

RAPPORT DE STAGE

—

BUT2 Informatique

Parcours A : réalisation d'applications

**“Développement d'une extension WordPress pour
l'utilisation des données personnelles”**

Steven TEA

Durée : 3 avril 2023 – 2 juin 2023

Lieu : Le Trente, 30 avenue Carnot, 91300 MASSY

Maître de stage : Jean-Christophe TOUVET

Tuteur pédagogique : Mourad OUZIRI

Executive summary

SIMPLOS est une plateforme innovante permettant aux petites organisations de s'abonner à un bouquet de services sécurisés liées à la gestion et à l'utilisation des données personnelles de leur clients ou adhérents, conformément à la loi européenne (RGPD). L'implémentation technique de cette solution s'appuie sur des mécanismes de sécurité brevetés, une application web, API et une application mobile.

Lors de mon stage, mon principal objectif est de développer une extension SIMPLOS pour WordPress. Cette extension à développer doit permettre l'intégration d'une fonctionnalité de récupération des données personnelles de la plateforme SIMPLOS sur n'importe quel site web marchand.

Je suis également responsable de l'ajout d'une fonctionnalité de notifications sur l'application mobile. Cette fonctionnalité permettra aux utilisateurs de recevoir des notifications sur leurs appareils mobiles pour les tenir informés de diverses informations de la plateforme SIMPLOS.

Le but principal du stage est donc de contribuer au développement et à l'amélioration de divers aspects du prototype du projet SIMPLOS.

Résumé & Abstract

Pendant mon stage, le travail qui m'a été confié est de développer une extension SIMPLOS pour le CMS WordPress et de développer la nouvelle fonctionnalité push sur le prototype d'application mobile de la société SIMPLOS.

L'objectif de l'extension est de permettre, pour un site internet marchand quelconque, de pouvoir se connecter à l'API de l'entreprise et utiliser les différentes fonctionnalités que propose la plateforme SIMPLOS et d'y garantir la sécurité. Pour pouvoir développer une telle extension, il faut avant cela que je développe un site marchand fictif en utilisant le CMS WordPress, qui me sert d'environnement de test pour l'extension que je développe. Cette extension est donc développée avec le langage de programmation qu'utilise WordPress, le langage PHP. Comme l'extension a pour objectif d'être portée et utilisée sur n'importe quel site marchand, je dois également m'assurer de la portabilité de l'extension sur d'autres sites marchands.

En ce qui concerne l'application mobile, je me charge de mettre en place une fonctionnalité push de notifications, consistant à recevoir des messages de l'API. Pour pouvoir implémenter cette nouvelle fonctionnalité, j'ai appris à utiliser le langage de programmation Dart et le framework Flutter, langage utilisé pour développer le prototype d'application mobile de l'entreprise.

Ces deux ajouts nécessitent également que j'apporte des changements dans l'API REST du prototype SIMPLOS.

During my internship, I was tasked with developing a plugin for the CMS WordPress and implementing a new push notification feature in the SIMPLOS' mobile application prototype.

The goal of the plugin is to allow any e-commerce website to connect to the company's API and use various features offered by the SIMPLOS' platform while ensuring security. Before starting to develop this plugin, I needed to create a fictional e-commerce website using the WordPress CMS which serves as a testing environment for the extension I am developing. This plugin is built using the PHP language, which is the native language of WordPress. I should also ensure compatibility with other e-commerce websites.

As for the mobile application, my responsibility is to add the push notifications feature that involves receiving notifications from the API. To implement this feature, I learned how to use the programming language Dart and the Flutter framework, which are used to develop the company's mobile application prototype.

These two new features require me to make changes to the SIMPLOS REST API prototype.

Mots-clés

Web, PHP

Extension / Plugin WordPress

Dart / Flutter

Cybersécurité

Données à caractère personnel

Remerciements

Je tiens à exprimer ma sincère gratitude envers Jean-Christophe TOUVET pour sa pédagogie et son investissement. Les nombreuses heures que j'ai passées à échanger avec lui ont été extrêmement bénéfiques. Son expertise en matière de cybersécurité et sa connaissance approfondie du domaine m'ont permis de renforcer mes connaissances et d'acquérir un point de vue différent sur la sécurité des données et des systèmes informatiques.

Je remercie également le personnel du Trente pour l'accueil chaleureux dans leurs locaux. Leur hospitalité a contribué à créer une ambiance favorable où j'ai pu me sentir à l'aise et me concentrer sur mes missions. Le professionnalisme et l'assistance du personnel m'ont été d'une grande aide tout au long de mon séjour. Je les remercie très sincèrement pour leur soutien précieux qui a grandement contribué à la réussite de mon stage.

Et je tiens finalement à remercier mon tuteur enseignant, Monsieur Mourad OUZIRI, pour son suivi de stage et pour ces deux dernières années d'enseignement à l'IUT de Paris qui m'ont permis d'acquérir de solides compétences en informatique.

Sommaire

I.	Introduction.....	6-7
II.	Contexte de l'entreprise SIMPLOS	8-10
1)	Historique de la société	8
2)	Clients et activités	8
3)	Environnement de travail.....	9
4)	Présentation du matériel utilisé	9
III.	Présentation des missions	11-21
1)	Analyse de l'existant.....	11
a)	Côté serveur	11
b)	Côté application mobile.....	11
2)	L'extension WordPress	12-16
a)	Objectif de la mission.....	12
b)	Difficultés rencontrées	12
c)	Réalisation de l'extension	13
d)	Modifications dans l'API.....	15
3)	La fonctionnalité "push"	16-22
a)	Objectif de la mission.....	16
b)	Problèmes rencontrés.....	17
c)	Ajout dans le serveur	17
d)	Solution utilisée.....	19
e)	Autres solutions possibles	21
IV.	Conclusion.....	23
V.	Table des figures	24
VI.	Annexes	25-26
VII.	Glossaire	27-28
VIII.	Webographie	29


I. Introduction

La société SIMPLOS développe une plateforme permettant aux TPE et associations de s'abonner à un ensemble de services sécurisés liés à la gestion et à l'utilisation des données à caractère personnel de leurs usagers. Cette plateforme est conforme à la loi européenne sur la protection des données, connue sous le nom de Règlement Général sur la Protection des Données (RGPD). Les usagers ont la possibilité de consulter les informations collectées par un abonné, de donner leur consentement pour leur utilisation. Ils peuvent également exercer leurs droits en matière de protection des données, tels que le droit d'accès, de rectification et de suppression.

Face aux enjeux croissants de protection des Données à Caractère Personnel et de la conformité au RGPD, les TPE et les associations rencontrent des difficultés pour gérer efficacement les informations sensibles de leurs utilisateurs.

C'est dans ce contexte que l'objectif central de mon stage est de proposer une extension WordPress pour les sites marchands voulant utiliser les services de la plateforme SIMPLOS, pour protéger les données personnelles des utilisateurs et des administrateurs de ces sites. Cette extension agit comme un intermédiaire entre les sites marchands et la plateforme SIMPLOS, permettant ainsi de déléguer la gestion des données à caractère personnel et la sécurité à SIMPLOS. Les données personnelles des usagers sont alors stockées dans les bases de données de SIMPLOS et le site marchand avec son compte d'abonné, utilisera l'identifiant de ces usagers, pour accéder à leurs DCP grâce à des tickets d'accès. L'ajout de la fonctionnalité "push" permet de recevoir des notifications de la plateforme SIMPLOS sur l'application mobile. Pour réaliser cela, je vais m'impliquer dans l'analyse de différentes solutions possibles et le développement d'une de ces solutions, ainsi que dans l'ajout et la modification des fonctionnalités de l'API existante.

