



Rapport Projet Application Web

EasyPrint

Octobre 2022 – Janvier 2023

Félix Bertagnolio

Florian Lachaux





Sommaire

PARTIE A - Introduction	3
PARTIE B - Architecture mise en place	4
I - Structure et Fonctionnement	4
II - Eléments supplémentaires	5
III - Voies d'amélioration	5
PARTIE C - Architecture Idéale	6
Architecture générale du service web	6
Architecture des composantes du service web	7
Architecture du réseau	7
Améliorations et Sécurité	8
API Octoprint	8
Session Administrateur	9
Sécurité	9
PARTIE D - FRAMEWORKS	10
Spring	10
Bootstrap	11
PARTIE E - conclusion	12
PARTIE F - Annexes	13



INTRODUCTION

Pour de ce projet portant sur la réalisation d'une application web, la solution retenue est de modéliser un site web pour une entreprise factice appelée *EasyPrint*. Cette entreprise de service basée dans le secteur des imprimantes 3D permettrait à ses clients de contrôler des imprimantes à distance. Ainsi, elle permet à n'importe qui possédant un compte client chez eux d'imprimer la pièce de leur choix depuis n'importe quel endroit sur Terre où une connexion à internet est présente. Cela est particulièrement utile pour les personnes ne travaillant pas dans un atelier dans lequel se trouvent les imprimantes. Dans le cadre de *robotech*, cela permettrait de lancer des impressions même lorsque l'école est fermée ou si l'étudiant n'est pas présent physiquement à l'école. A terme, l'utilisateur pourra avoir accès à plusieurs informations sur les imprimantes qu'il utilise, par exemple si celle-ci est libre ou occupée avec une autre impression, les temps restant d'impression, les erreurs éventuelles ayant lieu pendant l'impression etc.

Ce projet peut être décomposé en deux grandes parties, la première correspondant à la création et présentation d'un prototype fonctionnel réalisé durant les 4 mois de projet. La seconde constituant la conception et la présentation théorique de ce que devra être l'application web dans sa forme finale, prête à être mise en ligne.

Ce projet s'inscrit parfaitement dans l'industrie 4.0 dont la principale caractéristique est la virtualisation des commandes ainsi que l'interconnexion des machines. Si l'on reprend la définition exacte : « Les grandes promesses de cette quatrième révolution industrielle sont de séduire les consommateurs avec des produits uniques et personnalisés, et malgré de faibles volumes de fabrication, de maintenir des gains. Ces mêmes consommateurs peuvent ainsi communiquer avec les machines durant les phases de réalisation : ce type de production s'appelle *smart production* ». Ainsi, le projet *EasyPrint* s'inscrit parfaitement dans cette voie en permettant à ses utilisateurs de personnaliser leur commande avec la possibilité d'imprimer n'importe quelle pièce de leur choix en étant n'importe où dans le monde.



ARCHITECTURE MISE EN PLACE

Dans cette partie seront détaillés la structure et le fonctionnement du prototype de service web que nous avons créé ces quatre derniers mois. Ce prototype tourne actuellement en local sur une machine mais est adapté à une configuration pouvant être déployée sur les machines de l'école ou sur un serveur d'entreprise.

I - Structure et Fonctionnement

Dans un premier temps, le prototype a été pensé dans l'idée d'un déploiement au sein des locaux de Polytech. Le but est ainsi de mettre en place un service web permettant aux étudiants du projet robotech d'avoir accès, à distance, aux deux imprimantes situées dans les locaux de robotech.

La partie backend a été développée Java EE avec le Framework *Spring* et la partie frontend a été réalisée à l'aide des langages html, css avec le Framework Bootstrap et javascript. Le principe de fonctionnement de ces deux différents Framework seront explicités plus tard dans ce rapport. Les parties front et back constituent deux entités séparées qui sont liées grâce à des requêtes http. L'utilisation du langage javascript côté front est indispensable notamment dû au fait de l'utilisation de la fonction « fetch » qui permet de faire facilement des requêtes vers le serveur. Le service web est construit selon l'architecture MVC (Modèle, Vue, Contrôleur) qui sera détaillée dans la partie *Architecture Idéale*.

