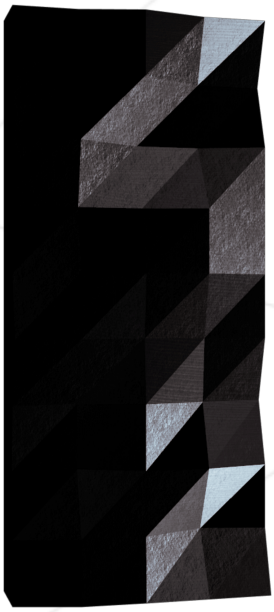


ft_pastebin

Ruche de Oueb Foulestacque

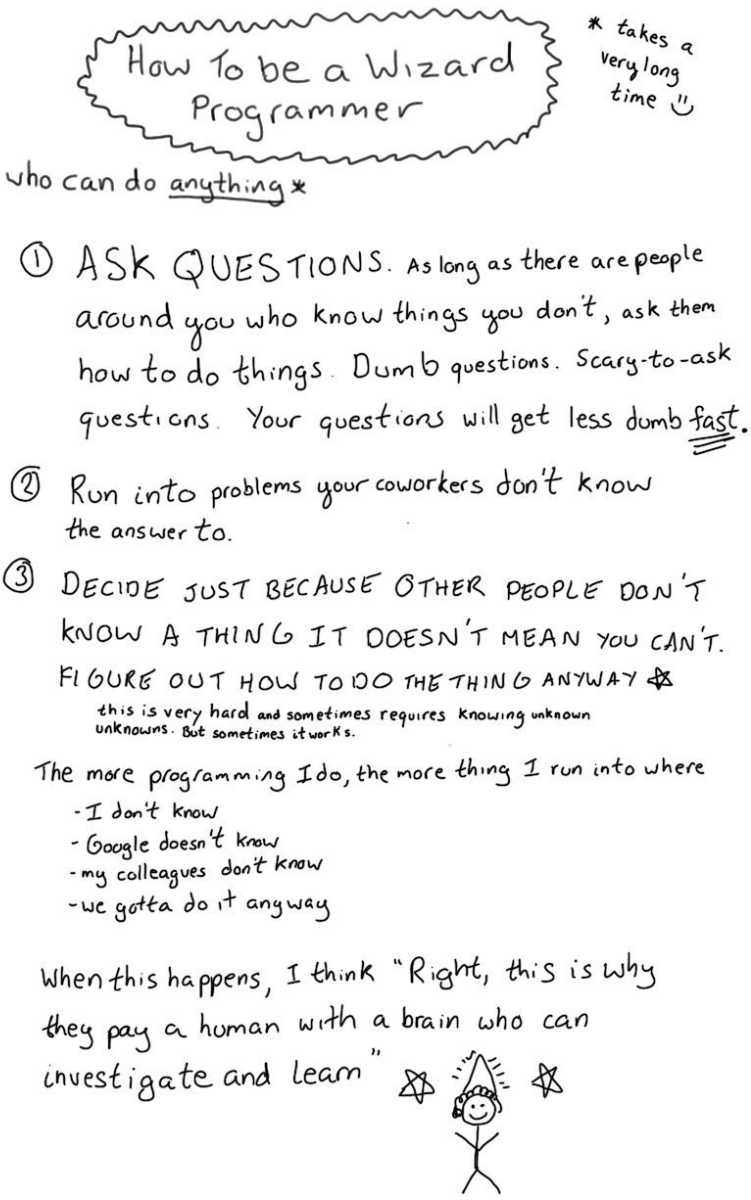
Staff 42 pedago@staff.42.fr
Coton coton@42.fr

Résumé: Ah bon ? le web c'est pas que du HTML et du CSS ? ?

Table des matières

Chapitre I

Préambule



Credit : @b0rk

Chapitre II

Introduction

En voila un rush qui a l'air cool, me diriez vous, mes amis. Du web... DU &%@#$ DE WEB !

Mais comme on est à 42, on ne fait jamais les choses à moitié. Donc pour le coup, on va vous mettre à l'épreuve sur un rush FullStack... comprendre que vous allez devoir concevoir une WebApp de la page index.html au script de mise en production. Et pour le coup, on vous a réservé de quoi vous occuper pendant le weekend !

