Corona Project

Arlindo Tavares, Célestin Piccin, Guillaume Duvoisin, Theo Yoshiura, Besjan Sejrani

arlindo.tavares@cpnv.ch, célestin.piccin@cpnv.ch, guillaume.duvoisin@cpnv.ch,
theo.yoshiura@cpnv.ch, besjan.sejrani@cpnv.ch

Table des Matières

Références

CPNV	
Prénom et nom	Xavier Carell
Email	xavier.carrel@cpnv.ch

Maîtres de classe	
Prénom et nom	Francis Varela
Email	francis.varela@cpnv.ch
Prénom et nom	Loïc Viret
Email	loic.viret@cpnv.ch

Elèves	
Prénom et nom	Arlindo Tavares
Email	arlindo.tavares@cpnv.ch
Prénom et nom	Célestin Piccin
Email	célestin.piccin@cpnv.ch
Prénom et nom	Guillaume Duvoisin
Email	guillaume.duvoisin@cpnv.ch
Prénom et nom	Theo Yoshiura
Email	theo.yoshiura@cpnv.ch
Prénom et nom	Besjan Sejrani
Email	besjan.sejrani@cpnv.ch

Analyse préliminaire

Introduction

Ce chapitre décrit brièvement le projet, le cadre dans lequel il est réalisé, les raisons de ce choix et ce qu'il peut apporter à l'élève ou à l'école. Il n'est pas nécessaire de rentrer dans les détails (ceux-ci seront abordés plus loin) mais cela doit être aussi clair et complet que possible (idées de solutions). Ce chapitre contient également l'inventaire et la description des travaux qui auraient déjà été effectués pour ce projet.

Ces éléments peuvent être repris des spécifications de départ.

Objectifs

Ce chapitre énumère les objectifs du projet. L'atteinte ou non de ceux-ci devra pouvoir être contrôlée à la fin du projet. Les objectifs pourront éventuellement être revus après l'analyse.

Ces éléments peuvent être repris des spécifications de départ.

Planification initiale

Ce chapitre montre la planification du projet. Celui-ci peut être découpé en tâches qui seront planifiées. Il s'agit de la première planification du projet, celle-ci devra être revue après l'analyse. Cette planification sera présentée sous la forme d'un diagramme.

Ces éléments peuvent être repris des spécifications de départ.

Analyse / Conception

Ressources	
Mandat	https://github.com/CPNV-INFO/Yearbook
Github	https://github.com/Corona-cpnv/Yearbook
Maquette interactive	xd.adobe.com/view/2b64c828-aa36-4649-41c1-226f5ae940ba-5d55/
MCD	
MLD	
Uses cases	
Objectifs	https://github.com/Corona-cpnv/Yearbook/projects

Différents environnements

Avant d'établir des stratégies de test, il serait probablement plus favorable de séparer les différentes phases pour la création d'un site web. AInsi, on aurait un environnement de développement en local, un environnement de test et un environnement de production.

Séparer la création en trois phases permet d'éviter idéalement des bugs en production, ainsi que de repérer des erreurs dans l'environnement de test.

Idéalement, on aurait deux base de données, une pour le développement et les tests ainsi qu'une en production. On pourrait alors effectuer autant de tests que l'on voudrait sans se soucier de l'implication de données réels.

Github | Gitlab

Peu importe quelle service web d'hébergement et de gestion de développement de logiciels nous utilisons, que cela soit GIthub ou GItlab, les données sensibles que l'on partage et sont mises en public ou en privées doivent être protégées pour la sécurité des utilisateurs de notre projet.

Mettre en clair, les identifiants de notre base de donnée avec le nom d'utilisateur et le mot de passe de celle-ci puis le partager, est un risque de sécurité. Peu importe que le projet soit en public ou en privé.

Pour résoudre ce souci, nous utilisons des variables d'environnements dans notre projet.
celles-ci injectent la valeur de nos données sensibles lors de l'exécution du programme. Une fois le programme terminé, les valeurs sont supprimées.

