Reykjavík University, Fall 2018
Course: T-302-HÖNN Hönnun og Smíði Hugbúnaðar
Teacher: Magnús Eðvald Björnsson
Assignment: Project I - Videotapes Galore, Part Three
Assignment Due: 26th of September 2018
Lab Class Group I

Videotapes Galore Prototype System

OVERALL DESIGN REPORT

26th of September, 2018

Edda Steinunn Rúnarsdóttir
kt. 241095-2909
eddasr15@ru.is

TABLE OF CONTENTS

SYSTEM OVERVIEW General Description

Videotapes Galore is a software that handles the management of serial borrows of videotapes for some videotape collection. The system stores information on borrows of videotapes and can ultimately give system managers various information and reports on the borrows registered within the system for a thorough review. In addition to this, Videotapes Galore offers videotape ratings and a recommendation service based on prior user ratings to give customers personalized recommendations of videotapes.

Although initially inspired as a software solution for a small, private business of borrowing videotapes, *Videotapes Galore* is a reliable, secure and effective system suitable for small-to-medium sized business as well as for individuals with extensive videotape collections. The system is easy to use and requires minimal manual intervention for carrying out necessary tasks related to borrowing videotapes, giving system managers greater oversight and control of their videotape collection with easy management and maintenance. The software introduces better safe-keeping of videotapes from collection by tracking borrows thus fulfilling responsibilities and reliability requirements of the borrow services that managers of videotape collections offer their customers. In addition to this, Videotapes Galore provides functionality for users to rank movies and to get recommendations based on these rankings for a personalized user experience with the potential to severely boost customer satisfaction as a result.

System users have defined roles corresponding to the nature of their access rights to information within the system. The user role with the least access rights is the *anonymous user* role and users of this role are limited only to viewing the videotape gallery in the system. An *authenticated user* role applies to users that have been authenticated into the system via Facebook authentication and users belonging to this role have more access rights than said anonymous users. If a user has been authenticated into the system they are able to add, view and modify their personal information within the system. The highest role for a user in terms of access-rights is the *admin user* role. Users of this role have full access to all resources and information stored within the system, that is they can access all information, add new information, update information and remove information from the system.

Videotapes Galore's functionality bases on managing and/or displaying the information that is registered into the system, mainly the information on all videotapes in collection, on all potential borrowers and record of borrows. The system is roughly made up of four major layers. The entry point to the system is an *Apache Tomcat web server t*hat gets requests from clients and responds to those requests. The requests are then routed to a *Java program* that processes the user requests, i.e. implements all the business logic behind processing user requests, authenticating users and serving data from the system's database to the web server to send back to user. The database for the system is a MySQL relational database that stores the system data. In addition to these three internal layers of the system it relies in addition on an external source, namely the authentication services of Facebook used to authenticate users into the system to which the said Java project communicates with for receiving authentication for users into the system.

This design report provides a visual overview on the how the system fits with said external service, roughly describes the internal structure of the system and importantly attempts to visualise the idea of design concept behind the Videotapes Galore system.

SYSTEM OVERVIEW Context Diagram

For a visual representation on the context of the Videotapes Galore system including user roles, possible user interactions with system and how system fits with existing systems refer to **Diagram 1** for a **Context Diagram** for the overall system design.

