focus.

Cahier des Charges

contact@focus.sh
https://focus.sh

Résumé

Ce document détaille les composantes techniques à réaliser pour mener à bien le projet Focus. Il s'engage à faire transparaitre le point de départ du projet, et le chemin nécessaire à sa réalisation. Le cahier des charges permet de donner une idée des besoins du projet. Plusieurs points importants sont couverts dans ce document. Le premier fait état des fonctionnalités présentes au cœur de Focus, en faisant le rapprochement entre les composants techniques et leur usage. Cette partie présente les différentes parties logicielles de Focus (Backend, Frontend, Daemon, Mobile). Est ensuite présenté l'environnement de réalisation, qui détaille les besoin et contraintes techniques de Focus. Les outils logiciels sont listés dans cette partie, ainsi qu'un point important traitant des plateformes sur lesquelles Focus devra fonctionner. Les normes sont aussi discutées, séparant les normes externes suivies par Focus de celles créées par l'équipe. Les technologies existantes utilisées par Focus sont traitées dans la partie des composants existants. Cette partie établie les pour et contres des technologies utilisées, leur part d'utilisation, leur coût ainsi que les implications techniques qu'elle apportent. La sécurité du projet occupe une partie importante du document. Elle décrit les moyens utilisés au sein de Focus pour prévenir les failles de sécurité, les précautions adoptées et les processus de développement (GitHub, pull requests, code review, etc). Elle traite aussi de la sensibilité des données obtenues et stockées par Focus, en abordant le sujet de l'encryption. Enfin, l'organisation du développement de Focus est énoncée. Les processus de travail, les responsabilités ainsi que les moyens de communication sont détaillés ici.

Description du document

Propriétés	Métadonnées
Titre	Cahier des Charges
Date	09/07/2017
Auteurs	Enzo AGUADO, Etienne PASTEUR, Hippolyte BARRAUD
Version du modèle	1.0
Mots clés	Focus, EIP, Epitech, Cahier des Charges, CDC

Table de révisions

Date	Version	Section(s)	Auteurs	Commentaires
09/04/2017	v0.1	Toutes	Etienne PASTEUR	Création initiale
09/04/2017	v1.0	Toutes	Enzo AGUADO, Etienne PASTEUR	Création du contenu
01/06/2017	v2.0	Toutes	Hippolyte BARRAUD	Refonte totale pour le rendu 2 + Intégration des retours

Table des matières

Rappel de l'EIP

Qu'est-ce qu'un EIP et Epitech

Epitech (École pour l'informatique et les nouvelles technologies) est un établissement d'enseignement privé français créé en 1999 par le groupe IONIS qui délivre un enseignement supérieur en informatique et nouvelles technologies.

L'école propose à ces étudiants, à partir de leur troisième année, un projet de fin d'études : l'EIP (pour Epitech Innovative Project).

A ce titre, les élèves doivent s'organiser en un groupe d'au moins six personnes et choisir un sujet innovant. L'EIP est un passage obligatoire et unique dans la scolarité de l'étudiant, de par son envergure (18 mois) et la préparation requise.

Un Epitech Innovative Project est conçu (comme toute la pédagogie Epitech fondée sur la méthode projets) à la manière d'un véritable projet entrepreneurial, dans toutes ses composantes : business, techno, design & communication. Un EIP est appelé à devenir une start-up viable à la fin du cursus de l'étudiant.

Sujet de l'EIP Focus

Focus est une solution visant à améliorer le workflow de travail des travailleurs en leur donnant plus de temps pour faire ce qui compte vraiment.

Grace a des rapports d'activité et des analyses avancées fournis par des outils installés sur les ordinateurs et smartphones des employés, Focus propulse des informations pertinentes sur l'activité de l'utilisateur. En s'appuyant sur ces données brutes, notre solution utilise des technologies de Machine Learning et compile un profil utilisateur complet : les outils utilisés, à quelle fréquence, de quelle manière et dans quel contexte.

Ces données permettent à Focus de comprendre comment vous travaillez et de regrouper les sessions de travail analogues (répondre à des courriels, rencontrer des clients, téléphoner, etc.), capturer et diffuser intelligemment les notifications au bon moment en fonction de leur contenu et de leur priorité et de rationaliser globalement le flux de travail.

La finalité est de permettre à nos utilisateurs d'être 100% focalisés sur une tâche unique et bien définie à la fois et de les préserver de toute distraction pendant qu'ils réalisent cette tâche.