Le site web est composé de 5 pages permettant chacune de rendre un service différent à l'utilisateur.

- **Page d'accueil** : C'est une page de présentation, simple, qui introduit le service auprès des utilisateurs. Depuis cette page il est possible d'avoir accès à la page de connexion et d'inscription.
- **Page d'inscription** : Cette page est constituée d'un formulaire d'inscription pour les nouveaux utilisateurs. Ils peuvent donc renseigner leurs informations d'identification (ici nom et mot de passe) ainsi que choisir, parmi les différentes imprimantes mises à disposition, celles sur lesquelles ils souhaitent travailler.
- **Page de connexion** : Cette page est constituée d'un formulaire de connexion grâce auquel les utilisateurs déjà enregistrés dans la base de données peuvent s'authentifier pour pouvoir utiliser les imprimantes.
- **Page client** : Une fois authentifiés, les utilisateurs arrivent sur la page client qui répertorie les imprimantes qu'ils ont choisis pour travailler ainsi que plusieurs informations concernant les imprimantes (le modèle, l'adresse IP). De plus une zone de dépôt de fichiers a été mise en place qui permettra, plus tard, d'envoyer des fichiers d'impression aux imprimantes.
- **Page admin** : Elle n'est accessible qu'avec un nom et un mot de passe spécial et permet d'ajouter une nouvelle imprimante à la base de données et permettra, à terme, de réaliser toute la partie configuration/modération du service web.

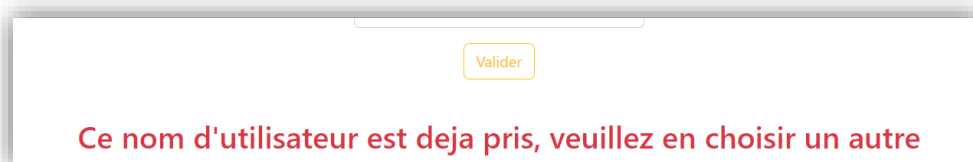


II - Éléments supplémentaires

La version du projet que nous avons actuellement est encore loin d'être un service web stable, sans failles de sécurité que nous pourrions mettre en place dans nos locaux ou chez un client. Cependant nous avons déjà commencer à ajouter quelques éléments permettant de renforcer la sécurité ainsi que d'éviter de gros bugs.

Pour cela, nous avons mis en place

- **Une validation des formulaires** : Pour cette partie nous allons vérifier que les informations rentrées dans les formulaires par les utilisateurs correspondent bien aux normes que nous attendons. Ainsi, il sera par exemple impossible de s'inscrire avec un nom d'utilisateur déjà utilisé. En effet, le nom d'utilisateur servant d'identifiant au sein de la base de données ne peut être le même pour deux personnes différentes car cela pourrait connecter des gens sur une session qui n'est pas la leur.



- **Une session administrateur** : Cette session n'est accessible qu'avec des identifiants et mots de passe définis. Elle permet, pour l'instant, d'accéder à une page de configuration dans laquelle il est possible d'ajouter une nouvelle imprimante à mettre à disposition des utilisateurs.
- **Bouton déconnection** : La page client est munie d'un bouton déconnection qui redirige vers la page d'accueil et qui permet d'effacer les données de session de l'utilisateur, c'est-à-dire ses informations d'authentification, et quelques informations sur les imprimantes qu'il utilise.

III - Voies d'amélioration

Comme énoncé précédemment, ce prototype nécessite encore de nombreuses améliorations pour être mis en place. Dans un premier temps il serait très intéressant d'administrer ce service web sur le matériel de l'école et non en local sur un ordinateur personnel afin de faire des tests et de permettre à d'autres étudiants de continuer le projet. Pour cela nous avons déjà administré la base de données ainsi que le serveur Apache sur une Raspberry PI qui sert à contrôler les imprimantes 3D de robotique. Cependant la création d'un fichier exécutable permettant à la Raspberry de compiler et d'exécuter le code écrit avec spring nous a posé quelques soucis étant donné la difficulté de configurer une telle installation uniquement en ligne de commande (la Raspberry PI fonctionne avec le système d'exploitation Linux – Debian).