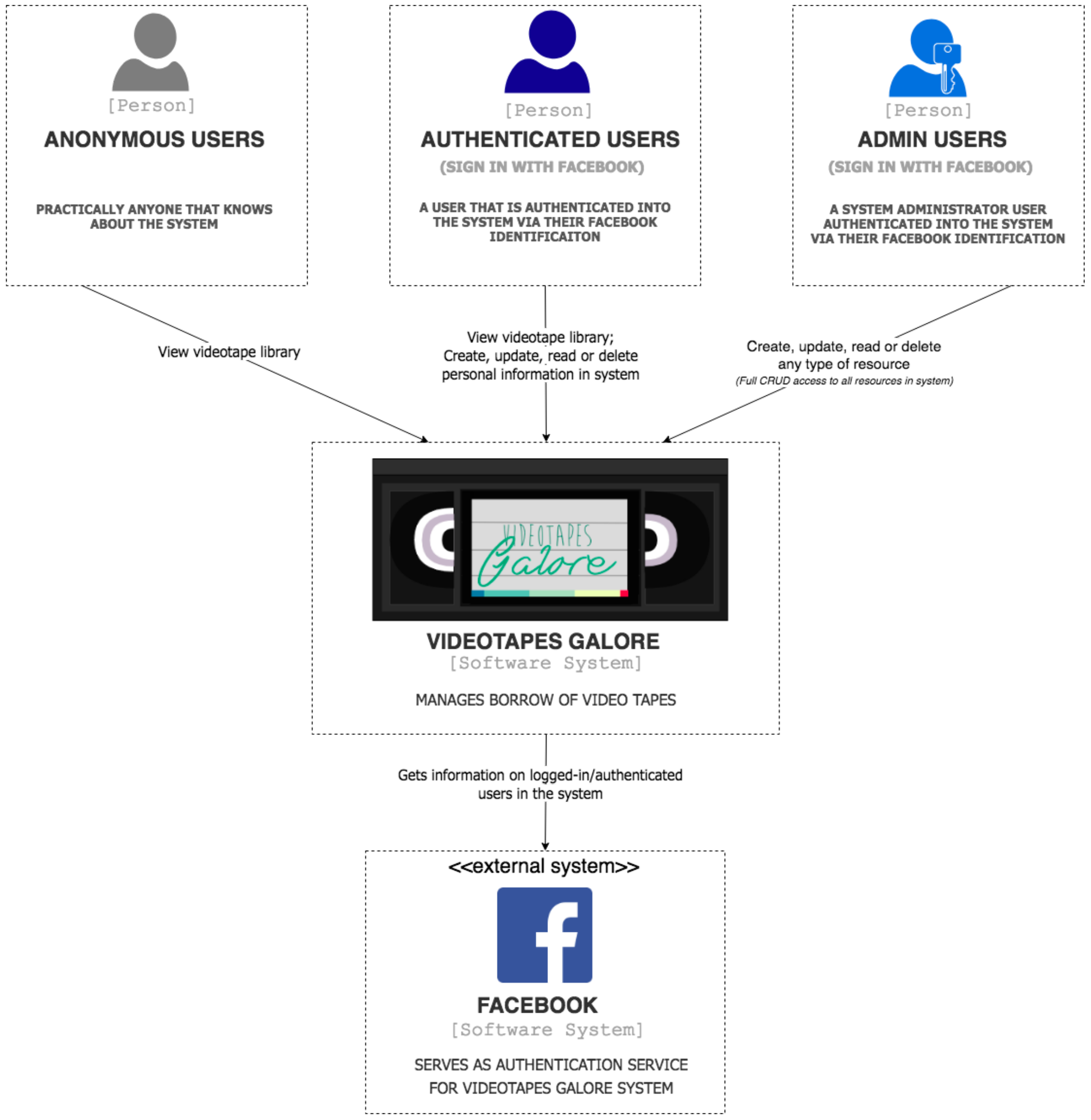

[Person]
ANONYMOUS USERS
PRACTICALLY ANYONE THAT KNOWS ABOUT THE SYSTEM
[Person]
AUTHENTICATED USERS
(SIGN IN WITH FACEBOOK)
A USER THAT IS AUTHENTICATED INTO THE SYSTEM VIA THEIR FACEBOOK IDENTIFICAITON
[Person]
ADMIN USERS
(SIGN IN WITH FACEBOOK)
A SYSTEM ADMINISTRATOR USER AUTHENTICATED INTO THE SYSTEM VIA THEIR FACEBOOK IDENTIFICATION
View videotape library
View videotape library; Create, update, read or delete personal information in system
Create, update, read or delete any type of resource
(Full CRUD access to all resources in system)
VIDEOTAPES Galore
VIDEOTAPES GALORE
[Software System]
MANAGES BORROW OF VIDEO TAPES
Gets information on logged-in/authenticated users in the system
<<external system>>
FACEBOOK
[Software System]
SERVES AS AUTHENTICATION SERVICE FOR VIDEOTAPES GALORE SYSTEM

DIAGRAM 1 Context Diagram for the Videotapes Galore system

SYSTEM OVERVIEW Container Diagram

For a better overview on the overall shape, responsibilities and key internal technology choices of the Videotapes Galore system design refer to **Diagram 2** for a **Container Diagram** for the overall system design.

