REMERCIEMENTS

La réalisation de ce projet n'aurait pas été possible sans l'aide de plusieurs personnes :

En commençant par remercier tout d'abord Monsieur Dominique Vilain et Monsieur Pierre-Antoine Delnatte pour la confiance qu'ils m'ont donné en acceptant que je réalise ce projet, pour les conseils qu'ils ont pu me donner et pour leur patience.

Je remercie aussi Monsieur Samuel Cardillo Lespes, employé chez Google, pour l'aide précieuse qu'il m'a donné sur Polymer.

Enfin, je remercie ma famille : mes parents, ma copine et mes proches de m'avoir soutenu tout au long de la réalisation de ce projet.

INTRODUCTION

Au terme de mes trois années d'études, je vous présente ici le rapport écrit de mon Projet de fin d'études. Au cours de ce rapport je vous présenterai les différentes étapes de développement d'une application exécutable multiplateforme : Analiz. Ce projet m'a permis d'élargir mes connaissances sur le langage JavaScript et Node.js et surtout de découvrir une nouvelle notion dans le développement web : l'utilisation des Web Components au sein du framework Polymer. Je tiens particulièrement à vous faire part des connaissances que j'ai acquises sur ces nouveaux outils durant le processus de développement.

L'application Analiz permet d'effectuer la validation, l'analyse et la synthèse de productions web dans une seule application. Cette application a d'abord été pensée pour les étudiants et les professeurs de la section Web mais peut convenir à tout développeur ayant besoin de procéder à de tels tests.

CHAPITRE 1

POURQUOI ANALIZ ?

L'idée vient à la base de mes deux principaux professeurs : Dominique Vilain et Pierre-Antoine Delnatte. Au début de l'année scolaire, ils ont présenté à l'ensemble de la classe la nécessité d'avoir une application pouvant servir tant aux élèves de la section qu'aux professeurs et l'ont proposée comme sujet de projet de fin d'études. Même si le principe de l'application ne m'a pas directement emballé, les outils web imposés pour le développement (NW.js et Node.js) ont attisé ma curiosité d'apprendre à les utiliser.

En effet, je vois la réalisation de ce projet de fin d'études comme un défi me permettant de tester mes capacités à apprendre de nouvelles technologies telles que les Web Components et à les appliquer au sein d'un projet concret.

NODE.JS

La création d'Analiz m'a permis de réaliser ce challenge tout en apprenant à développer sous Node.js, qui est une technologie JavaScript potentiellement très utile pour mon futur professionnel. Le JavaScript devenant de plus en plus important dans le monde du développement web, je me suis décidé à en apprendre le plus possible, au travers d'Analiz, pour pouvoir mettre en avant cet atout dans le monde du travail.

LA LIBERTÉ DU NAVIGATEUR

Une autre consigne était de réaliser une application exécutable à l'aide du framework Node Webkit, qui a été ensuite remplacé par Electron au cours du développement. Ces outils permettent de développer des applications natives en HTML et JavaScript dans un environnement Chromium[1] J'ai donc repris ce point technique à mon avantage car cela signifiait que je n'aurais pas de problème de compatibilité entre navigateurs. Mieux encore, je pouvais expérimenter des technologies comme Polymer ne fonctionnant que sur des navigateurs modernes.

POLYMER

Polymer fut une passionnante découverte. Avant de le proposer pour le développement d'Analiz, je n'avais lu que quelques articles à son propos mais son concept m'intéressait fortement. Il m'a permis de découvrir la notion de Web Components au travers d'un tout nouveau framework en constante évolution : en effet, la version 1.0 de Polymer est sortie au beau milieu de mon processus de développement. J'ai donc travaillé au sein d'une communauté active avec des technologies changeant au jour le jour. Ce fut une expérience passionnante de développer au sein de cette communauté en participant au développement avec ses intervenants grâce à Github et, par le hasard des choses, de rencontrer un des membres de l'équipe de développement de Polymer.

1 Chromium est un navigateur Web libre qui sert de base aux navigateurs Web propriétaires Google Chrome

CHAPITRE 2

BUT PREMIER DE L'APPLICATION

Comme dit dans l'introduction, mon Projet de fin d'études consiste en le développement d'une application permettant d'effectuer des rapports d'analyse sur des pages web statiques.

UNE APPLICATION SIMPLE D'UTILISATION

Monsieur Delnatte et Monsieur Vilain, que je considère ici comme mes clients, étant tous les deux enseignants dans le domaine du Web, souhaitaient que cette application serve de référence pour la validation et l'analyse des productions web réalisées par les élèves. Le but premier d'Analiz est donc de simplifier l'accès à ces outils d'analyse au sein d'une seule et unique interface, facile à utiliser pour des néophytes.

