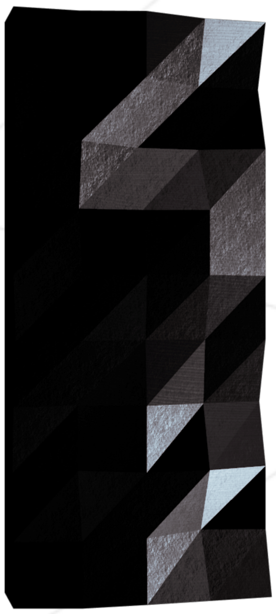

Web

Projet Hypertube

42 Staff pedago@staff.42.fr

Summary: a web app for the 21th century.

Contents

Chapter I

Foreword

The word “cafetière” [1] was originally the person that maintained a coffee house, a place where one could drink coffee. We have reasons to believe from notes of famous writers that the word existed already in the XVIII century when an old bishop gave 25 gold coins to his nephew to acquire a “beautiful cafetière” only to realize too late that she wore a skirt.

When coffee percolation was invented, the word “cafetière” became then the kitchen appliance we know today, ie: coffee maker.

Coffee makers in the 90’s have greatly contributed to the development of modern technologies..

For example, the researchers of the University of Cambridge in England, created in 1991 the first prototype of webcam, for the unique purpose of checking how much coffee was left in the coffee maker of the “trojan room” [2].

Figure I.1: La cafetière de la Trojan Room

In 1998, the Internet Society publishes a reference document [3] concerning the management of connected objects of type “hot brewerage electronic dispenser” [4] directly followed by the reference “Hyper Text Coffee Pot Control Protocol” (HTCPCP), defining a protocol of client-server communication, extension of the HTTP protocol, allowing the control, surveillance and diagnosis of the coffee maker.

[1] coffee maker

[2] http://www.cl.cam.ac.uk/coffee/coffee.html

[3] https://tools.ietf.org/html/rfc2325

[4] More precisely: “Definitions of Managed Objects for Drip-Type Heated Beverage Hardware Devices using SMIv2”

The requests of the coffee maker are then identified by the `coffee://` URI (or the name of the coffee in one of the 29 languages listed in the RFC including French) and supporting the following methods:

- `GET` gets the coffee from the coffee pot.[5].
- `POST` (or `BREW`) starts the infusion of coffee by the coffee pot.
- `PROPFIND` displays metadatas about the coffee.
- `WHEN` notifies the coffee pot to stop adding milk in the coffee (when relevant).

Some headers `Accept-Additions` can be added to define the type of milk, syrup, sugar substitute, spice or even alcohol (let's not forget about the Irish).[6]

To avoid any stupid mistake, a teapot would send back the error code 418 "I'm a teapot". The absence of a well-defined protocol for these started a strong argument from the tea lovers.

In 2014, Imran Nazar submitted to the Internet Engineering Task Force the reference of the "Hyper Text Coffee Pot Control Protocol for Tea Efflux Appliances" (HTCPCP-TEA), a variation of HTCPCP for teapots, ending at that moment the hostilities.

Although these RFC are April's fools, they are described with enough details to be implemented.

```
To this day not student from 42 has ever managed to create a "HTCPCP
compliant" coffee pot.
```

[5]This point is open to interpretation, Check RFC
[6]https://tools.ietf.org/html/rfc2324#section-2.2.2.1

Chapter II

Introduction

This project proposes to create a web application that allows the user to research and watch videos.

The player will be directly integrated to the site, and the videos will be downloaded through the BitTorrent protocol.

The research engine will interrogate multiple external sources of your choice, like for example http://www.legittorrents.info, or even https://archive.org.
Once the element selected, it will be downloaded from the server and streamed on the web player at the same time. Which means that the player won't only show the video once the download is completed, but will be able to stream directly the video feed.

Chapter III

General Instructions

- For this project you are free to use any language you choose.
- All the framework, micro-framework, libraries etc… are authorized within the limits where they are not used to create a video stream from a torrent, thus limiting the educational purpose of this project. For example, libraries such as `webtorrent`, `pulsar` and `peerflix` are forbidden.
- You are free to use the server of your choice, may it be `Apache`, `Nginx` ou même un `built-in web server`.
- Your whole application will have to be at minimum compatible with `Firefox` (>= 41) and `Chrome` (>= 46).
- You're free to use the web server you like most may it be `Apache`, `Nginx` or a `built-in web server`.
- Your whole app must be compatible at least with `Firefox` (>= 41) and `Chrome` (>= 46).
- Your website must have a decent layout: at least a header, a main section and a footer.
- Your website must be usable on a mobile phone and keep an acceptable layout on small resolutions.
- All your forms must have correct validations and the whole website must be secure. This part is mandatory and will be checked extensively in defense. To give you an idea, here are a few elements that are not considered secure:
    - To have a "plain text" password stored in your database.
    - To be able to inject HTML of "user" Javascript code in unprotected variables.
    - To be able to upload unwanted content.
    - To be able to alter a SQL request.
- You can ask your questions on the forum, on slack...