Ces deux missions m'ont particulièrement intéressé en plusieurs points. Tout d'abord, j'ai été enthousiasmé par l'opportunité de travailler avec des outils tels que le CMS WordPress et le langage de programmation PHP pour le développement de l'extension et dans l'API de SIMPLOS dans l'environnement Symfony. Ces technologies sont largement utilisées dans le domaine du développement web, et pouvoir les maîtriser représente une réelle valeur ajoutée à mon parcours. De plus, le domaine de la cybersécurité et de la protection des données à caractère personnel est d'une importance croissante dans le paysage numérique actuel, et le fait de pouvoir contribuer au développement de solutions respectueuses de la vie privée est également un aspect motivant de ce stage. Finalement, les locaux du Trente sont situés dans un environnement dynamique et agréable, offrant un cadre propice à l'acquisition de nouvelles compétences professionnelles.



Ce rapport vise à détailler les différentes étapes et les défis rencontrés lors de la réalisation de ce projet. Je vais tout d'abord présenter le contexte de la société SIMPLOS, en parlant de l'historique de l'entreprise, de ses clients et ses activités, puis sur les outils et le matériel utilisés. Ensuite, j'aborderai l'état de l'existant puis les missions qui m'ont été confiées, qui sont pour rappel, le développement de l'extension pour WordPress, en détaillant l'objectif de la mission, les difficultés que j'ai pu rencontrer lors du développement, puis les résultats obtenus et modifications dans l'API existante, et finalement je parlerai de l'implémentation de la fonctionnalité "push" sur l'application mobile, des difficultés que j'ai rencontrées et de la solution que j'ai choisie, puis d'autres solutions envisageables pour son optimisation.

II. Contexte de l'entreprise SIMPLOS

1) Historique de la société

Créée le 13 février 2019, SIMPLOS est une startup active depuis plus de quatre ans et dirigée par Jean-Christophe Touvet. Elle s'est imposée en tant qu'entreprise spécialisée dans la cybersécurité et la protection des données personnelles. Sous la forme juridique d'une Société par Action Simplifiée (SAS), SIMPLOS a effectué sa dernière modification du capital social le 22 septembre 2021 et est actuellement à 42 000€.

L'objectif est de pouvoir offrir aux utilisateurs de SIMPLOS un contrôle sur leurs données à caractère personnel en utilisant des tickets d'accès vers les données personnelles. L'utilisateur peut donner l'accès à une DCP à une organisation et pour cela, créer un ticket d'accès que l'organisation peut utiliser. Les données à caractère personnel sont chiffrées dans la base de données de SIMPLOS grâce à une clé de chiffrement AES 256 bits générée aléatoirement sur la plateforme spécifique à l'utilisateur. Pour les organisations, cela permet aussi de les protéger des cyberattaques en utilisant ces services sécurisés.

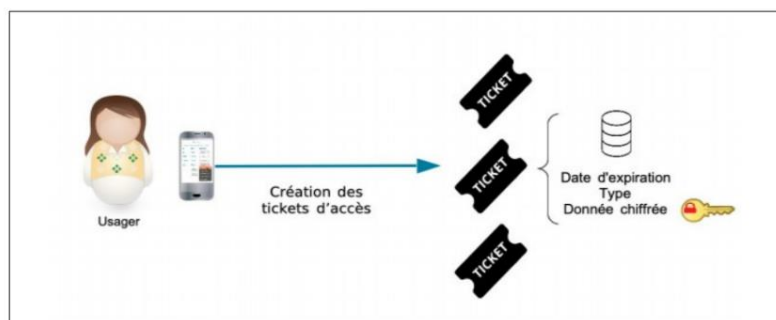


Figure 1 - Création de ticket d'accès

Étant donné que SIMPLOS est au stade de conception et développement d'un prototype, les missions données sont alors faites dans le cadre d'une démonstration de la plateforme et de l'application mobile.

2) Clients et activités

SIMPLOS est une société spécialisée dans le développement d'une solution technologique axée sur la protection des données personnelles et la gestion de la vie privée.

Elle vise principalement deux types d'utilisateurs :

- les particuliers voulant protéger et contrôler l'utilisation de leurs données personnelles, appelés les usagers.
- les organisations voulant utiliser les DCP de ces usagers en passant par la plateforme, appelés les abonnés.

Elle a développé un prototype comportant une application web et une API REST les deux développées dans l'environnement Symfony, ainsi qu'une application mobile développée en Dart dans l'environnement Flutter pour les appareils Android et iOS. L'application web est principalement utilisée par les abonnés afin de consulter les données personnelles de leurs usagers pour lesquels l'abonné a les autorisations, et l'application mobile est utilisée par les usagers de SIMPLOS pouvant gérer et contrôler l'utilisation de leurs DCP par les abonnés. L'API REST est la plateforme qui permet de connecter l'application mobile et web avec la base de données SIMPLOS pour y apporter des modifications des données ou leur consultation.

SIMPLOS accorde donc une importance primordiale à la confidentialité et au respect de la vie privée de ses usagers, ainsi qu'à la transparence et la responsabilité à travers ses diverses solutions.

3) Environnement de travail


L'entreprise SIMPLOS est représentée par son dirigeant et qui est en même temps mon tuteur de stage, Jean-Christophe TOUVET. SIMPLOS est située au Trente, un espace de coworking et de création d'entreprise à Massy.

C'est donc ici où j'ai pu travailler et profiter de l'espace pendant toute la durée de mon stage. Des points d'avancement ont été effectués très régulièrement, deux à trois fois par semaine avec mon tuteur de stage afin de constater mon avancement et voir les modifications et corrections à y apporter. Ces points d'avancement nous ont permis d'échanger sur les tâches à réaliser et les futures tâches et de mieux comprendre les valeurs de l'entreprise.

Le fait d'avoir pu travailler dans cet espace de travail a été très stimulant et j'ai pu davantage me concentrer sur les missions auxquelles j'ai été assigné.

4) Présentation du matériel utilisé

L'environnement matériel est assez simple, mon ordinateur et mon téléphone portable sous Android pour l'application mobile, ainsi qu'un serveur de développement distant hébergé chez OVH, contenant le prototype de SIMPLOS : l'API, ses bases de données, l'application web, ainsi que le serveur WordPress sur lequel je travaille.



Le serveur distant de la société utilise l'environnement LAMP, qui offre une grande flexibilité et stabilité. Étant donné que le serveur WordPress que je développe et l'API sont sur ce serveur distant, pour pouvoir réaliser mes missions, je dois pouvoir m'y connecter pour y faire des modifications. Pour cela, nous utilisons le protocole Secure Shell (SSH), qui permet d'établir une connexion chiffrée entre mon ordinateur et le serveur distant de l'entreprise. Le protocole SSH offre une grande sécurité pour la communication entre la machine locale et le serveur, il fournit notamment plusieurs méthodes d'authentification des utilisateurs, dans le cas de SIMPLOS, nous utilisons la méthode d'authentification à clé publique et privée.