Pour se mettre dans l'ambiance, voici ce que Wikipedia dit a propos d'un pastebin :

```
Un pastebin, connu également sous le nom de nopaste, est une application
web qui permet aux utilisateurs de mettre en ligne des morceaux de
textes, habituellement des extraits de code source, pour un affichage
public. Ils sont très populaires sur les canaux IRC où copier une
quantité importante de texte, y compris réparti en différentes lignes
successives, est considéré comme non conforme à la netiquette. Un
grand nombre de pastebins existent sur le web, remplissant différents
besoins et fournissant des fonctionnalités conçues pour les populations
auxquelles ils s'adressent.
```

Voila ce qu'il vous attend maintenant.

Chapitre III

Objectif

Comme vu plus haut, votre groupe devra créer une WebApp reprenant le fonctionnement d'un PasteBin.

Votre WebApp devra donc proposer un formulaire qui acceuillera du texte ou du code, qui sera envoyé sur une BDD. Ce texte sera disponible via une URL courte, pendant un laps de temps donné ou un nombre donné d'ouverture de cet URL (ou les deux).

Dans notre sacro-sainte bonté, nous vous laissons libre court à votre imagination pour concevoir cette app, c'est a dire que :

- Ce rush est en langage libre
- Vous pouvez utiliser n'importe quelle base de données, mais seulement en local (pas de BDD en ligne comme Firebase par exemple...)
- Vous pouvez utiliser les frameworks et les librairies que vous souhaitez. Aucune limitation (sauf pour les librairies recodant entierement un nopaste... Faut pas abuser non plus)
- La mise en place de la solution ainsi que sa mise en production est tout aussi libre.

De toute façon, vous aurez de quoi vous occuper tout au long de ce rush, car, nous allons vous demander d'intégrer beaucoup d'élements en très peu de temps, tout en gardant un ensemble cohérent.

Etant donné que vous serez en binôme pour ces prochaines 48h, nous vous conseillons de bien vous répartir les tâches entre **Front-end** et **Back-end** et de bien définir les technologies que vous souhaitez utiliser.

En cas de doute, vous pourrez toujours prendre PasteBin comme référence.

Chapitre IV

Partie obligatoire

IV.1 Tronc commun

`ft_pastebin` doit répondre aux contraintes suivantes :

- ft_pastebin sera constitué de deux sites séparés, un client et un admin, accessibles sur deux ports différents.
- La partie cliente servira à enregistrer du texte ou du code (qu'on appellera des "pastes") dans la base de données de la WebApp avec une contrainte de temps et/ou d'accès. Une fois enregistré, ce "paste" devra être accessible via une URL "courte" unique (ex : www.ft-pastebin.fr/ETgd475x).
- La partie admin servira à consulter l'état de la base de données à tout moment. Cette partie est protégée par mot de passe, et restreint à au moins un compte admin. Si vous comptez intégrer la gestion de compte dans votre rush [SPOILER ALERT], la gestion des users devra se faire aussi par la partie admin.
- La partie cliente sera dissociée du reste de l'application. Si vous comptez utiliser un framework MVC complet pour faire votre Back-end (type Rails ou Django), nous vous interdisons l'utilisation du système de templating pour rendre la partie cliente. Elle devra être servie par une entité à part.
- Seul le Back-end est autorisé à manipuler la Base de Données. Vous devrez faire en sorte de servir des données et les modifier uniquement en requêtant le Back-end.
- Vous allez monter vous même votre environnement de développement. Il n'y a pas de contraintes sur les logiciels et les librairies installées. Toutefois, assurez vous que cet environnement soit portable et installable rapidement lors d'une correction.
- Il ne devra pas y avoir de logs, d'erreurs ou de warnings dans la console de debug du navigateur, une fois votre application en production et/ou évalués. En revanche, du log sur le Back-end est conseillé.
- Un réel effort sera demandé sur la présentation et le design de la partie cliente `AU MINIMUM`. Comprenez bien que si vous avez le choix de la librairie CSS et/ou la charte graphique, autant avoir un minimum de goût.

IV.2 Front End

Partie Cliente

Objectif : Créer au minimum 2 vues, une pour la sauvegarde d'un "paste" et une autre pour l'affichage de ce "paste".