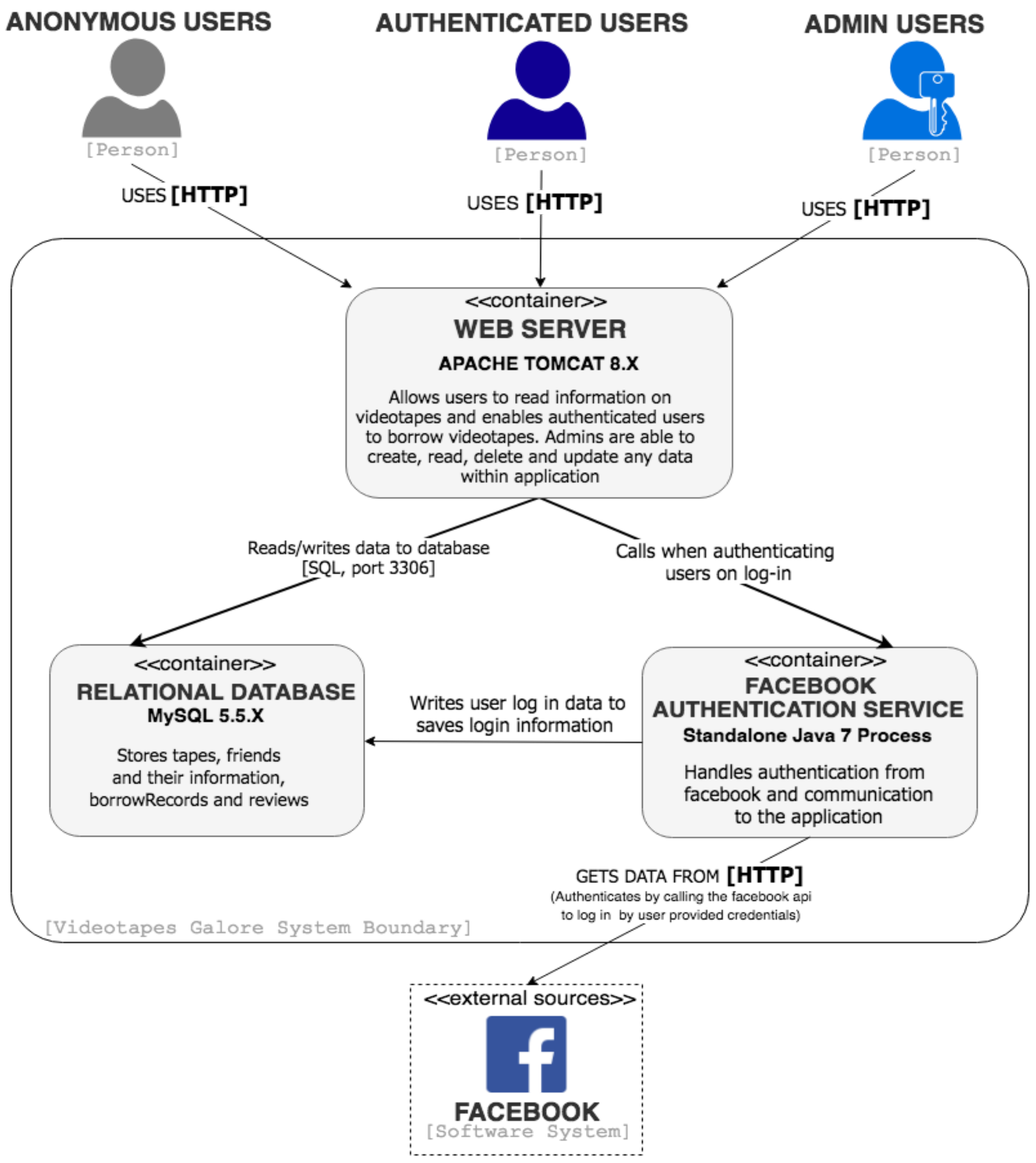

ANONYMOUS USERS
[Person]
USES [HTTP]
AUTHENTICATED USERS
[Person]
USES [HTTP]
ADMIN USERS
[Person]
USES [HTTP]
<<container>>
WEB SERVER
APACHE TOMCAT 8.X
Allows users to read information on videotapes and enables authenticated users to borrow videotapes. Admins are able to create, read, delete and update any data within application
Reads/writes data to database [SQL, port 3306]
Calls when authenticating users on log-in
<<container>>
RELATIONAL DATABASE
MySQL 5.5.X
Stores tapes, friends and their information, borrowRecords and reviews
Writes user log in data to saves login information
<<container>>
FACEBOOK AUTHENTICATION SERVICE
Standalone Java 7 Process
Handles authentication from facebook and communication to the application
GETS DATA FROM [HTTP]
(Authenticates by calling the facebook api to log in by user provided credentials)
[Videotapes Galore System Boundary]
<<external sources>>
FACEBOOK
[Software System]