Pour ce qui est de l'environnement de développement logiciel, j'ai utilisé l'éditeur de texte Nano, qui est disponible nativement sur de nombreuses distributions Linux et donc disponible sur le serveur de SIMPLOS, pour travailler sur le développement de l'extension et faire des modifications dans l'API. J'ai donc aussi utilisé le CMS WordPress pour pouvoir développer le site marchand qui sert d'environnement de test pour l'extension et j'ai utilisé le langage PHP pour développer l'extension car PHP est le langage utilisé par WordPress pour faire des sites internet. L'API de la plateforme, disponible sur le serveur, est également développée en PHP avec le framework Symfony et suit l'architecture REST. Pour récupérer ou modifier des informations dans l'API, je peux me connecter à l'API avec une requête HTTP.

En ce qui concerne l'application mobile avec la fonctionnalité "push", j'ai utilisé l'environnement de développement Android Studio, spécialement conçu pour des applications Android et offrant des outils avancés pour les émulateurs téléphone, comme la consultation de l'énergie consommée par l'appareil ou un traqueur réseau, pour voir les requêtes HTTP envoyées à l'API. Pour cette mission, j'ai eu l'occasion d'apprendre à utiliser le langage de programmation Dart avec le framework Flutter, qui assure une portabilité sur une multitude de plateformes, notamment iOS et Android.

Toute cette combinaison d'outils m'a permis de travailler de manière efficace, afin de pouvoir répondre aux exigences du projet.

III. Présentation des missions

1) Analyse de l'existant

a) Côté serveur

Avant mon stage, dans l'API de la plateforme, il était possible de créer, modifier, supprimer des tickets d'accès aux données personnelles des usagers et au niveau de la sécurité, les tickets sont également chiffrés dans la base de données SIMPLOS. Sur l'application web, utilisée par les abonnés, les tickets de leurs usagers sont également consultables. Il est également possible de créer des collaborateurs avec un abonné, celui-ci aura accès aux tickets de l'abonné qui l'a créé.

Cependant, il manquait une fonctionnalité dans l'API permettant aux abonnés d'accéder à certaines données personnelles des utilisateurs de SIMPLOS. Une telle fonctionnalité a pour objectif de pouvoir utiliser les données à caractère personnel des usagers tout en protégeant les clients et les organisations d'éventuelles cyberattaques en gardant les données à caractère personnel des usagers dans les bases de données de SIMPLOS.

b) Côté application mobile

Pour l'application mobile, du côté de la sécurité, l'utilisateur s'identifie sur son téléphone avec son empreinte digitale et les données sont chiffrées avec un système de clé, dont une partie se trouve du côté du serveur, dans la base de données et l'autre dans le stockage interne du téléphone. C'est grâce à la combinaison des deux moitiés de clé que les tickets sont déchiffrables pour que l'abonné puisse les lire. Au niveau des fonctionnalités, un usager peut ajouter et consulter ses DCP, modifier, créer ou supprimer les autorisations de ticket d'un abonné.

Cependant, il n'est pas possible d'informer un utilisateur si une erreur se produit dans l'API. Dans notre cas, si un abonné n'avait pas les autorisations nécessaires pour accéder ses données à caractère personnel, il faudrait pouvoir en informer l'utilisateur. Cela permet d'améliorer l'expérience des utilisateurs sur l'application mobile.

L'objectif de mon stage est donc, dans un premier temps, de développer l'extension WordPress, d'ajouter cette fonctionnalité de push dans l'application mobile, et d'apporter les modifications nécessaires dans l'API. Pour l'extension WordPress, ajouter une fonction de récupération des DCP d'un usager, et pour la fonctionnalité de push, ajouter une table de notifications dans la base de données et développer les services web qui seront utilisés par l'application mobile.

2) L'extension WordPress

a) Objectif de la mission

Afin que les abonnés puissent utiliser les données personnelles de leurs usagers, SIMPLOS a décidé de faire une extension pour le CMS WordPress, étant le CMS le plus utilisé et ayant une grande base de développeurs de site et d'extension, il a donc semblé plus judicieux d'utiliser ce CMS plutôt qu'un autre. L'objectif de cette extension est de pouvoir effectuer toutes les opérations nécessitant la consultation de données en utilisant la plateforme SIMPLOS comme passerelle. En plus de l'extension que je développe, je dois également fournir un mode d'emploi du module pour que les développeurs WordPress comprennent son fonctionnement et son utilisation.

b) Difficultés rencontrées

Lors du développement de l'extension, j'ai rencontré quelques difficultés, par exemple, sur le site marchand que je développe pour tester et utiliser l'extension, afin que le site ait un comportement de site d'e-commerce, j'utilise une extension appelée "Woocommerce", permettant d'enregistrer des produits, passer des commandes, consulter son panier... En utilisant cette extension, je suis alors contraint que l'extension que je développe soit dépendant de l'extension Woocommerce, mon but était alors d'éviter de trop dépendre de ce module pour pouvoir assurer la portabilité de l'extension sur divers sites marchands utilisant d'autres extensions pour faire de l'e-commerce.

De plus, dans le fonctionnement de l'extension, les DCP récupérées de l'utilisateur sont directement affichées à l'écran, donc si un utilisateur saisit l'identifiant d'un autre utilisateur, il peut alors voir et utiliser ses données à caractère personnel sans le consentement de cet utilisateur. Afin de prévenir de ce genre d'abus, nous avons opté pour une méthode de double authentification, c'est-à-dire, une fois qu'un utilisateur a entré un identifiant d'utilisateur, cet utilisateur va recevoir par SMS, un code confirmation qu'il va devoir entrer pour autoriser l'utilisation de ses DCP. Pour cela, nous utilisons Twilio, un service d'envoi de SMS à partir d'un serveur.

Ces difficultés rencontrées lors de la réalisation du plugin m'ont permis de mieux saisir le processus de développement d'extensions WordPress et les défis liés à l'intégration avec d'autres plugins. Également, cela m'a également permis de comprendre l'importance de la sécurité et de la protection des données à caractère personnel dans le développement de telles extensions.

c) Réalisation de l'extension

Afin de pouvoir identifier l'abonné qui utilise la plateforme SIMPLOS pour utiliser les données personnelles de ses usagers, l'organisation a besoin de se connecter sur la page de configuration de l'extension, disponible sur le tableau d'administration, et doit entrer son identifiant d'abonné ainsi que son mot de passe. Une vérification sera effectuée afin de voir si l'abonné existe et s'il peut donc utiliser les fonctionnalités de SIMPLOS ou non. L'abonné est informé de l'état de l'identification, succès ou échec.

Figure 2 displays two screenshots of the 'Mon identifiant d'abonné SIMPLOS' configuration form. The left screenshot shows a successful login with the message 'L'identifiant a été vérifié avec succès' (green bar), the identifier 'A000175', and a masked password. The right screenshot shows a failed login with the message 'L'identifiant ou le mot de passe n'est pas correct.' (red bar), the identifier 'A000000', and a masked password. Both forms include an 'Enregistrer' button.

