Rapport de stage

Société Prototypo

Evolution d'une application web d'édition typographique : Prototypo

Martin BOLOT

Période du 13 Juin au 26 août 2016

Diplôme préparé : DUT Informatique en Année Spéciale – IUT Lyon1

Responsable pédagogique : Mme Ariane Baron

Maître de stage : M. Louis-Rémi Babé

Prototypo, 25 Cours Albert Thomas, 69003 Lyon

IUT Lyon 1 - Site de Villeurbanne Doua1, rue de la Technologie, 69622 Villeurbanne Cedex

Fiche technique

Nom de l'entreprise : Prototypo

Activité de l'entreprise : Développement d'une application web d'édition typographique

Sujet : Aide à l'amélioration de l'interface utilisateur

Description :
Le travail effectué répond au besoin d'évolution de l'application Prototypo. Une refonte de l'interface est effectuée dans le but de mieux correspondre aux attentes des utilisateurs et doit être mise en production durant la première partie du mois d'août 2016, ce stage s'inscrivant dans ce projet d'évolution. Tous les utilisateurs possédant un compte sur l'application web Prototypo pourront bénéficier de la nouvelle version de cette dernière.

L'entreprise compte deux informaticiens : Louis-Rémi Babé, directeur technique et François Poizat, ingénieur informaticien.

Le stage s'est déroulé en autonomie mais avec l'aide constante des deux informaticiens de l'entreprise. Le projet global est commun à toute l'entreprise, à savoir le développement de l'application Prototypo, mais les tâches assignées sont effectuées de manière individuelle.

Le stage a été effectué à l'aide d'un ordinateur portable de type MacBook Pro, avec l'éditeur de code Atom. Le langage majoritairement utilisé a été le JavaScript, HTML et SCSS ont été employés dans une moindre mesure. J'étais déjà très familier avec le JavaScript donc aucun langage n'a dû être appris au cours de ce stage, seulement l'utilisation de certaines bibliothèques telles que React ou PaperJS.

A la fin du stage, toutes les tâches confiées ont pu être terminées. Une journée a été consacrée, au cours du stage, à la rédaction du rapport de stage. De nombreuses notes ont été prises durant l'exercice.

Mots-clés : Prototypo, Typographie, JavaScript, React, HTML, SCSS, Application web, User Interface, UX Design

Remerciements

Louis-Rémi Babé (Maître de stage, directeur technique) : Pour son accompagnement, le partage de ses connaissances en JavaScript et pour son suivi

François Poizat (Développeur) : Pour sa patience et les très nombreux conseils, méthodes et savoirs transmis au cours de mon stage

Yannick Mathey (Directeur artistique) : Pour les nombreuses informations transmises sur la typographie et pour son didactisme

Yann Guillet (Directeur marketing) : Pour son assistance face aux tâches orientées SEO et ses nombreux retours concernant les modules développés

Pierre Poizat (Directeur de l'incubateur Jean Moulin) : Pour sa bienveillance et l'accueil réservé au sein de l'incubateur

Ariane Baron (Responsable pédagogique) : Pour son suivi lors du stage et durant la période précédant la soutenance de stage

Pierre-Antoine Champin (Enseignant tuteur) : Pour son suivi et ses conseils au cours du projet tuteuré m'ayant aidé à obtenir mon stage de fin d'année

Sommaire

Après avoir travaillé plusieurs années dans le développement de sites web, l'année spéciale en IUT Informatique a été le moyen pour moi d'obtenir des bases solides et essentielles en programmation, de la manière la plus généraliste possible afin de pouvoir par la suite me spécialiser dans le domaine de l'informatique le plus approprié par rapport à mon profil. Le projet tuteuré était, quant à lui, l'occasion de se lancer en groupe dans une réalisation plus personnelle en plongeant dans une technologie particulière, le JavaScript pour notre cas dans le cadre du développement d'une application web d'édition d'automates à états finis. Ayant beaucoup appris de cette expérience, j'ai choisi lors de mes recherches de stage de cibler ce domaine précis. Je suis ainsi entré en contact avec Prototypo, une société développant une application web d'édition de fontes typographiques dont les grands concepts de fonctionnement étaient voisins de ceux de mon projet d'étude. La mission qui m'a été confiée, une fois ma candidature acceptée, était l'aide à l'évolution de l'application en elle-même, Prototypo. Cela représentait pour moi la promesse d'une plongée en profondeur dans la gestion d'une interface client complexe et l'approfondissement de mes connaissances en JavaScript via l'utilisation d'une bibliothèque de gestion d'interface utilisateur : React. J'ai été pendant trois ans employé en tant que développeur web dans une entreprise de type PME comptant dans son pôle web une vingtaine d'employés, il s'agissait pourtant de ma première expérience dans une entreprise de type « start-up ». Prototypo n'était formée à mon arrivée que depuis un an et constituée de quatre employés. J'allais être amené à observer de près la vie dans des entreprises composées d'équipes de taille très réduite car Prototypo est hébergée au moment de mon stage dans l'incubateur Jean Moulin regroupant une dizaine de start-ups.

Par la suite, nous allons voir dans un premier temps comment Prototypo, mon entreprise d'accueil pour ce stage, travaille en utilisant tous les outils les plus récents afin de proposer l'expérience utilisateur la plus riche possible. Puis, dans un second temps, il s'agira de définir les différentes tâches qui m'ont amené à l'amélioration de l'expérience des utilisateurs de l'application mais aussi celle de ses administrateurs. Enfin, dans une troisième partie, nous dresserons un bilan de cette expérience de stage en développement d'application web où des règles de développement strictes étaient en place pour fournir à l'utilisateur un outil de création le plus performant et intuitif possible.

I) Prototypo, une entreprise encore jeune qui profite des outils les plus récents

a) Présentation de l'entreprise

Prototypo est une entreprise fondée par Louis-Rémi Babé et Yannick Mathey, développant principalement une application web éponyme, Prototypo, ayant pour but de permettre l'édition de polices de caractères à l'aide d'une interface utilisateur pensée pour les designers. Cette application propose, de base, des fontes de caractères prédéfinies afin de faciliter la phase de création d'une police.

Le projet est né de la rencontre entre Yannick Mathey, designer passionné de typographie et Louis-Rémi Babé, développeur dans le domaine des technologies web et plus particulièrement dans le JavaScript, langage initialement prévu pour les navigateurs internet et leur interaction avec l'utilisateur. La problématique constatée par les deux futurs entrepreneurs de Prototypo au moment de la création du projet était l'absence de tout logiciel dédié à la création de fonte paramétrique. Les solutions possibles pour ce problème à ce moment-là se divisent principalement en deux catégories, à savoir pour la première l'utilisation d'un logiciel de dessin, laissant le champ libre à toute forme de création mais se révélant extrêmement fastidieux en raison de la nature fragmentée d'une fonte, composée de très nombreuses glyphes pour beaucoup interdépendantes. La seconde option consistait en l'utilisation de logiciels permettant l'élaboration de scripts destinés à alléger ce processus de création. Cette méthode nécessitait toutefois des compétences techniques élevées à la fois en design et en informatique, la réservant à une minorité de designers.

A donc germé l'idée d'un outil prévu pour les navigateurs qui permettrait aux graphistes de créer leurs propres polices de caractères, le but étant d'offrir de l'aide et de la simplicité sans pour autant brider l'utilisateur. En s'appuyant à la fois sur ce concept et sur les précédents travaux d'étude de Yannick Mathey, un prototype de logiciel d'édition paramétrique de polices de caractères, l'application Prototypo allait entrer en développement.

S'en est suivie la création d'un appel à la levée de fonds grâce à la plateforme de financement participatif Kickstarter, une entreprise qui propose la mise en relation d'un créateur directement avec les personnes souhaitant participer à la fondation de son projet. Ainsi, chaque individu intéressé peut apporter son aide financière à un projet afin qu'il voie le jour. Un objectif de somme à atteindre doit être fixé à l'avance par les créateurs du

projet, qui proposent la plupart du temps une compensation pour les personnes ayant participé au financement, avec très souvent différents paliers pour différentes sommes, représentant des compensations en accord avec l'argent investi. Le projet est parvenu en mai 2014 à réunir 34 133£ soit plus de 40 000€ à l'époque, pour un but initial de 12 000£, grâce à la participation de 1 670 contributeurs. Un donateur ayant dépensé 9£ ou plus se voyait garantir un an d'accès à la version payante de l'application tandis que donner 20£ ou plus permettait d'obtenir au choix deux ans d'abonnement ou un an d'abonnement accompagné d'une autre année d'abonnement à offrir. Les fonds récoltés ont permis de financer la première année de développement de l'application dont la conception avait déjà commencé avant la levée de fond Kickstarter. Le fait que les sommes aient été récoltées en livres sterling s'explique par l'absence, au moment de la levée de fonds, de branche française pour Kickstarter.

L'équipe de Prototypo s'est agrandie depuis la création de l'entreprise et du projet. Louis-Rémi Babé est actuellement directeur technique et s'occupe à la fois de la partie serveur de l'application tout en assurant son rôle de chef d'entreprise, tandis que Yannick Mathey est le directeur artistique de Prototypo, travaillant sur le design des fontes créées par l'entreprise, de leur intégration dans l'application mais également sur l'interface et l'ergonomie de Prototypo tout en étant co-gérant de l'entreprise avec toutes les tâches administratives et relationnelles que cela comprend.

Les deux fondateurs du projet ont été rejoints en avril 2015 par François Poizat, développeur spécialisé dans les technologies web front-end, à savoir HTML et CSS pour la couche de présentation, JavaScript pour la partie interactive des interfaces navigateur. Ses compétences ne se limitent toutefois pas au domaine du web, son parcours d'ingénieur en informatique à l'INSA fait de lui un informaticien polyvalent notamment habitué aux tâches liées à la gestion de projet telles que l'organisation de réunions autour de la technique et les estimations de temps de développement. Il a acquis avec le développement de Prototypo une maîtrise avancée de la bibliothèque React, permettant de développer plus facilement des interfaces web complexes, ainsi que de Flux, le concept de gestion d'état d'application web en lien avec React. Au mois de mars de l'année 2016, l'équipe s'est enrichie d'un nouveau collaborateur, Yann Guillet, au poste de directeur marketing. Il est en charge de la gestion du référencement web SEO (Search Engine Optimisation) de l'entreprise, mais aussi de la gestion de tous les canaux de communication dont dispose Prototypo pour se faire connaitre dans l'optique d'obtenir de nouveaux abonnés avec une place prépondérante des réseaux sociaux dans cette démarche. Dans le même registre, le SEA (Search Engine Advertisement) est un domaine qui lui permet d'effectuer de l'acquisition de trafic pour le

site internet et l'application de l'entreprise. C'est également à lui qu'incombe, en collaboration avec les dirigeants, la responsabilité d'établir les stratégies de communication : périodes pendant lesquelles communiquer le plus efficacement, fréquence des envois d'emails d'information par exemple.

Après avoir travaillé directement chez Yannick Mathey lorsque l'entreprise ne comptait que deux personnes, Prototypo s'est installée dans l'incubateur Jean Moulin début 2015. L'incubateur Jean Moulin accueille des start-ups au début de leur vie d'entreprise, selon certains critères de sélection comprenant la pertinence du projet présenté et ses chances de réussite, afin de fournir des locaux aux employés de ces petites structures, mais aussi des conseils et un accompagnement stratégique pour les créateurs d'entreprise. Cet incubateur dépend de l'Université Jean Moulin Lyon 3 et est divisé en deux parties : une partie « start » pour les entreprises tout juste créées, une partie « up » pour les projets ayant plus d'ancienneté. Prototypo fait partie de la section « up » de l'incubateur au moment de mon stage. Toutefois, l'incubateur n'offre l'hospice aux start-ups que l'espace d'un certain laps de temps et Prototypo devra à partir de 2017 disposer de ses propres locaux, l'entreprise arrivant au terme de sa période d'incubation.

Prototypo a doté son application d'un modèle économique qui a connu à partir d'août 2016 un changement majeur. Au lancement de l'application, Prototypo proposait à n'importe-quel utilisateur de pouvoir créer un compte gratuitement. Il obtenait alors l'accès à la version gratuite de l'application. Toutes les fonctionnalités d'édition de la police de caractères choisie étaient présentes à l'écran mais une partie d'entre elles était grisée et n'était pas fonctionnelle. Pour y avoir accès, il fallait souscrire à un compte payant, ce qui était possible soit directement depuis l'application, soit depuis le site internet de Prototypo. Différentes offres sont disponibles pour les comptes payants mais fonctionnant dans tous les cas par abonnement, au mois ou à l'année. Après avoir pris en compte plusieurs retours d'utilisateurs, il s'est avéré que la plupart des utilisateurs n'ayant pas un compte payant préféraient avoir accès à l'édition de tous les paramètres d'une police quitte à être bridés d'une autre manière. Ainsi, il a été décidé de laisser l'intégralité de l'application en libre-service pour les utilisateurs avec compte gratuit, en exigeant en revanche un paiement lors de l'export d'une fonte. L'application permet, en effet, l'export de la fonte de caractères éditée pour pouvoir être utilisée hors de l'application, sous différents formats. Depuis la mise à jour du 10 août 2016 (version 1.2), l'export de fonte est donc payant pour les utilisateurs classiques grâce à l'introduction d'un système de crédits où chaque crédit acheté permet d'effectuer un export. L'utilisateur disposant d'un abonnement payant ne verra

aucun changement concernant les fonctionnalités qu'il peut utiliser, l'export restant gratuit et illimité.