Afin d'éviter que les fichiers de variables d'environnement soient par mégarde envoyés sur Github ou Gitlab, il est généralement conseillé après avoir créé et assigné des valeurs à ces fichiers, de créer un nouveau fichier appelé **.gitignore,** spécifiant les ressources ne devant pas être envoyé par GIt sur Github ou Gitlab.

Afin de faciliter l'intégration des variables d'environnement dans notre projet, la librairie PHP phpdotenv[1] permet d'y arriver facilement. Celle-ci s'installe via Composer, c'est un package manager pour PHP, il permet de télécharger et d'intégrer automatiquement des projets PHP au nôtre.

[1] https://github.com/vlucas/phpdotenv

Environnement de développement

Le développement dans un environnement de développement s'effectue souvent en local, c'est-à-dire sur un ordinateur ayant un système d'exploitation Windows, Mac Os ou Linux installé (en fonction de ce que l'on développe, parfois il y a des restrictions sur les OS) ainsi que suffisamment de mémoire (RAM) et un navigateur récent.

Dépendamment de ce que l'on développe, un IDE (Integrated Development Environment) est recommandé.

Afin de pouvoir développer des projets PHP en local, il faut avoir préalablement installé et lancé Apache web server ainsi que MySQL ou MariaDB (clone open source de MySQL), de plus, il faut avoir également libéré ou modifié les ports nécessaire de la machine pour ces programmes.

WAMP et XAMPP sont des bundle de téléchargement très populaire permettant de télécharger tous les pré-requis en un seul téléchargement, le tout, généralement, déjà configuré.

Environnement de test

Un environnement de test est simplement un environnement de développement reporté sur une autre machine, puis testé par un testeur professionnel ou un collaborateur. Celui-ci ayant pour tâche de trouver des erreurs et les signaler au développeurs. L'application n'est alors autorisée à partir en production que lorsques toutes les erreurs soient corrigées.

Bien que Corona Projet est un projet d'études, on peut également introduire un environnement de test sans devoir acheter ou installer une nouvelle machine, sans rien dépenser, cela permet de garantir un minimum, la qualité du projet.

Heroku

Heroku est un PaaS (Plateforme as a service) proposé par SalesForce, permettant d'héberger gratuitement un serveur avec base de donnée en ligne, jusqu'à une certaine limite, une fois celle-ci dépassée, cela devient payant.

Pour des projets test, nécessitant une base de données et un serveur, c'est un parfait sandbox.

Heroku permet d'utiliser Java, Golang, Nodejs, PHP, Python, Ruby, Scala et Closure comme langage côté serveur.

Il supporte comme base de données, MongoDB, Redis et Postgres.

L'intégration avec Github / GitLab est très simple.

Bien que notre projet a pour base de donnée du MySQL et que sur Heroku ils ne supportent que Postgres, cela n'est pas un souci, puisque les deux c'est du SQL, la différence entre les deux est très petite. D'ailleurs, une fois que l'on apprend du SQL, on peut facilement les remplacer les unes avec les autres, sans devoir réécrire tout notre projet.

Environnement de production

L'environnement de production n'est autre que l'interaction réel de nos utilisateurs avec notre projet. Celui-ci doit être sécurisé et ne doit faire confiance à aucun des utilisateurs.

La communication avec le serveur de chez SwissCenter doit se faire uniquement via **SSH**[2] ou **SFTP**[3] pour le transfert de fichiers. La clé SSH générée initialement doit être placé en lieu sûr, de plus, l'ajout de collaborateurs doit se fait via la **création de compte sur le serveur, l'ajout de droits via un groupe ainsi que par la création de clés publiques et privées.** [4] Tous les ports non utilisés sur le serveur, doivent être fermées.

La communication entre le client et le serveur doit être assuré via un certificat TLS, permettant d'encrypter les informations envoyés et reçu par le client. Celle-ci peut être obtenu via **Let's Encrypt**[5].

