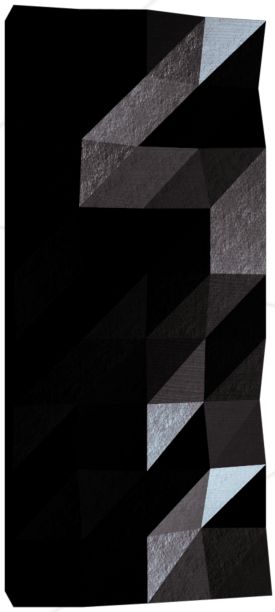


Camagru

Summary: The goal of this project is to build a web application.

Version: 4.1

Contents

Chapter I

Forewords

History of communication is as old as the history of Humanity and Mankind has managed to evolve it along centuries throughout incredible revolutions.

In 1794, Claude Chappe tried to resolve the issue of long-distance communication, that was limited to horse speed at that time. He set up an ingenious communication system of air telegraph during the French Revolution. Chappe's "tours" (towers) were covered by a mobile mast that could be seen with binoculars from the nearest neighbor tower located at approximately 10 to 15 km.

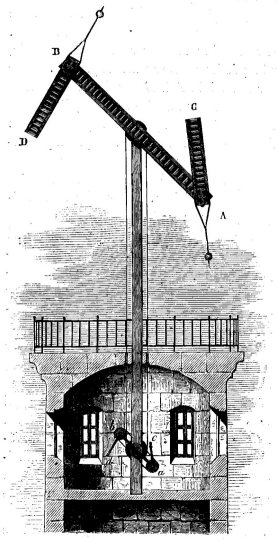


Fig. 19. — Télégraphe de Chappe.

The Paris-Lille line was operational the same year and allows transmissions between the two cities at a speed of 9 minutes per letter throughout 15 towers. Obviously, transmission time depends of its length.

In 1844, 534 towers were dispersed on the French territory linking on more than 5000km, the most important cities.

But, this system had 2 big disadvantages: it couldn't be functionnal by night because, obviously, of the bad visibility, and the number of operators per tower (2 every 15 km).

Fortunately, we are in the 21st century.

Chapter II

Introduction

Now you are ready to build your first web applications, like pros. If you didn't mind, the web is a vast and rich world, allowing you to release quickly data and content to everyone around the world.

Now you know the basics, here comes the time to leave those old fashioned to-do lists and eBusiness websites, and fly away toward bigger projects.

Also, here comes the time for you to discover new notions and the beauty of :

- Responsive design
- DOM Manipulation
- SQL Debugging
- Cross Site Request Forgery
- Cross Origin Resource Sharing
- ...

Chapter III

Objectives

This web project is challenging you to create a small web application allowing you to make basic photo and video editing using your webcam and some predefined images.

`Obviously, these images should have an alpha channel, otherwise your superposition would not have the expected effect !`

We will, for instance, picture the precise moment of an intergalactical cat launch, here's the evidence:

App's users should be able to select an image in a list of superposable images (for instance a picture frame, or other "we don't wanna know what you are using this for" objects), take a picture with his/her webcam and admire the result that should be mixing both pictures.

All captured images should be public, likeables and commentable.

Chapter IV

General instructions

- This project will be corrected by humans only. You're allowed to organise and name your files as you see fit, but you must follow the following rules.
- Your web application must produce no errors, no warning or log line in any console, server side and client side. Nonetheless, due to the lack of HTTPS, any error related to `getUserMedia()` are tolerated.
- You are free to use any language to create your server-side application, but, for every function that you use you must check that an equivalent exist in PHP standard library.
- Client-side, your pages must use HTML, CSS and JavaScript.
- Up to date containerization is a must.
- Remember that some choice can make you more attractive on the job market.
- Every framework, micro-framework or library that you don't create and without an equivalent in PHP standard library are totally forbidden, except for CSS frameworks that doesn't need `forbidden` JavaScript.
- Your application must be free of any security leak. You must handle at least cases mentioned in the mandatory part. Nonetheless, you are encouraged to go deeper into your application's safety, think about your data's privacy !
- You are free to use any webserver you want, like `Apache`, `Nginx` or even the `built-in webserver`[1].
- Your web application should be at least be compatible with `Firefox` (>= 41) and `Chrome` (>= 46).

`For obvious security reasons, any credentials, API keys, env variables etc... must be saved locally in a .env file and ignored by git. Publicly stored credentials will lead you directly to a failure of the project.`

[1] http://php.net/manual/en/features.commandline.webserver.php

Chapter V

Mandatory Part

V.1 Common features

You will develop a web application. Even if this is not required, try to structure your application (as a MVC application, for instance).

Your website should have a decent page layout (meaning at least a header, a main section and a footer), be able to display correctly on mobile devices and have an adapted layout on small resolutions.