Le public ciblé par Prototypo est principalement constitué de la branche de métiers liés au design, car les designers sont souvent concernés par l'utilisation de typographies propriétaires. Par extension, les écoles d'art et de design sont une cible de choix car elles sont susceptibles de chercher des outils à but éducatif, rôle que Prototypo se prédestine à pouvoir remplir, mais aussi parce qu'elles peuvent apporter via leurs étudiants de nouveaux utilisateurs potentiels. Ainsi, des tarifs spéciaux pour les groupes de type école étant prêts à acquérir plusieurs abonnements sont régulièrement discutés avec ces clients potentiels. Pour toute entreprise développant un outil web se pose la question de la compatibilité qui va être effectuée avec les différents navigateurs : cela peut aller de pair avec le public que l'on souhaite cibler. La ligne directrice de ce point de vue pour Prototypo a été de ne cibler que les dernières versions des deux navigateurs les plus utilisés : Google Chrome et Firefox. Cela peut se justifier par le fait que l'application nécessite une gestion optimale des ressources du navigateur, car elle a besoin de calculer en temps réel des modifications sur un jeu de glyphes complet. Editer l'apparence d'une lettre se répercutera immédiatement sur l'aspect de tout le jeu de caractères proposé par la fonte. De plus, les technologies utilisées pour le développement de l'application produisent un code nécessitant dans certains cas l'utilisation d'un navigateur récent. La compatibilité avec des versions plus anciennes des différents navigateurs n'est pas envisagée actuellement, la compatibilité avec le navigateur Opera est déjà très avancée et une version fonctionnant avec Safari est en cours de développement, même si ce n'est pas sur ce point que les priorités de l'équipe sont portées.

Nous avons précédemment décrit le modèle économique de l'application comprenant deux types d'accès, payant sur abonnement ou alors gratuit. La distinction des deux offres se fait uniquement par rapport au compte de l'utilisateur. En effet, la version de l'application est la même pour tout le monde et s'affichera de manière adaptative en fonction de l'utilisateur qui s'y connectera. Ce fonctionnement est en adéquation avec le modèle de gestion du code source de l'application. Celle-ci est entièrement en open-source et l'intégralité du code est accessible depuis GitHub. D'après Louis-Rémi Babé, co-fondateur et directeur technique étant responsable de la gestion des sources de l'application, cela pourrait bientôt ne plus être le cas. Selon lui, mais aussi selon l'équipe de développement, les bienfaits liés à l'accès libre aux sources par le public ne sont pas assez significatifs pour se priver de ceux liés à un code source privé. Pour l'instant, tout utilisateur possédant un compte GitHub peut faire

remonter des dysfonctionnements directement à Prototypo, ou encore proposer ses propres améliorations du code source via des requêtes d'incrémentation de code.

b) Technologies web mises en œuvre

Afin de pouvoir offrir une interface accessible à tous, Prototypo a été pensée et développée en tant qu'application web. Bien qu'il n'y ait pas de définition exacte de l'expression « application web », nous allons décrire le sens qu'elle prendra tout au long de ce rapport. Une application web ici désigne une page web unique qui va offrir à l'utilisateur une expérience de navigation sans aucun rechargement, mettant en avant l'interaction avec l'utilisateur et proposant des fonctionnalités que l'on retrouve habituellement dans un logiciel. Les pages sont composées de balises HTML qui structurent le document et définissent ses différentes parties (en-tête, corps, pied de page, éléments de navigation) et leur apparence est dictée par des règles CSS qui définissent l'affichage de chacun des éléments du document. Un site web classique se basera sur une architecture client-serveur où le client, représenté par l'utilisateur, se connecte au serveur en tapant l'URL du site dans la barre d'adresse de son navigateur puis navigue à travers les différentes pages en se connectant autant de fois que nécessaire à ce même serveur. Dans le cadre d'une application web comme Prototypo, le client ne se connectera qu'une seule fois au serveur de manière explicite, recevra toutes les informations nécessaires à sa navigation et à son interaction et la page web continuera son cycle de vie côté client de manière autonome jusqu'à la déconnexion de l'utilisateur. Typiquement, l'application Prototypo profite de son écran de chargement pour récupérer toutes les informations nécessaires en faisant patienter l'internaute, puis lui permet d'éditer librement sa fonte de caractères. La finalité de ce procédé est de proposer une interface proche d'un programme que l'on aurait installé sur son bureau, sans avoir à se soucier d'installer quoi que ce soit, uniquement en se connectant à un site internet depuis son navigateur.

Toutefois, l'utilisation d'un troisième langage est nécessaire à une application web. En effet, dans le schéma applicatif que nous avons ébauché plus haut, l'utilisateur n'a pas à demander au serveur de faire des calculs lorsqu'il effectue une action. Chaque interaction avec l'utilisateur va nécessiter, pour être interprétée, un calcul qui peut être fourni soit de la part du serveur, soit de la part du navigateur. L'option du serveur est à exclure ici ; pour un service qui souhaite proposer de l'interactivité, il serait impensable d'aller se connecter, en générant un temps de chargement, à chaque clic sur l'interface. C'est donc le navigateur qui va se charger de la majeure partie des calculs, vitaux pour l'application. Pour ce faire, Prototypo, de même que la majorité des applications web en la quasi-absence d'autres

options, utilise le langage JavaScript dont tout le cycle de vie est prévu pour se dérouler dans le navigateur.

Ce langage est au cœur de l'application car c'est lui qui permettra d'afficher, de modifier et d'exporter les polices de caractères dans Prototypo. Il s'avère qu'il est également présent du côté du serveur mais il s'agit plus, à ce niveau, du reflet des préférences de l'équipe de développement que d'un réel besoin, à l'inverse du navigateur où il est essentiel.

Le contexte technique de ce stage est donc presque exclusivement basé sur le JavaScript. Cependant, plus que du JavaScript traditionnel, Prototypo utilise un outil particulier nommé React. Si JavaScript offre des fonctionnalités événementielles permettant de gérer l'interface entre un utilisateur internaute et un navigateur, il n'en reste pas moins un langage de programmation informatique : il s'adresse à des informaticiens désireux d'exécuter des calculs et des algorithmes, mais n'a aucune inclinaison particulière. Créer des interfaces web complexes est l'un des buts qui peuvent être atteints grâce à une utilisation ingénieuse et spécifique du langage, mais cela implique une quantité de code potentiellement très élevée. De ce constat est né React, une bibliothèque de fonctions, c'est-à-dire un ensemble de fragments de code prêts à l'emploi qui se destine à aider les développeurs dans un but précis, ici créer des interfaces utilisateurs simples et robustes.

React est une bibliothèque sous licence BSD, ce qui signifie que toute entreprise, à but lucratif ou non, est autorisée à l'utiliser librement. Cet outil est le fruit des recherches et investissements de la société Facebook, basée aux Etats-Unis. Cette société étant un acteur majeur du web avec un poids économique très fort, la maintenance du logiciel, utilisé en interne par la firme américaine, a de forte probabilité d'être rigoureuse et de qualité. Cela en a fait un choix en tant qu'outil de développement validé par l'équipe de Prototypo, malgré le fait que React est encore actuellement à l'état de Beta et donc pas officiellement en production bien que très stable. L'intégralité de Prototypo reposant sur React, il était primordial à mon arrivée de maîtriser au plus vite les fonctionnalités proposées par cet outil. Le stage s'est effectué grâce à une machine ayant comme système d'exploitation OS X d'Apple. L'éditeur de code utilisé durant cette période était Atom, un éditeur permettant d'écrire du code dans n'importe quel langage mais spécialisé dans les langages web et plus particulièrement le JavaScript. Ce logiciel, écrit lui-même en JavaScript, fonctionne par briques que l'on peut ajouter à l'envi, souvent téléchargées depuis internet et apportant de nouvelles fonctionnalités à l'éditeur. Ces briques sont la plupart du temps développées par des particuliers et la communauté est très active ce qui permet d'avoir accès à un grand nombre d'outils d'aide à la conception. En particulier, l'un de ces modules permet une utilisation du JSX, format propre aux utilisateurs de React, qui m'a été d'une grande aide.

Comme beaucoup de langages informatiques, JavaScript est régulièrement mis à jour, via un système de spécifications correspondant peu ou prou à un système de version. Ainsi, après être resté plusieurs années sur la spécification ES5 (pour ECMAScript 5), JavaScript est sur le point de passer à ES6 et plusieurs outils permettent aux développeurs de commencer à écrire leur code selon ces nouvelles spécifications avant qu'elles ne soient officiellement disponibles, en traduisant simplement le code produit vers une version ES5 équivalente et donc compatible avec tous les navigateurs. Le code source de Prototypo est dans sa majeure partie écrit grâce à ES6, la dernière syntaxe disponible pour JavaScript. Le choix de cette norme a en partie été motivé par la lisibilité plus grande de ce type de code, par conséquent plus facile à maintenir. Pour un développeur ne connaissant pas toutes les subtilités de cette nouvelle version, cela ajoute une difficulté supplémentaire lorsqu'il découvre le code, mais ce dernier est plus concis et donc plus accessible à moyen terme. Un outil est inclus dans le projet permettant de souligner les éventuelles erreurs de syntaxe dans l'éditeur de texte de chacun, les règles étant les mêmes pour tous les intervenants. Cela permet à la fois d'apprendre aux nouveaux venus comment écrire du code sans erreur, mais aussi à l'ensemble des participants de respecter des règles propres au projet et définies de concert par l'équipe de développement.

Le nombre important de technologies très récentes présentes rend donc le projet assez difficile d'accès pour un stagiaire qui n'aurait pas eu d'expérience dans le web. Me concernant, les années de travail dans le milieu du web ont apporté un confort lors de la phase d'adaptation au travail avec l'application, même si ce sont principalement les projets personnels développés en JavaScript qui m'ont aidé lors de mon entrée en stage. En effet, ceux-ci m'ont permis d'obtenir des bases solides dans ce langage, nécessaires à l'appréhension du flot d'informations inhérent à mon arrivée sur le projet d'évolution de l'application Prototypo. Le système de composants au cœur de la philosophie de React ne m'était pas inconnu, de même que certaines des nouvelles normes d'écriture du JavaScript. La taille réduite de l'équipe, ainsi que l'organisation rigoureuse de la structure de l'application m'ont largement aidé dans mon assimilation du processus de développement propre à Prototypo. L'application est découpée en deux parties distinctes, l'une regroupant les scripts contenant la logique du code, l'autre les feuilles de styles. Chaque dossier comprend des utilitaires rangés selon leur fonction ainsi qu'un dossier reprenant l'ensemble des composants de l'application. Le nombre de fichiers de composants avoisinait la cinquantaine au moment de mon arrivée, mais du fait de leur découpage reprenant la logique de construction visuelle de l'application, même un néophyte peut facilement s'y repérer.

Bien qu'ayant des connaissances en développement web suffisantes pour pouvoir effectuer mon stage dans de bonnes conditions et apporter un véritable soutien à l'équipe de développement, mes connaissances en matière de typographie à mon arrivée chez Prototypo étaient extrêmement basiques. Malgré le fait que ce type de savoir n'ait pas été nécessaire pour moi en tant que stagiaire en développement informatique, les grands concepts reposant derrière la création d'une police de caractères sont un atout pour travailler avec Prototypo, une fois appréhendés. Les paramètres permettant de définir le comportement d'affichage des fontes sont définis dans plusieurs projets séparés du projet principal et majoritairement gérés par le directeur artistique Yannick Mathey. Ceux-ci sont écrits dans un langage dérivé du JavaScript, le CoffeeScript, et demandent de nombreuses explications pour être compris. Ces projets sont appelés des gabarits, ils sont au nombre de trois et permettent le fonctionnement des trois fontes pré-écrites proposées de base dans Prototypo. Elles se nomment Elzevir, John-Fell et Grotesk, possèdent chacune un style qui leur est propre et certaines propriétés exclusives par rapport aux autres gabarits. Chaque utilisateur possède relativement à son compte une collection de familles. Les familles sont un regroupement de fontes de caractères ayant la même base, justifiant l'utilisation originale du terme « police » de caractères, à savoir l'accumulation de plusieurs jeux de caractères. On emploie aujourd'hui indistinctement fontes de caractères et polices de caractères par abus de langage et à but de simplification, comme ce sera le cas dans ce rapport. Quand on descend dans la hiérarchie, chaque famille de la collection d'un utilisateur possède un certain nombre (une au minimum) de variantes. Exemple concret, Verdana Bold est une variante de la famille Verdana, qui se distinguera par sa graisse. Prototypo permet d'avoir un nombre illimité de variantes pour chaque famille typographique.

