JS UGR NODE.JS SERVER

API

USER MANAGEMENT API

/register (post)

- Requires req.body to contain: {user: “username”, pass: “password”, email: “user@domain”}.
- Creates an entry in the users database with the provided information and a 32 digit hexadecimal validation code and a validation flag set to false.
- Returns json {ok: *true*}. or {ok: *false*, error: *string*, reason: *string*}.

/confirm (post)

- Requires req.body to contain: {user: “username”, code: “validation code”}.
- Checks in the user database for an entry for “username” and checks the provided validation code with the one sent by mail. If they check it sets validate flag to true.
- Returns json {ok: *true*}. or {ok: *false*, error: *string*, reason: *string*}.

/request_pwd (post)

- Requires req.body to contain: {user: “username”}.
- Changes the validation code in the database for user “username” and sends an email to the user’s address with said validation code.
- Returns json {ok: *true*}. or {ok: *false*, error: *string*, reason: *string*}.

/reset_pwd (post)

- Requires req.body to contain: {user: “username”, pass: “password”, code: “validation code”}.
- Checks that the validation code is the same as in the database and changes the password.
- Returns json {ok: *true*}. or {ok: *false*, error: *string*, reason: *string*}.

/login (post)

- Requires req.body to contain: {user: “username”, pass: “password”}.
- Checks the user entry in the database and if the password matches it saves the username, a 32 character hexadecimal token and the login time to an array. Sessions expire after 15 minutes of inactivity.
- Returns json {ok: *true*, token: *string*} or {ok: *false*, error: *string*, reason: *string*}.

/logout (post)

- Requires req.body to contain: {user: “username”,token: “32char token”}.
- Checks if the origin ip is the same as the login request and if the token is the same as the one generated at login and deletes the username from the login array.
- Returns json {ok: *true*}. or {ok: *false*, error: *string*, reason: *string*}.

/check (post)

- Requires req.body to contain: {user: “username”,token: “32char token”}.
- Checks if the origin ip is the same as the login request, the token is the same as the one generated at login, the session has not expired, and if “username” is in the login array and updates the login time.
- Returns json {{ok: *true*, user: *string*}. or {ok: *false*, error: *string*, reason: *string*}.

FILE MANAGEMENT API

All the requests in this part of the api except /share require the user to be logged in. For this check, req.body has to contain a field “user:” and a field “token:” that is generated on login and should be stored in the client, in a cookie for example. If no request is made in 15 minutes, the session is closed, and the token is no longer valid. The user should have to authenticate again.

/publish (post)

- Requires req.body to contain:

```
{
"user" : user,
“token”: token,
"library": USER DEFINED - library to include,
"title": USER DEFINED - title of the website,
"html" : USER DEFINED - contents of index.html file,
"css" : USER DEFINED - contents of style.css file,
"js" : USER DEFINED - contents of script.js file,
}
```

- Publishes the files in the user directory on the server. references all defined files in head of index.html file.
- Returns json {ok: true, html: css, html: true, js: true} or {ok: false, html: true/false, js: true/false, css: true/false, error: …, reason: … }.

/save (post)

- Requires req.body to contain:

```
{
"user" : user,
“token”: token,
"library": USER DEFINED - library to include,
"title": USER DEFINED - title of the website,
"html" : USER DEFINED - contents of index.html file,
"css" : USER DEFINED - contents of style.css file,
"js" : USER DEFINED - contents of script.js file,
"time": LOCAL USER TIME,
"id": file id
}
```

- Saves all fields except token in a document in the couchdB docs database. If an id is set and there is a file on the server with said id, it will overwrite the file on the server if the owner is the one making the request.
- Returns json {ok: true, id: “file key”} or {ok: false, error: …, reason: … }. The file key is the key necessary to retrieve the document later on.

/delete (post)

- Requires req.body to contain:

```
{
"user" : user,
“token”: token,
id: “file id”
}
```

- Deletes file “file id” from the couchdB database if the file’s owner is the same as the sender of the request.
- Returns json {ok: true} or {ok: false, error: …, reason: … }.

/get_doc (post)

- Requires req.body to contain:

```
{
"user" : user,
“token”: token,
id: “file id”
}
```

- Returns the file “file id” from the couchdB database if the file’s owner is the same as the sender of the request.
- Returns json {ok: true, body:{“file contents”}} or {ok: false, error: …, reason: … }.

/all_docs (post)

- Requires req.body to contain:

```
{
"user" : user,
“token”: token,
}
```

- Returns a list of all file id’s, titles and time of creation for the specified user.
- Returns json {ok: true, body:{[{id: “file id”, key: “title”, value: “timeof file creation”},{…},…]} or {ok: false, error: …, reason: … }.

/share (post)

- Requires req.body to contain:

```
{
id: “file id”
}
```

- Returns the file “file id” from the couchdB database. The owner of this file has made the id public.
- Returns json {ok: true, body:{“file contents”}} or {ok: false, error: …, reason: … }.

MISC

/poll (post)

/results (post)

/raffle (post)

MODULES

EXTERNAL

express

fs

cradle

node-uuid

cron

forever

nodemailer

CUSTOM

jsUgrSessions

jsUgrRegistration

sha256

SECURITY

On registration

The registration process is two-step and requires the user to open an email sent by the system and copying a code into a registration form.

On Sessions

The sessions are maintained by the server in an array. The username, as well as a 32 hexadecimal characters long token and the login time are stored. On every petition, the token must be included in the petition body. If the token checks against the stored one for the given username and the last action wasn't more than 30 minutes ago the action may be executed, else an error is returned.

On Documents

Each document is identified by its uuid. All documents are stored in a single database, but may be modified only by its creator. The files can be read by anyone in possession of the uuid.

MAINTENANCE

The maintenance of the server is done with a module called cron.

When a user tries to register but does not confirm, the entry is left in the database. Each day at 5AM the entries in the user database that have expired validation codes are deleted.

When someone attempts to login but his password fails, and then doesn’t input the right password the entry in the security array on the server is left there. Every hour those entries are deleted.

If a user leaves the page without logging out, the entry (tho useless for an attacker because it’s expired) is left in the users array in the server. These entries are removed once every hour.