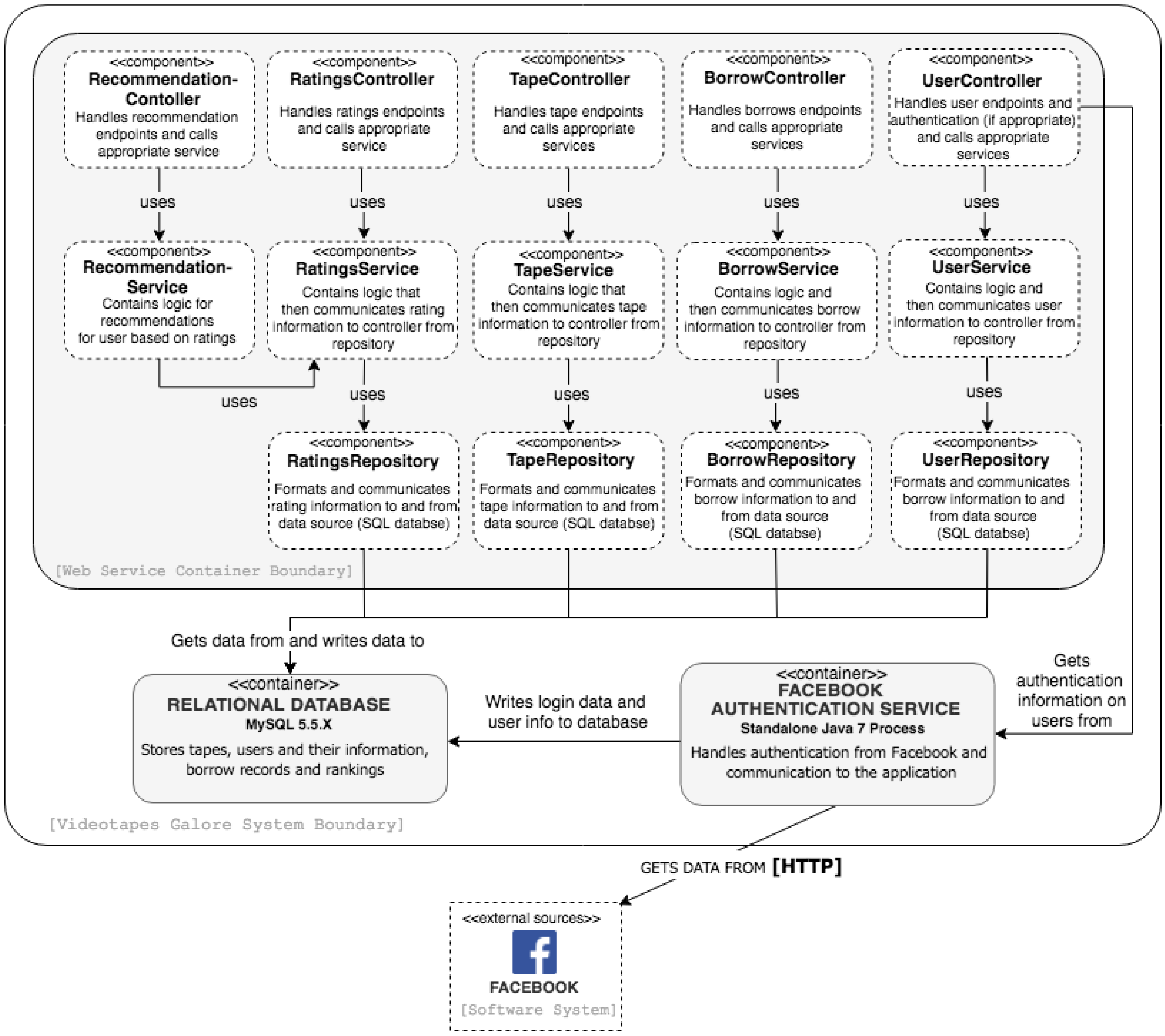
DIAGRAM 2 Container Diagram for the Videotapes Galore system

Note that as shown in Diagram 2, there are four major containers for the Videotapes Galore system all with defined roles for the system functionality, thereof three internal containers and one external source. These four major containers are as follows:

1. The **Web Server** is an Apache Tomcat web server that serves as the entry point that users send their requests to. This web server then serves data from or modifies the data of the relational database (depending on the nature of the user request) and sends back response to users.
2. The **Relational Database** is a a MySQL database storing all data in system such as all tapes, users, borrow records and ratings.
3. Then if user needs to be authenticated prior to web server processing the user request due to access restrictions, the web service must communicate with the **Facebook Authentication Service** which authenticates users into the system via external source and adds authentication information for that user to their user entity model and updates their user type/user role to redefine that user's access rights.
4. The Facebook authentication service uses **Facebook** as external source to which it communicates with to receive authentication for users.

SYSTEM OVERVIEW Component Diagram

For a better overview on major containers and their internal structure, refer to **Diagram 4** for **Component Diagram** on the **Web Service Container** of the system and to **Diagram 5** for a **Component Diagram** on the **Facebook Authentication Service Container**.