Figure 2 - Formulaire de configuration de l'extension

L'extension développée contient trois formulaires, respectivement un pour l'inscription des utilisateurs sur le site marchand, un pour la complétion automatique des données pour l'utilisateur facturé et l'autre, pour l'utilisateur qui est livré (si les deux utilisateurs sont différents). Les formulaires ont la même mise en forme, seul leur comportement change.

Pour le formulaire d'inscription des usagers, une fois le code d'authentification validé, il récupère l'adresse électronique de l'usager, l'unique donnée personnelle dont WordPress a besoin pour la création d'un compte. Une demande de réinitialisation de mot de passe est envoyée à l'adresse électronique saisie, avec cela l'utilisateur peut alors se connecter sur le site avec son identifiant SIMPLOS et le mot de passe entré.

Saisissez votre identifiant SIMPLOS ou entrez vos coordonnées ci-dessous.

Identifiant SIMPLOS *

Entrez votre code

Adresse mail *

Figure 3 - Formulaire d'inscription SIMPLOS

Utiliser cette fonctionnalité pour s'inscrire sur un site marchand au lieu de passer par la procédure classique d'inscription sur le site permet à l'utilisateur de ne pas saisir toutes les informations personnelles dont le site a besoin. Ainsi, voici le contenu de la base de données du serveur marchand avec un usager SIMPLOS et un utilisateur du site.

ID	user_login	user_pass	user_nicename	user_email
3	U000775	\$P\$BtqJ2ulFrPrR1a2mIYmd80MEDb1uKY1	u000775	ouistitive4@gmail.com
4	steven.tea@simplos.fr	\$P\$B59uO1rZal8xXA4lmaZc5CoBT9EUc21	steven-teasimplos-fr	steven.tea@simplos.fr

user_registered	user_activation_key	user_status	display_name
2023-05-22 09:02:57		0	U000775
2023-05-30 08:49:01		0	Steven Tea

Figure 4 - Données du site marchand d'un usager SIMPLOS et utilisateur simple

Lorsqu'un utilisateur s'inscrit par ce site marchand, dans la base de données sont stockées son nom, prénom et adresse électronique (les données personnelles requises pour l'inscription peuvent différer en fonction des sites), or, l'utilisateur passant par l'inscription SIMPLOS, son identifiant d'utilisateur (par exemple U000775) est enregistré et uniquement son adresse mail est nécessaire, pour recevoir l'email pour saisir le mot de passe de l'utilisateur pour la connexion.

En plus de cela, lorsque l'utilisateur s'inscrit et se connecte avec son identifiant SIMPLOS, il va utiliser les autres fonctionnalités de SIMPLOS, il n'aura pas besoin d'entrer le code de double authentification lorsqu'il saisit son identifiant d'utilisateur pour une autre opération une fois connecté.

Les deux autres formulaires respectivement pour l'utilisateur facturé et l'utilisateur livré permettent la complétion automatique des champs pour la facturation et la livraison des produits. Ces deux formulaires récupèrent les tickets d'accès nécessaires pour chacun des usagers, ainsi, pour l'utilisateur facturé, les DCP requises sont le nom, prénom, l'adresse avec le code postal et la ville, ainsi que l'adresse électronique, et le numéro de téléphone. Pour l'utilisateur livré, les DCP sont le nom, prénom, et l'adresse physique. Dans l'extension WooCommerce, ces données sont obligatoires pour pouvoir faire une livraison de produit, j'ai alors décidé de suivre ces obligations en récupérant ces informations personnelles.

Une fois l'extension développée, j'ai refait un site d'e-commerce sur un serveur vierge en utilisant WordPress, puis installé l'extension que j'ai développée sur le site et testé pour voir si les fonctionnalités de l'extension sont toujours utilisables sur un nouveau site marchand.

d) Modifications dans l'API

Du côté de l'API, pour développer cette extension, j'ai ajouté une fonction "lookup" qui est appelée lorsque l'un de ces formulaires SIMPLOS est soumis. Cette fonction permet de récupérer les types de DCP nécessaires passées en paramètre d'un usager à partir de l'identifiant d'utilisateur entré.

Avant qu'un abonné puisse récupérer les DCP de l'utilisateur, une vérification est faite pour identifier l'abonné grâce à un cookie de session, et si l'identifiant d'utilisateur entré existe ou non. Une fois les vérifications faites, le serveur stocke, dans une variable de session, un code généré aléatoirement qui sera envoyé par SMS à l'utilisateur qu'il va devoir saisir dans le formulaire de SIMPLOS. Le SMS est envoyé grâce à un client Twilio, une API qui peut envoyer des SMS depuis un serveur. Cela sert de double authentification pour l'utilisateur, dans le cas où un autre usager se tromperait dans son identifiant ou s'il essayait de récupérer des DCP d'autres usagers. Cet envoi de SMS est donc effectué uniquement si l'utilisateur n'est pas connecté ou si l'utilisateur connecté est différent de l'utilisateur dont les DCP sont demandées. Une fois le code entré et validé, la fonction renvoie sous forme d'un objet JSON associant le nom du type de la DCP avec sa valeur. Lorsqu'une des données personnelles n'a pas été autorisée pour l'abonné qui en a besoin, la valeur associée sera nulle, et la récupération des données se termine avec un échec de récupération des données et un message d'erreur. Cela évite alors d'offrir les données à un abonné alors qu'un usager n'a pas exprimé son accord total sur l'opération qu'il entreprend.

L'image ci-dessous représente le cas où la récupération des données personnelles a réussi. Par ailleurs, on peut remarquer que nom, prénom et adresse, code postal et ville sont séparés avec un retour à la ligne ('\n') : cela permet de remplir les champs qui sont séparés dans le formulaire de livraison et de facturation. La clé "sms" est associée à une valeur nulle car l'opération de récupération des données s'est passée sans erreur.

```
{  
  "nom": "Tea\nSteven",  
  "adresse": "143 avenue de Versailles\n75016\nParis",  
  "telephone": "xxxxxxxxxx",  
  "email": "steven.tea@simplos.fr",  
  "sms": null  
}
```

Figure 5 - Sortie d'un objet JSON de la fonction lookup

Avec cet ajout dans l'API, l'extension peut désormais récupérer les données personnelles d'un usager pour pouvoir les manipuler et faire les opérations de site marchand. Cette nouvelle fonction pourra également à l'avenir servir pour ajouter de nouveaux formulaires dans l'extension. Par exemple, pour faire un formulaire permettant de répondre à un formulaire de contact ou autres.

Cette fonction peut encore être améliorée, les données ont été récupérées en clair pour être utilisées, mais l'idéal aurait été d'utiliser la fonction d'insertion de DCP dans un document PDF, pour que les données à caractère personnel soient uniquement inscrites dans la fiche de livraison (pour que le livreur puisse lire les DCP) et qu'elles soient illisibles dans la base de données du site marchand. Mais à des fins de démonstration, nous avons opté de récupérer les DCP en clair directement.