Il serait également intéressant d'approfondir les fonctionnalités de la session administrateur, ce qui permettrait alors d'apporter de la modération à l'application web en ayant un réel contrôle sur les actions des utilisateurs offrant ainsi la possibilité de limiter certains utilisateurs ou certaines fonctionnalités que propose notre service web.

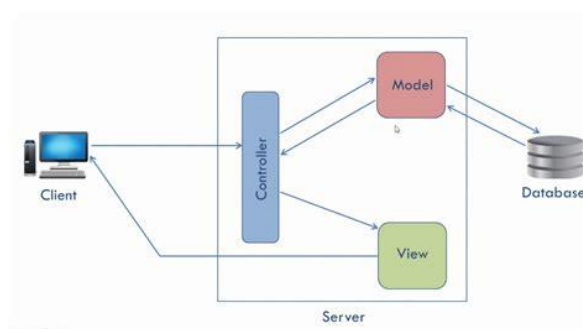
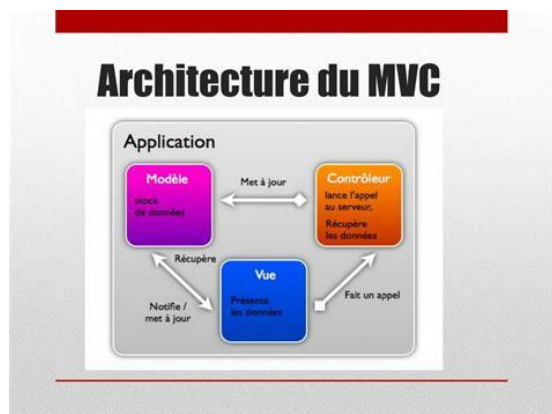


ARCHITECTURE IDEALE

Comme énoncé dans l'introduction, la seconde partie constitue une partie théorique dans laquelle sera détaillée l'architecture à mettre en place pour réaliser une version professionnelle de ce projet qui pourrait être utilisé dans un premier temps **dans le cadre de robotech** et qui serait donc fiable et sécurisé pour les utilisateurs.

Architecture générale du service web

Pour ce projet nous utiliserons une architecture MVC (Model View Controller)



Le modèle MVC est une norme permettant de développer de manière standardisée une application web. Elle se compose de 3 parties

- Le Modèle : Il contient les données à afficher (correspond à une retranscription de la base de données à laquelle il est lié). Dans le cas d'un développement avec spring il correspond aux classes du package *model* (qui sont elles-mêmes des retranscriptions des différentes tables de la base de données)
- La Vue : Elle correspond à l'interface graphique que va voir l'utilisateur. Dans le cas d'une application web elle correspond aux pages html (stylisée avec du css et rendu dynamique grâce au javascript) que l'utilisateur voit dans son navigateur.
- Le Contrôleur : Il sert à faire le lien entre la vue et le modèle. En effet, il va servir à réceptionner les requêtes HTTP envoyées par la vue, traiter les requêtes puis aller chercher des informations au niveau du modèle si nécessaire, et enfin envoyer une réponse à la vue avec les informations demandées. Dans notre cas cela est fait au niveau des classes du package *contrôleur*.

Un diagramme de l'architecture à mettre en place est disponible dans les annexes



Architecture des composantes du service web

Dans un premier temps, nous pouvons noter que nous utiliserons :

- Une base donnée MySQL
- Le serveur http Apache Tomcat
- Un code source écrit en Java EE avec le Framework spring pour la partie backend (côté serveur donc pour le modèle et le contrôleur)
- Un code source écrit en html, css avec le Framework Bootstrap et en javascript pour la partie frontend (côté Vue)
- L'application web est réalisée pour fonctionner sur tous les clients légers standards (google chrome, Firefox, safari etc.)