Enfin, il a fallu dans le contexte de ce stage se familiariser avec une autre bibliothèque de code, PaperJS. Cet outil propose, à l'instar de React, des fonctions prêtes à l'emploi à destination des développeurs web. Le domaine d'action de cette bibliothèque est le dessin vectoriel, c'est-à-dire le dessin de formes géométriques sur un plan 2D nommé « canvas » dans le contexte du navigateur. Ce dernier sert dans Prototypo à afficher la prévisualisation des lettres pour l'utilisateur. Celui-ci peut observer en temps réel sur une lettre précise, ainsi que sur un jeu de caractères, les répercussions des modifications qu'il effectue grâce aux boutons proposés. A chacune des modifications qu'il effectue, le plan en deux dimensions, représenté par le canvas est intégralement redessiné pour correspondre à la nouvelle apparence de la fonte. En vue d'une mission ultérieure liée au rendu des fontes et à leur export, il a donc fallu plonger dans la documentation, heureusement particulièrement complète et didactique, de cet outil jusque-là inconnu.

II) Amélioration constante de l'expérience utilisateur et administrateur

On pourra regrouper toutes les missions qui m'ont été confiées, diverses dans leur nature, sous un même but : améliorer l'expérience utilisateur et, accessoirement, celle des administrateurs, à savoir l'équipe design et marketing. Mon arrivée en stage s'est effectuée au moment où le lancement de la deuxième version de l'application s'apprêtait à être réalisé. Ainsi, cette période coïncidait avec le développement des derniers modules manquants par rapport à ce qui avait été planifié pour la refonte de l'interface utilisateur, décidée en fonction de retours et conseils aussi bien internes qu'externes.

a) Evolution d'une extension Google Chrome

Ma première mission s'est effectuée en dehors du contexte même de l'application. Cela constituait une bonne entrée en matière car la première étape, avant de pouvoir travailler sur un quelconque projet, était de prendre connaissance des méthodes de travail concernant la gestion de version du code source. Le code de l'application, mais aussi de chaque projet de l'entreprise, est stocké sur GitHub dans un répertoire commun. Afin de pouvoir travailler sur les fichiers hébergés dans ce répertoire, il a fallu appliquer les différentes méthodes propres à l'entreprise pour premièrement récupérer les fichiers, ensuite créer sa branche de travail et enfin apprendre comment envoyer vers le serveur son propre travail dans le but de le soumettre à la validation du référent technique. La méthodologie concernant la gestion de version du code m'a été transmise par François Poizat mais sa mise en place a été effectuée après une mise en commun des compétences à ce sujet avec Louis-Rémi Babé, le directeur technique, en tirant inspiration des grandes entreprises du web telles que Facebook ou Mozilla.

Le but de la mission était l'amélioration de l'extension Google Chrome dédiée à l'application Prototypo. Une extension pour Google Chrome est un module pouvant s'installer dans le navigateur développé par Google et ajoutant de nouvelles fonctionnalités au navigateur au-delà de celles prévues par Google. La firme à l'origine du navigateur fournit en effet un ensemble de fonctions, que l'on regroupe sous l'acronyme API, donnant au développeur l'accès aux pages visitées par l'utilisateur. Le projet avait été développé initialement par François Poizat et permettait de visualiser dans n'importe quelle page web, sous réserve d'avoir installé ladite extension sur son navigateur, une fonte éditée dans l'application Prototypo. Un onglet contenant l'application en état de marche est lui aussi nécessaire pour que l'extension fonctionne. L'extension tire alors partie de sa communication avec l'application pour pouvoir pré-visualiser directement dans une page web la police de

caractères que l'utilisateur est en train de créer, où les modifications effectuées sont appliquées en temps réel sur la page sélectionnée. La seule action requise de la part de l'utilisateur pour mener cette tâche à bien étant la sélection du texte qu'il souhaite voir mis à jour. Un exemple d'utilisation serait d'appliquer sa fonte personnalisée sur tous les paragraphes d'une page du site du « Monde ». Un moteur de sélection étant présent dans l'extension, il est possible de sélectionner tous les éléments de la page, tels que le titre, un ou plusieurs paragraphes, le pied de page. La sélection peut se faire manuellement grâce à un ciblage à la souris, ou bien grâce à du code directement inséré dans un champ de texte prévu à cet effet.

L'enjeu derrière l'évolution de cette extension Google Chrome comprenait une amélioration de la présentation de l'extension afin qu'elle se positionne dans la zone offerte par Google au sein du navigateur, plus conforme aux normes et plus accessible que précédemment, car elle se trouvait directement incluse dans la page web, pouvant ainsi gêner la navigation de l'internaute. La technique utilisée consistait en effet à injecter du code HTML dans la page de l'utilisateur, alors que l'utilisation de la fenêtre « pop-up » proposée nativement par Google permet d'utiliser un espace dédié, dans un document HTML séparé, pour la présentation de l'interface de notre extension. De plus, cette mise à jour visait à se conformer au mieux à la dernière version et aux dernières recommandations de l'API Google Chrome, en utilisant les fonctions recommandées par Google. Si du temps avait été prévu pour la refonte du code de l'extension malgré le fait que son fonctionnement était déjà tout à fait conforme aux spécifications, c'était pour espérer une plus grande compatibilité avec le navigateur Firefox de Mozilla, concurrent majeur de Google Chrome. En effet, une compatibilité inter-navigateurs est en développement du côté de Mozilla afin de garantir au mieux le transfert d'une extension Google Chrome vers Firefox, au vu de la popularité de ce type d'extension. Cette compatibilité sera toutefois garantie, à terme, uniquement sous condition du respect strict des recommandations de Google concernant le code, d'où l'intérêt d'une refonte précoce de l'extension.

La principale difficulté rencontrée lors de la réalisation de cette nouvelle version de l'extension fut l'organisation du code et la communication efficace entre ses différentes parties et l'application Prototypo. Pour faire passer la partie présentation du code de la page web vers la page pop-up, une réorganisation complète des sources a été nécessaire. Le code précédemment produit ne comprenait que très peu de communication entre les trois principaux scripts présents dans l'extension : l'avant-plan, l'arrière-plan et la fenêtre de pop-up. La partie pop-up est dédiée à l'interface utilisateur, la partie avant-plan va s'occuper de la gestion de la page web de l'utilisateur pour y appliquer, entre autres, la ou

les fontes éditées dans l'application, tandis que le script d'arrière-plan va gérer la communication entre tous les composants de l'extension. Il a donc fallu une certaine rigueur dans l'organisation du nouveau code de l'extension afin d'obtenir un résultat fonctionnel et maintenable, tout en apprenant au fur et à mesure les spécificités des fonctions offertes par Google pour mener à bien la mission.

Pour m'aider dans cette tâche, j'ai mis en place les méthodologies vues à l'IUT concernant l'ajout de commentaires dans le code, en annotant le code que je produisais mais également le code déjà existant afin de mieux l'assimiler. J'avais également assisté durant l'année dans le cadre de LyonJS, série d'événements organisés par une association regroupant la communauté des développeurs JavaScript lyonnais, à une conférence ayant pour thème la création d'extension Google Chrome dont j'avais retenu quelques concepts clés. La communication entre les composants est basée, dans l'API Google, sur un système de messages. Chaque script peut établir la communication avec un autre script via l'écoute ou l'envoi de message sur un port. L'utilisation judicieuse des ports avec la participation du script d'arrière-plan dans le rôle de centre de redistribution des messages était la clé pour obtenir un flux d'information logique et régulé, vital pour l'extension au vu du nombre d'éléments à échanger entre les composants et la fréquence d'envoi de messages dans le cadre d'une utilisation classique.

b) Création d'un outil d'affichage de fontes

Au terme de la seconde réunion hebdomadaire de répartition des tâches à laquelle j'ai pu assister, m'a été attribué la mission visant à développer une interface utilisateur dédiée aux administrateurs afin de visualiser les fontes créées par les utilisateurs de Prototypo. Pour pouvoir les afficher, en garder une trace sur l'un des serveurs de l'entreprise était nécessaire. L'idée était de se baser sur un serveur déjà existant qui se chargeait de transformer les fontes exportées par les utilisateurs pour fournir à ceux-ci la version correspondant au type d'export qu'ils avaient choisi.

Le but de ce nouvel outil était pour les administrateurs de pouvoir lister toutes les fontes exportées par les utilisateurs, en partant du principe que la plupart des fontes créées seraient exportées, en ayant directement dans l'interface un aperçu de chaque fonte. C'est la volonté de pouvoir en même temps repérer les créations des utilisateurs les plus originales, afin de promouvoir l'outil et de pouvoir repérer d'éventuels bugs d'affichages concernant les typographies, qui a motivé le démarrage de ce projet. A terme, il allait être possible de mettre en avant de manière promotionnelle les fontes créées par les utilisateurs, avec leur consentement, dans le but d'avoir des exemples concrets de produits créés avec

Prototypo. Il allait s'avérer par la suite que les deux buts seraient atteints : des polices originales ont été repérées grâce à l'outil, de même que certains bugs dans l'export de fontes.

Au départ, la technologie à utiliser pour faire fonctionner le serveur de récupération et d'affichage des fontes n'avait pas été définie par l'équipe de développement. En discutant du sujet lors de la réunion, nous sommes tombés d'accord très rapidement sur l'utilisation de NodeJS, c'est-à-dire la version serveur de JavaScript. Les compétences requises étant par ce choix limitées et l'interopérabilité augmentée du fait d'avoir le même langage du côté du client et de celui du serveur. Des modules sont disponibles sous NodeJS pour permettre la manipulation de fichiers qui allait être nécessaire afin de stocker les polices de caractères. La manipulation des fichiers est dans la plupart des langages une partie que j'avais du mal à appréhender mais qui, heureusement, avait largement été étudiée durant l'année de formation, notamment d'un point de vue objet grâce aux cours de Java. Dans un premier temps, il allait falloir créer une interface vide de sens qui permettrait de lister les fichiers présents dans un dossier donné du répertoire de développement puis, dans un second temps, lier l'export de fontes de l'application avec cette interface de manière à vérifier le bon fonctionnement du stockage avant de le tester dans l'environnement de développement.

Le développement de cet outil a été l'occasion de manipuler une nouvelle bibliothèque de fonctions, cette fois dédiée au JavaScript exécuté du côté du serveur, destinée dans notre cas à faciliter la création de serveur web. Son nom est Express, il s'agit d'un projet développé en open-source installable grâce à NPM, le gestionnaire de paquets de NodeJS. Son rôle a été d'aider à envoyer facilement des pages HTML, donc des pages web classiques, à l'interface qui allait être utilisée pour visualiser les fontes. Grâce à Jade, un module permettant de générer du HTML via Express tout en gardant une certaine simplicité dans la structure et la syntaxe du code, l'interface a été construite assez rapidement et contenait tout ce qu'il fallait pour proposer des fonctionnalités de filtrages des fontes et de chargement dynamique pour faciliter le travail des administrateurs qui allaient se connecter à cet outil.

Grâce à des modules complémentaires, les développeurs d'Express ayant choisi de ne garder dans la bibliothèque que les fonctions essentielles et de laisser le choix au développeur de télécharger ou non différents plugins auxiliaires, il a été possible de mettre en place un système de session afin de garder en mémoire les utilisateurs étant connectés à l'interface. Un accès restreint par mot de passe a été instauré pour limiter l'accès uniquement aux administrateurs autorisés. Un module additionnel a dû être une fois de plus installé et utilisé

pour pouvoir encrypter les mots de passe rentrés par l'utilisateur et comparer le tout à la version encryptée du mot de passe défini pour valider ou non l'accès.

Au final, l'outil de listage des fontes développé a été enrichi plus qu'il ne devait l'être à la base, notamment grâce à un surplus de temps pour ce projet. Ces ajouts ont été effectués du côté de l'interface, qui contenait un menu permettant de filtrer les fontes par utilisateur, un bouton en haut de la page permettant de charger plus de fontes depuis le serveur (via un appel AJAX ne nécessitant pas de rafraîchissement de la page), ainsi qu'un espace central contenant les fontes et l'identifiant de leur créateur. Le développeur de l'application a pu y ajouter un contenu éditable permettant de choisir quelles lettres pré-visualiser directement lors de l'utilisation de l'outil, ainsi que de plus nombreuses informations sur l'utilisateur à afficher dans l'espace central. J'ai, de mon côté, pu implémenter un bouton de téléchargement permettant à un administrateur de télécharger sur son ordinateur la fonte de son choix parmi celles qui étaient listées.

