Plan du rapport de stage :

Page de couverture  
Sommaire  
Intro

* Stage trouvé par intérêt pour le JS et obtenu en partie grâce au projet tuteuré de cette année
* Micro pitch : application pour graphistes d’édition de fontes
* Aide à l’évolution de l’application + missions annexes
* Promesse de plonger en profondeur dans la gestion d’une interface client complexe
* Déjà travaillé dans le milieu du web mais première expérience en startup, à l’incubateur Jean Moulin
* Annonce du plan

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

### a) Présentation de l’entreprise

* Présentation de Prototypo (historique + localisation + etc)
* Présentation de l’équipe et des compétences de chacun
* Environnement startup (incubateur), méthodes de travail différentes de celles observées en PME précédemment
* Fonctionnement entièrement en open-source

### b) Technologies web mises en oeuvre

* Utilisation des dernières technologies web FE et BE au service de l’édition de fontes
* Bases correctes autodidactes en JS, néophyte complet face à la typographie
* Familiarisation progressive avec les notions fondamentales de typographie, jamais bloquantes pour progresser dans le développement de l’application
* Prises de marques avec React, bibliothèque qui sera fondamentale pour tout le travail de développement effectué
* Découverte progressive de Paper.js en vue d’une mission particulière visant à la reproduction d’un bug pour obtenir un « reduced testcase » à soumettre au créateur de la bibliothèque

## II) Amélioration constante de l’expérience utilisateur et administrateur

### Evolution d’une extension Google Chrome

* Reprise d’un code existant afin de mieux correspondre au standards définis par Google pour la création d’extension (utiliser l’API « popup » au lieu d’injecter du balisage HTML dans les pages utilisateurs)
* But : faciliter l’interopérabilité (récupération du code) pour une compatibilité Firefox (minimiser la charge de travail pour rendre l’extension compatible)
* Mise en jeu de l’API fournie par Google + communication avec l’application
* But pas tout à fait atteint car les tests concernant la version Firefox n’ont pas été concluants

### Création d’un outil d’affichage de fontes

* But : avoir un affichage de toutes les fontes exportées par les utilisateurs de manière à pouvoir les monitorer => soit détecter les problèmes, soit repérer des créations pouvant faire la promotion de l’outil
* Utilisation d’Express, bibliothèque serveur pour NodeJS qui permet de récupérer les fontes puis de les envoyer au client
* Côté client, JavaScript classique pour une interface utilisateur minimaliste

### Soutien pour finaliser la nouvelle version de l’interface utilisateur

* A l’approche de la date de mise en recette de la nouvelle version de l’interface utilisateurs, beaucoup de petites tâches à effectuer / bugs à corriger
* Contexte adapté à la prise en main de l’outil, à l’apprentissage du fonctionnement de l’application et de son interface
* Formation aux procédures de gestion de versions avec GIT, différentes de celles que j’avais pu utiliser auparavant

### Tentative de déplacement d’une opération complexe du serveur au client

* Problématique : l’export des polices de caractère peut nécessiter une opération de « merge » (mise à plat) si demande de la part de l’utilisateur. Elle est faite côté serveur mais l’effectuer côté client augmenterait la fiabilité et diminuerait le temps de réponse. Malheureusement, un bug existait dans la précédente implémentation de cette opération, qui avait été mise de côté.
* Le but était de tenter de faire fonctionner l’opération du côté client et, en cas d’échec, déceler le bug responsable
* Après réussite de l’opération (il a fallu collaborer pour repérer les paramètres des fontes à modifier pour faire marcher l’export), constat de problèmes lors de la mise à plat : certaines formes ne sont pas supportées (pour la mise à plat) par la bibliothèque (Paper.js) en charge de l’opération
* Nouveau but : faire remonter le bug aux créateur de la bibliothèque utilisée (développée en open-source) via github. Toutefois, nécessité d’obtenir avant cela un « reduced testcase », cas de test le plus simple possible mettant en avant le bug rencontré afin de permettre à des développeurs tiers de le repérer puis de le résoudre

### Développement de tutoriels pour faciliter l’expérience utilisateur

* La complexité de l’application ne devant pas être un frein pour un nouveau client potentiel, certains points (les plus techniques) doivent être expliqués à l’utilisateur sous la forme de panneau superposés à l’interface qui ne doivent être affichés qu’à la première utilisation des fonctionnalités concernées
* Utilisation d’un contenu additionnel venant se greffer sur React, la bibliothèque utilisée pour gérer l’interface utilisateur, nommé React-joyride
* Gestion de la persistance des données, automatique dans l’application pour sauvegarder constamment les données utilisateur
* Apprentissage des bonnes pratiques pour la séparation de la logique et de la présentation, parfois ambiguë au sein d’un même composant React

## III) Application web et typographie : entre rigueur et liberté

* Interface complexe : nécessité de rigueur car maintenabilité fragile
* Interface web pour la typographie : possibilités nombreuses dans le but d’offrir de la créativité aux utilisateurs

## Conclusion

* Découverte d’une nouvelle manière de travailler dans un contexte d’application web
* Découverte des bases de la typographie
* Observation des rouages d’un framework dédié à l’interface utilisateur, de la mise en place de celui-ci au service d’une application (ce qui correspond aux attentes émises)
* Equipe réduite donc collaboration totale
* Rencontre de problématiques observables avec une utilisation poussée de certaines bibliothèques
* Progression avec React et prise d’autonomie, participation active au développement de l’application et à la résolution de bugs

## Annexes :