Pour une utilisation dans le cadre de robotech, les équipements disponibles sont deux imprimantes 3D reliées à une Raspberry PI. Ainsi c'est la Raspberry PI qui va servir à contrôler ces imprimantes et nous devront donc déplacer la base de données ainsi que le serveur web sur cette Raspberry PI. Pour cela une base de données MySQL sera créée directement sur la Raspberry qui fonctionnera en local. Ensuite, il faut mettre en place le serveur web qui fonctionne également sur la Raspberry PI, ce qui permet au serveur de communiquer facilement avec la base de données. Pour cela, le code source écrit en Java EE ainsi que celui servant à créer la vue (donc html, css, javascript) seront importés sur la Raspberry et le serveur HTTP Apache servira à faire la liaison entre le client (navigateur) et le contrôleur (serveur).

La base de données sera au minimum composée de deux tables :

- La table client : qui comporte les informations relatives aux clients, donc leur identifiant, mot de passe et l'identifiant des imprimantes sur lesquelles il a choisi de travailler
- La table imprimantes : qui comporte des informations sur les imprimantes que EasyPrint met à disposition comme leur identifiant, adresse IP, numéro de série, informations d'authentification etc.

Architecture du réseau

Comme vu dans la section précédente, la base de données ainsi que le serveur web sont tous deux implémentés sur la Raspberry Pi. Ainsi, cette Raspberry nous sert ici de routeur. Dans cette configuration cela permettrait à tous les étudiants de Polytech Nancy de pouvoir communiquer avec notre serveur web en « contactant » simplement la Raspberry Pi. Cela permet de ne pas avoir à installer un routeur externe qui engendrerait des coûts supplémentaires et compliquerait la mise en place du système.




Améliorations et Sécurité

API Octoprint

Le grand atout de ce projet est que les imprimantes qui sont mises à disposition dans les locaux de robotech sont contrôlées par une API publique : *Octoprint*. Sur la documentation en ligne de cette API sont renseignées toutes les différentes commandes et informations qu'il est possible d'obtenir de ces imprimantes. En effet, L'api Octoprint est une API REST et fonctionne donc comme un serveur web, de la même manière que le fait notre application. Ainsi, les commandes/informations qui sont exploitable au niveau de cette API et qui pourrait nous être utiles sont par exemple :

- Envoyer un fichier Gcode à distance (fichier lu par l'imprimante qui va lui indiquer la forme de la pièce qu'elle doit fabriquer)
- Lancer une impression à distance
- Savoir si une imprimante est libre ou non
- Si une impression est lancée, récupérer le pourcentage d'avancement de celle-ci
- Température de l'imprimante
- Si une erreur est apparue pendant l'impression etc.

La communication entre la Vue de notre application et le serveur de L'API Octoprint se fait sur le même système que pour la communication avec notre serveur, c'est-à-dire via des requêtes http.



Upload file or create folder

POST /api/files/(string: location)

```
POST /api/files/sdcard HTTP/1.1
Host: example.com
X-API-Key: abcdef...
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryDeC2E3iWbTv1PwMC
Content-Length: 430
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Location: http://example.com/api/files/local/20mm-umlaut-box.gcode

{
  "files": {
    "local": {
      "name": "20mm-umlaut-box",
      "origin": "local",
      "refs": {
        "resource": "http://example.com/api/files/local/whistle_v2.gcode",
        "download": "http://example.com/downloads/files/local/whistle_v2.gcode"
      }
    }
  },
  "done": true,
  "effectiveSelect": false,
  "effectivePrint": false
}
```

Query Parameters:

- **exclude** – An optional comma-separated list of fields to exclude from the response (e.g. if not needed by the client). Valid values to supply here are: `temperature`, `sd` and `state`.
- **history** – If set to `true` (or: `yes`, `y`, `1`), history information will be included in the response too. If no `limit` parameter is given, all available temperature history data will be returned.
- **limit** – If set to an integer (`n`), only the last `n` data points from the printer's temperature history will be returned. Will be ignored if `history` is not enabled.