Si la partie client de cette mission était intéressante et apportait une véritable plus-value pour l'équipe de Prototypo, c'est la partie serveur qui m'a le plus permis d'apprendre sur les techniques de développement, notamment grâce au système de validation de mot de passe et de reconnaissance de session, abordé en cours de PHP cette année mais ici transposée à NodeJS. Toutefois, le ressenti dominant concernant ce projet était la satisfaction au vu du retour des administrateurs utilisant l'outil développé, qui ont immédiatement pris plaisir à s'en servir et ont pu repérer assez rapidement, toujours grâce à cet outil, des fontes qu'ils pourraient plus tard mettre en avant par rapport aux utilisateurs si cette démarche se concrétise.

c) Soutien pour finaliser la nouvelle version de l'interface utilisateur

Après mes deux premières missions au sein de l'entreprise, il était temps de m'intégrer au processus de développement de l'application Prototypo en elle-même. Comme mon arrivée s'est faite peu de temps avant la sortie de la nouvelle version de l'application, de nombreuses petites modifications étaient à réaliser afin d'aider l'équipe de développement à maintenir un rythme d'avancée stable pour parvenir sereinement à la mise en production. J'avais déjà dû effectuer certaines modifications dans l'application, notamment pour permettre la communication entre celle-ci et le serveur d'affichage de fontes lors de l'export d'une police. De plus, il avait été décidé que je passerai une demi-journée en « pair programming » avec François Poizat, le développeur le plus expérimenté avec l'utilisation de React. Cette méthode de programmation a pour but, à l'origine, de faire collaborer deux codeurs avec l'un des deux écrivant le code et le second vérifiant le code produit et aidant

à la conception des algorithmes, partant du principe qu'on repère plus facilement des erreurs en observant qu'en écrivant. Dans notre cas, le but était plus celui de la formation. La demi-journée prévue s'est transformée en deux demi-journées, l'une lors de laquelle j'étais observateur et l'autre où j'étais acteur. Dans les deux cas, la personne qui composait le code devait impérativement décrire tout ce qu'elle écrivait en rapport avec l'algorithme lié à son code. J'ai beaucoup apprécié cette stratégie destinée à la formation des nouveaux arrivants. En matière de qualité de code produit, j'étais déjà favorable à une utilisation raisonnable et ponctuelle du « pair programming », cette nouvelle expérience a confirmé mon avis sur la question.

Pour travailler sur le projet, la première étape a été la récupération sur mon poste de travail des différents projets qui composent l'application. Ils sont au nombre de trois, le script Prototypo, le canvas Prototypo et PluminJS, dépendant chacun respectivement du suivant. Les trois projets ont été principalement développés par Louis-Rémi Babé et servent respectivement à représenter, afficher et traiter les fontes de caractères grâce à du code JavaScript. L'application Prototypo dépend, quant à elle, des trois projets à la fois, lesquels doivent être inclus dans le dossier des dépendances de l'application grâce à des liens symboliques, le but étant de s'assurer que la modification d'un des projets se répercutera bien sur l'application si besoin est. La récupération des dernières versions des projets se fait via Git, le logiciel de contrôle de version utilisé par l'équipe de développement de Prototypo. Cependant, là où la plupart des utilisateurs de Git pratiquent la fusion de branches pour procéder aux récupérations des modifications d'un contenu distant vers un contenu local, il a été décidé chez Prototypo de procéder à l'incrémentation des nouvelles modifications sans créer de nouvelle version. L'utilisation des commandes est ainsi différente de celles les plus habituelles et il est impératif de respecter cette méthodologie si on souhaite participer au projet sans créer de conflits dans le code déjà existant.

La nouvelle interface utilisateur propose une refonte du menu dans lequel se trouvent principalement les différentes options d'export de polices sur lesquels nous reviendrons plus tard, mais aussi l'ajout de nouvelles propriétés pour éditer les fontes, l'ajout de nouvelles options pour plus de confort lors de l'édition, comme par exemple l'affichage du texte en blanc sur noir, et surtout l'ajout de la possibilité d'éditer des groupes d'individualisation. A l'origine, les utilisateurs de Prototypo, en modifiant les paramètres de leur police, modifiaient l'intégralité des glyphes appartenant à cette police, soit l'intégralité de l'alphabet en y ajoutant les caractères spéciaux latins. L'apparition des groupes d'individualisation permet de spécifier un jeu de caractères pour lequel les valeurs des différents paramètres pourront différer. Il est possible de créer autant de groupes qu'on le

souhaite, mais un même caractère ne peut appartenir qu'à un seul groupe. Ce choix a été fait à la fois pour ne pas rendre le procédé trop compliqué aux yeux d'un utilisateur non averti mais aussi pour éviter l'apparition d'états incohérents pour un même caractère appartenant à plusieurs groupes et héritant de trop nombreuses propriétés. Pour illustrer l'utilisation des groupes d'individualisation : on pourra par exemple créer un groupe composé du caractère « a » et changer sa hauteur pour la mettre au double. Ainsi, tous les « a » d'un texte écrit dans cette typographie dépasseront de deux fois la taille des autres caractères. Il est possible d'éditer les paramètres d'un groupe de manière relative, c'est-à-dire en multipliant, pour un « a » deux fois plus grand dans notre exemple précédent. Il est aussi possible d'éditer les paramètres de manière absolue, en additionnant, pour obtenir un « a » deux unités typographiques de hauteur au-dessus du reste des caractères, pour continuer sur le même exemple.

La liste des tâches effectuées en soutien sur cette partie de la nouvelle interface utilisateur comprend l'ajout d'un bouton de confirmation lors de la suppression d'une famille ou d'une variante, l'ajout d'un composant permettant d'afficher des cases à cocher destinées à une utilisation future en lieu et place du curseur actuellement présent pour éditer certains paramètres, la correction de plusieurs bugs d'affichages, la création de « sprites » CSS pour faciliter la gestion des images de fond de certains éléments de l'interface, une modification des méthodes d'export des fontes pour les rendre non bloquantes pour l'utilisateur, plusieurs corrections fonctionnelles concernant le dédoublement des noms de variantes pour les fontes, ainsi que, à l'approche de la mise en production de la nouvelle version de l'interface, de l'aide à la recette fonctionnelle de l'application afin de vérifier la conformité au cahier des charges initialement établi et à l'estimation de temps de développement pour les bugs restant à traiter.

La participation à diverses tâches de développement lors d'une phase précédant la sortie d'une nouvelle version de l'interface a eu pour bénéfices de me faire découvrir la gestion et l'organisation du temps avec une date butoir de rendu rapprochée pour une équipe très réduite, tout en me faisant découvrir progressivement les différentes parties de l'application, la manière dont chaque composant agit sur l'ensemble des autres composants, la gestion de l'état de l'application globale ainsi que l'état de chaque composant, concept essentiel à la compréhension de React. Une fois cette phase passée, l'équipe de développement était plus confiante pour me laisser gérer seul ou presque des tâches plus importantes au niveau de l'interface. En outre, un aperçu des travaux nécessaires à la gestion de projet m'a été donné grâce aux missions de recette fonctionnelle de l'application

ainsi qu'à celles d'estimation du temps nécessaire à la correction des dysfonctionnements liés à l'interface.

d) Tentative de déplacement d'une opération complexe du serveur au client

La prochaine étape pour améliorer l'expérience des utilisateurs de Prototypo était amenée à être plus complexe que les précédentes et son issue était dès le départ incertaine. Louis-Rémi Babé, qui avait développé le module JavaScript permettant de traiter les fontes sous forme de dessins vectoriels en deux dimensions, avait déjà par le passé tenté une opération similaire et avait été stoppé dans sa démarche par des bugs liés directement à la conception de la bibliothèque utilisée pour gérer cette représentation vectorielle : PaperJS. L'objectif était de migrer tout le processus de calcul de mise à plat des glyphes d'une police de caractère lors de son export, effectuée par le serveur, du côté du client.

Pour comprendre la problématique à l'origine de ce besoin, il faut tout d'abord expliquer les deux grands types d'export de variantes dans l'application. Lors de l'édition d'une police de caractères dans l'application, ce que l'utilisateur édite est de manière plus précise une variante de la famille de police créée par ce dernier. Cette variante correspond la plupart du temps à un style particulier de police. Les plus courantes sont « Regular », « Bold », « Italic » par exemple, qui correspondent respectivement à une variante normale, grasse et italique de la police que l'on est en train de manipuler. Ce que l'on exporte à proprement parler est donc la variante d'une fonte et non la fonte en elle-même qui est en fait une famille, composée d'une ou plusieurs variantes. L'export d'une famille sous forme de dossier compressé est, au moment où je termine mon stage, en cours de développement.

Dans l'onglet « File » de l'application, plusieurs options d'export sont proposées, parmi lesquelles on distinguera les deux principales, l'export avec aplats et sans aplats. Les deux autres sont une variante de l'export avec aplat qui permet de changer le nom du fichier résultant de l'export mais observe le même mode de fonctionnement, ainsi qu'une option permettant d'exporter sa variante dans « Glyphr studio », une application en ligne tierce de création de police typographique n'ayant pas de lien direct avec Prototypo. L'export sans aplats consiste à télécharger la variante telle qu'elle est affichée dans l'éditeur de Prototypo. Le procédé d'export va récupérer chacun des glyphes qui composent la variante de la police de caractères pour créer un jeu de caractères complet. Le fichier en résultant va ensuite être téléchargé au format binaire sur l'ordinateur de l'internaute par l'intermédiaire du navigateur, avec l'extension OTF (pour Open Type Format). L'utilisateur pourra alors librement installer la police sur sa machine.

L'export avec aplats reprend le même principe que le premier type d'export mais ajoute une étape préliminaire, l'aplatissement des différentes couches qui composent chaque glyphe de la fonte, aussi appelé « merge » en anglais. Il s'agit en fait de la fusion des différentes formes composant un glyphe, via un ensemble d'opérations booléennes permettant de définir les zones à dessiner ou non à partir d'une suite de points et de vecteurs. Cette opération est effectuée du côté du serveur par un logiciel nommé FontForge, outil de dessin vectoriel destiné aux polices de caractère. Celui-ci permet d'être agrémenté de script dont celui écrit par Prototypo qui s'occupe d'aplatir les fontes. Cette méthode est parfaitement fonctionnelle mais a l'inconvénient de nécessiter un appel au serveur pour être exécutée. La solution envisagée était donc de se fier au moteur PaperJS pour s'occuper de la mise à plat des fontes directement du côté du client, sans avoir à faire d'appel vers l'extérieur. Toutefois, au moment de l'instauration de cette solution, un dysfonctionnement avait rendu impossible son utilisation. Ce bug venait de PaperJS et provoquait des erreurs d'interprétations dans les formes lors des opérations booléennes, causant principalement l'apparition de trous et la disparition de certains composants dans plusieurs glyphes de la police exportée.

Ma mission était de remettre en œuvre la mise à plat du côté du client, en testant bien sûr le fonctionnement en local sur ma machine de travail, après avoir mis à jour la version de PaperJS dans les dépendances spécifiées pour Prototypo. Dans chaque projet JavaScript tel que Prototypo, un fichier nommé « package.json » liste en effet l'intégralité des dépendances du projet, ainsi que les versions pour chacune des dépendances. Il incombe aux développeurs du projet de maintenir à jour cette liste. A l'installation du projet, toutes les dépendances listées dans ce fichier sont récupérées en ligne. Je devais donc mettre à jour celle de PaperJS pour obtenir la dernière version de la bibliothèque, avant de vérifier si le bug répertorié plus tôt par l'équipe était toujours d'actualité.

Il s'est avéré, après une mise à jour assez fastidieuse au vu du nombre de dépendances du projet et du fait que PaperJS était nécessaire au fonctionnement de tous les projets de l'entreprise dont dépendait l'application Prototypo, que le bug sévissait toujours. Faire fonctionner l'export en lui-même était d'autant plus complexe que nous nous sommes rendu compte de la présence de certains bugs liés aux gabarits de deux des trois polices développées en interne par Prototypo. La résolution de ce problème a donc été obligatoire, en collaboration avec les directeurs techniques et artistiques dont les compétences étaient nécessaires, avant de pouvoir constater le bug en question. La prochaine étape était alors, toujours en restant sur la branche de développement afin de ne pas impacter les autres versions de l'application, de cibler le plus précisément possible le bug dans le but de le faire

remonter à l'équipe de développement de la bibliothèque PaperJS. Le projet est, en effet, développé lui aussi en open-source où toute personne possédant un compte GitHub a la possibilité soit de soumettre un bug aux développeurs, soit de proposer lui-même une solution pour résoudre un problème repéré ou encore pour apporter une amélioration dans le code. Les gérants du projet avaient pour habitude de répondre aux questions et de prendre en charge très rapidement les requêtes de l'équipe de Prototypo qui avait déjà eu à faire remonter certains problèmes par le passé.

De manière à pouvoir soumettre le bug aux développeurs de PaperJS, il fallait pouvoir isoler le plus précisément possible le bug afin de minimiser le temps nécessaire à l'équipe pour comprendre l'origine du problème. Je me suis donc attelé à la réalisation de cette tâche en apprenant la dénomination d'un tel exemple de dysfonctionnement : un « reduced testcase », à savoir le cas de reproduction d'un bug le plus simple possible. Il s'agissait donc de décomposer le problème petit à petit. La première étape était, bien sûr, la lecture de la documentation de PaperJS car je n'avais encore jamais eu affaire à cette bibliothèque de code. Une fois cela fait, le prochain objectif était de plonger au cœur des opérations booléennes effectuées par l'opération de mise à plat en local pour en isoler le code et le sortir de l'application. Pour reproduire le bug le plus fidèlement possible, la stratégie mise en œuvre a été la suivante : repérer un glyphe posant problème, le « x » minuscule dans notre cas, puis reproduire sa forme avec PaperJS en utilisant les mêmes fonctions que celles trouvées dans l'application. Pour reproduire une forme, il n'y avait pas d'autre solution que de récupérer les coordonnées de chaque point pour les recréer dans un plan en deux dimensions, créé pour l'occasion. La tâche était d'autant plus longue que tous les points séparés par une courbe étaient représentés à la fois par un point en deux dimensions (défini par deux nombres entiers, la coordonnée sur l'axe des abscisses et celle sur l'axe des ordonnées) et par un segment servant de vecteur donnant l'allure de la courbe établie entre les deux points, ce segment étant lui-même composé de deux points. La représentation de tous ces éléments (points, segments…) se fait en JavaScript dans PaperJS et utilise la Programmation Orientée Objet pour créer différentes instances dans lesquelles un même objet de type Point peut être réutilisé plusieurs fois. Néanmoins, la quantité de code générée pour reproduire le bug sur la forme du « x » était trop grande par rapport à la simplicité que l'on souhaitait obtenir afin de soumettre le bug aux développeurs de la bibliothèque. Nous avons par la suite réussi à isoler, parmi les composants du « x », ceux qui posaient problème afin d'avoir, pour de bon, la démonstration la plus simple possible pour un même dysfonctionnement.