Voici vos directives pour les 2 vues demandées :

Vue Sauvegarde - (Route : `'/'`)

- Vous devez proposer un formulaire comportant :
  - une aire de composition pour rédiger un "paste". Elle doit pouvoir béneficier de la coloration syntaxique si du code est rédigé dans un langage précis (indiqué dans le formulaire).
  - deux champs : titre et auteur
  - le langage utilisé pour la coloration syntaxique
  - une liste de choix de durées cohérentes
  - une liste de nombre limite d'accès
  - une checkbox privé / publique
- Vous devez aussi proposer une liste des derniers "pastes" publics postés sur l'app.
- Vous devez avoir une gestion d'erreur cohérente et logique... ainsi qu'une vérification du formulaire. Si le formulaire est OK, il est envoyé via une requete POST au Back-end

Vue Affichage - (Route : `'/:id'`)

- Le "paste" doit etre affiché dans une zone spécifique. Si il s'agit de code, cette zone doit indiquer les numeros de lignes. Cette zone ne doit pas être modifiable.
- L'auteur et le titre du paste doivent être affichés, ainsi que la date ou le temps ecoulé depuis la création du "paste"
- Des options sont disponibles pour :
  - télécharger le "paste"
  - afficher uniquement le "paste" dans un format brut (sans html)
  - récupérer le "paste" dans le presse papier
- Les informations sont récupérés via une requete GET au Back-end
- Un ENORME MESSAGE d'indisponibilité si le délai est dépassé ou si le nombre d'accès est dépassé.

Ces deux vues doivent être responsive.

Partie Admin

Objectif : Créer un ensemble de vues pouvant administrer ft_pastebin de manière simple, efficace et sécurisé.

La partie Admin doit pouvoir :

- Consulter la base de données sans faire de requetes type SQL ou autre.
- Modifier des entrées de la base.
- Supprimer ou rendre indisponible a la volée un paste
- Gérer les comptes admins (ou user si bonus)

Pas de règles de design en particulier, c'est une partie Admin apres tout. Il faut que ca soit fonctionnel !

... Ah non, faut que ca soit responsive un minimum pour le coup.

IV.3 Back End

Objectif : Garantir une continuité du service et une couche de sécurité pour votre application.

Pour le Back-end, vous allez devoir gérer pleins de choses.

Vous êtes architecte de votre solution, de ce fait, vous devez monter vous-même la(les) table(s) dans la base de données pour loguer toutes les informations nécessaire au bon fonctionnement de l'application.

Vous devrez rendre disponible 2 routes spécifiques :

- Une route qui ne prendra que les requêtes GET et qui enverra les toutes les informations d'un "paste" selon l'ID donné en paramètre. Si cette route accède à un paste ayant une limitation sur le nombre d'accès, vous devrez faire en sorte que cela soit comptabilisé. Vous devrez invalider le "paste" si le nombre d'accès est supérieur ou égal à la limite fixée.
- Une route qui ne prendra que les requêtes POST et qui receptionnera le formulaire de la vue Sauvegarde et loguer le "paste" dans la base de données.

Ces deux routes devront renvoyer le bon code d'erreur HTTP si une méthode non autorisée est utilisée pour accéder à l'une des routes.

Vous êtes aussi garant de la sécurité de votre application. Vous devrez donc faire en sorte que seul le Front-end puisse POSTer des "pastes". Un simple `curl` ne doit pas pouvoir créer de "paste" grâce au Back-end.

En plus de cela, vous devrez mettre en place une solution qui effectuera un check **TOUTES LES MINUTES** de la base de données et qui invalidera un "paste" dont le délai est dépassé. L'invalidation du "paste" ne devra pas être causé par un accès à ce "paste" mais bel et bien par cette solution.

Invalider un "paste" revient au moins à supprimer le contenu du paste. Libre à vous de compléter cette règle si cela vous semble cohérent.

IV.4 Mise en production

Une fois votre application prête, vous devrez mettre en production votre code dans une VM et configurer un serveur Web de votre choix (et vous allez l'avoir entre nginx, apache, traefik...) afin que l'application soit accessible sur 2 ports differents :

- port 80 : partie cliente
- port 4242 : partie admin

En revanche, nous vous imposons ces directives :

- Vous utiliserez `Vagrant` pour générer la VM.
- La VM doit embarquer une distribution Debian (Jessie Stable au minimum).
- Vous devrez rendre le VagrantFile et les scripts de mise en production nécessaire à la configuration de cette VM.
- L'ensemble du code produit lors de ce rush doit être mis à disposition dans la VM dans des dossiers cohérents (mettre votre WebApp dans le /bin de votre VM conduira à un -42 et une insertion forcée de bouts de bambous sous les ongles).

Chapitre V

Partie Bonus

Ici vous pouvez vous lâcher pour les bonus. Mettez tout ce qui vous parait utile pour votre app. Par exemple un back office plus avancé, une gestion des users avec inscription, historique des derniers pastes vus, derniers pastes écrits, etc...

Chapitre VI

Rendu et peer-évaluation

Rendez votre travail sur votre dépot `GiT` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance.
Dans un souci de taille, nous vous demandons de ne rendre QUE vos sources ainsi que vos scripts de mise en productions