Chapter IV

Mandatory part

You will need to create a Web App with the following features:

IV.1 User part

- The app must allow a user to register asking at least an email address, a username, a last name, a first name and a password that is somehow protected.

- The user must be able to register and connect via Omniauth. You must then implement at least 2 strategies: the `42` strategy and another one of your choice.
- The user must then be able to connect with his/her username and password. He/She must be able to receive an email allowing him/her to re-initialize his/her password should the first one be forgotten.
- The user must be able to disconnect with 1 click from any pages on the site.
- The user must be able to select a preferred language that will be English by default.

A user will also be able to:

- Modify the email address, profile picture and information.
- Consult the profile of any other user, ie see the profile picture and information. The email address however will remain private.

IV.2 Library part

This part can only be accessible to connected users.

This part will have at minimum:

- A research field.
- A thumbnails list.

IV.2.1 Research

The search engine will interrogate at least two external sources of your choice, [1], and return the ensemble of results in thumbnails forms.

You will limit the research to videos only.

IV.2.2 Thumbnails

If a research has been done, the results will show as thumbnails sorted by names.

If no research was done, you will show the most popular medias from your external sources, sorted as per the criteria of your choice (downloads, peers, seeders, etc...).

In addition to the name of the video, a thumbnail must be composed, if available, of its production year, its IMDb note and a cover image.

You will differentiate the videos watched from unwatched, as you prefer.

The list will be paginated, at the end of each page. The following one must be automatically charged asynchronically. That means there cannot be a link from each page.

The page will be sortable and filtered according to criteria such as name, genre, the IMDb grade, the gap of production year etc...

[1]comme par exemple `http://www.legittorrents.info`, ou encore `https://archive.org`

IV.3 Video Part

⚠ `This part can only be accessible to connected users.`

This section will present the details of a video, ie show the player of the video as well as – if available - the summary, casting (at least producer, director, main cast etc...) the production year, length, IMDb grade, a cover story and anything you think relevant.

You will also give the users the option of leaving a comment on the video, and show the list of prior comments.

To launch the video on the server we must - if the file wasn't downloaded prior – launch the download from the associated torrent on the server, and stream the video flux from that one as soon as enough data has been downloaded to ensure a seamless watching of the video. Of course, any treatment must be done in the background in a non-blocking manner.

Once the movie is entirely downloaded, it must be saved on the server, so that we don't need to re-download the movie again. If a movie is unwatched for a month, it will have to be erased.

If English subtitles are available for this video, they will need to be downloaded and available for the video player. In addition, if the language of the video does not match the preferred language of the user, and some subtitles are available for this video, they will need to be downloaded and selectable as well.

If the video is not natively readable for the browser [2], you will convert it on the fly in an acceptable format. The `mkv` support is a minimum.

[2]That means it isn't either `mp4`, nor `webm`.

Chapter V

Bonus part

If the mandatory part is entirely done and is perfect, you can now add the bonuses you wish; they will be evaluated at the discretion of your correctors. You will however need to respect the basic constraints. For example, to download a torrent will have to happen from the server's side in the background.

If you are needing some inspiration, here are a few ideas:

- Some additional Omniauth strategies.
- Manage various video resolutions.
- Develop an RESTful API.
- Stream the video via the MediaStream API.

Chapter VI

Submission and peer correction

The following instructions will be part of your defense. Be cautious when you apply them as they will be graded with a non-negotiable 0.

VI.1 Eliminatory instructions

- Your code cannot produce any errors, warnings or notices either from the server or the client side in the web console.
- Anything not specifically authorized is forbidden.
- The slightest security breach will give you 0. You must at least manage what is indicated in the general instructions, ie NOT have plain text passwords stored in your database, be protected against SQL injections, and have a validation of all the forms and upload.
- Finally, you will submit at the root of your repository an author file containing your logins, one per line in this manner:

```
$>cat -e auteur
xlogin$
ylogin$
zlogin$
alogin$
$>
```

You can ask your questions on the forum, on slack...

Good luck everyone!