Au terme de nos recherches, nous avons réussi à découvrir deux bugs différents mettant en cause une même fonction présente dans la bibliothèque PaperJS. Armés de notre cas de dysfonctionnement le plus réduit possible, le « reduced testcase », nous avons fait remonter le bug aux développeurs de PaperJS via GitHub. L'anglais était de rigueur pour pouvoir s'adresser à une communauté disséminée dans le monde entier et reliée via internet. Très rapidement (quelques dizaines de minutes plus tard), une réponse nous a été fournie et une mise à jour de la bibliothèque a suivi durant la journée, en réponse à la remontée d'un de nos deux bugs. Malheureusement, on ne peut pas vraiment parler de réussite pour la finalité de cette mission : la mise à plat des fontes en local n'a pas été mise en production car différents bugs subsistent en dehors du problème de rendu et du fait du manque de fiabilité de PaperJS pour effectuer cette opération de mise à plat qui n'a pas encore été assez suffisamment pour être considérée comme complètement fonctionnelle.

Malgré tout, cette mission reste pour moi la plus intéressante parmi la liste des tâches qui m'ont été confiées en vue d'améliorer l'expérience des utilisateurs de l'application. Les compétences techniques requises n'étaient pas particulièrement élevées et le but initial n'a pas pu être atteint, mais j'ai pu rentrer en contact avec l'équipe de développement d'une bibliothèque utilisée par de nombreux développeurs web en JavaScript (1700 téléchargements sur le NPM entre juillet et août 2016), PaperJS, m'apercevant ainsi de la réactivité et du professionnalisme des personnes qui maintiennent ce projet. De la même manière, il m'a été donné la possibilité d'observer comment un outil open-source peut être utilisé de manière extrêmement poussée, amené dans ses derniers retranchements par l'équipe de développement d'une application se basant sur cette technologie. Cela m'a poussé à réfléchir sur l'importance des choix à faire au niveau technologique lors du lancement d'un projet, ainsi qu'à constater les raisons derrière le choix fait par l'équipe de Prototypo pour cette bibliothèque en particulier : il s'agit d'un outil largement éprouvé même si pas encore parfait, possédant une communauté très active et offrant un support de qualité en cas de découverte d'un bug.

e) Développement de tutoriels contextuels

Le postulat de base de Prototypo est d'offrir la possibilité à ses utilisateurs de créer leurs propres fontes sans avoir besoin de connaissances techniques particulières ni d'un point de vue informatique, ni d'un point de vue typographique. Mais l'application en elle-même, pour pouvoir proposer un maximum d'options et de paramètres d'édition de fonte s'est, au cours du temps, considérablement complexifiée. C'est à partir de cette contradiction qu'est né le besoin de fournir aux nouveaux utilisateurs de l'application des tutoriels dans le but de les accompagner textuellement dans les étapes les moins intuitives de la conception d'une

police de caractères. Bien sûr, l'équipe de développement en collaboration avec le directeur artistique travaille constamment sur ce que le monde du web appelle communément l'« UX Design», soit l' « User eXperience Design», le design d'expérience utilisateur en français, sorte de discipline inhérente à la création d'interfaces utilisateurs qui vise à maintenir toujours accessible et intuitive cette dernière en analysant les réactions des utilisateurs face à l'interface, notamment en définissant différentes actions desquelles suivre la trace via des scripts JavaScript.

Malgré tous les efforts fournis pour simplifier au maximum l'interface des Prototypo, certains concepts liés à la typographie ou aux spécificités de l'application ont dû être expliqués via des panneaux contextuels à faire apparaitre au moment où l'utilisateur effectue pour la première fois des actions prédéfinies. Ces actions sont au nombre de quatre : l'ouverture de l'onglet « File » qui contient tous les types d'export, l'ouverture de la partie « Collection » qui permet de gérer les familles et variantes précédemment créées, l'activation du mode « Individualisation » qui permet d'individualiser un ensemble glyphes, ainsi que l'entrée dans un groupe d'individualisation présentant une interface dans laquelle on peut modifier les paramètres des glyphes présents dans le groupe. Au niveau de l'onglet contenant les exports, le tutoriel vise à expliquer les différents types d'export qui ont été définis précédemment dans ce rapport, plus précisément, les deux principaux modes d'export, à savoir avec ou sans aplats. Pour les collections, il fallait expliquer la différence entre familles et variantes, présenter le concept des collections, indiquer l'utilité de chaque panneau (liste des familles, listes des variantes de la famille sélectionnée puis variante sélectionnée) et enfin lister les différentes actions qu'il est possible d'effectuer pour chaque panneau. Le tutoriel du mode d'individualisation consiste, quant à lui, simplement à expliquer les concepts d'individualisation d'un ou plusieurs glyphes, en faire comprendre l'intérêt à l'utilisateur ainsi que définir les modalités de création de groupes, le fait que l'on puisse créer plusieurs groupes, que l'on ne puisse pas créer de groupe vide. Une fois le tutoriel d'individualisation effectué et un premier groupe d'individualisation créé et ouvert dans Prototypo par l'utilisateur, une nouvelle fenêtre modale aura pour but de lui expliquer les deux modes d'édition des paramètres d'une police. Ces deux modes sont le mode « relatif » et le mode « absolu » qui permettent d'augmenter ou diminuer les valeurs d'un paramètre de manière respectivement multiplicative ou additive.

Pour mener cette tâche à bien, une bibliothèque avait été présélectionnée par l'équipe de développement. Il s'agit de « React-Joyride », un module additionnel venant s'ajouter à React et s'utilisant directement dans les composants déjà créés sans nécessiter de réécriture du code. Le module en lui-même constitue un composant qui définira la fenêtre modale à

afficher pour montrer le tutoriel, avec un découpage en plusieurs phases dans lesquelles l'utilisateur peut naviguer. Il est du ressort du développeur de définir le moment auquel s'affichera le tutoriel, l'apparence qu'il prendra ainsi que son cycle de vie. Un composant géré par React doit en effet décrire les actions, dans le code, qui seront effectuées aux différents moments de sa présence dans la page : sa création, son montage ou démontage (en fonction de s'il est visible ou non), la mise à jour de son état ou de ses propriétés.

Pour faire en sorte de déclencher l'apparition des tutoriels, il a fallu mettre en place le système de communication inter-composants déjà appréhendé lors des précédentes tâches concernant la nouvelle interface utilisateur. La problématique ici était de trouver comment faire savoir à notre composant de tutoriel quand s'afficher, donc se déclencher et plus particulièrement quel tutoriel jouer parmi les quatre disponibles. Il fallait donc prendre chacun des composants nécessitant un tutoriel et envoyer une action dictant au tutoriel de s'afficher et décrivant quel tutoriel lancer. Pour mettre en place ce système correctement, la clé était l'utilisation de l'état des composants géré nativement par React. Il faut bien séparer un composant qui va émettre une action, ici par exemple le composant « Collection » qui émettra l'action « je viens de m'ouvrir pour la première fois », et un composant qui va écouter une action, ici notre composant de tutoriel. Pour être plus précis, l'action émise ira modifier une valeur dans la liste des valeurs de l'application, elle aussi gérée par React. Dans la méthode de création du composant de tutoriel, on mettra en écoute la modification d'une telle valeur qui viendra automatiquement modifier l'état de ce même composant. Enfin, on vérifiera en fonction de ce nouvel état si le tutoriel doit être lancé ou non. Par exemple, si la valeur spécifiant que l'outil de collection a été ouvert mais qu'on est au courant qu'il a déjà été ouvert par le passé, il ne faudra pas lancer le tutoriel correspondant. La gestion des états par React est automatique et la bibliothèque se chargera à notre place de décider quand l'état d'un composant doit être mis à jour et si ce composant nécessite d'être régénéré.

Enfin, la gestion de la persistance des données est un aspect nouveau à ce moment du stage qu'il fallait introduire pour être certain de n'afficher qu'une seule fois chaque tutoriel. Nous allions donc initialiser une valeur à « vraie » pour les quatre tutoriels différents qui représenterait si c'était oui ou non, la première ouverture du composant mise en cause et ce pour chaque utilisateur connecté. Si j'avais été habitué à stocker des données non-persistantes au sein de l'application, principalement pour gérer l'état des composants qui devait être réinitialisé à chaque rechargement ou fermeture de l'application, stocker des données durablement était sensiblement plus complexe. Il ne fallait plus se contenter de faire appel à de simples actions mais gérer le système de « stores ». Ces « stores », qui

contiennent toutes les données persistantes de l'application liées au compte d'un utilisateur, sont au cœur de la gestion de données de React et suivent les grandes lignes d'une philosophie instaurée par React et nommée Flux. Flux représente principalement un schéma de gestion des données et indique comment organiser leur évolution de manière pérenne quelle que soit la taille de l'application React en cours de développement. Les principales différences venaient de l'initialisation des données, qu'il fallait précédemment déclarer dans un fichier spécifique dédié, ainsi que dans les actions à appeler pour stocker les données dont l'utilisation pouvait varier par rapport aux actions simples.

A des fins de tests, j'ai pu avoir accès au système de gestion de base de données de l'application, ou tout du moins à sa version de développement contenant tous les comptes utilisateurs sur la plateforme de test. Ce système était CouchDB, dédié à gérer les bases NoSQL. J'ai pu y éditer les données de mon propre compte afin de pouvoir vérifier plusieurs fois si les bons tutoriels se déclenchaient au bon moment selon les bonnes actions, et ce fut peut-être la partie la plus importante de cette mission, celles concernant les tests fonctionnels. En effet, l'ajout d'une fonctionnalité telle que l'apparition de tutoriels requiert d'être certain que ceux-ci n'apparaitront qu'une seule et unique fois, et qu'ils n'apparaitront pas à un moment inopportun, ce qui irait à l'encontre du but original de cet ajout qui est d'aider au maximum l'utilisateur dans sa compréhension du fonctionnement de l'application et de faciliter son expérience avec l'outil développé par Prototypo. Au sortir de cette phase de test, je pense pouvoir affirmer que les méthodes rigoureuses acquises à l'IUT ajoutées à mon expérience professionnelle passée m'ont permis d'être plus serein face à ces séries de tests fonctionnels à exécuter lorsqu'on travaille sur une interface, qui pouvaient parfois me poser problème dans le cadre du développement de site web.

f) Modification manuelle de l'espacement de caractères

Ma dernière mission lors de ce stage de fin d'études a été la plus complexe à mettre en œuvre, celle impliquant également le plus de réflexion concernant l'expérience utilisateur. Mon assignement de cette tâche trouvait son origine dans le souhait des dirigeants de Prototypo de proposer à leurs utilisateurs une manière visuelle d'éditer l'espacement des caractères directement depuis le texte. L'application Prototypo propose trois vues permettant d'observer sur la police de caractères les modifications effectuées, ces trois vues pouvant être activées et désactivées à l'envi. La première est la présentation d'un texte court censé ne contenir qu'un mot sur une seule ligne éditable, la seconde est la présentation d'un caractère précis sélectionné par l'utilisateur, la troisième présente, quant à elle, un texte plus long dans une zone de texte éditable avec retour à la ligne possible, proposant un remplissage automatique si besoin.

La mission consistait en l'ajout, pour la première zone contenant un texte court éditable sur une ligne, de deux barres pouvant apparaitre de chaque côté d'une lettre et qui permettraient à l'utilisateur de régler manuellement l'espacement à appliquer autour de la lettre ciblée. Le fonctionnement de base était décrit tel que nous le faisons ici, sachant qu'une maquette avait été réalisée auparavant, permettant ainsi de savoir comment articuler visuellement ce nouvel élément. Les modularités d'apparition de cet outil n'étaient pas clairement définies et j'ai dû prendre la décision de créer un « mode » dans lequel il était possible d'éditer l'espacement d'un caractère, mais dans lequel on ne pourrait plus éditer le mot présent. En effet, la cohabitation des deux fonctionnalités aurait été potentiellement conflictuelle, React lui-même émettant un avertissement en cas de tentative de la part du développeur de gérer l'édition de contenu à son insu. Bien entendu, toute idée ou décision était validée par toute l'équipe de développement de l'application, parfois également par les directeurs artistiques et marketing.