<<component>> Recommendation-Contoller Handles recommendation endpoints and calls appropriate service
<<component>> RatingsController Handles ratings endpoints and calls appropriate service
<<component>> TapeController Handles tape endpoints and calls appropriate services
<<component>> BorrowController Handles borrows endpoints and calls appropriate services
<<component>> UserController Handles user endpoints and authentication (if appropriate) and calls appropriate services
uses
<<component>> Recommendation-Service Contains logic for recommendations for user based on ratings
<<component>> RatingsService Contains logic that then communicates rating information to controller from repository
<<component>> TapeService Contains logic that then communicates tape information to controller from repository
<<component>> BorrowService Contains logic and then communicates borrow information to controller from repository
<<component>> UserService Contains logic and then communicates user information to controller from repository
<<component>> RatingsRepository Formats and communicates rating information to and from data source (SQL databse)
<<component>> TapeRepository Formats and communicates tape information to and from data source (SQL databse)
<<component>> BorrowRepository Formats and communicates borrow information to and from data source (SQL databse)
<<component>> UserRepository Formats and communicates borrow information to and from data source (SQL databse)
[Web Service Container Boundary]
Gets data from and writes data to
<<container>> RELATIONAL DATABASE MySQL 5.5.X Stores tapes, users and their information, borrow records and rankings
Writes login data and user info to database
<<container>> FACEBOOK AUTHENTICATION SERVICE Standalone Java 7 Process Handles authentication from Facebook and communication to the application
Gets authentication information on users from
[Videotapes Galore System Boundary]
GETS DATA FROM [HTTP]
<<external sources>> FACEBOOK [Software System]

DIAGRAM 4 Component Diagram for system in respect to the Web Service Container

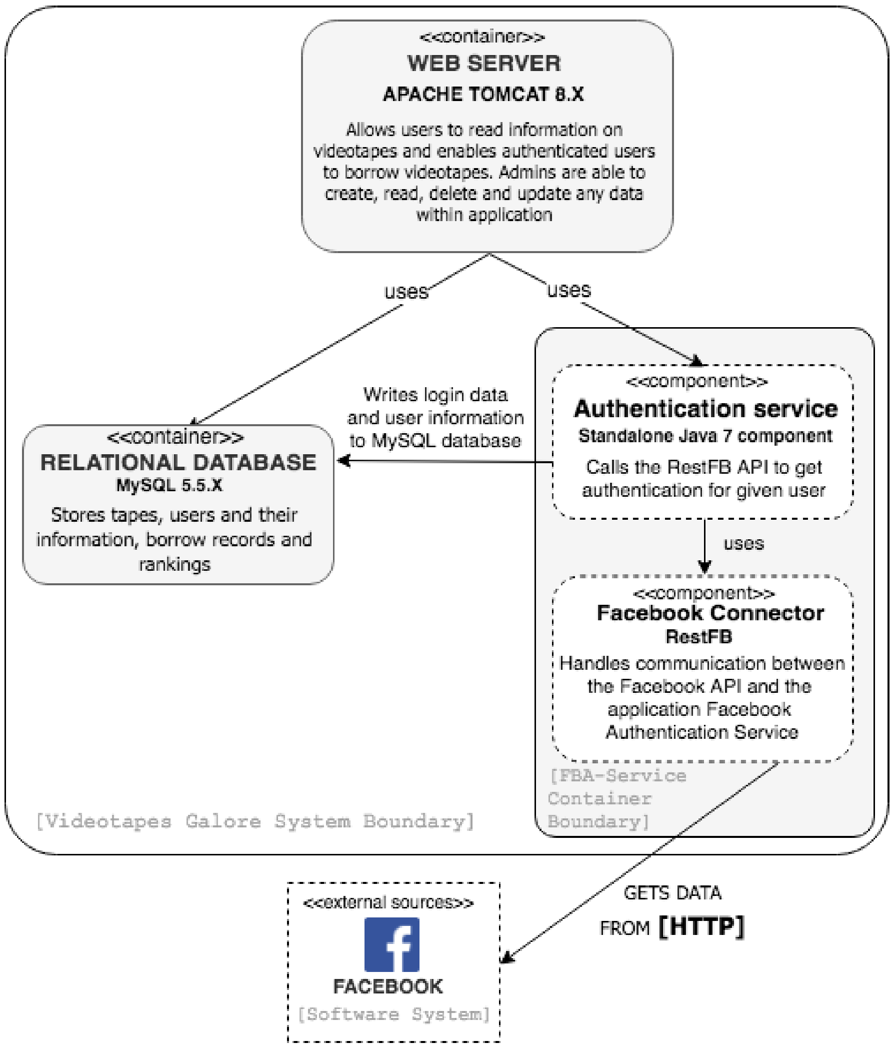

<<container>>
WEB SERVER
APACHE TOMCAT 8.X
Allows users to read information on videotapes and enables authenticated users to borrow videotapes. Admins are able to create, read, delete and update any data within application
uses
uses
Writes login data and user information to MySQL database
<<container>>
RELATIONAL DATABASE
MySQL 5.5.X
Stores tapes, users and their information, borrow records and rankings
<<component>>
Authentication service
Standalone Java 7 component
Calls the RestFB API to get authentication for given user
uses
<<component>>
Facebook Connector
RestFB
Handles communication between the Facebook API and the application Facebook Authentication Service
[FBA-Service Container Boundary]
[Videotapes Galore System Boundary]
GETS DATA FROM [HTTP]
<<external sources>>
FACEBOOK
[Software System]

