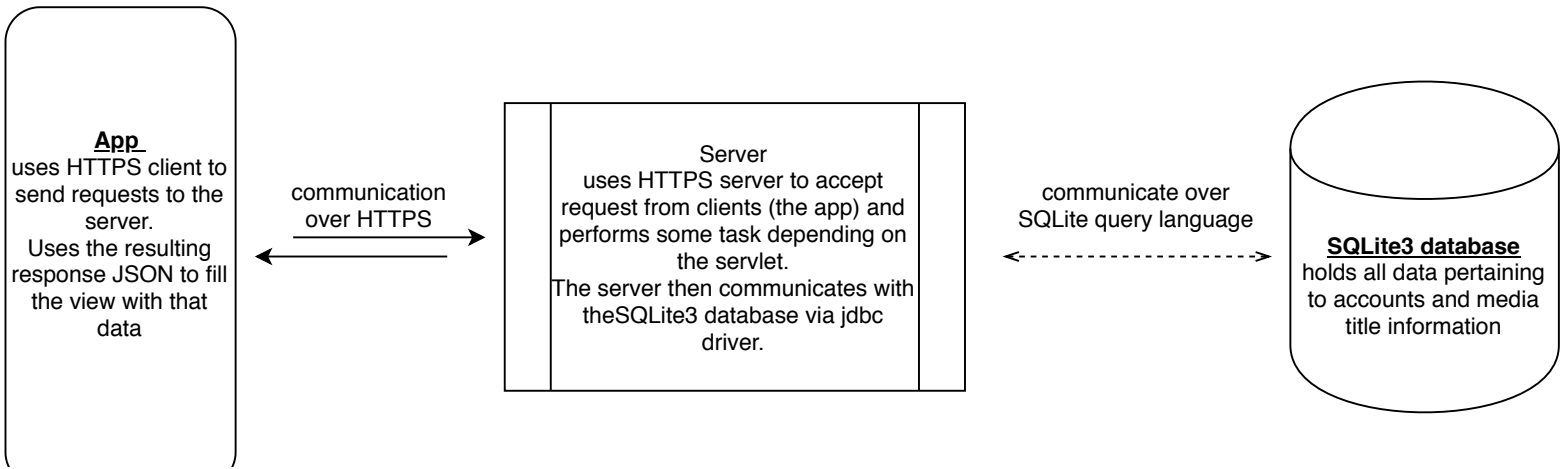
Basics


App
uses HTTPS client to send requests to the server.
Uses the resulting response JSON to fill the view with that data
communication over HTTPS
Server
uses HTTPS server to accept request from clients (the app) and performs some task depending on the servlet.
The server then communicates with theSQLite3 database via jdbc driver.
communicate over SQLite query language
SQLite3 database
holds all data pertaining to accounts and media title information

Database (this is a work in progress)

'accounts' : User Accounts

userID PRIMARY KEY, INTEGER
username TEXT NOT NULL UNIQUE
email TEXT NOT NULL UNIQUE
passwordhash TEXT NOT NULL
favorites TEXT DEFAULT ""
bookmarks TEXT DEFAULT ""

'admins' : Admin Accounts(child of user accounts)

adminID INTEGER PRIMARY KEY REFERENCES accounts(userID)
adminLevel SHORT NOT NULL DEFAULT 1

'ratings' : Rating relation (user)--[rated on]-->(media title)

ratingID INTEGER PRIMARY KEY
rating FLOAT NOT NULL
userID INTEGER NOT NULL
FOREIGN KEY (userID) REFERENCES accounts (userID)
mediaID INTEGER NOT NULL
FOREIGN KEY (mediaID) REFERENCES mediaTitles (mediaID)

'reports' : Reports relation (user)--[reported on]-->(review)

reportID INTEGER PRIMARY KEY
reviewID INTEGER NOT NULL
FOREIGN KEY (reviewID) REFERENCES
reviews (review_ID)
user TEXT NOT NULL
FOREIGN KEY (user) REFERENCES
accounts (userID)

'reviews' : Reviews relation (user)-[reviewed]->(mediaTitle)

review_ID INTEGER PRIMARY KEY,
user_ID INTEGER NOT NULL
FOREIGN KEY (user_ID) REFERENCES
accounts (userID),
media_ID INTEGER NOT NULL
FOREIGN KEY (media_ID) REFERENCES
mediaTitles (mediaID),
review_text TEXT (500) DEFAULT "",
review_title TEXT (60) DEFAULT "",
report TEXT (1000) DEFAULT "",
numReports INTEGER DEFAULT 0