C'est pour cela que, lors de l'établissement du cahier des charges, mes clients m'ont donné quelques contraintes techniques pour le développement d'Analiz. L'application doit être utilisable au moins partiellement en hors-ligne et utilisable comme une application native à leur système d'exploitation. Monsieur Delnatte m'a donc conseillé de la réaliser en utilisant

le framework NW.js[2], permettant, comme dit précédemment, de créer des applications de bureau multiplateformes en JavaScript. J'ai dû, pour des raisons techniques que j'aborderai dans un prochain chapitre, passer de NW.js à Electron qui est un outil analogue.

UNE APPLICATION MODULAIRE

Il y a énormément d'attentes et d'idées concernant les fonctionnalités d'Analiz. Mes clients m'ont fourni un cahier des charges contenant une liste non exhaustive des différentes analyses que l'application pourrait effectuer :

- Validation de documents HTML et CSS
- Analyses syntaxiques de documents
- Affichage du plan du document
- Rapports sur les feuilles de styles
- Rapport sur les contrastes
- Rapport sur les images utilisées

Dans le but de rendre l'application évolutive et dynamique chaque analyse est en fait un plugin externe pouvant être développé par un tiers, cela permet donc l'ajout de nouvelles fonctionnalités assez facilement. Ces plugins utilisent des modules Node.js permettant de faire l'analyse et les formatent pour les envoyer à Analiz qui s'occupe du rendu.

2 **Anciennement appelé Node Webkit**

CHAPITRE 3

LE PUBLIC CIBLE

A l'origine, l'idée de l'application vient de la constatation par Monsieur Delnatte et Monsieur Vilain que les étudiants arrivant en premier bachelier ne sont pas tous familiarisés avec le Web et ses règles. En effet, la plupart des étudiants qui s'inscrivent à l'INPRES ne se destinent pas à l'option Web et souvent ne comprennent pas grand chose à cette technologie. C'est pourquoi ils ont pensé à Analiz, un outil leur permettant de réaliser facilement des tâches souvent peu ou mal comprises.

LES ÉLÈVES

Tous les élèves d'infographie seront ainsi amenés à utiliser Analiz comme une référence pour la validation et l'analyse de leurs productions web au sein de l'école.

D'une part, l'application s'adresse aux étudiants novices par rapport aux technologies du Web qui recevront probablement la liste de filtres à appliquer par leur professeur et ne chercheront peut-être pas à comprendre les différentes options qui leur sont proposées. Ce qui leur importe sont les résultats de leurs analyses, qui doivent être clairs et précis. La page qu'ils chercheront principalement sera donc la page d'audit. Un code couleur des différentes catégories et une visualisation rapide du nombre d'erreurs leur seront donc indispensables.

D'autre part, l'application s'adresse aussi à un type d'utilisateur qui est déjà familier avec les technologies du Web. Pour lui, l'application peut se révéler un nouveau terrain de jeu, une nouvelle façon d'apprendre de façon autodidacte : en lançant à la volée plusieurs analyses et en modifiant quelques options, il pourra, plus rapidement qu'avec la technique traditionnelle, se rendre compte de ses erreurs et des modifications qu'il pourrait apporter pour enrichir son projet, à la manière d'un petit chimiste qui joue dans un laboratoire et peut toucher librement à tout sans risque pour son projet. Et ceci de manière beaucoup plus rapide et visuelle. La page qu'ils consulteront le plus fréquemment sera vraisemblablement celle des configurations des plugins.

LES PROFESSEURS

Par ailleurs, les professeurs pourront utiliser Analiz comme outil normatif de correction et d'évaluation à appliquer aux productions Web des étudiants. Les tests et tâches réalisés par Analiz sont fiables et stables, ce qui permet une continuité dans l'évaluation tant par les élèves que les professeurs.

La première utilité qu'ils trouveront dans l'application est de donner aux élèves la liste des analyses ainsi que les options choisies pour ces dernières. Cette liste sera donnée oralement ou par écrit lors d'un cours. Ceci permettrait à tous les élèves de réaliser le même travail, selon les mêmes critères de cotation et ou de correction. Ce système pourra éventuellement, selon les desiderata du professeur, participer à une grille de cotation.

La seconde est de vérifier les travaux des élèves de façon rapide et automatisée. Un gain de temps pour le professeur et une objectivisation dans la cotation.

UN DÉVELOPPEUR LAMBDA

Enfin, dans une moindre mesure, l'application peut s'adresser à toute personne qui ne ferait pas partie de l'INPRES et qui, travaillant seule ou sans une aide pouvant le guider et le corriger se verrait aidé grâce à cette application. D'une certaine manière, Analiz pourra jouer le rôle du professeur et aura donc un rôle essentiellement correctif, similaire au professeur guidant l'étudiant novice.