Status Codes:

- **200 OK** – No error
- **409 Conflict** – If the printer is not operational.



Session Administrateur

La configuration/mise en place de ce genre de service web demandant quelques qualifications et un système de modération étant toujours le bienvenu, un autre élément essentiel d'une bonne application web et de mettre en place une session administrateur. Ainsi, certaines personnes de confiance (dans le cadre de robotech, cela pourrait être un professeur) se verront attribué des identifiants et mots de passe spéciaux reconnus par le serveur comme utilisateurs administrateurs. Le rôle de ces administrateurs est d'une part de mettre en place le système en réalisant les configurations initiales nécessaires au lancement de l'application et d'une autre part d'effectuer un travail de modération sur les utilisateurs malveillants à l'égard des imprimantes ou de la sécurité du service web.

Pour résumer, un utilisateur lambda pourra :

- Se connecter à sa session utilisateur personnelle
- Utiliser les imprimantes mises à sa disposition lorsque celles-ci sont libres
- Avoir des informations sur l'état d'impression de ses pièces

Un utilisateur Administrateur pourra lui :

- Faire les mêmes choses qu'un utilisateur lambda
- Configurer les imprimantes déjà disponibles sur le site
- Ajouter de nouvelles imprimantes
- Arrêter une impression à tout moment
- Bannir un utilisateur suspect, qui spam des demandes ou qui ne respecte pas le matériel

Sécurité

La Sécurité est un élément primordial de ce type de projet qui s'inscrit dans l'industrie 4.0. Laisser une tierce personne avoir accès aux serveurs de notre application web reviendrait à lui donner tous les droits donc de faire ce qu'elle souhaite avec le matériel. Une personne mal intentionnée pourrait par exemple endommager gravement le matériel jusqu'à le rendre totalement inutilisable en plus d'avoir accès à toutes les données des utilisateurs enregistrés dans la base de données comme leurs adresses mail, mots de passe etc. Pour cela, il est possible de mettre en place quelques éléments, parfois très simples, mais qui permettent de renforcer très efficacement la sécurité de notre application web.

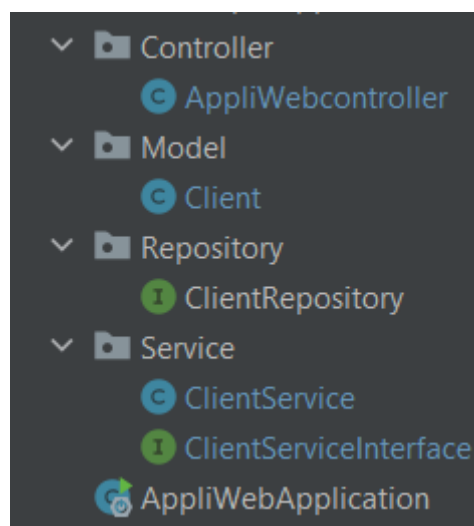
- Validation des formulaires : cela permet de contrôler les données saisies par les utilisateurs dans les formulaires et donc d'empêcher qu'un utilisateur s'inscrive avec les mêmes identifiants qu'un autre, qu'un compte soit créé avec un mot de passe vide etc.
- Validation d'un administrateur : dans le même principe que la validation des formulaires, cela agirait comme une double authentification avec un contrôle d'un administrateur sur chaque nouvelle inscription et sur les actions menées par les utilisateurs sur le site web.
- Crypter les mots de passe : le but est ici de crypter les mots de passe au sein de la base de données pour éviter une trop grosse fuite d'informations en cas d'intrusion dans le serveur par une personne tierce.
- Rester dans un réseau local : rester dans un réseau local permet aux administrateurs d'avoir la main mise sur les utilisateurs du service web et donc de pouvoir remonter jusqu'à la personne en cas de perturbations ou de tentatives d'intrusion sur le serveur