'musicInfo' : Music Titles (unique attributes / columns)

mediaID INTEGER PRIMARY KEY
REFERENCES mediaTitles(mediaID)
producer STRING DEFAULT "
director STRING DEFAULT "
writter STRING DEFAULT "
editor STRING DEFAULT "
starring STRING DEFAULT "
productionCompanies STRING DEFAULT "
distributors STRING DEFAULT "
releaseDate STRING DEFAULT "
runningTime STRING DEFAULT "
language STRING DEFAULT "
musicBy STRING DEFAULT "
cinematography STRING DEFAULT "
budget STRING DEFAULT "
boxOffice STRING DEFAULT "

'mediaTitles' : Media Titles (shared attributes / columns)

mediaID INTEGER PRIMARY KEY
title TEXT NOT NULL
mediaType TEXT NOT NULL
summary TEXT
genres TEXT NOT NULL
tags TEXT DEFAULT "
rating FLOAT DEFAULT 0.0
numRaters INTEGER DEFAULT 0
links STRING DEFAULT "
image BLOB

use media types {'cinema', 'music', 'tv-series', 'video game', 'novel'}

genres depend on media type: example 'action,horror'

tags depend on media title ex. 'swords,blood,wheels'

'cinemaInfo' : Cinema Titles (unique attributes)

mediaID INTEGER PRIMARY KEY
REFERENCES mediaTitles(mediaID)
album STRING DEFAULT "
artist STRING DEFAULT "
duration FLOAT DEFAULT 0.0
year STRING DEFAULT "

'tvseriesInfo' : TV-series Titles (unique attributes)

mediaID INTEGER PRIMARY KEY
REFERENCES mediaTitles(mediaID)
producer STRING DEFAULT ''
director STRING DEFAULT ''
writter STRING DEFAULT ''
starring STRING DEFAULT ''
cinematography STRING DEFAULT ''
releases STRING DEFAULT ''
seasons STRING DEFAULT ''
episodes STRING DEFAULT ''
episodeLength STRING DEFAULT ''
language STRING DEFAULT ''

year STRING DEFAULT

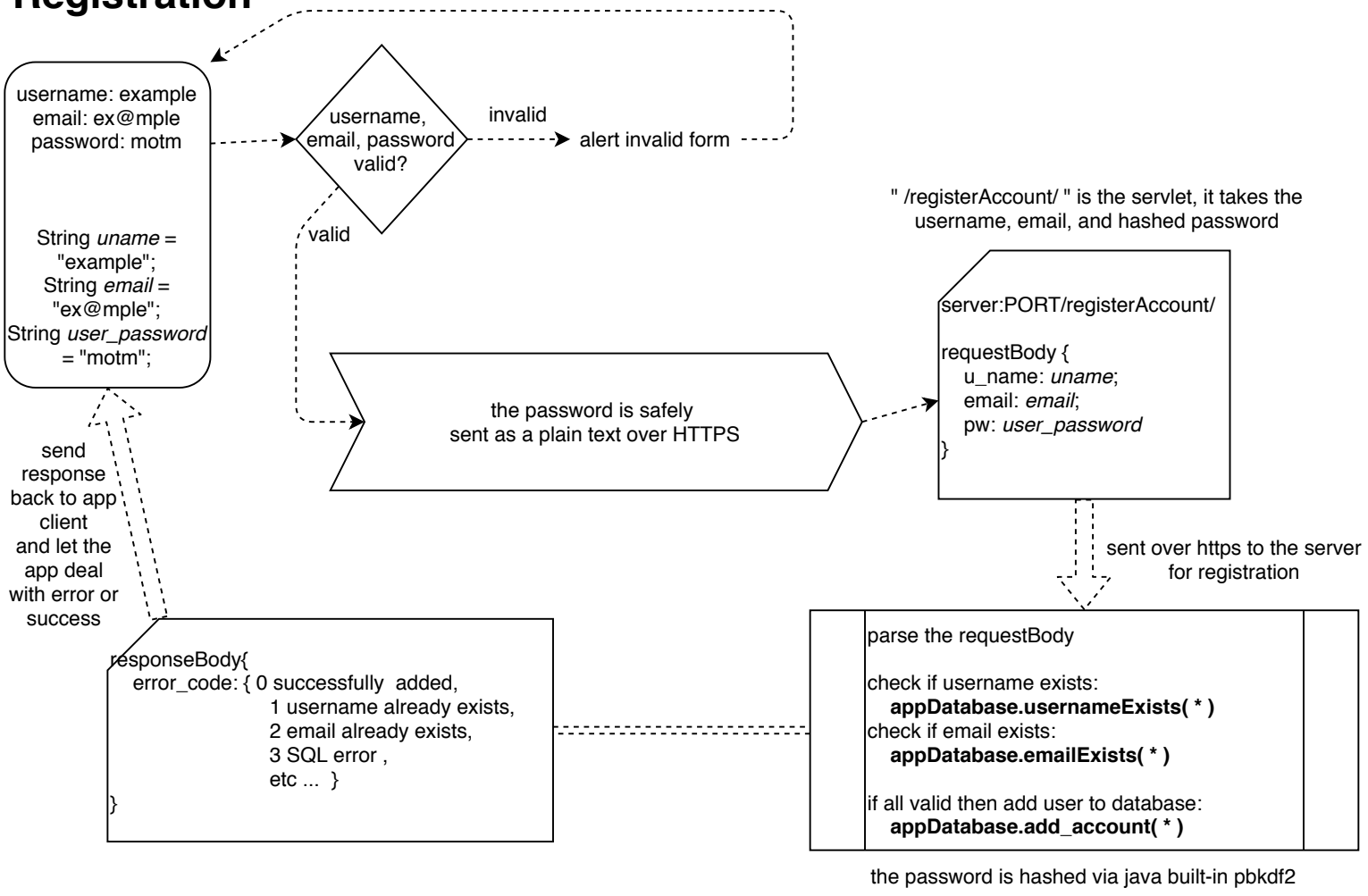
'videogameInfo' : Video Game Titles (unique attributes)