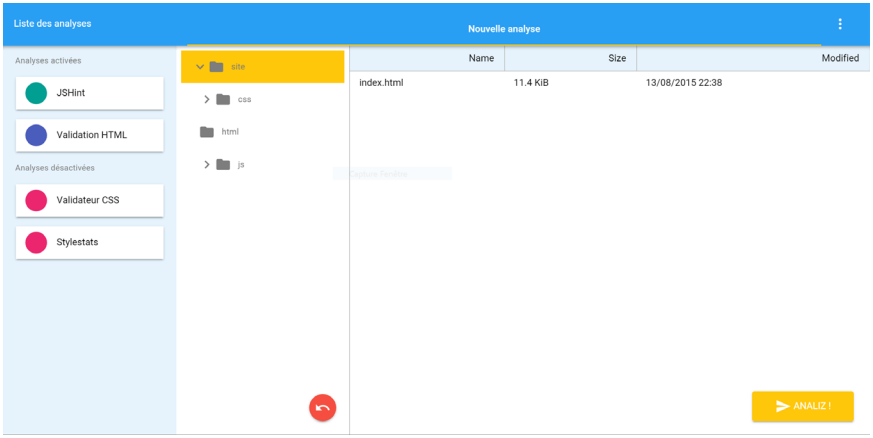
CHAPITRE 4

INSPIRATION

Il existe quelques applications Web permettant l'audit de sites web. Je pense à site-analyzer.com, woorank.com parmi d'autres. Néanmoins, ces applications d'audit déjà existantes ne permettent d'analyser et d'évaluer des sites Web que selon leurs paramètres prédéfinis. La plus-value d'Analiz est de pouvoir choisir, réinitialiser et adapter facilement ces critères à chaque analyse. Je n'ai rien pu retirer de concluant et d'utile pour la mise en page et le développement d'Analiz de ces sites vu la différence notable de mise en forme des résultats entre ceux-ci et mon application ainsi que l'absence de modularité des choix d'analyse de ces applications concurrentes. (cf: Annexe 1)

Par ailleurs, Polymer m'a permis d'intégrer le Material Design créé par Google à mon projet grâce à une librairie d'éléments déjà construite. C'est grâce à cette ressource que j'ai trouvé quelques idées d'éléments. La structure d'Analiz provient d'un élément nommé Drawer panel, c'est grâce à cette structure que j'ai divisé l'écran de l'application en deux sections :

- La première est le **bloc latéral** de gauche contenant les informations sur les analyses et leur configuration;
- Le second est le **bloc principal** contenant les données sur les fichiers à analyser et sur leur résultat suite à l'analyse.



Application Analiz

Les règles du Material Design ont été initialement pensées pour des applications mobiles, et celles-ci étant à leurs balbutiements dans le Web bureautique actuel, je n'avais aucun exemple concret ou inspiration potentielle de site préexistant qui aurait pu m'aider pour le design d'Analiz. J'ai utilisé cette absence de repère pour laisser libre cours à mon imagination, néanmoins dans le respect des règles du Material Design, qui sont extrêmement bien documentées par Google[3].

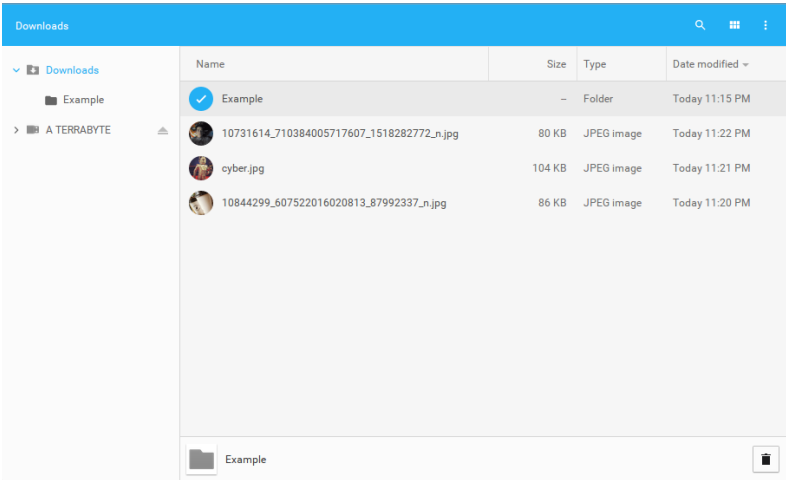
Pour avoir une application esthétique, personnelle mais qui devait avant tout respecter la charte graphique du Material Design, j'ai trouvé mon inspiration sur le site materialup.com. Ce site communautaire référence une multitude d'idées de graphistes professionnels ou amateurs sur le Material Design, le faisant vivre et évoluer.