FRAMEWORKS

Spring

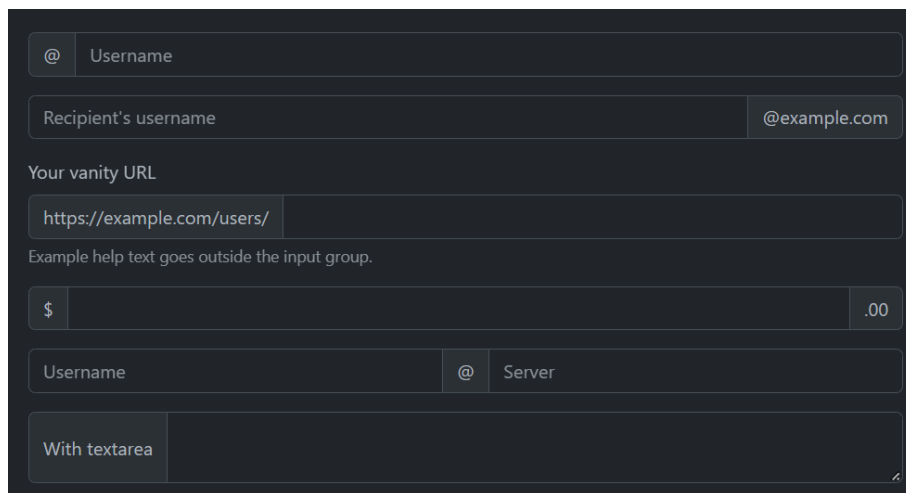
Le Framework que nous avons utilisé côté backend est *Spring*. Spring est un Framework basé sur le langage Java EE, qui est une extension du langage Java permettant notamment de créer des services web et d'en réaliser la partie backend (côté serveur). Le composant principal de Spring que nous allons utiliser est *spring boot* qui facilite énormément la mise en place d'un service web de type REST. En effet, ce Framework simplifie le plus possible les premières configurations permettant de mettre en place une API REST (une API correspondant à la partie backend d'un service web). De plus, il facilite les tâches des développeurs en prenant en charge les différents types de données et de liaisons de données au sein du code grâce à un système d'annotations. Le but étant de permettre aux développeurs de ne pas perdre de temps sur une configuration compliquée et de leur permettre de se concentrer uniquement sur la mise en place des différents services. Le Framework permet également de créer une structure de code standard et cohérente avec l'architecture MVC qui est composée de 4 packages. Le package *Model* dont les classes servent à modéliser la structure de la base de données utilisée dans une forme compréhensible par le langage Java. Le package *Repository* qui comprend une interface qui va servir de référentiel pour la base donnée. Le package *Service* dont les classes sont constituées des différents services que l'on va proposer à l'utilisateur qui sont, dans le cas de ce projet, codées sous forme de méthodes Java. Enfin le dernier package se nomme *Controller*, les classes vont faire les liaisons entre les services que l'on propose dans notre API et les différentes requêtes effectuées par un client léger (un utilisateur sur un site web). Le Framework Spring permet donc de simplifier grandement la mise en place d'un service web en Java EE et permet donc aux développeurs de se concentrer sur la partie service d'une API.





Bootstrap

Le Framework Bootstrap est utilisé dans la partie frontend du service web. En effet, bootstrap est un Framework CSS qui permet de faciliter le design de notre application web du point de vue du navigateur, donc pour l'interface graphique que va voir l'utilisateur. Pour se servir de ce Framework on va pouvoir ajouter/modifier certains attributs de la section « class » se trouvant dans l'en-tête des différentes balises HTML (comme <p>, <a> ...). Ainsi on va pouvoir créer certains designs déjà « tout-faits » générés par le Framework et les personnaliser par la suite si on le veut. Ce Framework permet donc de gagner énormément de temps lors du design de la vue (frontend) de notre application web. Ce Framework, comme tous les Framework, a ses limites et dans notre cas on peut assez rapidement être bloqués ou obtenir des erreurs si l'on veut trop personnaliser/modifier les composants générés par bootstrap.



```
<div class="car d-flex flex-column justify-content-center">
```





CONCLUSION

Pour conclure, même s'il est vrai que nous n'avons pas totalement fini le projet étant donné que le prototype n'est pas encore dans sa version définitive, nous pouvons considérer que ce projet est une réussite. En effet, nous avons réussi à créer, en l'espace de quatre mois, un prototype fonctionnel basé sur la même architecture que la solution finale envisagée. Maintenant il ne reste plus qu'à étoffer le prototype et faire des séries de tests pour s'assurer que l'application web est fiable et sécurisée afin de pouvoir être mise en place au sein des locaux de l'école.