DIAGRAM 5 Component Diagram for the Videotapes Galore system in respect to the Facebook Authentication Service Container

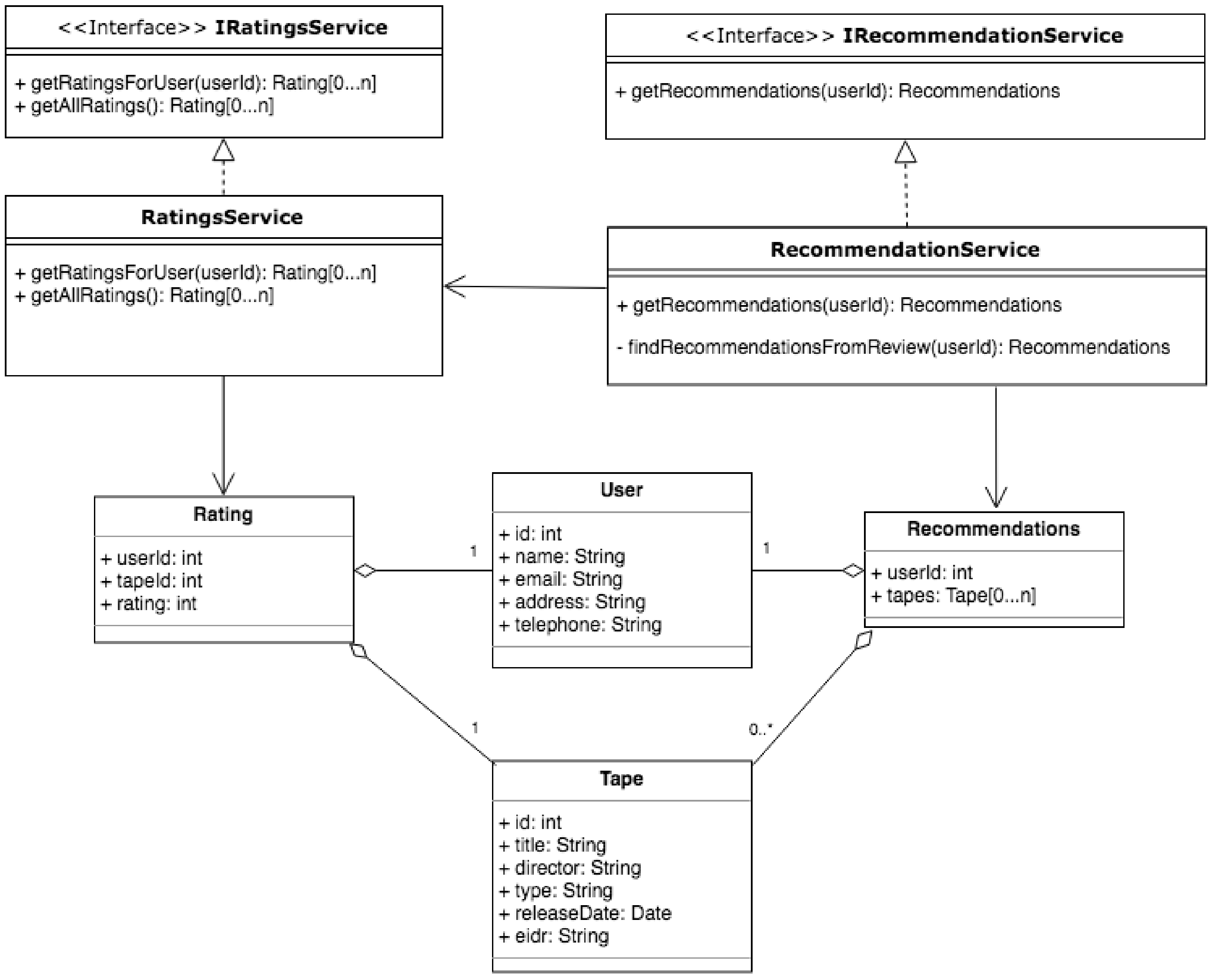
Note that the reason for presenting system's component diagram in two separate ones is that component diagrams tend to get cluttered if many containers are visualized in their componental structure, as well as by convention contents of only a single container is shown per component diagram. Furthermore, the reason for showing component diagrams for these two specific containers over the other containers in system was that their internal structure give the best overview of the system in terms of overall internal structure and functionality, whereas the internal structures of the containers that were not set up in component diagram do not add much informational context to the overall design.