3 https://www.google.com/design/spec/material-design/introduction.html

ÉCRAN DE CONFIGURATION D'ANALYSE

Les applications web citées précédemment proposaient une analyse à partir d'une URL d'un site déjà en ligne. J'ai pensé qu'il serait plus profitable et plus simple pour le public cible que nous visions de pouvoir le faire sur des fichiers locaux qu'il donnerait à l'application. Cette sélection devait être intuitive et implicite pour l'utilisateur, c'est pourquoi j'ai permis à l'utilisateur un ajout de dossiers via drag'n drop (ou glisser-déposer).

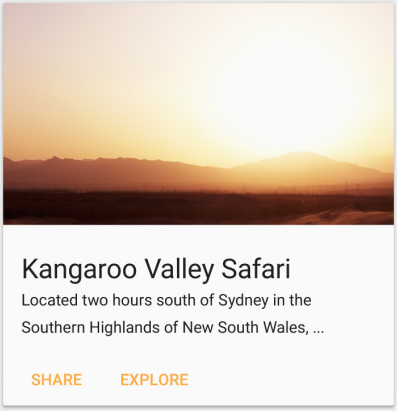
Les fichiers étant locaux, l'utilisateur doit pouvoir avoir un support visuel du dossier qu'il veut analyser. Je me suis inspiré de l'interface du gestionnaire de fichiers que l'on retrouve dans le système d'exploitation Chrome OS. Ce dernier étant entièrement réalisé en Material Design, il devenait donc une source d'inspiration très intéressante pour moi. J'ai eu l'opportunité de tester ce design et de me rendre compte de certaines réactions telles que les animations ou la navigation dans l'arborescence des dossiers qui semblaient convenir parfaitement à mon projet.



Gestionnaire de fichier sur un Chromebook

ÉCRAN DES SYNTHÈSES DES RÉSULTATS

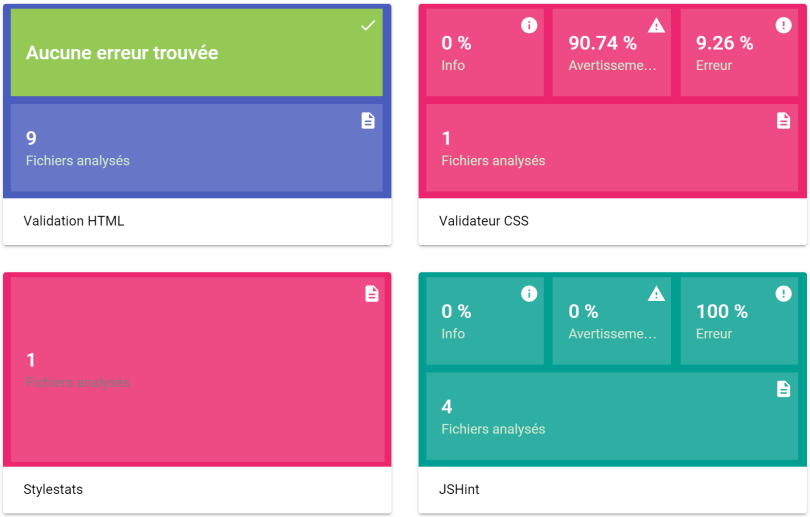
Contrairement aux applications concurrentes, mon application permet un affichage des résultats par plugins. Je tenais à garder leur idée de trier ces résultats en fonction de leur catégorie d'analyse. Ainsi, j'ai permis une compréhension rapide et claire des résultats de chaque plugin grâce à ce que Google appelle "une collection de cartes" dans les guidelines du Material Design.



Exemple de carte à la Material Design

Ces dernières sont colorées, chaque couleur correspondant à une catégorie d'analyse, et contiennent une brève synthèse des résultats de l'analyse de ce plugin (le s analyses plus complètes pouvant être obtenues en cliquant sur la carte associée). Si les guidelines m'ont permis de mettre en page les plugins par cartes, c'est grâce à materialup.com que j'ai trouvé comment styliser la synthèse qu'ils contiennent. En effet, beaucoup de personnes utilisent la couleur de fond d'un élément avec une

teinte plus claire ou plus foncée pour créer un nouveaux système de carte à l'intérieur même d'une carte, permettant une intégration des synthèses dans un environnement graphique proche du Material Design utilisé dans toute l'application.



Cartes de synthèse des résultats dans Analiz

ÉCRAN DES RÉSULTATS

Chaque feuille de résultat est propre à sa catégorie d'analyse car elles proviennent toutes de modules différents et donc codées par des développeurs différents ayant un style différent ainsi qu'un but informatif différent. Ainsi, on obtient par exemple une liste d'erreurs si l'on sélectionne un module de validation, ou un rapport d'analyse des résultats de la carte si on demande des statistiques sur le code.

