Háskólinn í Reykjavík

Tölvunarfræði deild

Hönnun og smíði hugbúnaðar

RuTube

Design document

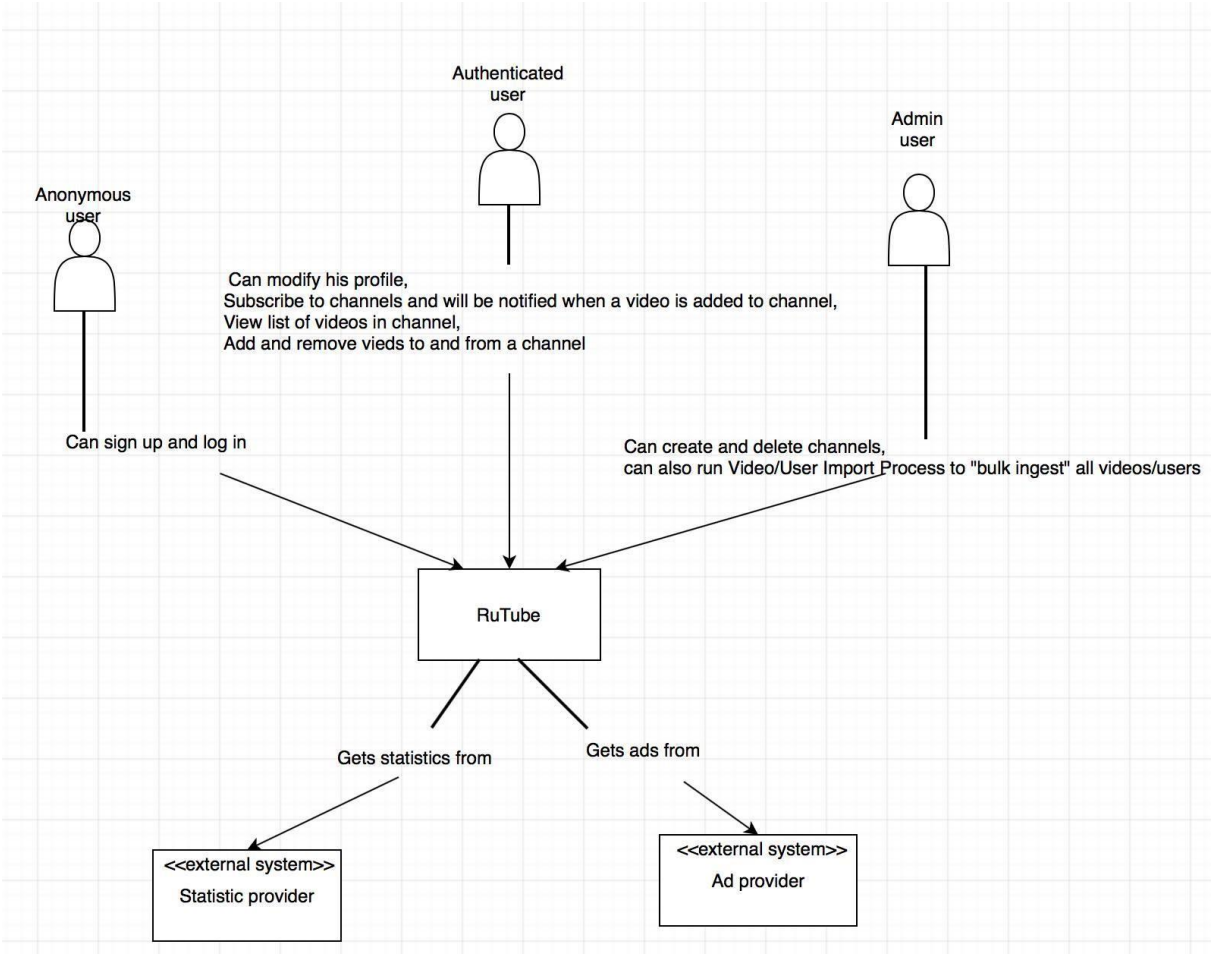
1.11.2016

Steinar Marinó Hilmarsson og Egill Gautur Steingrímsson

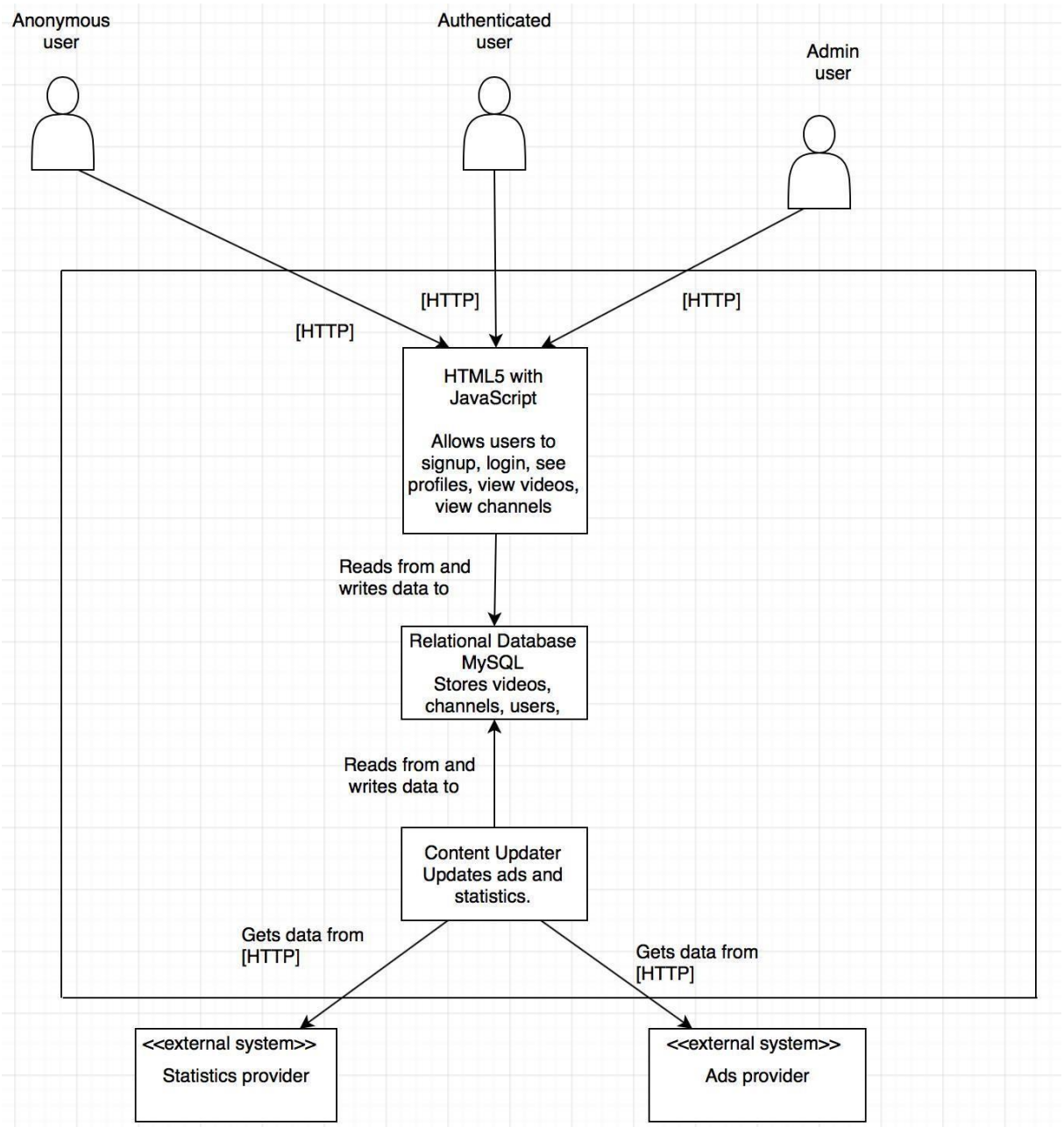
Introduction

RuTube is a video website where users can upload and view videos. The Website allows administrator to group videos into channels. RuTube is HTML-based website with a microservice architecture backend.