mediaID INTEGER PRIMARY KEY
REFERENCES mediaTitles(mediaID)
developers STRING DEFAULT ''
publishers STRING DEFAULT ''
directors STRING DEFAULT ''
designers STRING DEFAULT ''
programmers STRING DEFAULT ''
artists STRING DEFAULT ''
writers STRING DEFAULT ''
composers STRING DEFAULT ''
engine STRING DEFAULT ''
platforms STRING DEFAULT ''
release STRING DEFAULT ''
modesOfPlay STRING DEFAULT ''

'novelInfo' : Novel Titles (unique attributes / columns)

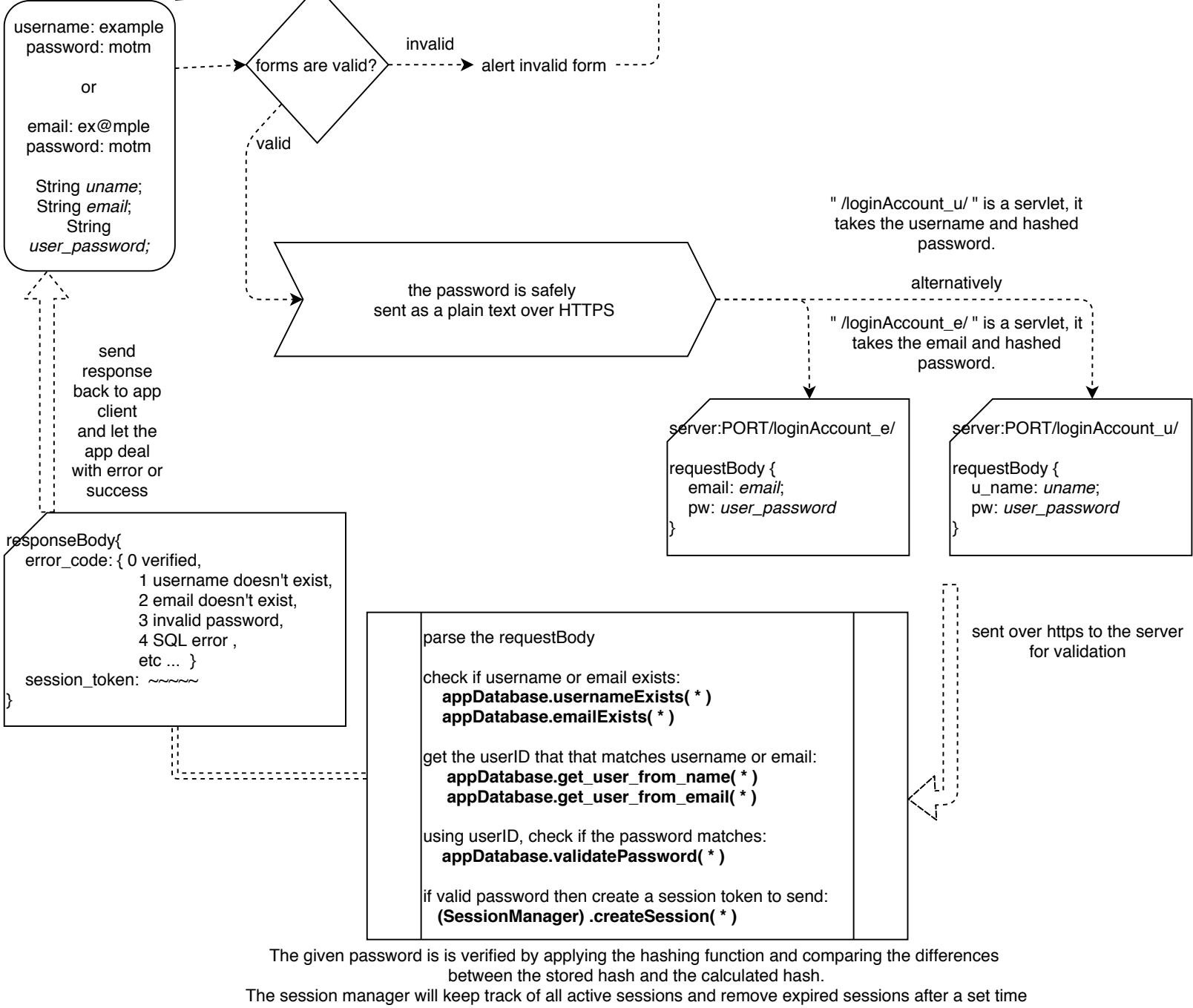
mediaID INTEGER PRIMARY KEY
REFERENCES mediaTitles(mediaID)
author STRING DEFAULT ''
country STRING DEFAULT ''
language STRING DEFAULT ''
publisher STRING DEFAULT ''
publishedDate STRING DEFAULT ''
numBooks SHORT DEFAULT 0
numChapters SHORT DEFAULT 0
status STRING DEFAULT ''