La première étape a été l'installation purement visuelle des barres apposées à une lettre, la première pour commencer, ainsi que des chiffres qui allaient représenter les valeurs liées à l'espacement et à la largeur du caractère. Ensuite est venue la partie plus délicate consistant à mettre en place, toujours en gardant de fausses valeurs, un système de « drag and drop » permettant de déplacer les deux barres indépendamment. Un système de ce genre existait déjà au sein de l'application et était implémenté pour gérer les règles permettant de modifier les paramètres d'une fonte. Toutefois, la fonctionnalité à mettre en place ici était différente et ne se limitait pas à déplacer un objet mais deux, possédant des valeurs indépendantes. De plus, le développeur de l'application m'a fait remarquer en amont que, si les accroches aux côtés des lettres ne mesuraient qu'un pixel sur les maquettes, il allait falloir être ingénieux et prévoir une taille réelle plus grande qu'un pixel si on souhaitait que l'utilisateur puisse facilement l'attraper à l'aide de sa souris.

Ensuite s'est posé le problème plus épineux à gérer de la relativité du déplacement de l'espacement. En effet, la zone de texte où le système doit être installé ne contient qu'une ligne, or il est possible pour l'utilisateur d'ajouter dans cette partie autant de texte qu'il le souhaite. Pour pouvoir respecter ces deux impératifs, il a été mis en place un système qui adapte la taille de la police proportionnellement à la quantité de texte ajouté, où plus le texte s'accumule, plus la taille de la police rétrécit. De ce fait, déplacer l'une des deux barres d'espacement d'un pixel, en fonction de la taille de la police au moment où ce geste est effectué, ne devait pas modifier l'espacement du même nombre d'unités typographiques. La réponse à ce problème a été le calcul d'un ratio trouvé à partir de deux éléments, la taille du caractère en pixels et sa taille totale en unités typographiques. On

obtenait alors, en unité typographique par pixel, le ratio à multiplier au nombre de pixels parcourus pour obtenir le nombre d'unités typographiques à ajouter ou retirer au caractère.

Après avoir repensé et revu le système drag and drop plusieurs fois avec l'assistance de François Poizat, le développeur principal de l'application, nous sommes arrivés à un système satisfaisant et facile d'accès pour les utilisateurs. Cette mission aura engendré des modifications dans plusieurs projets de Prototypo en dehors de l'application elle-même et m'aura offert la possibilité de créer un composant ex nihilo, sans reprendre de modèle existant, en gérant tout son cycle de vie ainsi que les différents autres composants dont il dépendait. De la rigueur a donc été nécessaire dans l'organisation du fichier contenant l'ensemble des outils liés à l'espacement des caractères et des choix ont dû être faits dans la gestion des méthodes présentes pour décider à quel composant revenait l'appartenance de la fonctionnalité correspondante. Par exemple, le traitement des mouvements de la souris devait-il dépendre de la zone texte au global ou de la barre à déplacer elle-même ? De même pour la gestion du clic avant le déplacement de la souris, entre autres. J'ai pu à cette occasion m'appuyer sur les techniques que j'avais utilisées durant l'année lors du développement de mon projet tuteuré qui présentait, à l'époque, des problématiques ressemblant à celles rencontrées dans le cadre de ce stage, que nous avions pu résoudre en équipe.

Du fait de la complexité de l'opération et du flux de données à mettre en place, nous avons dû, à plusieurs reprises, avoir recours aux fonctionnalités de debug de l'interpréteur JavaScript du navigateur, outils qui m'étaient inconnus avant de commencer mon stage. Le langage principalement enseigné à l'IUT est le Java, qui est, comme son nom ne l'indique pas, très différent du JavaScript. En étudiant le Java, nous avons appris à effectuer des debugs pas à pas, c'est-à-dire à stopper virtuellement le programme en cours d'exécution pour observer son état à un instant précis, souvent dans le but de connaitre la valeur d'une variable précise pour identifier un problème. Dans le cas du Java, le programme étant compilé par un IDE, logiciel proposant un environnement de développement intégré, il suffit de placer des points d'arrêt dans son code source comme le propose NetBeans par exemple, plateforme utilisée durant les travaux pratiques à l'IUT. Pour le JavaScript, il est nécessaire d'ouvrir la console développeur au sein du navigateur puis d'aller chercher les fichiers sources dans le bon onglet pour placer des points d'arrêt et stopper le script au moment voulu. Des outils de profilage puissants sont également proposés et nous avons dû y avoir recours pour mettre en cause une fonction exécutée de manière répétée qui ralentissait l'application. Le navigateur utilisé était Google Chrome, mais on peut constater de manière générale que, petit à petit, les éditeurs de navigateur web se mettent à proposer des outils

rivalisant en complexité et en nombre de services fournis avec ceux dédiés au développement d'applications compilées comme NetBeans ou Eclipse pour le Java.

III) Application web et typographie : Interface complexe au service de la créativité

a) Interface complexe, maintenabilité fragile

Alors que le JavaScript était un langage initialement destiné à créer de l'interactivité dans des pages web statiques, on le retrouve aujourd'hui dans de très nombreuses situations qui n'avaient jamais été prévues par les créateurs du langage, ne serait-ce qu'avec l'arrivée du JavaScript côté serveur grâce à NodeJS. Ce langage bénéficie d'une communauté très active, d'un soutien par des acteurs majeurs d'internet tels que Google ou Facebook et a évolué pour devenir particulièrement complexe. Certains de ses utilisateurs font preuve d'un zèle prononcé : des centaines de bibliothèques de fonctions offrant des fonctionnalités extrêmement variées ont vu le jour afin de répondre à des besoins toujours grandissant d'interaction entre les internautes et les pages web qu'ils consultent. Des compilateurs ont été créés pour que les développeurs puissent profiter des spécifications du langage que les navigateurs les plus utilisés n'ont pas fini d'implémenter et dont certaines sont encore en phase de validation par le comité en charge des spécifications ECMAScript, dont JavaScript est une implémentation. En résulte un foisonnement de technologies basées sur un même langage et souvent interdépendantes dans lequel un néophyte peut se faire submerger assez rapidement.

Au cœur de ce bouillonnement est né React, ainsi que plusieurs autres bibliothèques comme AngularJS, afin de permettre aux développeurs de sortir plus facilement du modèle classique de pages web statiques et offrir à leurs utilisateurs de véritables applications interactives. Comme plusieurs bibliothèques ayant un but similaire de gestion de l'interface utilisateur, elles sont nées de solutions internes de la part de leurs entreprises génitrices avant d'être rendues publiques via un passage à l'open-source. Cela confirme l'existence de la problématique à laquelle de nombreux projets web ont dû faire face au moment du développement d'interfaces complexes : la gestion du flux de données et de l'état de ce type de page web.

Dans le contexte de mon stage, j'ai pu constater à la fois la fragilité de la maintenabilité d'un tel type d'interface, ainsi que les bénéfices apportés par des solutions comme React. Plus que de solution, il faudrait parler d'outil car il reste tout à fait possible, avec une utilisation irréfléchie de l'outil, d'aboutir à une solution instable et d'aller à la rencontre de nombreux problèmes de maintenabilité, même au sein d'une équipe de développement à la taille réduite. La solution mise en œuvre pour le développement de l'application Prototypo repose en effet sur une architecture respectant pleinement les règles proposées

par les développeurs de React et allant même jusqu'à s'en imposer de nouvelles pour obtenir plus de fiabilité. Dans les « stores » de Prototypo, où sont stockées toutes les données de l'application, un système interne de gestion de version est utilisé. A l'instar du fonctionnement de Git, chaque changement résulte en un patch qui est ajouté à l'historique des données, de manière à garantir à tout moment la cohérence d'un éventuel retour en arrière.

La mise en place des dernières syntaxes disponibles pour le langage JavaScript met une fois de plus en évidence un souci de maintenabilité qui semble être source de réflexion au sein de l'entreprise. L'équipe est actuellement composée principalement de deux développeurs travaillant à temps plein sur l'application, mais ce paramétrage des ressources pourrait être amené à changer, il faudra donc qu'une personne intégrant l'équipe soit capable de lire et de comprendre rapidement le code derrière l'application si on souhaite l'intégrer le plus rapidement possible au cycle de développement de Prototypo. De même, les développeurs m'ont plusieurs fois repris sur une syntaxe pourtant valide, mais qui n'aurait pas permis une réécriture du code facile si quelqu'un était amené à revoir une méthode ou un passage du code. Mon expérience a pris place au sein d'une start-up, entreprise de très petite taille et possédant peu d'ancienneté. Celle-ci a choisi d'utiliser les outils les plus pointus et récents disponibles dans le domaine du web, ce qui n'est pas sans risque. Malgré la présence de grandes compagnies au support derrière certaines des bibliothèques utilisées, aucune garantie de maintenance n'est établie par rapport à ces technologies. Ainsi, un revirement de situation ou un développement qui ne serait plus maintenu pourrait éventuellement forcer l'équipe à réécrire une partie de l'application dans un laps de temps très limité. La grande force d'une start-up est de pouvoir se permettre de prendre le risque de se retrouver face à une telle situation, là où une entreprise avec une base de code beaucoup plus volumineuse et un grand nombre de développeurs ne pourrait pas répondre à une telle urgence et ne se serait donc probablement pas engagée sur l'utilisation de ces outils pourtant très performants. Le fait que le code soit lisible et facile à réécrire reste néanmoins un prérequis pour pouvoir faire face aux différentes situations qui peuvent se présenter à l'avenir. Dans cette même optique, on peut relever l'utilisation de la syntaxe JSX propre à React qui permet d'écrire ses composants dans un langage cousin du HTML, très facilement compréhensible par n'importe quel développeur web et particulièrement lisible sous réserve d'avoir un éditeur de code prenant le JSX en charge.

Cette notion de maintenabilité est renforcée sur les projets autour de Prototypo ainsi que sur l'application elle-même par le fait d'avoir défini des règles strictes auxquelles les intervenants doivent se soumettre. Ces règles ne peuvent concerner que la syntaxe mais

pourront être bloquantes. Au moment de générer les sources d'un projet à partir du dossier de travail, le processus sera pour la plupart des projets bloqué si l'une des nombreuses conditions à respecter concernant la mise en forme du code n'est pas respectée. En résulte un code dont on aura la garantie qu'il sera uniforme quel que soit le nombre de personnes qui travailleront dessus à l'avenir. De la même manière, si certains des tests unitaires définis dans le projet ne sont pas validés au moment de la compilation des sources, une erreur surviendra et le procédé n'aboutira pas. La mise en place de tests unitaires n'est pour l'instant pas aussi rigide que les règles de syntaxe. Il est tout à fait possible de tester les composants de l'application Prototypo, mais les tests les plus importants concernant l'interface restent pour le moment les tests fonctionnels réalisés à la main par l'équipe dans sa globalité. Il s'avère que l'équipe de développement a, au moment où mon stage touche à sa fin, pour prochain objectif de mettre en place l'automatisation totale de ce genre de tests ainsi que des procédés de normalisation similaires à ceux régissant la syntaxe du code de l'application.

La modularité du code, qui est prépondérante dans Prototypo, sera l'un des échafaudages nécessaires à la mise en place des futurs tests. Le code est divisé en composants tous séparés en fichiers puis importés dans un fichier principal qui constitue le cœur de l'application. En outre, différentes fonctions d'aide au développement sont factorisées dans des dossiers dédiés. Il n'est pas possible de forcer un développeur à rendre son code modulaire à l'aide de règles de relecture, mais l'équipe de développement fait en sorte de former tout nouvel arrivant de manière à ce que ce soit toujours le cas. Cette manière de séparer différentes parties du code fait partie des bonnes pratiques générales de la programmation objet auxquelles le JavaScript n'échappe pas.

Il est intéressant d'observer la mise en place de la gestion de projet au sein d'une entreprise telle que Prototypo. Les inspirations sont puisées du côté d'Agile, et on peut y voir une implémentation minimaliste des règles qui y sont dictées. En effet, les méthodes Agile ne sont pas plus qu'une liste de règles de gestion de projet à mettre en place en fonction de la taille de son équipe, du mode de fonctionnement que l'on souhaite privilégier et des priorités décidées. Dans notre cas, les réunions stand-up se font sous la forme d'une réunion hebdomadaire le lundi matin où chaque membre de l'équipe décrit ce qu'il a fait la semaine passée et ce qu'il lui reste à faire. Le reste des meetings sont pour la plupart informels et se produisent quand le besoin en est ressenti. La gestion des retours, des bugs et des estimations de temps se font sur la plateforme GitHub, en ajoutant des extensions Google Chrome aux fonctionnalités de base lorsque celles-ci ne suffisent pas. Cela permet d'inclure la communauté potentielle liée à la nature en sources libres des projets de Prototypo.

