### Service’s Architecture

Service architecture choice:

The architecture of choice in this project is multi-tier, this choice was made as we needed easily expandable and maintainable service. N-tier architecture is more suitable for above mentioned tasks, than other widely used architecture types (classic web architecture and classic client/server architecture), because they aren’t as flexible.

Project decisions:

For our project we have decided to pursue “high cohesion and low coupling”, this allowed us to maintain and expand existing program easier, thus resulting in time gained, which can later be used on other tasks. Deploying this program doesn’t take a lot of time or energy, all that’s needed for client is network connection and access to one of the two releases, web or dedicated client. Dedicated client is more advanced, and the newest features are implemented at first for this release and after time if there are no problems, web client is expanded with these features. The only drawback for dedicated client is that it is supported only for windows machines. Deployment of the service isn’t as easy as for clients, but that’s to be expected, since it will most likely require some small changes depending on host.

Pros:

* Program is easily updatable
* No duplication of business code
* Simple to maintain
* Easy to add new client
* Client doesn’t access database directly

Cons:

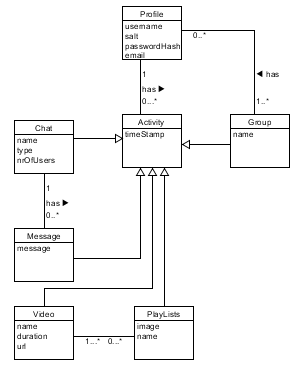
* Requires more time to create new

Domain model

Work on domain model:

First iteration of the domain model was finished by the end of sprint 0, development of it continued, but in the middle of sprint 1 it was finalized, resulting in this domain model. At first it was created overly complicated, but as the idea of the project changed in the middle of sprint 1, changes were made for domain model, both optimizing it for new tasks and simplifying it.

Domain model diagram:



Domain model explanation:

Profile- all user related information.

Activity- holds a timestamp and reference to user, to know who and when created something.

Group- collection of users, created by a specific user.

Chat- holds messages, created by a specific user;

Message- text message created by a specific user.

Video- holds the reference of video to be played, created by a specific user.

Playlist- users created playlist, is accessible by anyone.

Architecture Diagram

Work on diagram:

Work on architecture diagram as many other diagrams started early in development process at first producing simple diagram and later diagram was updated to make it more specific. As seen in image below we have separated service from both clients, thus making service self-containing, and clients can’t react with database directly, but they must use services to make any changes to it.

Design class diagram:

//update it later

