) *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.
- (a.2) *Party*: This entity is used to manage the master data of a party: name, venue, start and end time, party code, etc.
- (a.3) *Playlist*: The playlist is the collection of all tracks suggested for a party. The playlist is clearly assigned to a party.
- (a.4) *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).
- (a.5) *Vote*: This entity stores which user voted for which track and when.
- (a.6) *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).

b) HiveStudio.Server

- (b.1) *User administration*: The server component is responsible for the administration of the user master data (add, update, delete).
- (b.2) *Administration of the parties*: It must at least be possible to create new parties, changing and deleting the party master data is optional.
- (b.3) *Management of playlists*: playlists must be expandable with new tracks and existing entries must be able to be deleted (with all associated data). For each entry it is necessary to include who proposed this track and when, or who voted for it. Users should be able to suggest songs and videos in advance of a party. The prerequisite for this is that the party organizer (an administrator) sends you an appropriate invitation via email, which contains the access data. The server must send the e-mail invitation (see also (b.6)).
- (b.4) *Management of the party twitter-like timeline*: party guests could send short messages (party tweets) during a party, which are displayed on the *HiveStudio.PlayBox*. The server component must offer functions for storing and retrieving these tweets.
- (b.5) *Queries*: It must be possible to query the user and party master data. The heart of the application are playlists, which the clients have to access according to very different criteria. Therefore, especially in this area, special attention should be paid to a well-structured, flexible access API. Which queries the server component must support in detail can be read from the requirements of the various clients.
- (b.6) *Generation of party and QR codes*: Access to the playlist of a party is only possible if a user is in possession of a corresponding party code. This code can be forwarded to party guests in two ways. Either it is sent to a guest at the initiative of an administrator via email (see (b.3)) or the party code is sent to the guest in the form of a QR code - which is displayed in the *HiveStudio.PlayBox* (see (d.4)). Both the sending and the generation of the QR code are tasks of *HiveStudio.Server*.

c) HiveStudio.WebService

The functionality required for *HiveStudio.WebVote*/*HiveStudio.SimpleVote* is to be exported in the form of a web service. The functionality required for *HiveStudio.PlayBox* can also be exported as a web service.

d) HiveStudio.PlayBox

This client allows the user group administrators to manage the user master data, create parties and invite users to parties. Specifically, the following functions are to be implemented:

- (d.1) *Login*: An administrator must log in with his username and password before he can gain access to the system.
- (d.2) *Administration of parties*: A function is to be provided with which new parties can be created and existing ones can optionally be edited and deleted.
- (d.3) *Editing the playlist*: It must be possible to add or remove tracks from the playlist at any time. This feature must also be available before the party starts.
- (d.4) *QR code*: It must be possible to generate a QR code for a party and to display it. Party guests can use this code to join a party.
- (d.5) *Administration of user data*: Input options must also be created for creating and updating users. When creating a new user, the access data must be sent to him by email.

- (d.6) *Inviting to a party*: Administrators can invite guests to parties who will be notified of this via email. It must also be possible to send the e-mail with the party code directly via the *HiveStudio.PlayBox* too.
- (d.7) *Starting a party*: After starting a party, the playlist should be processed and the party twitter timeline and the party QR code should be displayed. The change of the playlist through the vote of the guests must be considered continuously (the currently played track cannot be changed by this). It should be possible to pause or continue processing.
- (d.8) *Visualization*: The processing of the playlist should be displayed accordingly (current track). Furthermore, the video (if available) should be displayed for the current playback. For each entry in the playlist, the total of the votes cast should be visible and the playlist should be sorted accordingly. It should be possible to hide the display of the video.
- (d.9) *Party twitter timeline*: The party twitter timeline should allow the guests of a party to post short messages (party tweets). The party twitter timeline should be displayed in play mode on the *HiveStudio.PlayBox*. The party twitter timeline should also represent the casting of votes accordingly.

e) HiveStudio.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.

- (e.1) *Party overview*: All parties in the future are listed on the entry page. It should be possible to set the time range for which parties are displayed. The upcoming parties in the coming month are displayed by default. You can check-in by selecting a party (see (e.2)).
- (e.2) *Check-In*: In order to vote for an existing track in the playlist or to propose a new track, the user must select a party and check-in using a QR code or party code. Only then does the current playlist become visible.
- (e.3) *Add title*: After check-in, the user also has the option of suggesting additional titles, if they have received a corresponding invitation in advance.
- (e.4) *Voting*: After checking in, the user can vote for certain titles. Each user can only vote once for each track in the playlist.
- (e.5) *Party tweet*: In the party tweet of a party, users can write short, simple comments. An entry is also added here every time the user has voted for a title (see Facebook "Max Mustermann likes ...").
- (e.6) *Timeline*: Like on Facebook, Twitter, Instagram, etc… the individual party tweets should be displayed in a timeline.

f) HiveStudio.SimpleVote

The web service interface is required for the implementation of the web client. However, in order to be able to test the *HiveStudio.PlayBox* component at an early stage, a simple client has to be developed that enables you to log in with the party code, suggest tracks, vote for existing titles and send tweets.

Hand-In

The project should be designed using the arc42 template. There is no need to implement the project in Java. Fill out every (useful) section in the template.

Name your document Group_{GroupNumber}_Hive.pdf

Upload the file to moodle and prepare a corresponding presentation for the presentation-lecture.

Consider to use a cloud tool that can handle concurrent usage like Word in Microsoft365 (e.g.: in combination with MS-Teams) or Google-Docs.