To further explain the component diagram for the web service (Diagram 4), the components are separated into three layers. On the top is the controller layer. The controller layer handles API requests that come into the system and validates if they are formatted correctly and/or contain the correct data. The **UserController** also handles communication with the AuthenticationService to authenticate logged in users (admins and authenticated users) if user's request requires authentication. Each controller communicates with the appropriate service (middle layer) once the request has been processed. The middle layer are the services that contain the business logic that the system uses. From there the services communicate with the repository layer which has the sole purpose of handling communications to the data source and formatting the data written to and sent from the repository layer. Note that this does not apply to the **RecommendationService** because the recommendations themselves are not stored in the data source of the system but are calculated/estimated based on ratings. Therefore the RecommendationService needs only to call the RatingService and not a repository (the implementation of the RecommendationService implementation is explained in more detail in next section via class diagram).

To further explain the component diagram for the Facebook Authentication Service (Diagram 5), it shows that the container is split up in two components; the first one, the **Authentication Service**, is the entry point to the Facebook Authentication Service container. The Web Service container calls the Authentication Service to perform user authentication which then calls to the other component within the Facebook Authentication Service container for authenticating the user.That other component is a RestFB component called **Facebook Connector** and therefore contains functions imported from external sources to implement RestFB functionality (a minimal Facebook GraphAPI client that connects to facebook). Said Facebook Connector then connects to Facebook to and authenticates user. Therefore, the Authentication Service component is a mere entry point whereas the Facebook Connector essentially conducts all functionality and external connections for the overall container.

SERVICE OVERVIEW Recommendation Service Class Diagram

For a better overview and visual representation on how the recommendation service functions, how it communicates and relates with other classes, it's member variables and methods refer to **Diagram 5** for a **Class Diagram** of the recommendation service.



<<Interface>> IRatingsService
+ getRatingsForUser(userId): Rating[0...n]
+ getAllRatings(): Rating[0...n]
<<Interface>> IRecommendationService
+ getRecommendations(userId): Recommendations
RatingsService
+ getRatingsForUser(userId): Rating[0...n]
+ getAllRatings(): Rating[0...n]
RecommendationService
+ getRecommendations(userId): Recommendations
- findRecommendationsFromReview(userId): Recommendations
Rating
+ userId: int
+ tapeId: int
+ rating: int
User
+ id: int
+ name: String
+ email: String
+ address: String
+ telephone: String
Recommendations
+ userId: int
+ tapes: Tape[0...n]
Tape
+ id: int
+ title: String
+ director: String
+ type: String
+ releaseDate: Date
+ eidr: String
1
1
1
0..*

DIAGRAM 5 Class Diagram for the Videotapes Galore system's Recommendation Service

The conceptual design behind the implementation of the Recommendation Service is explained the class diagram Diagram 5; in said class diagram the **Recommendation Service** bases the recommendation for videotapes for user on the functionality of the **Rating Service** by exploring user ratings, and therefore has unidirectional association to that service. Rating Service serves **Ratings** to the Recommendation Service, and Ratings have a aggregate relationship with one **Tape** and one **User** (a single rating applies to exactly one tape and a is rated by exactly one user). Then, based on this Rating, the Recommendation Service generates a **Recommendation** for user based on some business logic conducted within function findRecommendationsFromReview(). This Recommendation has an aggregate relationship with one or more Tapes (the tapes that are recommended) for that one User that recommendation was requested for (with whom the Recommendation also has aggregate relationship with).

Note that the interfaces **IRatingsService** and **IReccommendationService** included in the diagram are not particularly relevant for showing any class relationships nor for explaining the design concept, but kept in diagram to show that the system aims to honor good practises and implementation to interfaces for maintainability and extensibility.

DATABASE OVERVIEW Database Schema

For a better overview and visual representation on how the back-end database is structured for the Videotapes Galore system, on the relation between data and key data models and choices of the system's resource design please refer to **Diagram 6** for a **Database Schema** for visual representation of the back-end database.