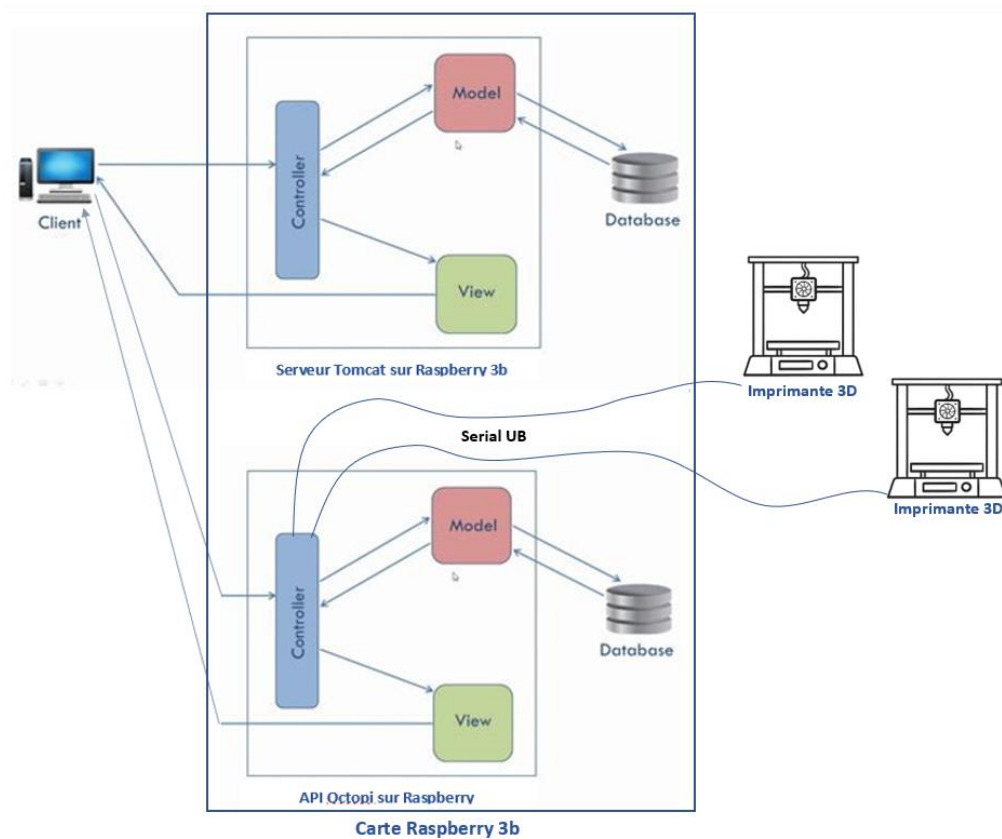
Au-delà du point de vue purement technique, ce projet nous a permis de mieux comprendre les liens entre les systèmes de robotique (à petite ou grande échelle) et l'informatique ainsi que de comprendre comment est faite la structure d'un service web et le rôle que jouent les différents éléments constituant le service web.

Enfin c'est aussi un plaisir de pouvoir créer un service web qui sera, à terme, utilisé par d'autres étudiants de Polytech Nancy dans le cadre de robotech, et qui ajouteront sans doute de nouvelles fonctionnalités à ce service afin de le rendre encore plus performant et donc de continuer sans cesse d'innover.