We are planing to use the following technologies: HTML5 with JavaScript, RuFramework2.0 and the latest Spring Framework, Import from a statistic provider(using API with JSON content), Ads from an Ad provider(using API with JSON content) and a MySQL server as the underlying database. We are going to use three services, Account service, which will be handeling signups, logins and anything to do with access. It stores usernames and passwords. The user service will handle the user profile and account and the video service will handle video registration and meta-data updates.
We will use four different diagrams of the system to show how our design will be. A context diagram, which is a high-level diagram that shows actors and system dependencies. Container diagram, which is a high-level diagram that shows technology choices and responsibilities, and a component diagram, which is showed for each container, what are the key logical components and their relationships. A database scheme will show relationship between schemes and a Rest URI design shows the routes for the HTTP requests.

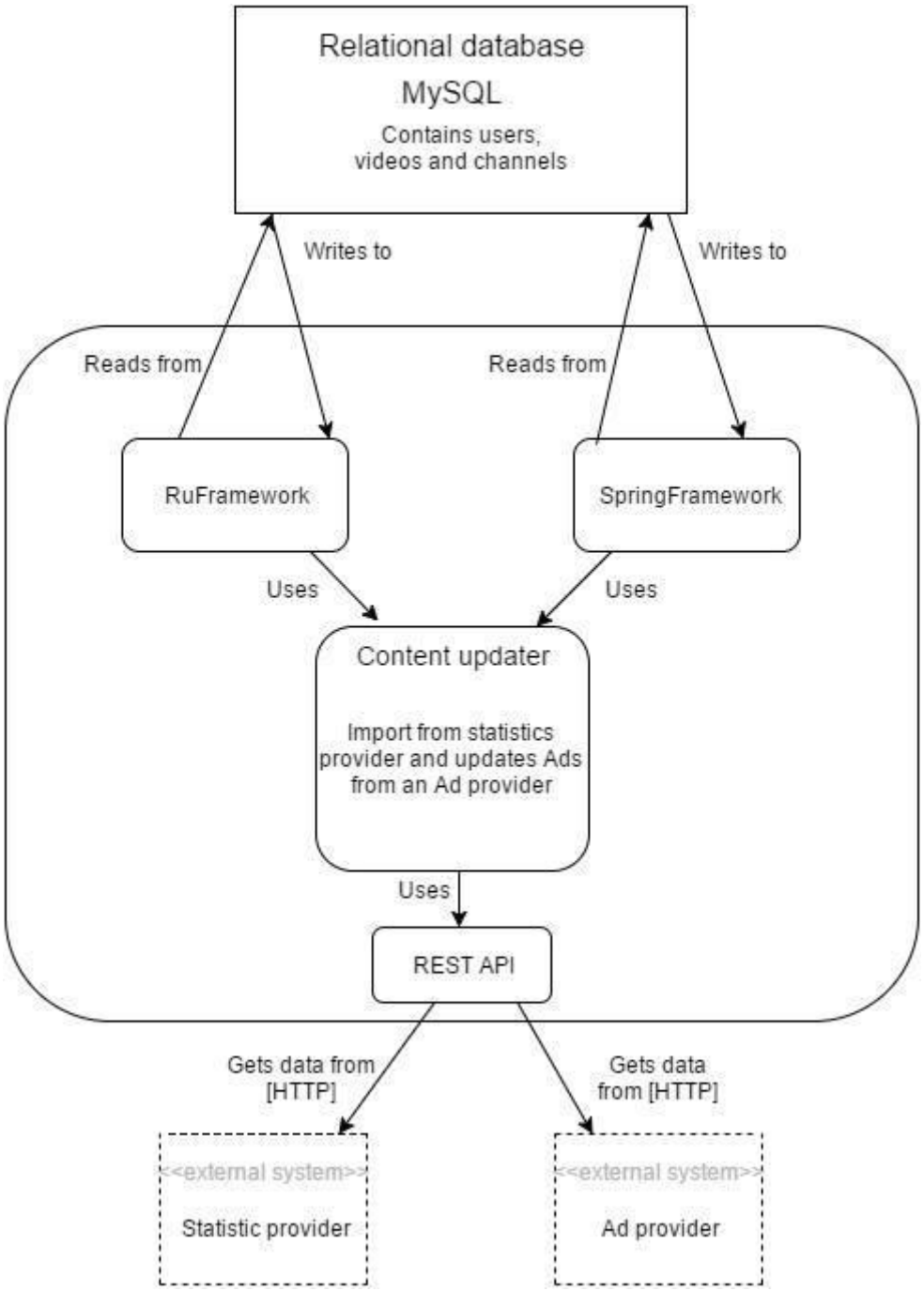
The Context diagram:



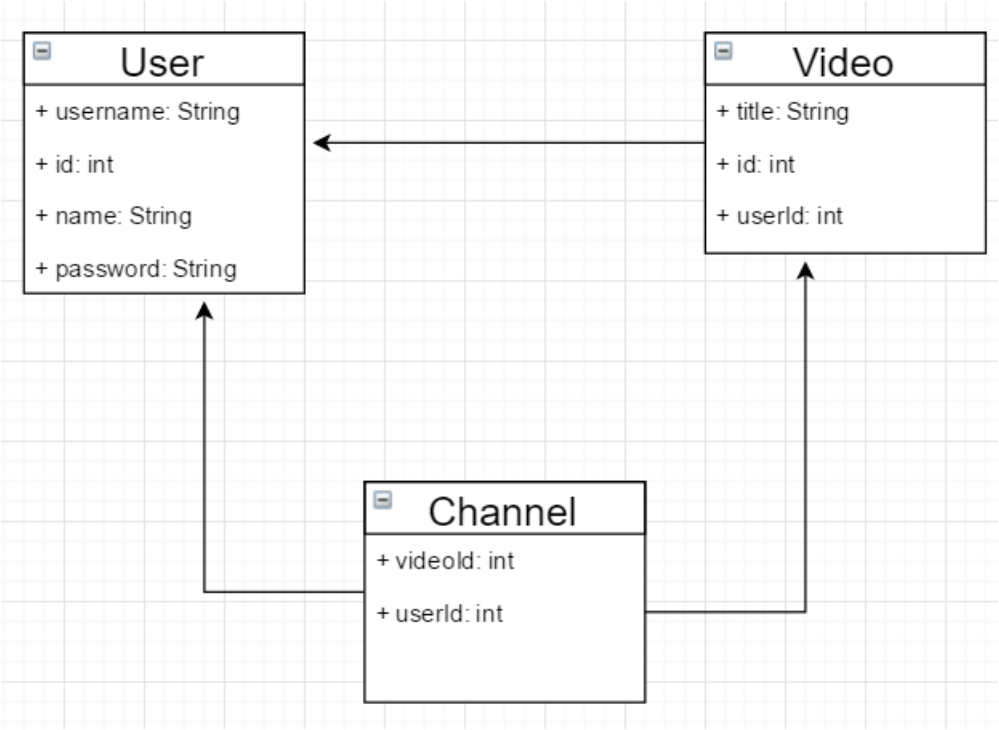
The Container diagram:



The Component diagram:


Relational database
MySQL
Contains users,
videos and channels
Writes to
Writes to
Reads from
Reads from
RuFramework
SpringFramework
Uses
Uses
Content updater
Import from statistics
provider and updates Ads
from an Ad provider
Uses
REST API
Gets data from
[HTTP]
Gets data
from [HTTP]
<<external system>>
Statistic provider
<<external system>>
Ad provider

The database schema:



The Rest URI design:

URI	HTTP method	Action
/users	GET	Gets list of all users in the database
/users	POST	Creates a new user and saves him in the database
/users/example	PATCH	Updates the password of the user example and saves the changes in the database

/users/example	DELETE	Deletes the user from the database
/users/example	GET	Gets the user from the database
/videos	GET	Gets list of all videos in the database
/videos	POST	Creates a new video and adds it to the database
/videos/example	GET	Gets the video example from the database
/videos/example	DELETE	Deletes the video example from the database
/users/example/channel	GET	Gets list of videos from the channel of the user example from the database
/users/example/channel	POST	Creates a new video and adds it to the database and to the channel of the user