Lorsqu'un utilisateur s'inscrit sur notre site, il utilise probablement le même mot de passe pour se connecter à sa banque, Paypal, Facebook ou Amazon, après tout, c'est plus facile à retenir. Notre responsabilité en tant que développeur et de ne jamais stocker tel quel les mots de passe sans les avoir au préalablement encrypté, jamais ! Pour y arriver, nous

[2] https://fr.wikipedia.org/wiki/Secure_Shell
[3] https://fr.wikipedia.org/wiki/SFTP
[4] https://linuxtechlab.com/restrict-ssh-access-on-linux/
[5] Let's Encrypt est une autorité de certification, elle fournit des certificats gratuits X.509 pour le protocole cryptographique TLS au moyen d'un processus automatisé.

pouvons utiliser des fonctions de hachage ayant fait leur preuves, telles que **Bcrypt**[6] ou **Scrypt**[7].

La base de données doit également être sécurisé, l'utilisateur par défaut de celle-ci doit être remplacé par un nouvel utilisateur possédant un mot de passe sécurisé, huit caractères étant au minimum conseillé.

Côté client, les champs de saisie doivent être aseptisé afin que des utilisateurs malveillants ne puissent pas effectuer des attaques **XSS**[8] ou **Cross-site Request forgery**[9] à l'aide de javascript.

Côté serveur, les requêtes reçues doivent être vérifiées par le serveur, si un mot de passe est demandé, l'utilisateur ne doit pas pouvoir envoyer autre qu'une chaîne de caractères alpha numérique et le serveur n'est pas sensé accepté une chaîne de caractères qui n'est pas conforme.

La manipulation de données au sein de l'application doit être implémenté d'une façon particulière lorsque l'on à affaire à une base de données SQL. Si l'on ne prépare pas les requêtes SQL à l'avance, il y a un risque d'**injection SQL**[10], c'est un procédé simple permettant à un utilisateur non connecté ou non autorisé d'obtenir des informations précises de notre base de données.

[6] https://fr.wikipedia.org/wiki/Bcrypt
[7] https://fr.wikipedia.org/wiki/Scrypt
[8] https://de.wikipedia.org/wiki/Cross-Site-Scripting
[9] https://en.wikipedia.org/wiki/Cross-site_request_forgery
[10] https://en.wikipedia.org/wiki/SQL_injection

Stratégie de test

Durant le développement, en environnement de développement, jusqu'à un certain degré, des tests manuelles peuvent être effectués. Une fois que le projet prend de l'ampleur, il serait judicieux pour le long terme de commencer à implémenter des tests unitaires afin d'éviter des régressions.

Tests unitaires

Les tests unitaires ont pour but de tester spécifiquement qu'une fonctionnalité à la fois. C'est pour cela qu'il est important lorsque l'on crée des fonctions, de les limiter à une seule fonctionnalité, afin de pouvoir mieux les tester. Ce concept est également appelé **Single responsibility**, il est généralement accompagné d'une autre convention appelé **Separation of concerns**, où le principe est de découper son code en différentes parties afin de le rendre modulaire pour que l'on puisse le réutiliser.

Tests d'intégration

Une fois que les tests unitaires sont en place, implémenter des tests d'intégration afin de tester l'intégration de différents éléments ensemble paraît nécessaire. Cela peut notamment s'apparenter à de la réception de données ou l'envoie de données via des requêtes HTTP.

Il est important à noter que pour les tests d'intégrations, nous falsifions les requêtes HTTP car celles-ci peuvent prendre considérablement de temps dans l'ensemble, surtout si l'on a 300 tests. Cela s'appelle également du **data mocking.**

Tests automatiques

Après avoir parlé des différents environnements pour la création de site web, il serait tout de même génial si l'on pouvait, via un mécanisme, envoyer notre code sur Github ou Gitlab, puis de là, qu'un serveur prenne notre code, le test et s'il passe, il est envoyé ensuite automatiquement sur notre serveur en production ou notre serveur test. Et s'il ne passe pas, que l'on reçoive une notification nous expliquons ce qui a échoué. Ce principe est le **CI / CD**, et nous pouvons l'implémenter.

CI / CD

Continuous integration et continuous deployment, sont les standards du bon développement d'une application.

CI permet de récupérer le code, ensuite d'effectuer les tests s'il y en a, si tout se passe bien, le CD push l'application sur un serveur test ou sur un serveur de production que l'on aura au préalable défini.