Pour garder une continuité visuelle entre les pages de résultats, je me suis intéressé au fonctionnement des animations

entre les pages avec Polymer. C'est en lisant la documentation sur les animations que j'ai pu voir les exemples proposés dont un correspondait à ce que je voulais faire[4].

L'écran de résultats est obtenu en cliquant sur la carte d'intérêt sur l'écran de synthèses de résultats (ou "Audits"). Il est propre à chaque carte et reprend une synthèse très complète de l'analyse. Chaque écran est donc différent, mais son style graphique est constant : le fond est là quelques teintes plus claire de la carte sélectionnée et la feuille de résultat est bordée de la couleur de base. (cf: Annexe 2)

4 https://elements.polymer-project.org/bower_components/neon-animation/demo/grid/index.html

CHAPITRE 5

LA RECHERCHE D'INTERFACE

La mise en page d'Analiz fut un réel défi pour moi.

Tout d'abord, car il s'agit d'une application et non d'un site. En effet, les enjeux ne sont pas les mêmes et demandent de penser l'interface avec un point de vue complètement différent. Une application n'est pas forcément connectée au Web, il faut une connexion internet pour la télécharger mais elle n'est pas obligatoirement nécessaire pour l'utiliser. Pour un site, c'est indispensable. En terme d'usage, une application n'a pas le même but qu'un site mobile. Une application a plutôt une fonction d'action et non d'information comme pourrait l'avoir un site.

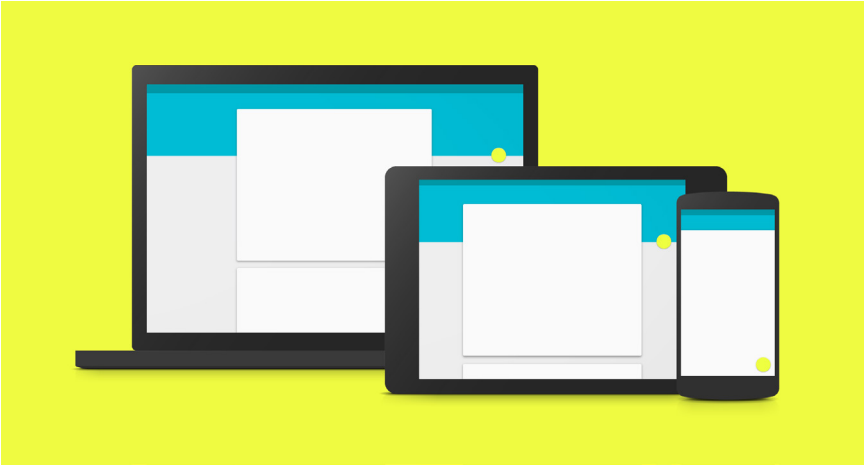
Mais aussi parce que cette dernière doit respecter les guidelines de design de Google : le Material Design. Ces dernières étant très nombreuses, j'ai du en apprendre les règles et le fonctionnement grâce à une documentation magnifiquement bien écrite et illustrée d'exemples concrets[5]. J'ai donc eu quelques contraintes quant à la mise en page de mon application mais

5 https://www.google.com/design/spec/material-design/introduction.html

aucune d'entre elle n'a posé problème, que du contraire ces lignes de conduite m'ont souvent aidé à réaliser une mise en page accessible et fluide.

POURQUOI AVOIR CHOISI D'UTILISER LE MATERIAL DESIGN ?

Google a créé le Material Design pour avoir un système permettant à l'utilisateur de pouvoir avoir la même expérience de navigation quelque soit l'environnement dans lequel il est utilisé.



Exemple de Material Design

Le but premier de ce design est de mettre en avant le contenu de manière consistante et réelle. Pour cela, il représente le contenant comme des feuilles de papier qui seraient superposées les unes sur les autres en jouant avec des effets de matière et d'ombrage. Le contenu est donc l'encre sur ces différentes couches. Le dernier point important du Material Design est ses animations. Elles se veulent significatives et appropriées à

l'action de l'utilisateur lui permettant de rester concentré sur le contenu tout en maintenant une continuité cohérente entre les éléments.

De plus, Polymer contient une librairie d'éléments créée à partir des règles du Material Design. C'était donc une opportunité à ne pas manquer.

MAQUETTAGE

Lors de la toute première recherche graphique, j'ai pensé à certaines fonctionnalités absentes dans la dernière version de l'application. Soit parce qu'elles se sont avérées non nécessaires, soit parce qu'elles ont été prévues comme futures améliorations à Analiz.

J'avais donc pensé l'interface pour faciliter l'utilisation pour les néophytes du Web avec un mode basique et un mode avancé. Il aurait été possible de changer de mode grâce à un bouton activable en haut à droite (constamment présent).