3) La fonctionnalité "push"

a) Objectif de la mission

Une fois l'extension WordPress développée, et la fonction "lookup" fonctionnelle, et donc que les données à caractère personnel des usagers sont récupérables et utilisables par les abonnés, cette fonctionnalité de push de notification va de pair avec l'extension et l'utilisation des DCP car celle-ci a pour objectif d'informer les usagers que leurs données personnelles ne sont pas autorisées pour un abonné ou qu'elles sont manquantes. Cela permet alors d'améliorer l'expérience des utilisateurs de l'application mobile. L'ajout de cette fonctionnalité peut également servir à notifier les utilisateurs d'autres types de notifications qui peuvent être développés à l'avenir.

b) Problèmes rencontrés

Avant l'implémentation de la fonctionnalité "push" développée, une des solutions possibles que nous avons trouvées pour faire un push de notifications est d'utiliser Firebase Cloud Messaging (FCM), un service d'envoi des messages de notification optimisé et disponible sur iOS et Android développé par Google.

Cependant, l'utilisation de ce service cloud est pour plusieurs raisons, une option que nous avons écartée :

- Firebase Cloud Messaging pour l'environnement serveur est disponible dans les langages de programmation : Go, C#, Python, Java et Node. Or, l'API de SIMPLOS est développée en PHP, un langage non pris en charge par Google, cela nous force à passer par un tiers non-officiel pour pouvoir utiliser ce service.
- FCM étant développé par Google, l'idée que les données transitent par un tiers et que celui-ci ne respecte la vie privée de ses utilisateurs et que Google risque d'enregistrer les données dans leurs bases de données, nous a convaincu que l'utilisation d'un service de push de notifications développé par un tiers n'était pas une solution viable.

Nous sommes donc venus à la conclusion que nous allions nous-mêmes développer un service de push. J'ai alors pu proposer différentes solutions pour pouvoir intégrer une fonctionnalité de push dans l'application :

- 1) Permettre à l'API d'envoyer un SMS à l'utilisateur lorsqu'une notification a été poussée par le serveur.
- 2) Faire dans l'application mobile, une fonction en arrière-plan qui va appeler l'API pour récupérer les notifications dans la base de données.

c) Ajout dans le serveur

Pour mettre en place cette fonctionnalité de push de notifications, j'ai apporté des changements au niveau de l'API de SIMPLOS, ainsi que dans la base de données, pour pouvoir lire les notifications sur l'application.

Dans la base de données, j'ai ajouté une nouvelle table correspondant aux informations nécessaires à la notification :

Nom	Commentaire
id	Clé primaire de la notification, s'auto-incrémente
usager_unique_id	La clé primaire de l'utilisateur recevant la notification
abonne_unique_id	La clé primaire de l'abonné qui envoie la notification. Cette donnée peut être nulle et modifie les actions dans l'application mobile
message	Le contenu du texte qui est affiché

Figure 6 - Structure de la table de notification

Avec l'ajout de cette nouvelle table dans la base de données, afin de pouvoir manipuler des objets notifications dans l'API avec le framework Symfony, j'ai alors créé une entité "Notification", un repository de notification pour pouvoir parcourir et sélectionner une notification parmi celles présentes dans la base de données, ainsi qu'un contrôleur afin de faire des services web qui sont appelés par l'application mobile.

Dans le contrôleur de notification, j'ai ajouté deux fonctions de gestion des notifications, une pour sélectionner et renvoyer une notification, et une autre pour supprimer une notification.

L'image ci-dessous, est un extrait de la fonction permettant de récupérer une notification de l'API. Avant le retour de la notification, une identification de l'utilisateur est faite, avec son identifiant d'utilisateur et sa demi-clé stockée localement sur le téléphone. Une fois l'authentification réussie, nous faisons une recherche dans le repository de notification avec l'identifiant de l'utilisateur, pour trouver une notification de l'utilisateur. Une fois la notification récupérée, l'identifiant de l'abonné est vérifié et est mis à null si l'abonné n'existe pas, puis la notification est retournée au client.

```
$notification = $this->notification_repository->findOneBy(["usager_unique_id" => $usager_id]);

$company = $this->company_repository->findOneBy([
    'id' => $notification->getAbonneUniqueId(),
]);

$notification->setAbonneUniqueId($company != null ? $company->getUniqueId() : null);

return $this->json($notification);
```

Figure 7 - Extrait de code de la sélection d'une notification

La notification sélectionnée est la première dans l'ordre séquentiel de la clé primaire, étant donné qu'une notification n'est pas faite pour rester en permanence dans la base de données,

j'ai décidé d'utiliser la méthode FIFO pour gérer les notifications. Ainsi, si plusieurs notifications sont reçues d'un usager, il recevra en premier lieu la plus ancienne notification.

Du côté de l'application mobile, il faut pouvoir récupérer une notification de l'API, pour cela, je fais un appel à l'API en méthode POST, et dans le corps de la requête, je mets en paramètres l'identifiant d'usager et la demi-clé, pour faire l'authentification de l'usager côté serveur, puis si une notification a été trouvée dans le repository, la notification est reçue du côté client sous forme d'un objet JSON. La notification est ensuite enregistrée dans un modèle de notification et va être traitée dans l'application mobile.

```
Map data = {
    "usager_id": "${prefs.get('id')}",
    "secondhalf_key": "${prefs.get('secondhalf_key')}"
};
var body = jsonEncode(data);

await http.post(Uri.parse(notificationUrl + "list"), body: body).then((response) {
    if (response.statusCode == 200) {
        _jsonResponse = response.body;

        Map<String, dynamic> body = jsonDecode(response.body);

        print(body['usagerUniqueId']);
        notif = new NotificationModel(
            id: body['id'],
            usager_unique_id: int.parse(body['usagerUniqueId']),
            abonne_unique_id: body['abonneUniqueId'],
            message: body['message'],
        );

        print(" Notification get : " + notif.toString());
    }
});
```

Figure 8 - Extrait de code de l'appel à l'API pour la récupération de notification

d) Solution utilisée

Afin de consulter les notifications de l'utilisateur, une nouvelle page a été créée pour consulter les notifications en cours d'un usager. Un usager peut alors directement voir sur cette page s'il a une notification en attente de réponse. Lorsque l'utilisateur ouvre la page de notification, l'application fait un appel à l'API pour récupérer une notification, si une notification a été trouvée dans la base de données, un bouton apparaît et une boîte de dialogue se lance avec le message contenu dans la notification.