Toujours sur cette plateforme, des « milestones » sont définies, pierres angulaires permettant de définir des périodes équivalant à des sprints et d'y assigner des bugs ou des tâches. Trello est également un outil de gestion de projet qui est utilisé en parallèle à GitHub au sein de l'entreprise, mais qui se destine plus aux tâches qui ne seraient pas en rapport direct avec le code telles que l'organisation pour participer à une conférence, des modifications à apporter du côté typographique ou des tâches liées à la vie de l'entreprise. Le trait particulier que je noterai après avoir observé durant dix semaines le fonctionnement d'une équipe de la taille de celle de Prototypo est le potentiel de souplesse et de flexibilité extrême qui émane de l'organisation. L'apparition du moindre problème lors de la conception de l'application ou de la moindre divergence d'opinion peut donner lieu à un changement d'organisation qui ne posera pas de problème. J'ai trouvé intéressant le fait de pouvoir comparer ce mode de fonctionnement aux principes fondamentaux des méthodes Agile qui préconisent exactement ce genre d'approche, privilégiant la réactivité et la spontanéité de l'équipe au détriment de rituels fixes prolongés qui pourraient signifier une perte de temps pour certains des acteurs. On distinguera simplement le fait que dans la configuration de Prototypo, l'accent ne peut pas réellement être mis sur la satisfaction du client avec sa collaboration, étant donné que les retours clients ne sont pas disponibles comme ressource immédiate. La gestion s'axera donc plutôt sur une extrapolation des envies et besoins des utilisateurs afin d'essayer de produire une application le plus en accord possible avec le public ciblé.

b) Des possibilités nombreuses pour offrir de la créativité aux utilisateurs

Comme on a pu le constater au vu des missions décrites dans la seconde partie de ce rapport, l'interface utilisateur de Prototypo est en constante évolution et a pour but de répondre toujours mieux aux besoins de l'utilisateur. Car répondre aux besoins d'un utilisateur, c'est créer l'opportunité d'en faire un nouveau client grâce au processus d'abonnement payant. Ces incrémentations constantes dans le but d'améliorer l'outil proposé reflètent donc le modèle économique général établi par Prototypo pour son application, qui souhaite retenir l'attention de l'internaute pour l'inciter à s'abonner.

Au cours de mon stage, j'ai eu la chance d'assister à un changement majeur à la fois dans l'interface utilisateur et dans le modèle économique de l'application. L'interface a été refondue de manière à laisser la possibilité à un utilisateur non abonné d'avoir accès de manière immédiate à tous les paramètres d'édition pour sa fonte de caractères. Un changement dans le modèle économique a rapidement été mis en place du côté administratif, impliquant de nombreux changement dans une interface qui doit impérativement s'accorder à la manière dont les utilisateurs vont pouvoir consommer

l'application. Chaque changement apporté à l'application en lien avec le paiement entraine par ailleurs une batterie de tests en vue de ne pas flouer un utilisateur devenu client, ni d'entrainer de pertes pour l'entreprise suite à un éventuel dysfonctionnement.

J'ai pu observer la complexité de trouver une offre commerciale optimale qui poussera l'utilisateur à investir son argent dans l'application. Le passage d'un modèle gratuit pour une utilisation simple et payant pour une utilisation plus poussée vers un modèle gratuit pour toute l'utilisation mais payant pour l'exploitation du produit créé portera ou non ses fruits dans un avenir plus ou moins proche qu'il ne me sera pas donné d'observer de l'intérieur de l'entreprise. Malgré tout, il a été très intéressant de suivre la démarche des créateurs qui, suivant les statistiques et écoutant les retours de leurs utilisateurs et collaborateurs ont été amenés à revoir en profondeur un système pourtant décidé de manière réfléchie au lancement de l'application. Une fois de plus, la taille réduite de l'entreprise offre la possibilité d'une remise en question régulière quant à la meilleure manière d'obtenir l'adhésion des utilisateurs.

Ce stage a été l'occasion pour moi de découvrir le design d'interface utilisateur, science complexe et raffinée. Bien que les concepts de base en UI (user interface) m'eussent servi pour la conception de mon projet d'étude tuteuré, j'ai pu me rendre compte durant ce stage de la rigueur nécessaire au bon fonctionnement d'une interface. Les tests sur les sujets les plus nombreux possibles sont essentiels pour se rendre compte des habitudes et réflexes que peuvent avoir des néophytes par rapport à l'interface, différents bien souvent de ceux de l'équipe de développement ayant créé l'outil. De même, le fonctionnement de l'application avec des « trakers », scripts permettant de compter et répertorier les différentes actions menées par les utilisateurs en vue de les analyser par la suite est primordial afin de comprendre son public cible.

Après avoir choisi le type d'utilisateurs le plus susceptible d'être convaincus par l'application, reste la difficulté créée par leur diversité et la diversité des outils qu'il est possible de leur offrir. Prototypo a, en effet, été financée via la plateforme Kickstarter par des utilisateurs venant du monde entier. Ces différentes cultures qui viennent se confondre dans une même interface imposent des contraintes, à commencer par l'utilisation constante de la langue internationale, l'anglais. Il ne sera jamais évident de connaitre les envies et les besoins de différents utilisateurs de Prototypo du fait de la segmentation du public ciblé. On peut toutefois voir cela comme étant également une force, un moyen de multiplier les retours d'expérience, de permettre la création de fontes moins conventionnelles que celles qui auraient pu être créées par un public exclusivement français.

Le paramétrage des fontes de caractères, effectué principalement par Yannick Mathey, le directeur artistique de Prototypo, foisonne lui aussi de possibilités. Il est possible pour lui de définir toutes les propriétés qu'une fonte pourra revêtir, de les combiner afin de créer une police de caractères unique. Les outils utilisés sont une combinaison de logiciels de dessin et de fichier JavaScript écrits en CoffeeScript, un dérivé du JavaScript facilitant la rédaction d'objets de grandes tailles dont font partie les fichiers de description de fontes de Prototypo. Les différentes propriétés présentes dans le code se basent sur celles qui définissent une fonte de manière typographique : l'espacement entre les caractères, leur hauteur, largeur, la présence ou non d'empâtements. Chaque propriété ajoutée dans ce contexte aura pour effet d'offrir de nouvelles possibilités de modifications aux utilisateurs. Celles-ci évoluent constamment depuis la création de Prototypo. Il incombera par la suite aux développeurs de permettre leur édition et de vérifier que les valeurs pouvant être réglées depuis l'application ne portent pas préjudice à la police générée. Malgré cela, notamment grâce à la nature de Prototypo dont les calculs sont pour la plupart effectués du côté du client, l'utilisateur a toujours le pouvoir et la liberté de créer des fontes ayant les formes les plus extravagantes et ne respectant aucune des recommandations fournies par l'application.

J'ai également eu la possibilité d'entrevoir un nouvel outil qui sera développé dans le futur par Prototypo, interface externe à l'application destinée à l'assistance au développement de fichiers de polices tels que ceux définissant les trois gabarits de typographies de base proposées par Prototypo. L'exercice est long et fastidieux et demande de nombreuses compétences techniques à la fois en développement et en typographie. Cet outil permettra de paramétrer de manière visuelle les différents points de la base d'une fonte qui générera un fichier utilisable dans le projet Prototypo. On pourra donc créer soi-même, dans une page web, une police paramétrable. L'outil semble se destiner aux administrateurs de Prototypo, tout du moins dans un premier temps, mais son utilisation pourrait aller plus loin et proposer une interface intuitive aux utilisateurs souhaitant créer leur fonte sans se baser sur un gabarit existant pour remplir leur objectif. L'équipe envisage déjà, à terme, l'apparition d'une plateforme d'échange en ligne permettant aux créateurs de partager ou de monnayer leurs polices de caractères facilement et librement.

L'expérience que je retire de mon stage au sein de Prototypo est très positive et tout à fait conforme à ce que j'en attendais au départ. C'était pour moi la promesse de plonger dans les rouages d'une interface utilisateur complexe, maintenue grâce à une bibliothèque dédiée. J'attendais du contexte d'une start-up un environnement à l'écoute mais aussi exigeant, où l'ensemble des problématiques liées à l'application pourraient être observées de près. J'espérais également mener à bien cette mission en acquérant des notions basées sur la typographie et des compétences plus concrètes dans la maîtrise du framework React. Le lien a pu être fait naturellement pour moi entre mes missions et la formation à l'IUT, à la fois grâce à mon projet tuteuré qui m'a préparé au mieux à la réalisation de ce stage, mais également aux différentes méthodologies de tests, commentaires, gestion de projet et programmation objet, réutilisées intuitivement au moment de la mise en pratique que représente le stage de fin d'étude.

J'ai donc pu observer le fonctionnement d'une start-up ayant plus d'une année d'ancienneté, mais aussi être immergé dans un incubateur regroupant plusieurs entreprises en démarrage, souvent confrontées aux mêmes problématiques de recherche d'un modèle économique qui convienne le mieux possible à un public cible et qui permette de concrétiser le plus de ventes ou d'abonnements possible, mais aussi faisant face à l'importance d'une communication réussie pour le lancement ou la maintenance d'un projet avec l'investissement en temps et en ressources correspondant.

La taille réduite de l'équipe, à savoir deux développeurs informaticiens, un directeur artistique et un directeur marketing, permet lors d'un stage d'observer très facilement le fonctionnement d'une organisation dans son ensemble et de bénéficier d'un soutien technique efficace. La mise en place de méthodes de gestion de projet de type Agile dans ce genre de configuration d'équipe est particulièrement intéressante à suivre car je pense que ces dernières ont été créées dans le but de reproduire, potentiellement à grande échelle, ce type de microcosme d'une extrême souplesse pouvant s'adapter très rapidement à des besoins changeants et limitant au maximum les pertes de temps.

S'intégrer dans l'application Prototypo s'est fait sans que je rencontre de véritable problème. Je mettrai cela au crédit de la qualité du code source produit par l'équipe de développement, limpide dans la plupart des cas, mais aussi de la qualité de l'accompagnement fourni au regard des compétences que peut nécessiter l'évolution d'une interface web de ce type. J'ai toutefois pu confirmer le sentiment que j'avais par rapport à la complexité des bibliothèques dédiées à l'interface utilisateur, ainsi que la rigueur qu'elles nécessitent afin de garantir l'obtention d'une application sûre, solide et maintenable. Il faut gérer avec efficacité ses différents états, veiller à la factorisation de ses composants, ne

pas tomber dans le piège de faire vivre trop de logique métier dans des composants de présentation. Les écueils peuvent devenir nombreux et la distinction est très claire avec un site web classique tel que j'ai pu en développer de nombreux par le passé. J'ai le sentiment d'avoir connu une véritable prise d'autonomie par rapport à React, bibliothèque autour de laquelle je possédais quelques notions mais que j'ai pu analyser de manière beaucoup plus sérieuse au cours de ce stage. La chance m'a également été donnée d'observer directement, dans un contexte professionnel, des cas d'utilisation très poussés de bibliothèques développées en open-source et fournissant pourtant un support aux entreprises ayant choisi de s'appuyer dessus, en dépit de tout contrat de maintenance.

Dans un avenir proche, je ne pense pas avoir de nouveau l'occasion de pratiquer le JavaScript, ni de me retrouver dans une configuration d'équipe aussi réduite. En effet, j'entame à partir de la rentrée 2016 une alternance au sein d'une entreprise comptant plus d'une centaine d'employés. Je suis donc très satisfait d'avoir pu travailler le temps d'un stage avec ce langage qui me passionne, au sein d'une équipe extrêmement compétente dans ce domaine et qui m'a fait confiance pour le développement de fonctionnalités importantes. Devenir entrepreneur est un projet que je n'exclue pas pour l'avenir, cette première expérience en start-up me confortant dans l'idée de liberté que cette voie peut offrir. Prototypo est une société qui va probablement être amenée à s'agrandir sous peu et qui risque donc de devoir s'adapter au mieux pour rester aussi efficace qu'elle l'est actuellement, quitte à changer certaines des méthodes à l'origine de cette efficacité. Au-delà de la réussite que je lui souhaite, je pense que cette entreprise réussira à atteindre ses objectifs et, selon moi, le bon déroulement de ce stage démontre la stabilité et la maintenabilité de Prototypo, outil pourtant complexe qui saura sûrement rester solide lors de ses futures évolutions.

Glossaire

API : Interface de programmation, ensemble de fonctions fournies par l'éditeur d'un programme ou d'un service informatique à des utilisateurs pour pouvoir interagir avec ce programme ou service.

Agile : Ensemble de méthodes de gestion de projet définies par un manifeste en énonçant les grands principes, qui se basent sur l'interaction entre les individus, la livraison de logiciels fonctionnels, la collaboration avec le client et l'adaptation au changement.

Bibliothèque : « Library » en anglais, diminutif pour « bibliothèque de fonctions », ensemble de fonctions écrites dans un langage informatique particulier qui se destinent à aider le développeur dans une tâche précise.

CoffeeScript : Langage destiné à simplifier la syntaxe JavaScript, permettant d'écrire du code source traduit par la suite vers du JavaScript classique.

CSS : Langage de feuilles de styles en cascade, responsable de la description visuelle des pages internet.

ECMAScript : Langage de programmation orienté script tirant ses spécifications de la société ECMA et dont JavaScript est une implémentation.

Framework : « Cadre de travail » en français, nom généralement donné à une bibliothèque qui, en plus de son rôle habituel, va imposer à son utilisateur des règles à suivre pour son bon fonctionnement. La définition de la différence entre un framework et une bibliothèque variera selon les sources.

Git : Logiciel de gestion de version permettant entre autres de garder une trace de toutes les modifications effectuées sur un ensemble de fichiers.

GitHub : Plateforme en ligne d'hébergement de code source, utilisable conjointement à Git.

HTML : Langage de structuration des pages internet, fonctionnant grâce à un système de balisage.

JavaScript : Langage interprété par les navigateurs, de nature évènementielle, qui permet de gérer l'interaction avec les utilisateurs d'une page web de manière dynamique, côté client, sans rafraîchissement de page.

NetBeans : Logiciel d'édition de code permettant de compiler un programme de haut niveau vers du code machine, tout en l'écrivant. Offre de nombreux outils d'aide à la conception du code.