- Dans le **mode basique**, l'analyse est un ensemble de règles établies par défaut dans Analiz. L'utilisateur peut donc voir directement une analyse de ses fichiers sans pour autant connaître les outils d'analyse ainsi que les options à choisir.
- Dans le **mode avancé**, l'utilisateur peut donc choisir les analyses ainsi que leurs options

Pour mieux visualiser les différents écrans et différentes tâches auxquels l'utilisateur serait confronté, j'avais mis au point des wireframes des deux modes de l'application (cf : Annexe 3). Lors du développement et de quelques tests d'utili-

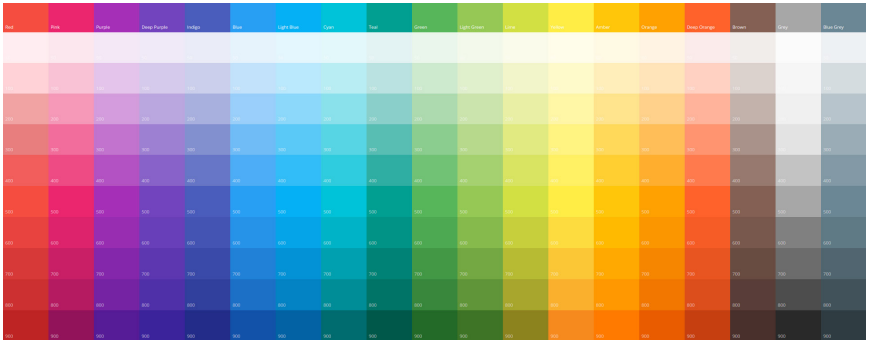
sations, je me suis rendu compte qu'ajouter un mode basique pourrait compliquer la compréhension de l'utilisateur étudiant en ne lui montrant pas directement certaines options qu'il devrait activer suite à une demande du professeur.

LES COULEURS

Toutes les couleurs qui sont utilisées dans Analiz proviennent de la palette de couleur donnée par le Material Design. Cette palette est composée de 19 couleurs ayant chacune 14 teintes.

Lors du choix d'une couleur pour une application, il faut choisir une teinte de base et une autre couleur secondaire appelée couleur accent. Elle est à utiliser pour les boutons d'actions et les éléments d'état tels que les check-boxes, les sliders, …

Je n'ai malgré tout pas utilisé l'entièreté de la palette de couleurs : en effet, pour les daltoniens, certains couples de couleurs étaient perçus comme identiques ou trop ressemblant (ex : mauve et indigo). Je n'ai donc gardé qu'un ensemble de couleurs assez contrastées les une des autres.



Palette de couleur Material Design

LE LOGO

Avant même que je commence le projet, Monsieur Delnatte avait déjà réalisé une première version du logo. Cette dernière représentait un microscope vu de profil mais de façon minimaliste. J'ai longtemps hésité à changer le logo, mais après avoir testé plusieurs personnes sur sa signification je me suis décidé à garder celui proposé comme base de travail. En effet, toutes les personnes reconnaissaient bien le microscope et les mots-clés revenant le plus souvent était "recherche" et "analyse".

Toujours dans le but de garder une ligne directrice dans mon design, j'ai décidé de retravailler le logo dans le style Material Design. J'ai donc décomposé chaque forme du logo en les imaginant comme des feuilles de papier grâce à des effets d'ombre.

Ancien et nouveau logo

ICÔNES

Les guidelines Material Design apportent un lot d'icônes assez complet permettant de répondre à presque tous les besoins rencontrés dans une application Web. Cela m'a permis d'utiliser un set d'icônes cohérent avec l'ensemble du design mais surtout entres elles. En effet ces icônes répondent toutes aux quelques règles de forme et d'espace que le Material Design impose.

TYPOGRAPHIE

Encore une fois, je n'ai pas eu trop de choix possibles, Google ne permettant que l'utilisation de sa police phare : Roboto. Même si cette police est décriée pour n'être qu'une mauvaise réplique de Helvetica ou est surnommée "Le Arial de Google", mon leitmotive était de respecter au mieux les règles du Material Desgin pour rester cohérent dans mon choix graphique, je n'ai donc pas eu d'autre choix que de prendre cette police.

Roboto Light

Roboto Regular

Roboto Bold

ADAPTABILITÉ

Analiz est une application et non un site, je l'ai déjà dit mais c'est un point que j'ai pris en compte durant toute la mise en place de l'interface. Il ne sera donc pas utilisable sur un appareil mobile mais doit proposer pour tout contexte de redimensionnement de la fenêtre une interface lisible et dans la mesure du possible utilisable.