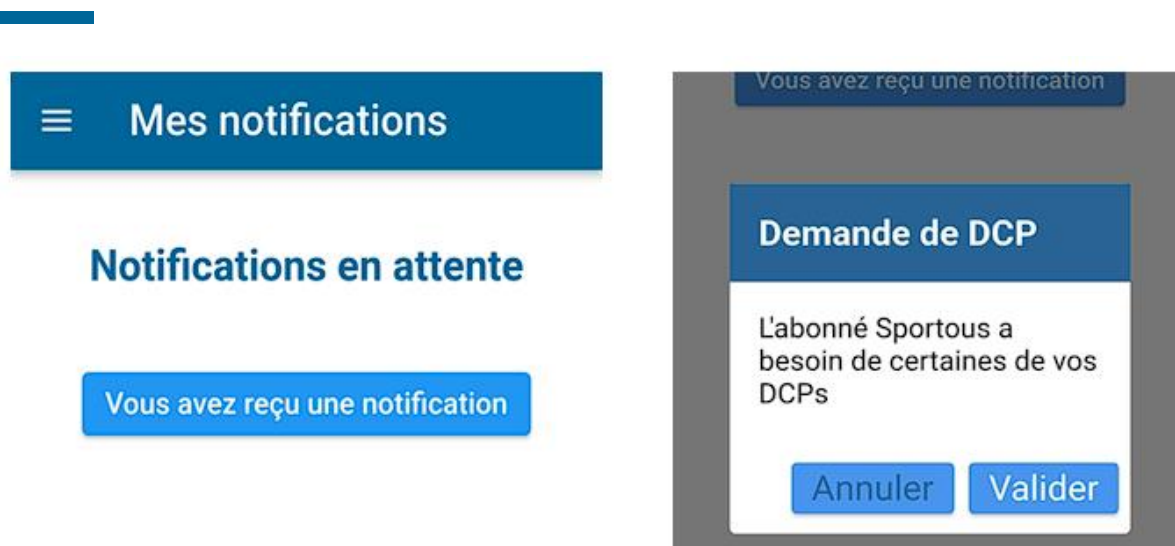


Figure 9 - Page d'une notification reçue et popup

Cela permet alors de lire les notifications que l'utilisateur reçoit et il peut donc donner les autorisations à l'abonné qui en a besoin pour faire l'opération.

Maintenant que les notifications sont lisibles par les usagers, je peux faire en sorte que l'application puisse notifier l'utilisateur lorsqu'une notification a été envoyée à l'API et reçue sur l'application mobile. Dans l'objectif de faire cette fonctionnalité de push de notifications sans passer par un tiers comme Firebase Cloud Messaging, j'ai décidé de faire une tâche planifiée qui tourne en arrière-plan qui, à chaque minute, fait une requête à l'API, et si une nouvelle notification a été récupérée, l'utilisateur en sera directement informé.

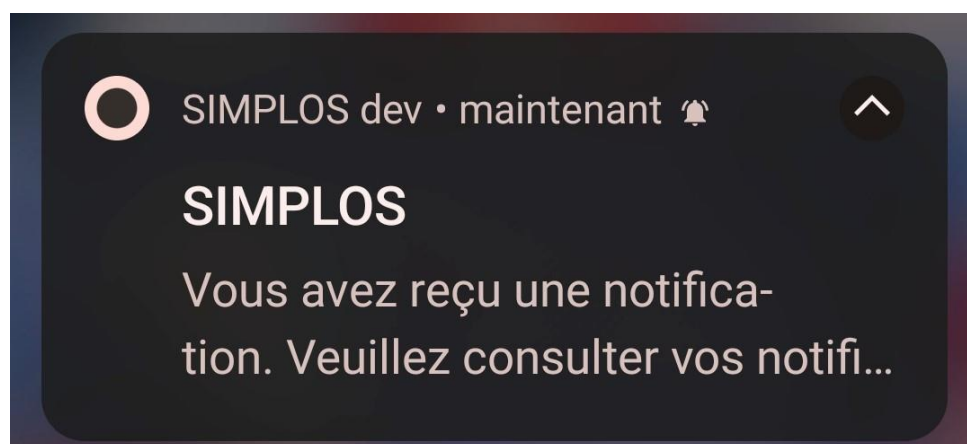


Figure 10 -- Notification reçue de l'application mobile

Ce système de push de notification sert de porte d'entrée pour la société et peut encore être développé à l'avenir. En effet, une fois la fonctionnalité de notifications en place, elle peut être utilisée pour informer l'utilisateur de divers types de messages différents, hors du manque de DCP de l'utilisateur. Par exemple, le code envoyé par SMS qui sert de double authentification pour utiliser les DCP peut-être envoyé avec une notification avec un identifiant d'abonné nul.

Cette méthode de push de notifications développée est fonctionnelle mais présente quelques problèmes. Elle est peu réactive, en effet, la notification peut prendre au maximum une minute entre le moment où le serveur pousse une notification dans la base de données et l'application mobile appelle l'API pour recevoir la notification.

e) Autres solutions possibles


Étant donné que la solution développée n'est pas la plus réactive et n'ayant pas le temps de pouvoir mettre en place une nouvelle solution, je recherche sur internet d'autres méthodes envisageables pour implémenter un système de push plus optimisé et plus réactif.

- Web Sockets : créer une connexion bidirectionnelle continue avec un Web socket Server (WSS), cette connexion continue permettrait donc de recevoir des notifications instantanément sur l'application mobile sans avoir besoin de répéter des appels à l'API, cependant, n'étant pas une requête HTTP, les web sockets ne sont pas utilisables avec l'architecture REST de l'API déjà en place et nécessiterait un changement conséquent.
- "Long polling" : une méthode permettant de maintenir une connexion HTTP ouverte avec le serveur pendant une période de temps prolongée, jusqu'à ce qu'une notification ait été reçue, cette méthode utilisant une connexion HTTP, elle est donc compatible avec l'API REST.
- "Server Sent-Event" (SSE) : une technologie servant à une application de recevoir des notifications dans une période de temps prolongée du serveur grâce à une unique connexion HTTP.

Ces deux dernières méthodes qui peuvent être mises en place sont plus réactives que la fonctionnalité push que j'ai implémentée, étant donné que lorsqu'une notification est reçue sur le serveur, elle peut être directement envoyée à l'application mobile grâce au maintien de la connexion ouverte.

Cependant, le maintien d'une connexion ouverte entre l'application mobile et le serveur peut être lourd du côté du client ou du serveur. Pour le long polling, l'appel en permanence à l'API peut être lourd pour le client, et pour le server sent-event, l'appel à l'API n'est fait qu'une fois mais c'est le serveur qui maintient la connexion jusqu'à l'arrêt du client, c'est alors plus lourd du côté du serveur que du client.

Je pense alors que le long polling est une solution plus adaptée dans le cadre du push car la technologie SSE est plus limitée. Le fait que la connexion doit être maintenue du côté du serveur peut limiter le nombre de connexions à l'API, alors il serait plus judicieux que ce



soit le client qui appelle le serveur. De plus, en cas d'erreur de connexion à l'API avec le SSE, le client ne rappelle pas l'API, alors que le long polling va refaire la connexion à l'API qu'il y ait une erreur de connexion ou non.

IV. Conclusion

Durant mon stage, j'ai réalisé deux principales missions, l'une est de réaliser une extension WordPress, pour cela, à partir d'un serveur WordPress vierge, j'ai développé un site marchand "Sportous", qui sert d'environnement de test pour mon plugin. Une fois le site web terminé, je passe au développement de l'extension. L'extension contient trois différents formulaires qui sont utilisés pour faire les principales opérations dans un site marchand, dont un, pour l'inscription des usagers, et les deux autres pour la complétion automatique pour la facturation et de la livraison. L'extension est utilisable mais elle peut être encore améliorée, en ajoutant de nouveaux formulaires pour couvrir d'autres opérations sur d'autres genres de sites. Lors de cette mission, en faisant le site marchand, j'ai pu me familiariser avec l'environnement WordPress et avec le développement de plugins dans ce CMS avec le langage PHP.