Description des différentes parties du programme

L'activité de Focus se présente sous la forme d'une déclinaison de **micrologiciels multiplateformes** à installer sur ses appareils de travail (ordinateurs, tablettes et smartphones). Un des buts de ces logiciels est d'enregistrer les actions de l'utilisateur. Nombre de frappes par minute, fréquence de changement de fenêtre, nature et contenu des documents affichés et bien d'autres encore ; toutes ces informations sont recueillies, sécurisées, et envoyées au **backend** de Focus afin d'être analysées.

Le backend applique en temps réel des algorithmes poussés de machine learning et donne du sens à ces actions. En sont inférés le sujet d'activité de l'utilisateur (travaille-t-il sur un dossier client ? Est-il en déplacement ? etc), son niveau d'implication dans ce sujet (à quel point est-il concentré ?) et l'importance de cette activité (est-ce critique à son activité ?).

Grâce à ces informations sur le niveau de concentration de l'utilisateur, les logiciels installés sur tous les appareils sont en mesure de **contrôler intelligemment les notifications** rentrantes en ne laissant passer que les plus pertinentes. Si l'utilisateur est concentré sur une tâche, les notifications non pertinentes à cette dernière seront capturées. Elles ne seront relâchées que lorsque l'utilisateur sera moins concentré, et plus propice aux interruptions.

Enfin, un **site web** permet à l'utilisateur de bénéficier de rapports sur ses activités. Focus est en mesure de restituer précisément tous les éléments constituant la journée de l'utilisateur : combien de temps a-t-il passé sur tel ou tel dossier, pendant combien de temps était-il en déplacement, etc. L'utilisateur a donc une vue macro de ses activités l'aidant à mieux prendre conscience de ses activités.

Backend

Le Backend est l'infrastructure qui sous-tend Focus. Son rôle est de faire la liaison entre les Daemons, chargés de récolter les données, avec les différents Frontend dont les rôles sont de mettre à disposition les données aux utilisateurs. Il représente un composant fonctionnel essentiel : il permet la synchronisation des autres éléments de la solution les uns avec les autres.

Ses responsabilités sont vastes mais son apport fonctionnel est central : le Backend se charge d'inférer un sens aux méta-data relayées par les Daemons. En se basant sur des algorithmes de cross-referencing entre tous nos utilisateurs, ainsi que des règles métiers inscrites en dur, il détecte des optimisations possibles. Parmi celles-ci on compte :

- Conseiller à l'utilisateur d'appliquer une méthodologie de gestion du temps spécifique (Ex. Pomodoro) et l'aider dans sa démarche.

- Bloquer temporairement les notifications de son ordinateur/de son portable quand une activité spécifique est en cours.

- Inciter à prendre des pauses lors d'une concentration soutenue durant une période trop longue.

- Différer intelligemment la réception de mails jusqu'aux périodes de temps mort de productivité.

- Dynamiquement mettre à jour le statut Office365 et autres en fonction de l'activité en cours. Si l'utilisateur semble très concentré sur une tâche, il vaut mieux qu'il ne soit pas dérangé.

- Bloquer temporairement certaines sources de distractions extra-professionnelles.

Le Backend est un élément qui, bien qu'invisible pour l'utilisateur, est déterminant dans la réussite du projet. Il représente effectivement une part significative des développements et conditionne une partie des coûts de fonctionnement de Focus d'autre part.

Considérant que les membres fondateurs de Focus ont un bagage technique élevée, le développement du Backend représente une opportunité à saisir de réduire nos coûts opérationnels.

Récolte des évènements utilisateur

Le backend mettra à disposition des différends deamon des APIs permettant de récolter des métriques liées aux usages des utilisateurs. Ces endpoints répondront à un protocole propriétaire. La communication s'effectuera via websocket. Le choix d'une communication websocket permet de limiter les différences de développement entre deamon et frontend via un protocole unique.

A la réception, ces métriques sont temporairement stockées dans un cache, puis récolté par un service qui va mettre à jour les métriques globales de productivité pour l'utilisateur concerné. Enfin, les métriques sont combinées par activité (toutes les métriques concernant une même activité) sont fusionnés en une seule. Celle-ci est alors stocké dans une base de données gelée.

Toutes les métriques seront préfixées par un entête comprenant :

- Un token d'identification (cf. Authentification)
- Le UUID du device émetteur
- Horodatage
- Un type de métrique

Sont définis différends type de métrique, chacun comprenant des informations spécifiques.

