app.js - backend

a class for encrypt/decrypt, hash, interaction with the DB, generate keys

UserRoute

-
makeKeyPair() - make an asymmetric key
-encryptData() - encrypt by the public Key
-decryptData() - decrypt by the private Key
-hashPassword() - generate a hash
-comparePassword() - compare - is the password corect
-saveUsr() - save a new user into the DB
-updateUsr() - update a user data (password,name)
-isUserInDB() - checking: is the the user exists in the DB
-getSessionStatus() - checking: is the session active or not
-setSession() - set a session status (active or not)
-getUsr() - get a user data
-readKeys() read a public key and a private key

<< interfaces >>

async pRegisterUser (x = { usrId: '', password: '', usrName: '' }) - register a new user
@return { status: 'succ', result: rows.affectedRows } status can be only 'fail'/'succ'

async pUpdateCommonKeys (tablename = 'commonkeys', psw = 'x512') - generate a new asymmetric key and sav
@return mySQL rows after the update operation

async pGenerateCryptoCookie (usrId, cryptoPassword = 'x512') - encrypt currentDATE+usrId
@return { status: 'succ', result: encryptedCookie } result is a <Buffer>

async pValidateCryptoCookie (crCookie = Buffer.from('123'), cryptoPassword = 'x512')
@return {ststus, usrId, created} status:'unauthorized','fail','succ'

async pStartSession (x = { usrId: '', password: '' }, cryptoPassword)
1)checking the password
2)write 'active' in the SQL - users.sessionId
3)encoded timestamp+usrId
@return {status:'succ', encodedCookie:<Buffer>}

a class for interaction a student with the DB during the exem

Exem

async pWriteUserResponse (usrId, qId, qKey, tablename = 'exem') - write a rsponse into the DB
@return { status: 'succ' }

async pSetUsrTime (usrId, tablename = 'exemTime') - set a User`s start time in the DB
@return { status: 'succ' } or 'fail'

async pGetUserScore (usrId)
@return: { status: 'succ', result: rows[0] }

async pGetTicketsNew (tabname = 'question') - get a tickets and variants of responses
@return rows[]

async pGetUsrTimeout (usrId) - take a start exem Time of a User
@return { status: 'succ', result: rows[0].startTime }