La deuxième mission que j'ai réalisée est une fonctionnalité de push de notifications dans l'application mobile de l'entreprise. Cette mission a pour objectif d'informer un usager lorsqu'il reçoit un message de la plateforme SIMPLOS. J'ai pu lier cette mission avec la première en faisant en sorte d'envoyer une notification à l'application mobile lorsque l'utilisateur n'a pas donné toutes les autorisations nécessaires à un abonné pour faire une des opérations disponibles dans l'extension. Ce système de notification peut être réutilisé pour informer l'utilisateur d'autres messages venant de la plateforme. Par exemple, le code de double authentification pour valider son identité peut être envoyé sur l'application mobile plutôt que par SMS et cela permet d'ajouter une couche de sécurité car pour accéder à l'application, l'utilisateur doit mettre son empreinte digitale.

En beaucoup de points, ces deux missions ont été très enrichissantes techniquement pour moi, j'ai pu me familiariser avec l'environnement Symfony dans l'API de SIMPLOS, et revoir le langage PHP dans l'API et dans le développement de l'extension WordPress. J'ai également pu apprendre à utiliser un nouveau langage, Dart et l'environnement Flutter dans l'application mobile. De plus, j'ai appris l'importance de la communication entre moi et mon tuteur, ayant eu pas mal de difficultés lors du développement du site et de l'extension WordPress, cela m'a permis de retirer mes doutes et zones d'ombre sur cette mission. Je pense alors que cette première insertion dans le milieu professionnel a été concluante.

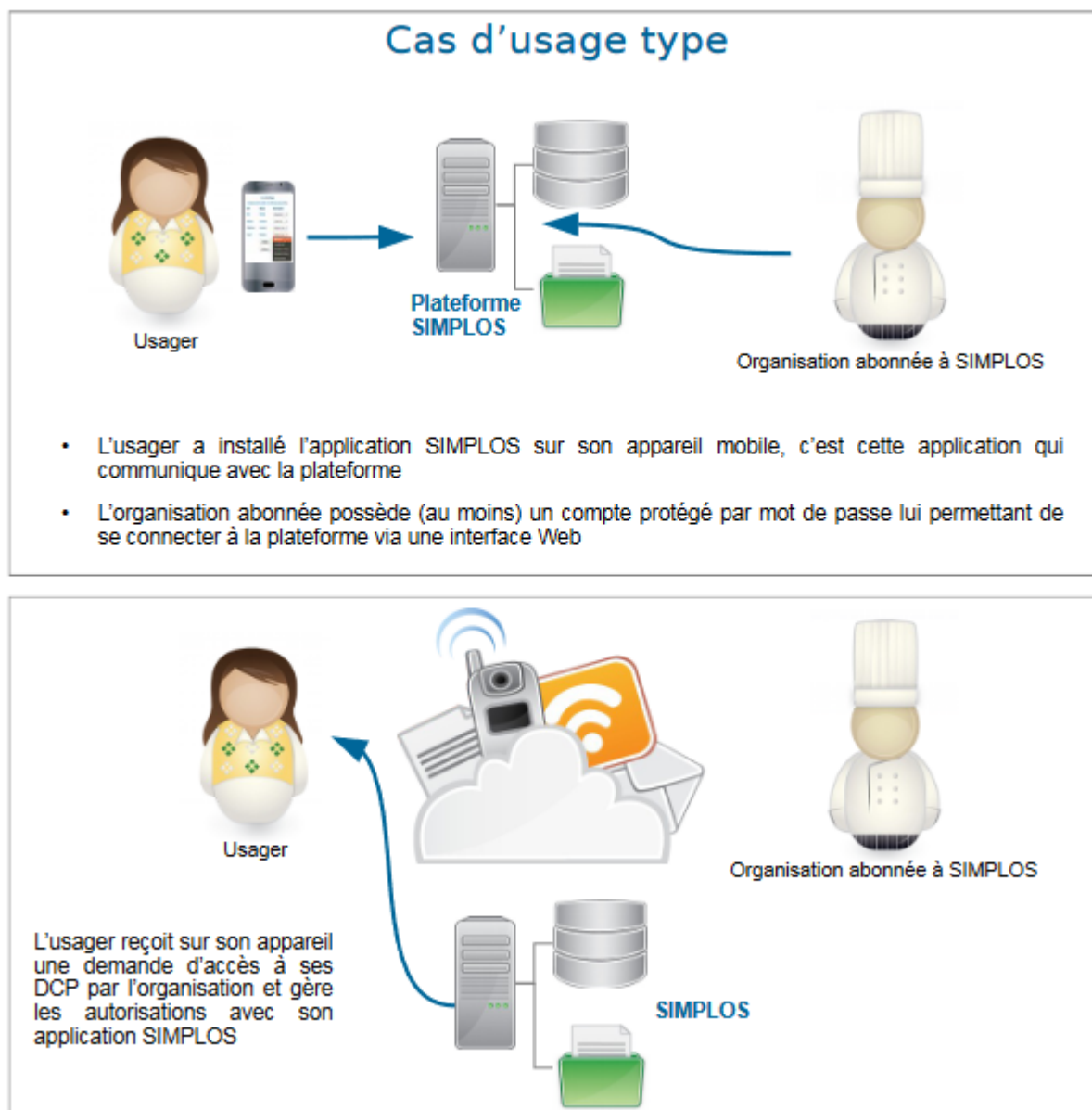
Pour moi, ce stage m'a conforté dans l'idée qu'après ma troisième année de BUT informatique de poursuivre en école d'ingénieurs en développement de logiciels. Ayant beaucoup apprécié la deuxième partie de mon stage, sur l'ajout de la fonctionnalité push de notification sur l'application mobile, cela m'a conforté dans l'idée d'approfondir mes connaissances en développement et en architecture logicielle.