ANNEXES

Architecture Idéale



Pages du site Web

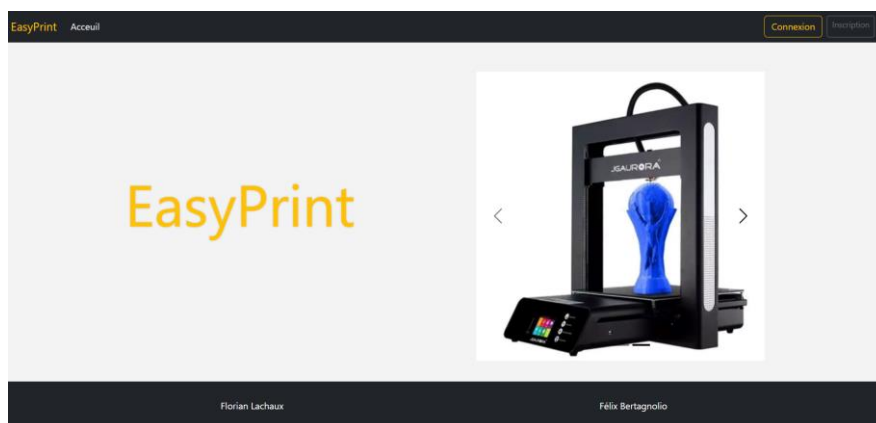


Figure 1 : page d'accueil



Bienvenue loxxx

Imprimante	Modèle	Adresse IP
1	model a	1.1.1.1

Fichier à envoyer à l'imprimante 1

Choisir un fichier

Aucun fichier choisi

Envoyer

Imprimante	Modèle	Adresse IP
2	model b	2.2.2.2

Fichier à envoyer à l'imprimante 2

Choisir un fichier

Aucun fichier choisi

Envoyer

Figure 2 : page client

EasyPrint Accueil

Inscrivez vous chez EasyPrint

Identifiant

nom d'utilisateur

Mot de passe

mot de passe

Modèle imprimante 1

Modèle imprimante 2

Valider

Figure 3 : page d'inscription



Le code des parties Backend et Frontend sont disponible sur GitHub aux adresses suivantes

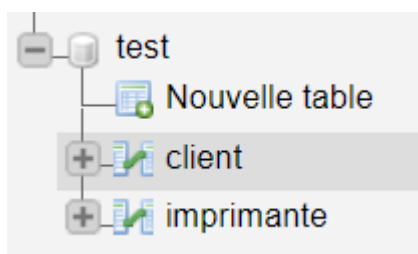
Back :

https://github.com/Feloxladetox/Back_AppliWeb

Front :

https://github.com/Feloxladetox/Front_AppliWeb

Base de données



La base de données « test » est composée de la table client ainsi que la table imprimante

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Extra	Action
1	id	bigint(20)			Non	Aucune	AUTO_INCREMENT	Modifier Supprimer Primaire Unique Index Spatial Texte entier plus
2	mdp	varchar(255) latin1_swedish_ci			Oui	NULL		Modifier Supprimer Primaire Unique Index Spatial Texte entier plus
3	nom	varchar(255) latin1_swedish_ci			Oui	NULL		Modifier Supprimer Primaire Unique Index Spatial Texte entier plus
4	modim1	varchar(255) utf8mb4_unicode_ci			Oui	NULL		Modifier Supprimer Primaire Unique Index Spatial Texte entier plus
5	modim2	varchar(255) utf8mb4_unicode_ci			Oui	NULL		Modifier Supprimer Primaire Unique Index Spatial Texte entier plus

Figure 4 : Structure de la table client

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Extra	Action
1	id	int(11)			Non	Aucune	AUTO_INCREMENT	Modifier Supprimer Primaire Unique Index Spatial Texte entier plus
2	model	varchar(255) latin1_swedish_ci			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier plus
3	adresse	varchar(255) latin1_swedish_ci			Non	Aucune		Modifier Supprimer Primaire Unique Index Spatial Texte entier plus

Figure 5 : Structure de la table imprimante



Api Octoprint

<https://docs.octoprint.org/en/master/api/>