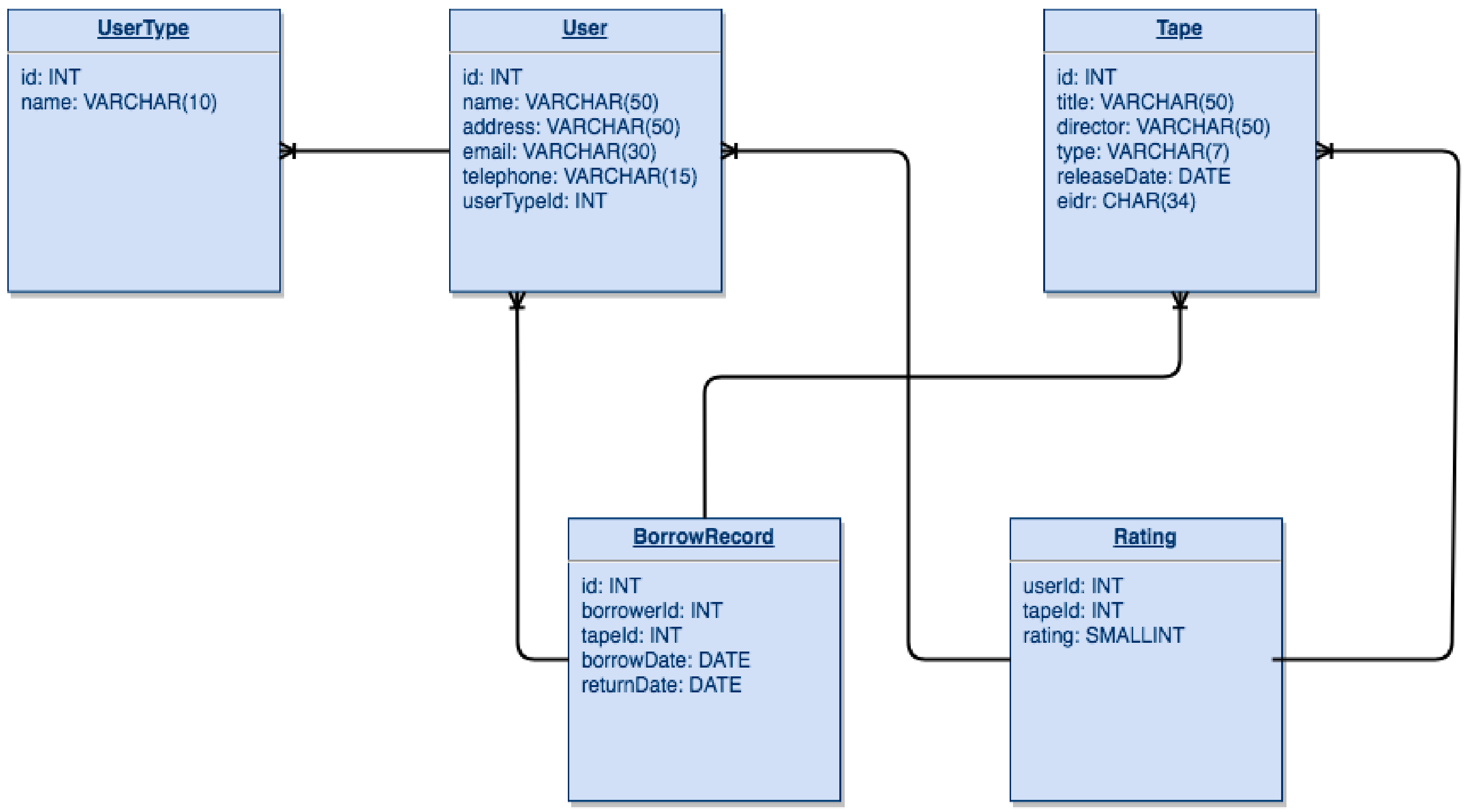


DIAGRAM 6 Database Schema for Videotapes Galore system's back end

To further clarify, the Diagram 6. depicts the five types of entity models in our database and the relation between them. To further clarify the diagram, the entity models shown and their relationships are as follows:

1. The **User** entity model which represents user of the system which has relevant personal information and a user type id associated with it defining that user's role.That user type id is a foreign key referencing the UserType entity model.
2. The **UserType** entity model is a simple and independent entity model that has an identification and name of specific user group associated with user role and user access rights (i.e. whether user is anonymous, authenticated or an admin).
3. The Tape entity model is an independent entity model representing videotape in the system's library of videotapes and contains relevant information on the tape.
4. The **BorrowRecord** entity model represents a record of some tape borrowed to a user in the system and therefore has the borrower's user id number as a foreign key referencing User entity model and the borrowed tape's id number as a foreign key referencing the Tape entity model as well. In addition to this it stores further relevant information on the borrow, e.g. date of borrow and date of return.
5. The **Rating** entity model stores information on the rating that a user gives a videotape and therefore has the rater's user id number as a foreign key referencing the User entity model and the rated tape's tape id number as a foreign key referencing the Tape entity model. In addition to this the rating value is of course stored.

These entity models and their relations enable the functionality of connecting user to a role, ratings to users and tapes and borrow record to users and tapes. Therefore the database entity connections are crucial as these connection enable the functionality of granting access rights to users correctly, storing a borrow record in system as necessary and rating a given videotape.