C'est pendant cette phase de développement que j'ai découvert un grand avantage des Web Components : Certains éléments Polymer proposent d'eux même des comportements responsive. Par exemple, un menu latéral se cache tout seul en dessous d'une certaine largeur et affiche un bouton pour faire basculer son état de visible à caché.

QU'AI-JE APPRIS DE CETTE RECHERCHE D'INTERFACE ?

C'était la première fois pour moi que je travaillais avec des guidelines aussi spécifiques et heureusement pour moi, ces dernières étaient très détaillées. L'utilisation de Polymer m'a grandement aidé pour respecter le plus possible ces règles. J'ai appris beaucoup sur le placement intelligent des éléments d'action dans une application pour permettre à l'utilisateur de savoir où se trouve l'action primaire à sa tâche.

Le Material Design peut sembler très sévère mais permet une véritable unification d'interface entre différents supports mais aussi entre différentes applications. On peut même supposer que Google veuille faire une sorte d'apprentissage externe entre chaque application suivant ces règles.

CHAPITRE 6

POLYMER, ELECTRON ET NODE.JS

Je parle de Polymer depuis le début de ce rapport, mais à quoi sert-il réellement ?

Polymer est un framework créé par Google qui utilise une toute nouvelle spécification en cours de standardisation par le W3C : Les Web Components.

LES WEB COMPONENTS

Ces derniers changent complètement la façon dont nous réalisons des sites Web en ajoutant une approche modulaire au HTML. A la manière d'un module Node.js, les développeurs peuvent créer, partager et importer des modules qui sont ici

des balises HTML personnalisées et réutilisables. Il est donc possible d'assembler plusieurs éléments appartenant au HTML déjà connu mais aussi provenant de source externe pour créer son propre élément HTML ayant des attributs, des propriétés CSS et un comportement propre à lui.

Les Web Components sont en fait un ensemble de nouvelles spécifications du W3C. Trois d'entre elles sont les piliers de cette technologie.

Premièrement, la **création d'éléments personnalisés** (Custom Elements) permet aux développeurs de définir leur propres éléments HTML personnalisés. Ils peuvent donc associer du code JavaScript avec les balises personnalisées et ensuite utiliser ces balises comme ils le feraient avec n'importe quelle autre balise standard.

Ensuite, **l'importation de ces éléments** (HTML Imports) est une façon d'inclure et de réutiliser des documents HTML dans d'autres documents HTML. Comme une balise <script> inclut du JavaScript externe dans une page, l'importation permet de charger des ressources HTML complètes. En particulier l'importation de Custom Elements externe.

Finalement, un **gestionnaire de DOM privé et public** (Shadow Dom) pour chaque Custom Elements leur permet d'obtenir un nouveau type de noeud HTML appelé «shadow root». Un élément ayant un noeud «shadow root» associé est appelé un «shadow host». Le contenu d'un «shadow host» n'est pas affiché, c'est celui du "shadow root" qui le sera à la place.

Depuis sa création, le W3C a travaillé sur d'autres librairies utiles et simplifiant la création d'éléments. On retrouve entre autres :

- Model Driven Views qui fournit un système d'affichage du contenu dynamique,
- Web Animations offre des APIs pour mettre en œuvre des animations complexes.
- Pointer Events qui unifie les événements souris, tactile et stylet.

Nouvelles spécificités W3C riment avec problèmes de compatibilité entre les différents navigateurs. C'est pourquoi un ensemble de polyfills ont été créés sous le nom « webcomponents.js[6] » pour permettre une compatibilité un peu plus large avec les dernières versions des navigateurs.

POLYMER

Comme je viens de le dire, les Web Components apportent tout un lot d'outils utiles mais pour ne pas me lancer la tête la première dans cette nouvelle technologie j'ai choisi d'utiliser un framework facilitant l'utilisation de ces outils. En effet, Polymer ajoute un tas de fonctions pratiques aux Web Components, à la manière d'un JQuery pour JavaScript,. De plus, Polymer contient un catalogue de Custom Elements très complet pour une application web en plus de ceux dédiés au Material Design.

6 https://github.com/webcomponents/webcomponentsjs

Au début du développement d'Analiz, Polymer était toujours en version 0.5. Il avait été présenté presque un an plus tôt lors de la conférence annuelle de Google. C'était encore une version contenant énormément de bugs et qui n'était pas très optimisée. Pourtant j'ai beaucoup aimé cette étape de développement, j'avais l'impression de ré-apprendre à développer en HTML. La documentation de Polymer et de certains éléments du catalogue n'étant pas complète il m'a fallu quelques temps pour comprendre le principe de base d'importation et de déclaration de Custom Element.