Si le CI ne passe pas, l'utilisateur ayant commis le push reçoit une notification d'erreur.

A noter qu'aujourd'hui la relation entre CI / CD est devenue flou, la plupart des outils sur le marché permettent d'effectuer environ les mêmes fonctionnalités.

Outils gratuit recommandés : **Travis CI, Circle CI**

TDD

Le Test Driven Development, aussi appelé TDD, est une approche différente lorsque l'on développe. L'objectif est d'écrire le test avant d'écrire notre code, cela favorise une approche plus réfléchie et permet dans certains cas de gagner du temps.

Nous n'allons pas l'implémenter, cela demande une certaine expérience.

Can I use

Can i use[11] est un site web listant l'adoption des différents navigateurs par rapport aux nouvelles fonctionnalités de disponibles, cela permet notamment de savoir si Internet Explorer est compatible avec les nouveautés du CSS tel que flexbox ou grid.

Librairies de test

Pour la réalisation des tests mentionnés ci-dessus, nous utiliserons deux librairies, une en PHP et une en javascript, **PHPUnit[12]** et **Jest[13].**

[11] https://caniuse.com/
[12] https://phpunit.readthedocs.io/en/9.1/
[13] https://jestjs.io/

Couverture des tests

Une couverture de tests à 100% n'est forcément synonyme de tests efficients. Cela montre simplement qu'on a effectué une grande quantité de tests sur un sujet précis.

Effectuer des tests oui, mais se fixer comme objectif 100% c'est une perte de temps.

Test externe

Une fois que l'environnement de test sur Heroku est mis en place, ainsi que le CI / CD avec Travis CI, nous avons la possibilité de faire tester notre application à des personnes externes, en avant première, via le lien de Heroku sur lequel se trouve notre environnement de test. Nous pourrons alors analyser les premières impressions.

Risques techniques

Compte tenu de la situation exceptionnelle du Coronavirus en cette période de mois de mai 2020, l'apprentissage se fait depuis la maison via l'ordinateur, pour certains élèves c'est plus difficile d'apprendre qu'en présentielle en cours.

De plus, par souci d'enseignement, tous les modules web ont été réunis ensemble afin de pouvoir être enseigné à distance puisque les projets système ne pouvaient pas avoir lieu dû à la complexité de la pandémie.

Ainsi, les élèves apprennent au même moment PHP, l'intégration d'une base de données à une application web et le développement d'une application web.

De toute évidence, durant ce projet il y aura un manque de compétences ainsi que de nombreuses problématiques liées à ce dernier qu'il faudra résoudre.

Solutions

A tout problème, il y une solution. Nous sommes 5 élèves par groupe, ainsi le travail peut être partagé de manière homogène.

De plus, il y a une grande quantité de ressources disponibles sur Internet, documentations officiels, documentation fourni, tutoriels, cours, articles de blog, podcasts, aide d'un professeur. Le plus difficile est simplement de savoir chercher, ce qui n'est pas évident.

Priorités

Parfois pour résoudre un problème, il n'y pas besoin de réinventer une solution.

Tout au long de cette première année d'apprentissage ainsi que de part les petits projets que l'on a réalisé, nous avons déjà les connaissances nécessaires à la réalisation d'un tel projet.

- Programmation procédurale
- Exécution de mandats
- Encryptage et système de codification
- Réaliser et publier un site web
- Graphisme web
- Application C
- Implémentation d'un modèle de données
- Réaliser un petit projet informatique
- Développer des interfaces graphiques
- Appui en programmation

Par simplicité, nous utiliserons ce que l'on a déjà appris.

Technologies utilisées

Pour la réalisation de notre projet, Corona Project, nous utiliserons ce que nous avons appris en classe.

L'architecture sera de type MVC (Model View Controller), elle est généralement la plus utilisée pour la création de site web.

Le modèle se traduit généralement par le modèle logique de données couplé à un ORM[14] (nous n'utilisons pas de ORM pour le projet) pour faciliter les interactions avec la base de données.

Le view est simplement le client, le frontend[15] qui interagit avec le serveur.