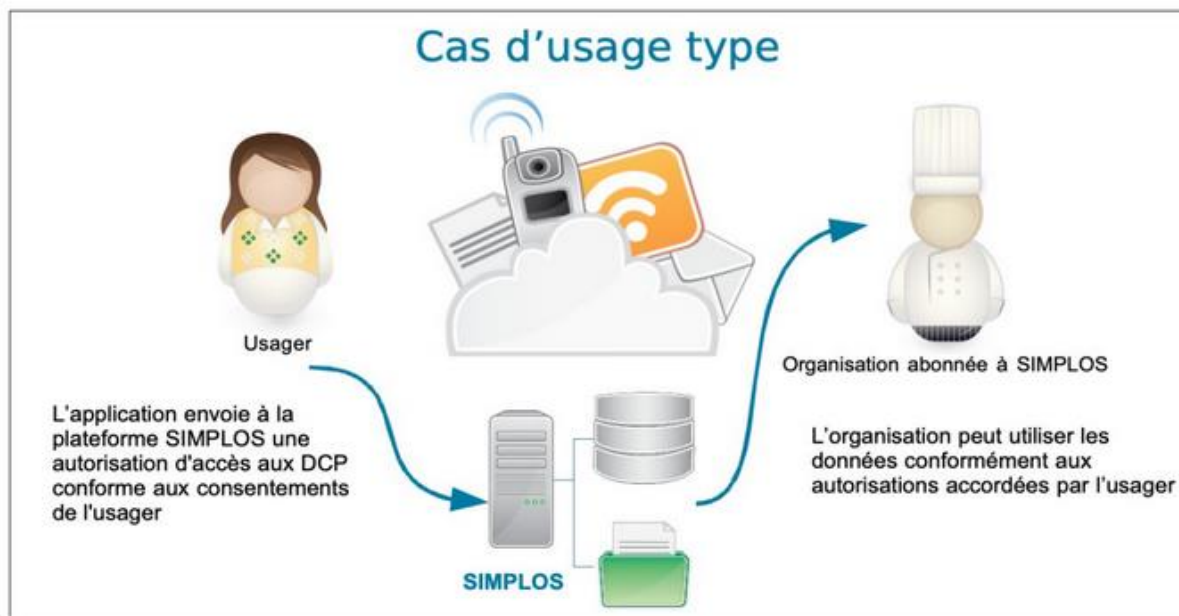
V. Table des figures

Figure 1 - Création de ticket d'accès	8
Figure 2 - Formulaire de configuration de l'extension.....	13
Figure 3 - Formulaire d'inscription SIMPLOS	14
Figure 4 - Données du site marchand d'un usager SIMPLOS et utilisateur simple	14
Figure 5 - Sortie d'un objet JSON de la fonction lookup	16
Figure 6 - Structure de la table de notification	18
Figure 7 - Extrait de code de la sélection d'une notification.....	18
Figure 8 - Extrait de code de l'appel à l'API pour la récupération de notification	19
Figure 9 - Page d'une notification reçue et popup	20
Figure 10 - Notification reçue de l'application mobile	20

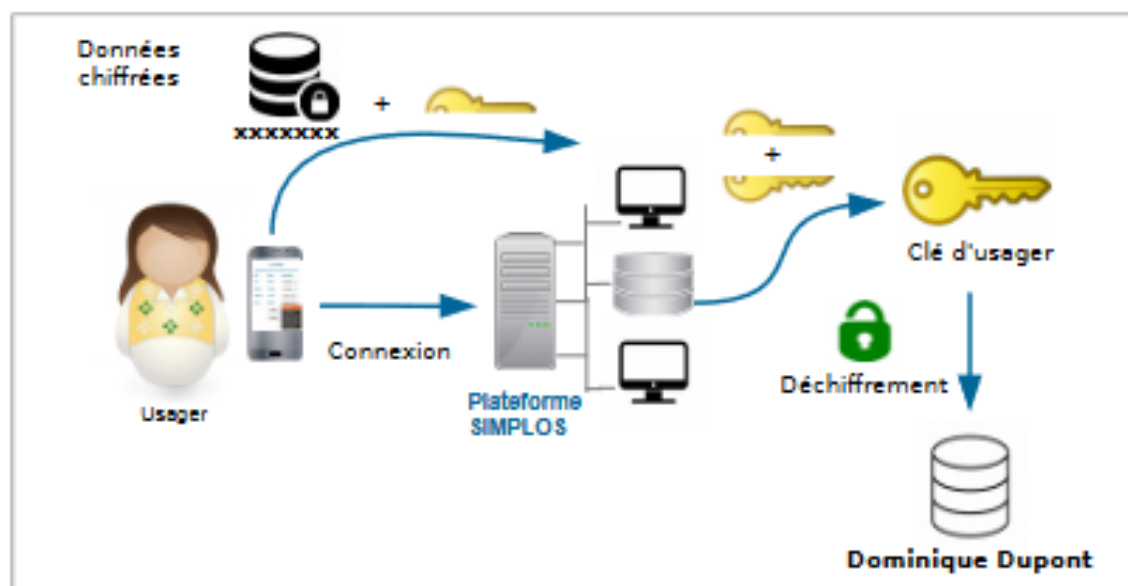
VI. Annexes

Principe du fonctionnement de la plateforme SIMPLOS





Mécanisme de connexion



VII. Glossaire

Usager : utilisateur de l'application mobile SIMPLOS.

Abonné : TPE ou association client de la plateforme SIMPLOS.

DCP : donnée à caractère personnel (ex : nom/prénom, adresse email, adresse physique, numéro de téléphone...).

Extension WordPress : module logiciel supplémentaire qui peut être ajouté à un site WordPress pour étendre ses fonctionnalités.

Ticket : ticket d'accès à une DCP avec sa durée d'expiration.

Chiffrement AES (Advanced Encryption Standard) : Algorithme de chiffrement symétrique utilisant une clé secrète (dans notre cas, la clé fait 256 bits) pour chiffrer et déchiffrer.

Notification Push : fonctionnalité servant à recevoir un message de notification d'une application sur le téléphone lorsque celle-ci n'est pas en cours d'utilisation.

LAMP (L = Linux, A = Apache, M = MySQL, P = PHP) : ensemble de logiciels libres permettant le développement de serveurs de site internet.


SSH (Secure Shell) : protocole de communication sécurisé utilisé pour établir une connexion chiffrée entre un ordinateur local et un serveur distant.

Authentification à clé publique et privée : méthode d'authentification utilisée dans le protocole SSH, où deux clés cryptographiques, une publique et une privée, sont utilisées pour vérifier l'identité de l'utilisateur.

Symfony : framework dans l'architecture MVC (Model View Controller) développé en PHP, il fournit des modules permettant l'accélération du développement d'un site web.

API (Application Programming Interface) : une interface logicielle qui permet de connecter un logiciel à un autre logiciel ou service afin d'échanger des données ou utiliser ses fonctionnalités.

REST (API) : ensemble de règles et de protocoles qui permettent à différentes applications de communiquer entre elles via le protocole HTTP, utilisé pour développer des services web.



Requête HTTP (Hypertext Transfer Protocol) : demande envoyée par un client à un serveur web pour obtenir des informations ou effectuer une action.

Objet JSON (JavaScript Object Notation) : format de données textuel dérivé du JavaScript.

Tâche planifiée : fonctionnalité qui permet de programmer une action qui se déclenche à intervalles réguliers ou à des moments spécifiques.

CMS (Content Management System) : système de gestion de contenu, un programme informatique servant à créer des sites internet.

Dart / Flutter : Flutter est un kit de développement développé par Google utilisant le langage Dart, il permet l'utilisation du logiciel sur une multitude de plateformes comprenant navigateur web, Windows, Android etc...

FIFO (First In First Out) : structure de données dans laquelle le premier élément inséré est le premier à en sortir.

VIII. Webographie

- Historique et contexte de la société

<https://www.societe.com/societe/simplos-848425120.html>

<https://www.verif.com/societe/SIMPLOS-848425120/>

- Sécurité et utilisation de WordPress

https://help-beta.ovhcloud.com/csm/fr-web-hosting-1-click-module-management?id=kb_article_view&sysparm_article=KB0052416

https://codex.wordpress.org/fr:Premiers_pas_avec_WordPress

- Développement du plugin sur WordPress

<https://developer.wordpress.org/plugins/intro/>

<https://wpformation.com/creer-plugin-wordpress/>

https://developer.wordpress.org/reference/functions/wp_insert_user/

- Compilation et mise à jour de Flutter

<https://flutter.dev/>

<https://medium.com/swlh/convert-your-flutter-app-to-enjoy-null-safety-69632aa62d7a>

- Notifications push

<https://firebase.google.com/docs/cloud-messaging?hl=fr>

<https://stackoverflow.com/questions/60108468/how-to-show-push-notification-in-flutter-without-firebase>

<https://codeburst.io/polling-vs-sse-vs-websocket-how-to-choose-the-right-one-1859e4e13bd9>