Changement de contexte

Ce type de métrique correspond à un changement d'activité par l'utilisateur. Par exemple, celui-ci vient de quitter une application, d'ouvrir un fichier, ou éteins son appareil.

Le Backend ne répond pas à ces métriques.

Nouvelle interruption

Quand le daemon détecte une interruption quelconque (mail, notification, etc.), il retient automatiquement et transmet l'information au Backend. Parmi les informations transmissent :

- Le UUID de l'émetteur (si applicable)
- L'application responsable de la nuisance
- Un set de déterminants (noms propres, sujets contenus dans le message, noms des pièces jointes, domaines des liens hypertextes, etc.)

Le Backend répond à ces métriques en donnant un UUID à cette nuisance. Par la suite, le Backend instruira les Daemons sur comment traiter cette nuisance via celui-ci.

Discrimination des nuisances

Quand le Backend est notifié par un Daemon de la présence d'une nuisance, le Backend contextualise celle-ci avec le dernier état connu de l'activité de l'utilisateur. Le contexte en question peut comporter les éléments suivants :

- L'application en cours d'utilisation.
- Les fichiers en cours d'utilisation ou éditées récemment.
- L'emploi du temps de l'utilisateur (si connectée).
- L'indice flottant d'interactivité courant (cf. poste analogue)

Sur la base de ces informations, le Backend va décider s'il est pertinent de décaler la réception de la nuisance (un temps maximum de décalage sera établi, permettant de mitiger les erreurs de traitement, a priori celui-ci sera *sui generis*).

Par exemple, un mail comprenant des mots clés présents dans le document en cours d'édition sera classé comme ayant une forte affinité avec le workflow en cours et sera délivré immédiatement. De même si son expéditeur est lié à un rendez-vous ayant lieu dans la journée de l'utilisateur.

Chaque décision prise par Focus en la matière sera consignée et les éléments de décisions connexes seront misent à disposition de l'utilisateur.

Machine Learning

À chaque décision prise par Focus concernant le traitement d'une nuisance est attachée un poids.

Dans l'historique des décisions concernant les nuisances passées, disponible sur son dashboard, l'utilisateur peut les marquer comme conforme ou non-conforme. Le marquage par l'utilisateur entraine une modification du poids de l'élément concerné.

À la prochaine réception d'une nuisance similaire, le Backend va prendre comme facteur les poids des décisions passées dans son choix. Ainsi, la discrimination des nuisances devient progressivement de plus en plus précise à mesure que les mauvaises décisions sont marquées comme telle par nos utilisateurs.

Ce fonctionnement est analogue à celui des filtre antispam. De la même manière, notre solution va tirer parti des feedbacks de tous les utilisateurs, de sorte que la solution devienne plus performante pour l'ensemble de nos utilisateurs sans requérir de chacun un investissement trop important.

Frontend Web et Mobile (applications)

Les Frontend Web et Mobile mettent à disposition de l'utilisateur les informations collectées par le Daemon au travers d'un Dashboard. Les challenges posés par les Frontend sont multiples. En effet, les données collectées et mises à disposition n'ont de valeur que si Focus arrive à les présenter de manière à leur donner un sens.

En outre, une attention particulière est portée à l'expérience utilisateur. Mettre en avant l'information utile, tout en laissant le pouvoir aux utilisateurs d'avoir accès à toutes ces données récoltées. De plus, il s'agit de la partie de Focus la plus exposée au regard et aux opinions des utilisateurs. En toute logique, c'est Focus tout entier qui sera jugé sûr cet élément.

Historique des activités

L'utilisateur pourra accéder à un historique de ses activités passées. Pour chaque activité, les informations suivantes seront disponibles :

- Horodatage
- Application utilisée
- Fichier édité (si applicable)
- Projet concerné
- Appareil
- Géolocalisation de l'événement

L'historique sera présenté via une vue calendrier. L'historique devra être à précision variable, notamment via :

- Un contrôle de l'échelle temporelle (de la vision sur une année à la semaine courante)
- Des filtres (par projet, par contexte, par clients)
- Un moteur de recherche

Le but étant de permettre à l'utilisateur de répondre à une question du type « Qu'est-ce que j'ai fait tel jour pour client X ? » en un minimum d'action utilisateur.

