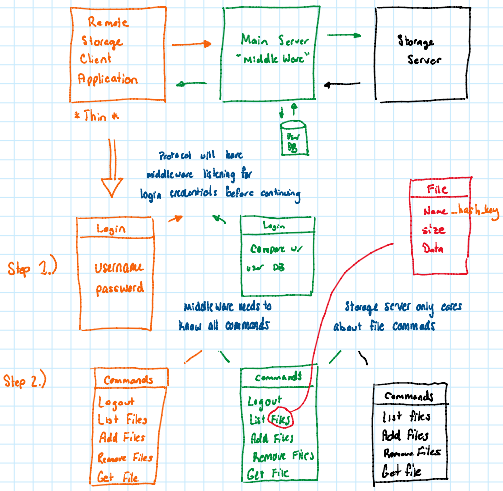
Proj: Part One

Wednesday, February 8, 2023 8:33 PM

Data Model:



★ Main Menu will be sent on success connection from middleware to client ★

Login :
lgn : size_of_creds : username pass

Logout :
lgot

List :
list

★ Will client manually enter sizes, or will we implement helper function to calculate name/data sizes ★

add :
push : size_of_data : data

Remove :
Rmve : size_of_name : name

Get :
Pull : size_of_name : name

Client only recieves:
Good : Size : data
or
Erro

Login:

Business Logic

1.) Connect to middleware

2.) send "lgn : size : login_cred"
Send to middleware

1.) If success,
"good : size : menu"
else
"erro" return step 1

2.) Unmarshal user input
compare to DB on server
If true :
send menu
else :
return step 2, 3 tries, return step 1.

hash username + password to reference files

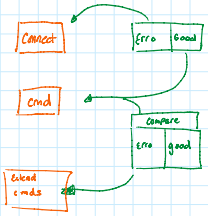
App Protocol

2.) Establish Connection → 1.) "good : size : data" or "erro"

whole = input ("cmd")
lgn : size : usr passw
mess = whole.encode (utf-8)

2.) Decode message
whole = read x bytes after ":"
compare to DB results
Success :
Send "Good : size : data"
close :
Send "Erro"

Step 2 in Data Model



Logout:

Business Logic

1.) Send logout command

2.) Re-login or close connection

1.) send to login page on success, else error

Application Protocol

1.) cmd = input ("Enter command :")
mess = cmd.encode (utf-8)

1.) If bytes [0-3] = "Logo"
↳ send "good : size : menu_data" login protocol
else
↳ send "erro"

2.) Analyze Menu + decide

List:

Business Logic

1.) Send List command →
1.) Check User's hash value
2.) Look for files w/ hash value
3.) Good ERRO
2.) Read, continue commands

Store, Retrieve, + Delete will follow same as List. The client will not get a response until the middleware connects w/ the DB Server. If middleware receives "good" or "error" from DB, it needs to send that info to client. Client never uses these commands so no problems there.

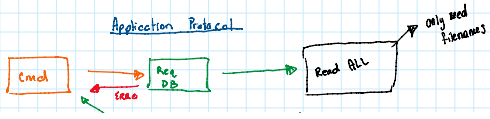
→ use hashing algorithm to generate file names

Store:

1. Send command
↳ "push : size : data"

1.) Receive command
↳ Add user's hash to filename
Add to local DB
Send : "push : size : data"

1.) Receive Command
↳ send "good :0 :"
or
send "ERRO"



1. Send command
   ↳ "push : size : data

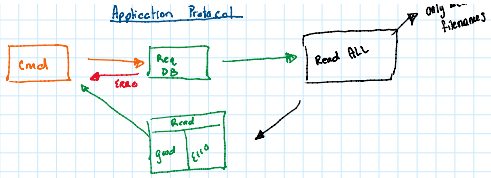
2. Receive message from middleware
   ↳ Continue w/ commands

1.) Receive command
   ↳ Add users hash to filename
   Add to local DB
   Send : "push : size : data"

2.) Receive Message From DB
   ↳ Send "Good : 0 : NULL" to client
   or "ERRO"

1.) Receive Command
   ↳ send "good : 0 : "
   or
   send "ERRO"

Application Protocol



Retrieve :