SCSS : Bibliothèque JavaScript permettant d'émuler le fonctionnement du CSS en lui ajoutant de nouvelles fonctions telles que la déclaration de variables ou la création de fonctions.

SEA : « Search Engine Advertising », discipline de l'optimisation d'un site internet pour les moteurs de recherche par tous les moyens payants comme les liens sponsorisés, de manière à augmenter son trafic.

SEO : « Search Engine Optimisation », discipline de l'optimisation d'un site internet pour les moteurs de recherche, dans le but de lui obtenir le meilleur positionnement possible dans les résultats de recherche pour un jeu de mots-clés donné, par tous les moyens « naturels » c'est-à-dire gratuit.

Sources

Webographie :

- A JavaScript library for building user interfaces | React. *Facebook.github.io (en ligne)*. Consulté le 30 juillet 2016. Disponible sur internet : https://facebook.github.io/react/
- Application Web. *Wikipédia (en ligne)*. Consulté le 5 août 2016. Disponible sur internet : https://fr.wikipedia.org/wiki/Application_web
- CoffeeScript. *CoffeeScript (en ligne)*. Consulté le 20 juin 2016. Disponible sur internet : http://coffeescript.org/
- Démarrez votre projet – Kickstarter. *Kickstarter (en ligne)*. Consulté le 26 août 2016. Disponible sur internet : https://www.kickstarter.com/learn
- Git – Documentation. *Git SCM (en ligne)*. Consulté le 26 août 2016. Disponible sur internet : https://git-scm.com/doc
- Incubateur Jean Moulin : Accompagnement d'Entrepreneurs à Lyon. *Université Lyon3 Entreprendre (en ligne)*. Consulté le 26 août 2016. Disponible sur internet : http://entreprendre.univ-lyon3.fr/incubateur-jean-moulin/
- JavaScript APIs - Google Chrome. *Chrome Developer (en ligne)*. Consulté le 15 juillet 2016. Disponible sur internet : https://developer.chrome.com/extensions/api_index
- Manifesto for Agile Software Development. *Agile Manifesto (en ligne)*. Consulté le 26 août 2016. Disponible sur internet : http://www.agilemanifesto.org/
- Paper.js. *PaperJS (en ligne)*. Consulté le 4 août 2016. Disponible sur internet : http://paperjs.org/
- Référence JavaScript - JavaScript | MDN. *Mozilla Developer Network (en ligne)*. Consulté le 26 août 2016. Disponible sur internet : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference
- TC39 – ECMAScript. *ECMA International (en ligne)*. Consulté le 26 août 2016. Disponible sur internet : http://www.ecma-international.org/memento/TC39.htm
- The 4.4BSD Copyright. *FreeBSD (en ligne)*. Consulté le 26 août 2016. Disponible sur internet : http://www.freebsd.org/copyright/license.html

Annexes

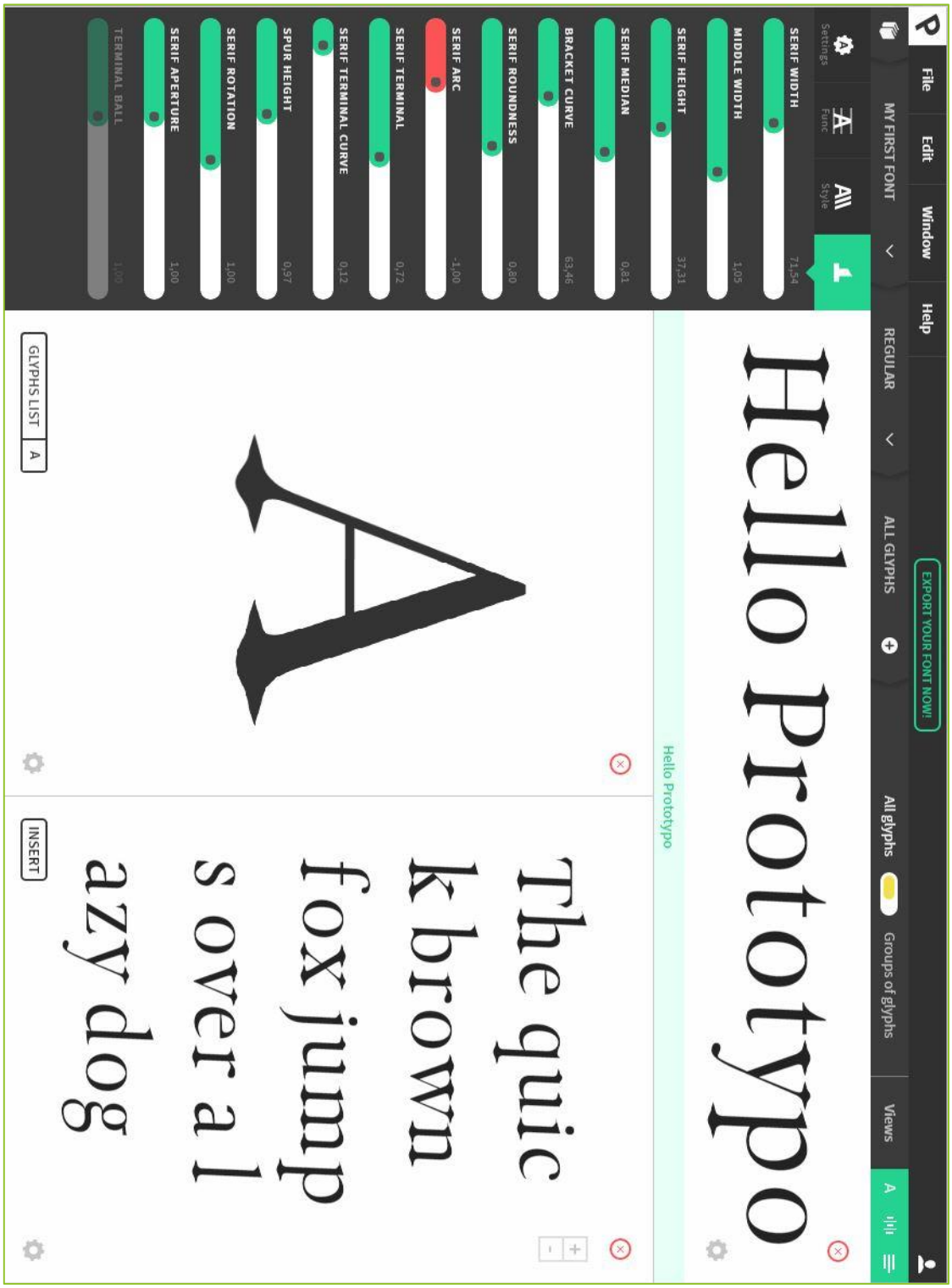
1) Image de l'interface de l'application Prototypo

2) Image de l'interface de l'extension Google Chrome pour Prototypo

3) Image de l'interface de listage de fontes

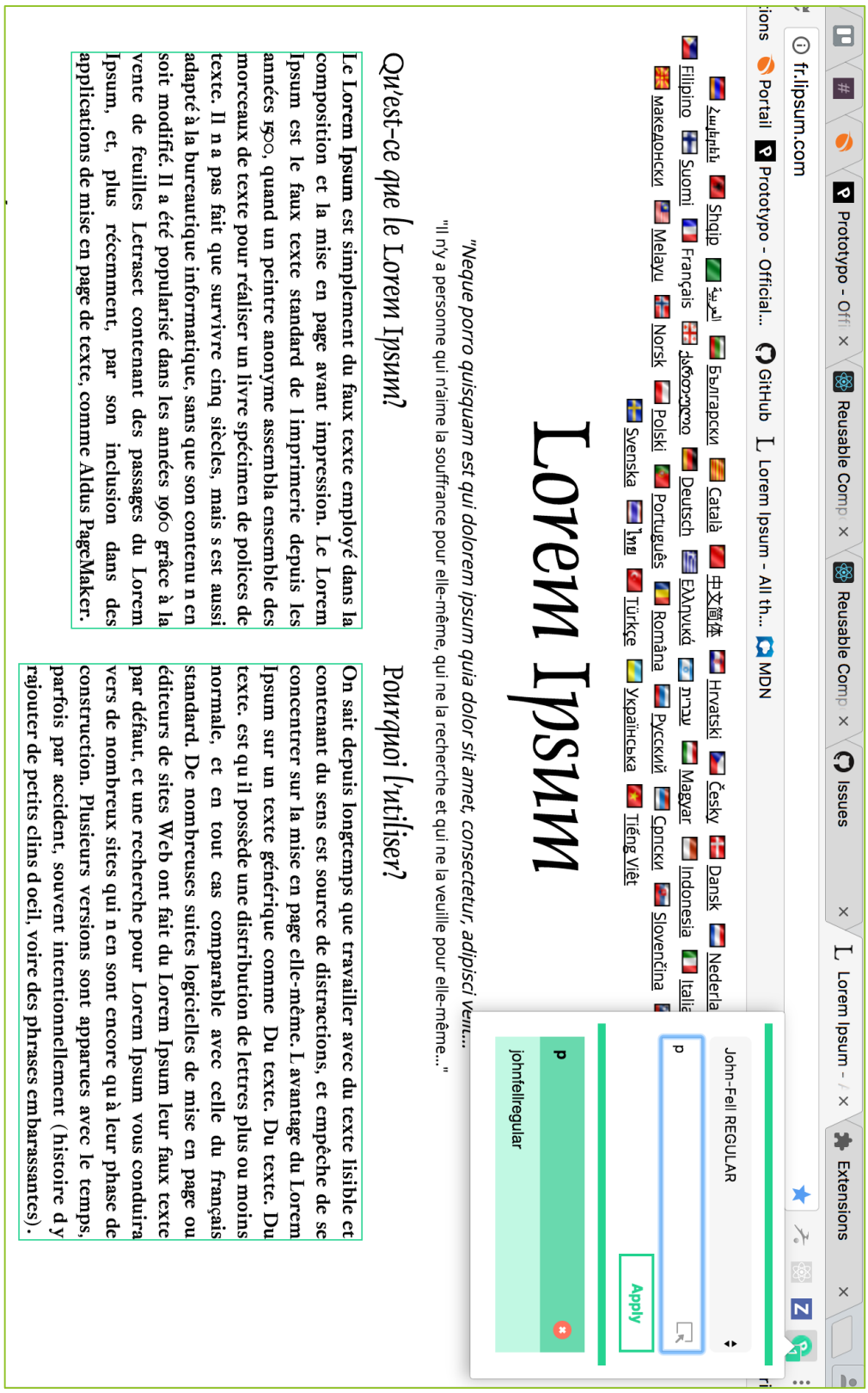
4) Exemple de code d'un composant React

1) Image de l’interface de l’application Prototypo



Capture d'écran de l'interface utilisateur de Prototypo

2) Image de l’interface de l’extension Google Chrome pour Prototypo



Capture d’écran de l’extension Google Chrome de Prototypo

3) Image de l’interface de listage de fontes

Capture d’écran de l’interface de listage de fontes

4) Exemple de code d’un composant React

```
/**
 * Component : a regular letter
 * @extends React.Component
 */
export default class RegularLetter extends React.Component {
    constructor(props) {
        super(props);
        this.shouldComponentUpdate = PureRenderMixin.shouldComponentUpdate.bind(this);

        // function bindings
        this.selectLetter = this.selectLetter.bind(this);
    }

    componentWillMount() {
        this.client = LocalClient.instance();
        this.lifespan = new Lifespan();
    }

    componentWillUnmount() {
        this.lifespan.release();
    }

    selectLetter() {
        this.client.dispatchAction('/store-value', {uiWordSelection: this.props.index});
        this.client.dispatchAction('/update-letter-spacing-value', {
            letter: this.props.letter.charCodeAt(0),
            valueList: ['advanceWidth', 'spacingLeft', 'spacingRight', 'baseSpacingLeft', 'baseSpacingRight']
        });
    }

    render() {
        return (
            <span className="letter-wrap" onDoubleClick={this.selectLetter}>
                {this.props.letter}
            </span>
        );
    }
}
```

Capture d’écran d’une partie du code d’un composant React

Rapport de stage - Société Prototypo

Evolution d'une application web d'édition typographique : Prototypo

Elève : Martin BOLOT – IUT Informatique Lyon 1 – Année Spéciale Promotion 2015/2016

Mots-clés : Prototypo, Typographie, JavaScript, React, HTML, SCSS, Application web, User Interface, UX Design

Outils technologiques : JavaScript, React, HTML, SCSS, PaperJS, NodeJS

Résumé : Ce stage s'est déroulé au sein de Prototypo, une start-up qui développe une application d'édition typographique. Elle permet de créer sa propre police de caractères en se basant sur des gabarits proposés par l'entreprise. L'application est entièrement écrite en JavaScript et utilise principalement la bibliothèque React pour la gestion de son interface utilisateur. Les missions réalisées pendant le stage visaient à améliorer l'expérience utilisateur et administrateur avec l'application. Les tâches effectuées incluent la maintenance d'une extension Google Chrome, le création d'un serveur NodeJS de listage de fichiers et d'une interface de listage de fontes, la création de tutoriels interactifs avec React et la manipulation de PaperJS dans le but de faire remonter un dysfonctionnement particulier aux développeurs de la bibliothèque.