Indicateurs de productivité

A partir des métriques récoltées depuis les appareils de l'utilisateur, des métriques de productivités seront mises à disposition :

- Le nombre de mails dont la réception a été retardée
- La répartition des activités par projet, par application et par type de fichier
- Le nombre de pain-points recensés.
- L'indice flottant d'interactivité (cf. poste analogue)

Pour chaque métrique, les dérivés/évolution sur périodes donnée seront disponibles.

Daemon

Le Daemon est l'élément en charge de collecter toutes les données relatives à l'activité numérique de l'utilisateur. Il doit requérir le moins de configuration possible et opérer en autonomie complète. Il détecte et enregistre un certain nombre de métriques et d'événements générés par l'ordinateur de l'utilisateur. À partir de ces informations, le Daemon génère des méta-datas qui décrivent des comportements spécifiques correspondants à des scénarios préétablis (Ex. « l'utilisateur consulte ses mails », ou bien « L'utilisateur change très fréquemment de fenêtre »). Le Daemon se charge alors de rediriger cette information vers le Backend.

Connexion au Backend

Le Daemon est continuellement en communication avec le serveur via une liaison WebSocket perpétuelle.

Si aucune connexion n'est disponible, le Daemon est agilité à prendre des décisions en autonomie sur la base des réponses précédentes du Backend.

Détection du contexte d'utilisateur

En tirant parti des APIs misent à disposition par le développeur responsable de la plateforme sur laquelle il s'exécute, le Daemon va récolter une série d'informations sur l'utilisation qui est faite de l'appareil.

Parmi ces informations on trouve :

- Les horodatages de log
- Les applications ouvertes
- Les applications ayant le focus (si lieu)
- Les fichiers couramment ouverts
- Les sites web visités
- Les activités souri (mètres/minutes, clics/minutes)
- Les activités clavier (caractères par minutes)
- Configuration réseau (SSID wifi, proxy, mode de connectivité)

Régulièrement, le Daemon va transmettre au Backend l'état courant, comprenant ces informations.

Blocage des nuisances

Le Daemon va se connecter au service système des notifications et s'inscrire comme gestionnaire des notifications système. En ce comportant comme proxy, le Daemon bénéficie d'opportunité de traitement des notifications avant que celles-ci ne soient affiché à l'utilisateur.

Pour chaque notification, des métadonnées là concernant seront générées et transmises au Backend. Celui-ci va répondre en assignant un UUID à la nuisance. Dès que l'analyse de la nuisance a été

effectué par le serveur, celui va avertir de Daemon de la façon dont elle doit être traitée (livraison immédiate ou différé).

Le Daemon est chargé de stocker les nuisances en attendant d'avoir une réponse du Backend. En l'absence de réponse suivant un délais *sui generis*, la nuisance est immédiatement délivrée.

L'indice flottant d'interactivité

Le Daemon va également calculer continuellement un Indice d'Interactivité Utilisateur (IIU) qui va grossièrement quantifier l'intensité des interactions qu'entretient l'utilisateur avec son appareil.

Il prend en compte notamment :

- La fréquence des frappes clavier
- Le nombre de login/logout par heure
- La distance parcourue avec la souris
- Le nombre de changement de contexte par heure
- L'application courante

Le IIU va permettre au Backend de détecter des moments de temps morts de productivité (Ex. Sur le navigateur internet, peu de frappes clavier depuis 10 minutes). Ces moments sont des opportunités de modifier le traitement courant des nuisances et déclencher immédiatement leur livraison.

Présentation de l'environnement de réalisation

Environnement de réalisation

Les environnements de réalisations sont multiples UWP, MacOs, Linux, Android, iOS. Par conséquent les outils de développement le sont aussi.

Windows et l'une de nos plateformes principales, en effet, notre clientèle est majoritairement positionnée sur ce système d'exploitation, le support de ce dernier sera donc, dans un premier temps, notre priorité.

Le logiciel de gestion de version Git est utilisé, afin de profiter d'un système de branches efficace et évitant ainsi les conflits dans le code. Les dépôts Git sont hébergés sur le site Github, offrant des fonctionnalités annexes telles que les **Issues** ou le **Wiki**, proposant ainsi un espace clair pour discuter de différents points du projet, gérer les contributions et rédiger la documentation.

Aucun environnement de développement n'est imposé. Il est donc important que les dépôts restent vides de tous fichiers liés aux éditeurs de textes.