All your forms should have correct validations and the whole site should be secured. This point is MANDATORY and shall be verified when your application would be evaluated. To have an idea, here are some stuff that is NOT considered as SECURE:

- Store plain or unencrypted passwords in the database.
- Offer the ability to inject HTML or "user" JavaScript in badly protected variables.
- Offer the ability to upload unwanted content on the server.
- Offer the possibility of altering an SQL query.
- Use an extern form to manipulate so-called private data

V.2 User features

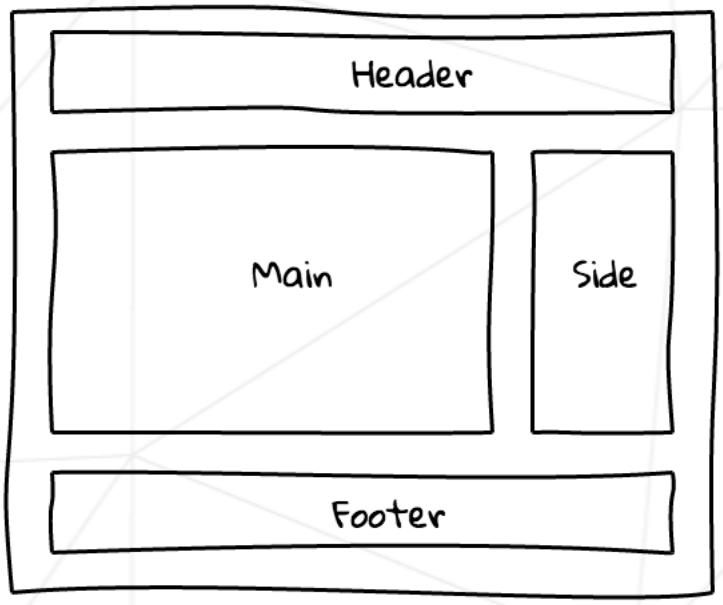
- The application should allow a user to sign up by asking at least a valid email address, an username and a password with at least a minimum level of complexity.
- At the end of the registration process, an user should confirm his account via a unique link sent at the email address filled in the registration form.
- The user should then be able to connect to your application, using his username and his password. He also should be able to tell the application to send a password reinitialisation mail, if he forget his password.
- The user should be able to disconnect in one click at any time on any page.
- Once connected, an user should modify his username, mail address or password.

V.3 Gallery features

- This part is to be public and must display all the images edited by all the users, ordered by date of creation. It should also allow (only) a connected user to like them and/or comment them.
- When an image receives a new comment, the author of the image should be notified by email. This preference must be set as true by default but can be deactivated in user's preferences.
- The list of images must be paginated, with at least 5 elements per page.

V.4 Editing features

Figure V.1: Just an idea of layout for the editing page



This part should be accessible only to users who are authenticated/connected and should politely reject all other users who attempt to access it without being successfully logged in.

This page should contain 2 sections:

- A main section containing the preview of the user's webcam, the list of superposable images and a button allowing to capture a picture.
- A side section displaying thumbnails of all previous pictures taken.

Your page layout should normally look like in Figure V.1.

- Superposable images must be selectable and the button allowing to take the picture should be inactive (not clickable) as long as no superposable image has been selected.
- The creation of the final image (so among others the superposing of the two images) must be done on the server side.
- Because not everyone has a webcam, you should allow the upload of a user image instead of capturing one with the webcam.
- The user should be able to delete his edited images, but only his, not other users' creations.

V.5 Constraints and Mandatory things

To sum up things, your Über application should respect the following technologic choices:

- Authorized languages:
  - [**Server**] Any (limited to PHP standard library)
  - [**Client**] HTML - CSS - JavaScript (only with browser natives API)
- Authorized frameworks:
  - [**Server**] Any (up to PHP standard library)
  - [**Client**] CSS Frameworks tolerated, unless it adds forbidden JavaScript.

Your project must imperatively contain:

- One (or more) container to deploy your site with one command. anything equivalent to docker-compose is ok.

Chapter VI

Bonus part

If the required part is done entirely and perfectly, you can add any bonus you wish.

The evaluation sheet offers bonus points with a granularity equivalent to five medium-sized bonus features.
A single substantial bonus may justify several points, while multiple minor bonus features may justify fewer points.

To be considered as a valid bonus, a feature must be:

- Useful and relevant to the project.
- Fully functional.
- Technically sound.
- Coherent with the overall application.

Incomplete, broken or superficial features (with no functional or technical impact) will not be considered as bonus.
If you lack inspiration, here are some leads:

- "AJAXify" exchanges with the server.
- Propose a live preview of the edited result, directly on the webcam preview. We should note that this is much easier than it looks.
- Do an infinite pagination of the gallery part of the site.
- Offer the possibility to a user to share his images on social networks.
- Render an animated GIF.

The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning.

Chapter VII

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double-check the names of your folders and files to ensure they are correct.