Le controller est une action déclenchant une fonctionnalité précise côté serveur.

Le langage côté serveur sera du PHP, il permettra d'envoyer des fichiers HTML, CSS et Javascript au client afin que l'utilisateur puisse interagir avec. Dans ce contexte, PHP est utilisé sous forme de template[16].

Pour la structure, nous utiliserons du HTML.

Pour le design de la page, en production, nous utiliserons du CSS alors qu'en développement, nous utiliserons Sass.

Sass est un préprocesseur CSS permettant d'ajouter des fonctionnalités productives en phase de développement, une fois compilé, Sass devient du CSS normal.

[14] ORM (Object relational mapping) est généralement une librairie permettant de faciliter la communication avec les bases de données SQL, ils nous évitent de devoir écrire de larges requêtes en SQL.
[15] Navigateur, client.
[16] PHP peut être utilisé de deux manière différentes, comme template ou comme API REST, la seul différence réside dans ce qui est envoyé au client. En template, des fichiers sont envoyés alors qu'en API REST se sont des données qui sont envoyées, celle-ci permet de communiquer avec d'autres serveurs.

Avantages

- Permet d'utiliser des variables pour désigner du CSS
- Permet d'importer des fichiers SCSS entre eux
- Permet de mieux structurer les projets contenant beaucoup de CSS
- Permet d'utiliser des fonctions avec Sass
- Permet de compiler uniquement un seul fichier css, évitant des requêtes HTTP inutiles.

Inconvénients

- Légère courbe d'apprentissage

Sachant que ce projet implique un nombre significatif de pages, utiliser Sass peut être un bon choix pour la réalisation et la maintenance du projet.

Comme framework[17] CSS, nous utiliserons Bootstrap, il a déjà été vu en classe et est facile d'utilisation. Bootstrap permettra de gagner du temps en développement.

Javascript sera également utilisé via les interactions dynamique du client, notamment les barres de recherches, afficher conditionnellement du contenu, création de contenu HTML et CSS, …

Pour Corona Project, javascript sera utilisé sans librairies ou framework, éventuellement à l'exception d'une libraire test tel que Jest.

Jest est une librairie de tests, d'assertion, de data mocking et de data coverage.

[17] Un framework est simplement du code écrit par une personne tierce ou une entreprise qui utilise votre code, elle a déjà une structure de fonctionnement. Elle a pour but de vous aider dans le processus du développement. Dans le cadre de Bootstrap, le framework est maintenu par Twitter et permet d'utiliser des "composants" déjà créé que l'on peut facilement intégrer à nos projets.

Pour la base de données, nous utiliserons MySQL sans ORM, avec comme seules connaissances, le modèle conceptuelle de données et le modèle logique de données.

Le tout sera hébergé chez SwissCenter, probablement dans un serveur Linux ou l'on devra nous même installer et configurer Apache pour PHP.

Planification

Révision de la planification initiale du projet :

- *planning indiquant les dates de début et de fin du projet ainsi que le découpage connu des diverses phases.*
- *partage des tâches en cas de travail à plusieurs.*

*Il s'agit en principe de la planification **définitive du projet**. Elle peut être ensuite affinée (découpage des tâches). Si les délais doivent être ensuite modifiés, le responsable de projet doit être avisé, et les raisons doivent être expliquées dans l'historique.*

Dossier de conception

Fournir tous les document de conception:

- *le choix du matériel HW*
- *le choix des systèmes d'exploitation pour la réalisation et l'utilisation*
- *le choix des outils logiciels pour la réalisation et l'utilisation*
- *site web: réaliser les maquettes avec un logiciel, décrire toutes les animations sur papier, définir les mots-clés, choisir une formule d'hébergement, définir la méthode de mise à jour, …*
- *bases de données: décrire le modèle relationnel, le contenu détaillé des tables (caractéristiques de chaque champs) et les requêtes.*
- *programmation et scripts: organigramme, architecture du programme, découpage modulaire, entrées-sorties des modules, pseudo-code / structogramme…*

Le dossier de conception devrait permettre de sous-traiter la réalisation du projet !