La majeure partie du projet est réalisée avec le langage C++, reconnu pour sa flexibilité et sa robustesse. Tout le code C++ produit doit être en respect avec nos Guidelines, imposant diverses règles qui amènent à un code propre et agréable à lire, et qui fera office de convention au sein des développeurs.

Dans la mesure du possible, le **deamon** doit être portable sur une majorité de plateformes (Windows, MacOS, Linux).

Afin d'assurer la documentation du projet, nous nous servons du site Read The Docs, et notre documentation est générée avec l'outil Sphinx.

Composants existants

Nous nous servons au sein de notre projet des bibliothèques et des API mises à disposition par les organisations gérant les services que Focus utilise, par le biais de ses **drivers**.

Ce choix est logique dans le sens où il nous permet d'utiliser des briques logicielles dont la qualité a été testée par les organisations elles-mêmes, ce qui nous en évite la maintenance.

Cependant, le choix d'user de ces bibliothèques reste libre et doit être raisonnable et ne pas aller à l'encontre des contraintes de portabilité.

Nous nous servons, pour la base de données de Focus, d'une interface C++ à Cassandra, qui est une base de données open source maintenue par Apache. Son principal intérêt pour notre application est qu'il s'agit d'un logiciel NoSQL, sans modèle relationnel, il est important pour nous d'avoir une base de données flexible pouvant répondre à plusieurs schémas différents.

Nous utiliserons les différentes API systèmes pour le développement du daemon afin d'être en mesure de collecter les informations nécessaires au fonctionnement de Focus à savoir Win32, X et Cocoa.

Enfin, pour la partie Web nous utiliserons des librairies et des framework connus, tel que Bootstrap pour sa grande flexibilité et la rapidité qu'il apportera au développement, D3.js pour gérer nos graphiques ainsi que JQuery et ReactJS.

Gestion de la sécurité

La gestion de la sécurité est présente à plusieurs endroits dans le cycle de vie d'une application.

La première chose qui vient à l'esprit est de sécuriser le logiciel lui-même. C'est souvent par des attaques lorsque le logiciel est en production que celui-ci est compromis. Mais ce n'est pas le seul risque. En effet, il faut prêter la même attention au processus de développement lui-même, la gestion des différentes ressources qui lui sont liées, et à la façon dont le projet va être maintenu.

Infrastructure & maintenance

Le processus de développement s'organise autour de Git et en particulier Github. Les membres du groupe d'EIP Focus ont un accès administrateur sur cette plateforme. Cela leur permet automatiquement de contribuer au code et d'utiliser les différents outils fournis par Github. De plus Git permet d'avoir une traçabilité des modifications du code car chaque modification est signée et authentifiée.

Le code source est hébergé sur les serveurs de Github, mais chaque développeur possède une copie locale du dépôt à tout instant, donc une perte de données de la part de l'hébergeur n'est aucunement dramatique.

Tous les contributions aux dépôts de Focus se font via les **pull-requests**. Il s'agit d'une demande d'intégration au dépôt du code développé par une personne. Si une **pull-request** est faite, le code sera audité pour s'assurer de sa qualité, mais aussi pour vérifier qu'il n'enlève rien à l'efficacité des solutions déjà en place.

Processus de développement

La majeure partie du projet est développée en C++. De ce fait, il nous faudra apporter une certaine rigueur au cours du développement afin de prévenir les vulnérabilités les plus classiques que l'on rencontre dans des projets développés avec des langages de bas niveau, notamment concernant la gestion de la mémoire qui est à la charge du développeur.

Les interactions avec les bases de données et les différents protocoles sont faits au travers de bibliothèques proposant un usage simple et moins propice aux erreurs qu'un accès direct aux ressources.

Cependant, un certain nombre de vulnérabilités ne peuvent être évitées que par du bon sens et un respect des meilleures pratiques. C'est notamment le cas pour tout ce qui concerne les fonctions qui permettent d'accéder aux ressources système. L'appel à ces fonctions devra donc être particulièrement surveillé et justifié, plus encore lorsqu'elles peuvent être influencées par des entrées utilisateurs.

Des phases d'audit seront organisées, où l'ensemble du code sera vérifié par des développeurs n'ayant pas travaillé sur ces parties, de façon à avoir un deuxième avis objectif sur la question de la sécurité.