Registration

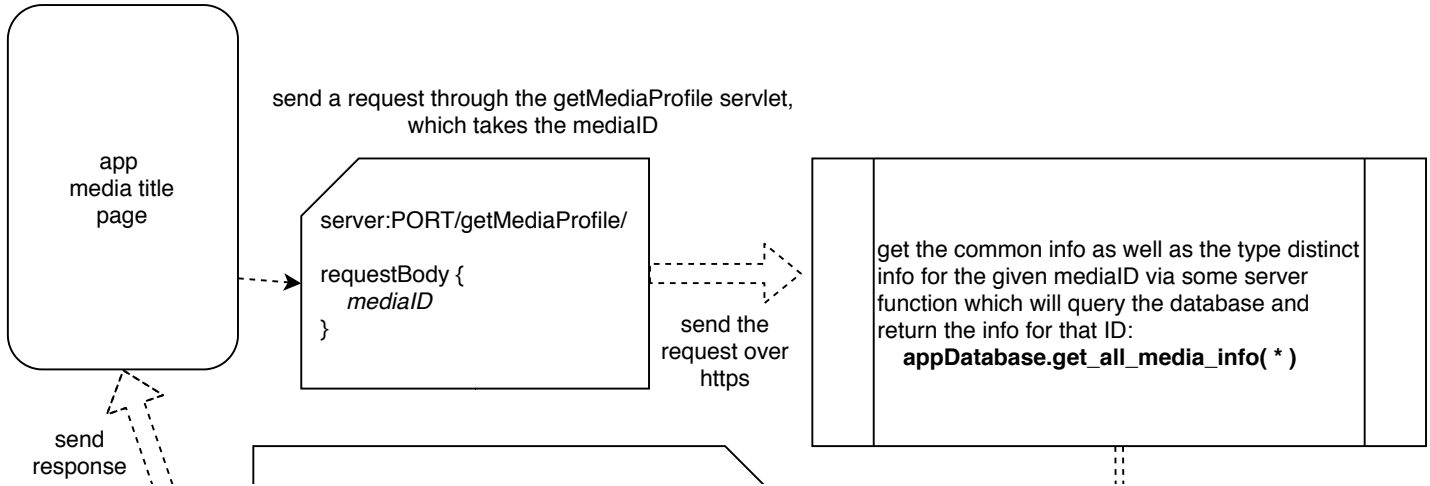


the password is hashed via java built-in pbkdf2

Validation



get Media Profile



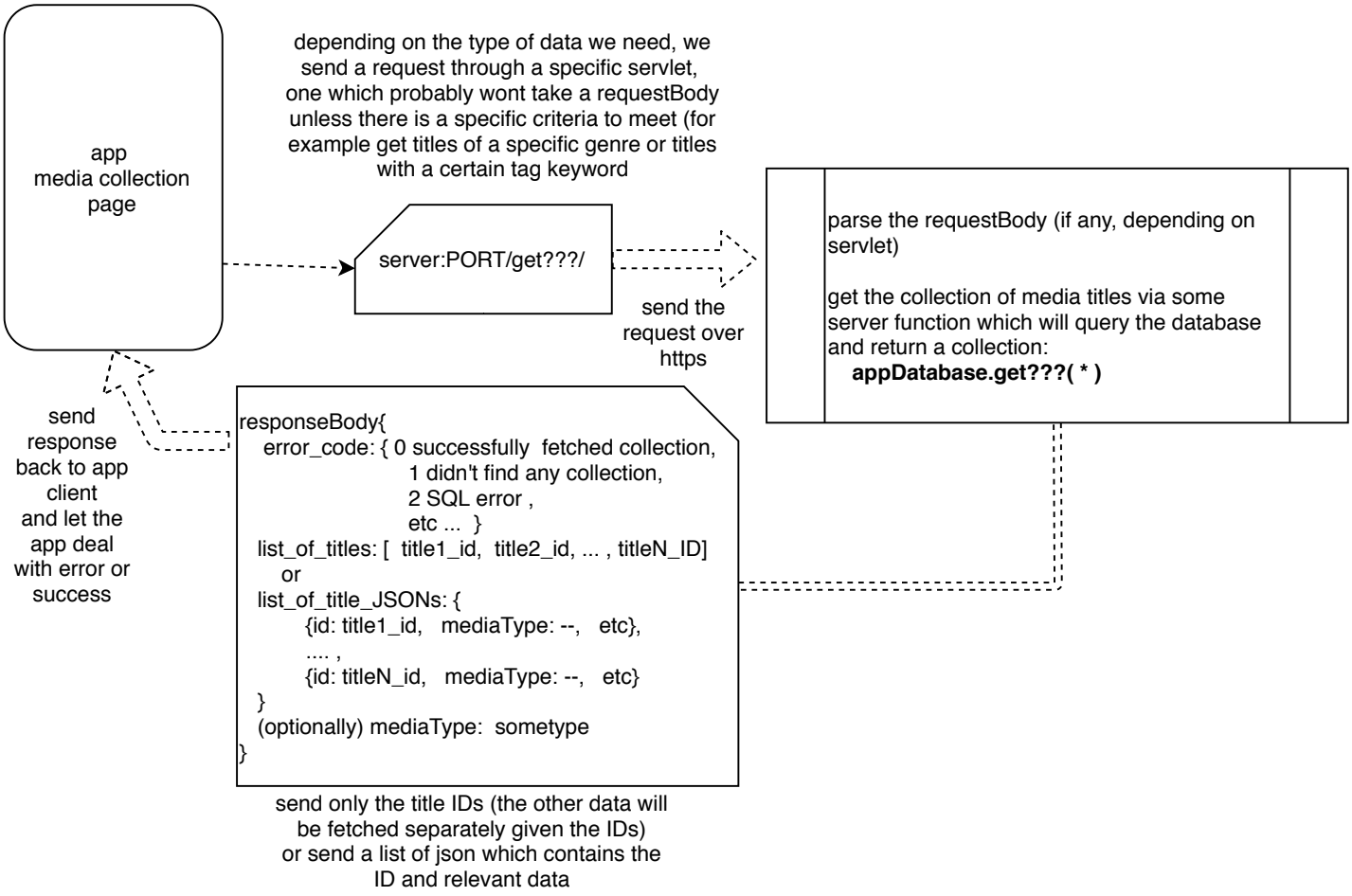
send response back to app client and let the app deal with error or success

responseBody{
error_code: { 0 successfully fetched collection,
1 invalid mediaID,
2 critical server error ,
etc ... }
common: {~:~, , , }
distinct: {~:~, , , }
}

cover_img (sent separately)