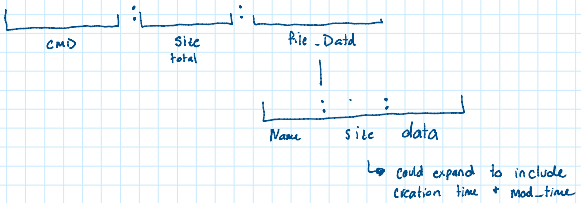
1. Send command
   ↳ "Pull : size : name"

2. Receive cmd
   ↳ Continue sending commands.

1.) Receive Message From DB
   ↳ verify hash of filename
   Send to DB "Pull : size : name"
   ERRO or Permission Denied

2.) Receive Message from DB
   ↳ Send to client
   "Good : size : data"
   or
   "ERRO"

1.) Receive Command
   Perform Actions
   Search for file
   Get name, size, & data
   Send "Good : size : data"
   or
   "ERRO"



↳ could expand to include creation time & mod_time

Delete :

1.) Send to Middleware
   "Rmve : size : filename"

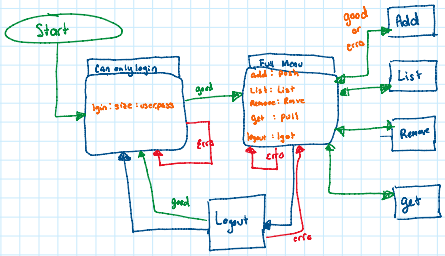
2.) Receive Message
   ↳ continue commands

1.) Receive Command
   ↳ verify user & file hash
   Send "Rmve : size : filename" to DB
   error or no permission
   or ERRO back to client

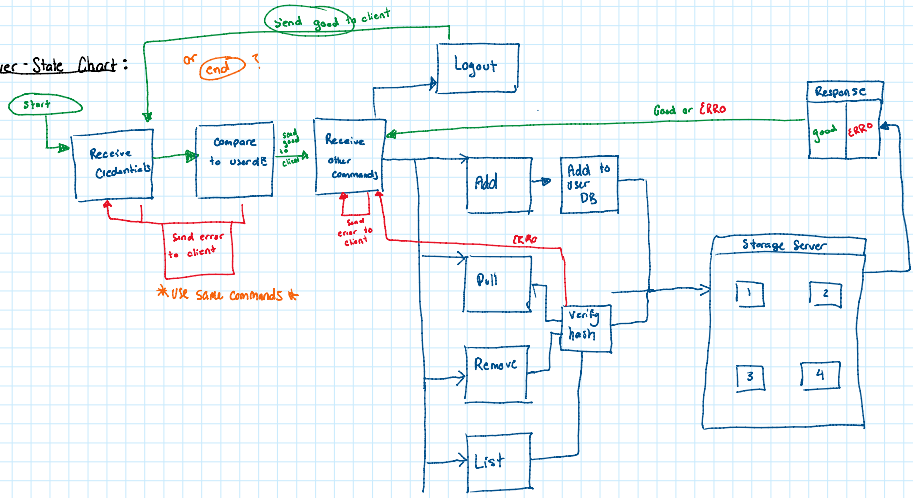
2.) Receive Message
   ↳ Send to Client
   Update local DB by removing file

1.) Receive command
   ↳ Find file & Delete
   Send "Good : 0 : "
   or
   "ERRO"

Client - State Chart :



Server - State Chart :



Keep Concurrency between user DB & DB Server

Ex.) User attempting to list all of their stored files.

Client : Connect to middleware

Middleware : Send "good : size : menu_data"

Client : Send "lgin : size : user password"

Middleware : Send "good : 0 : NULL"

Client : Send "list"

Middleware : Send "list" to server

Server : Find Files, Send "good : size : data" to Middleware

Middleware : Send "good : size : data" to client

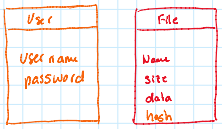
Client : Send next command

* Important Note *

- Stay consistent w/ commands & messages to prevent reading old data or not enough data
- Keep Client thin, should only send commands & decode incoming messages

messages

- keep file information format consistent



In order to verify that users can only view their files, we will include a hash to all saved files that is unique to each user. Upon login, their hash will be used to authenticate commands.

The user is unaware of these hash values.

Ex.) Client : push : 4a : data

Middleware : generate file name

↓

Name = hash + filename

↓

Send response "Good : 0:"

Client : pull : 4a : filename

Middleware : Find file that contains user's search

↓

Compare user hash w/ file hash

↓

Good Bad

Client : Next command