Sécurité vis-à-vis de l'utilisation de Focus

La nature des informations que Focus va collecter sur ces utilisateurs nous pousse à donner une place centrale à la sécurité au sein de notre projet. En effet, les données et les métadonnées que nos outils vont collecter sont de nature sensible, il n'est pas envisageable que quiconque en dehors de l'utilisateur lui-même ait accès à ces données. Notre politique de confidentialité se devra d'être à la hauteur des attentes de nos utilisateurs. Tous les échanges entre nos services (backend) et les utilisateurs finaux (client desktop) seront bien évidements cryptés de manière sûre.

Points sensibles

Focus doit être réalisé dans un délai de deux ans et demi. Pendant cette période, plusieurs points sensibles sont à surveiller.

Le premier concerne la portabilité. En effet, cela entraîne de nombreux tests sur un maximum de plateformes possibles (UWP, MacOs, Linux, Android, iOS), et le risque de devoir se passer d'une bibliothèque ou d'une fonctionnalité car elle n'est pas supportée partout. Si une plateforme pose trop de problèmes et n'est pas utilisée par beaucoup d'utilisateurs, l'abandon de son support est à envisager.

Le second point sensible concerne les services externes utilisés par Focus. L'intérêt de Focus est d'autant plus grand que l'application est capable d'interagir avec un grand nombre de services différents.

Mais ces services étant en grande majorité hors de notre contrôle, la fermeture d'un service est toujours une possibilité à ne pas exclure. La fermeture d'un service rendu compatible avec Focus est donc problématique, dans le sens où le travail fourni pour communiquer avec ce service devient obsolète, et que l'intérêt général de Focus diminue en conséquence.

De manière générale, le développement d'un service tel que Focus est toujours un sujet délicat à étudier avec attention. De nombreux tests de différentes catégories doivent être effectués régulièrement, et les retours des utilisateurs doivent être écoutés et pris en compte.

Organisation projet

Notre groupe d'EIP est composé de 6 membres. Chacun dispose d'un tronc commun de compétences propre à l'ensemble du groupe, les principales étant la gestion de projet et le développement C++. Parallèlement à cela, chaque membre dispose de compétences qui lui sont propres et pourra ainsi prendre la responsabilité d'une partie du projet. Pour être plus clair, prenons l'exemple de Robin CHARPENTIER : ce dernier dispose de compétences dans le développement d'applications mobiles, c'est pourquoi c'est lui qui est chargé de cette partie du projet. Il lui appartient de découper sa partie en petites briques fonctionnelles (voir WBS/Gantt) et de rédiger un certain nombre de documentations appelées Guideline permettant à n'importe quel autre membre du groupe de le rejoindre sur le développement de sa partie.

Le projet d'EIP nous semble être parfait pour appliquer des méthodes de travail agiles. Le découpage en partie fonctionnelles (voir WBS) est un premier travail qui nous a permis de mettre en place un tableau agile. Le but premier de cette organisation et de travailler en sprints courts de 2 à 3 semaines et de pouvoir rebondir plus rapidement et plus facilement en cas de problèmes. Chaque « chef de projet » (responsable d'une partie du développement) aura à sa charge la gestion et la mise à jour continue du tableau agile de sa partie.

Nous avons mis en place un certain nombre d'outils afin de faciliter et d'améliorer les conditions de réalisations de notre projet. Les plus importants sont :

- Le Discord (logiciel de communications vocales et écrites comparable à Slack)
- Le Cloud permettant de partager des fichiers entre nous, gérer nos tableaux Kanban, nos mails, nos calendriers, nos rappels, etc.

De plus, nous avons mis en place au sein du groupe un « processus de résolutions de conflit » afin que chacun puisse remettre en cause les actions d'un autre membre, cela dans le but d'éviter les futurs problèmes et tensions inutiles qui pourraient nuire au bon déroulé du projet.

Annexes

Le nom de domaine **focus.sh** a été dors est déjà réservé. Un ensemble de sous-domaine a été mis en place afin de rediriger correctement nos utilisateurs.

- **focus.sh** : accès à la landing page de Focus.
- **dashboard.focus.sh** : permet d'accéder à son profil et de retrouver toutes ses informations.
- **api.focus.sh** : l'Endpoind principal permettant aux daemon et Frontend de communiquer avec le Backend.
- **cloud.focus.sh** : permet à l'équipe de retrouver tout le contenu partagé au sein du groupe (fichiers, calendrier, rappels, tableau agile, etc.)
- **mail.focus.sh** : serveur mail mis en place pour que l'équipe ai des adresses email à leur disposition.

D'autres viendront s'ajouter à la liste en fonction des besoins futurs encore non-envisagés.