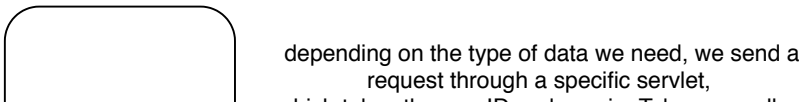
send all necessary data (image will probably have to be sent separately due to http functionality)

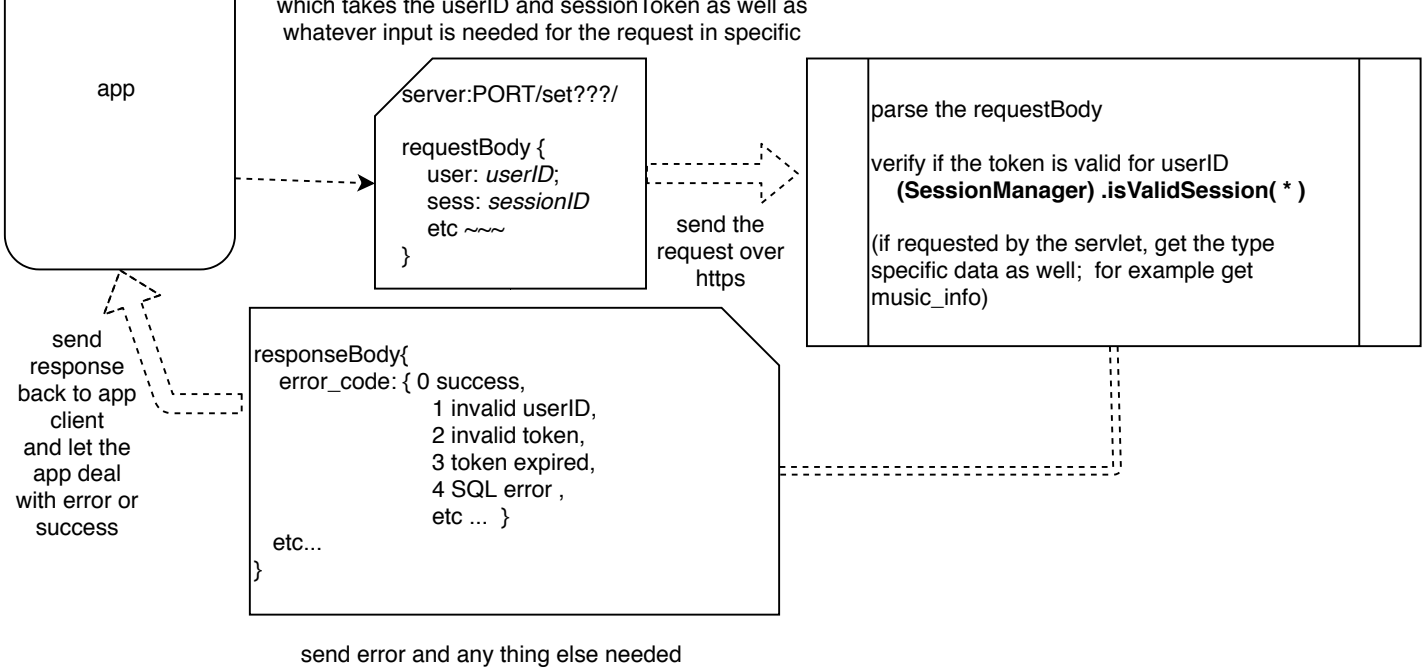
example of a general endpoint
ex: get collection of Media titles



account sensitive request (example)

(post review, set rating, make changes to account page, etc)




which takes the userID and sessionToken as well as
whatever input is needed for the request in specific
app
server:PORT/set???/
requestBody {
user: userID;
sess: sessionID
etc ~~~
}
send the
request over
https
parse the requestBody
verify if the token is valid for userID
(SessionManager) .isValidSession(*)
(if requested by the servlet, get the type
specific data as well; for example get
music_info)
send
response
back to app
client
and let the
app deal
with error or
success
responseBody{
error_code: { 0 success,
1 invalid userID,
2 invalid token,
3 token expired,
4 SQL error ,
etc ... }
etc...
}
send error and any thing else needed