De plus, la version 1.0 de Polymer est sortie en plein milieu de mon processus de développement. Cette version étant enfin stable et surtout beaucoup plus rapide, le choix de recommencer mon développement sur une base saine n'a pas été très difficile. C'est aussi à ce moment-là que je suis passé de NW.js à Electron pour des raisons aussi de stabilité et de facilité d'intégration que je détaillerai dans le prochain sous-chapitre. Un autre point très positif de cette mise à jour, est l'apparition d'une documentation complète et composée d'exemples instructifs et inspirants.

Alors que je recommençais mon projet avec de nouveaux outils, j'ai eu l'occasion de rencontrer Samuel Cardillo Lespes. Employé chez Google et surtout membre de l'équipe de développement de Polymer. C'est avec plaisir qu'il s'est proposé de m'aider à mieux comprendre l'utilisation de Polymer. Son aide à d'ailleurs été très utile pour la compréhension des comportements et des éléments d'animations au sein de Polymer.

Je l'ai déjà mentionné mais j'ai vraiment aimé la dynamique créée par la nouveauté de ce framework. Les éléments

du catalogue sont mis à jour presque tous les jours en prenant en compte les retours des autres développeurs. J'ai moi-même pu participer à plusieurs demandes de correction de bugs ou d'ajouts de fonctionnalités.

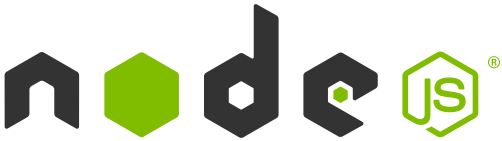
ELECTRON

Monsieur Pierre-Antoine Delnatte m'avait conseillé l'utilisation de NW.js comme compilateur d'applications pour la réalisation d'Analiz.

NW.js demandant un fichier HTML comme point d'entrée de l'application, je devais afficher un chargement à l'exécution de l'application le temps de charger le code JavaScript et de lancer un serveur Node.js pour faire tourner l'application.

Après en avoir discuté avec Monsieur Delnatte, il m'a parlé d'Electron qui était plus adapté à mon application. Dans Electron, le point d'entrée est un script JavaScript. Au lieu de fournir directement une URL, il faut créer une fenêtre de navigateur et ensuite charger un fichier HTML via l'API fournie. Cela crée donc directement un environnement serveur Node.js utilisable.

L'ENVIRONNEMENT NODE.JS



Analiz a été mon premier projet développé entièrement sous Node.js, cela m'a permis de découvrir cette technologie et surtout d'apprendre d'importants concepts de programmation.

Une notion qui m'était jusque là inconnue est celle d'exécution synchrone et asynchrone du code. Il m'a fallu du temps et surtout l'aide du plugin Node.js «Async» pour réussir à gérer mes données durant la gestion de fichiers ou encore durant le processus d'analyse. A l'heure actuelle je ne suis toujours pas tout à fait familier avec ce concept et je compte en apprendre plus pour améliorer les performances d'Analiz.

Même si il m'était déjà arrivé de prendre un script ou deux sur un site ou d'utiliser une librairie de fonctions pour réaliser certaines tâches, l'utilisation de dépendances externes grâce à un gestionnaire comme npm fut très intéressante. Il y a presque toujours un plugin pour ce que l'on veut faire ou ce dont on a besoin comme le montre l'exemple cité plus haut du module «Async» qui m'avait été proposé par Monsieur Delnatte. C'est d'ailleurs ce système de modules que nous avons utilisé pour les plugins d'Analiz.

CHAPITRE 7

API DE PLUGIN EN NODE.JS

Chaque analyse que l'on retrouve dans Analiz est en fait un module Node.js développé en dehors de l'application. Cela permet de rendre l'application extensible à souhait en fonction des nouveaux outils d'analyse qui pourraient sortir. J'ai développé quelques analyses mais j'ai surtout préparé l'API permettant à un développeur externe de créer ses propres analyses. Cette API consiste en deux propriétés que chaque plugins doit exposer à Analiz qui sont une **configuration** et une **fonction** faisant l'analyse.

J'ai également créé un plugin générique[7], servant de base de travail pour la création d'autres plugins.

LA CONFIGURATION

Chaque plugin donne des informations de base qui permettent à Analiz d'afficher correctement ses options et les résultats de l'analyse. Parmi ces informations, on retrouve le nom

7 https://github.com/analiz-app/analiz-plugin-generic

du plugin, les extensions de fichiers qu'il peut analyser, sa catégorie, le type d'affichage pour les résultats et les options que l'utilisateur choisit pour personnaliser son analyse.

Pour l'instant, je prends en compte deux types d'affichage pour les résultats :

- Une **validation de code** à un ensemble de règles, le résultat est donc **composé d'erreurs** pouvant avoir jusqu'à trois niveaux : Informatif, avertissement et erreur.
- Une **analyse du code** pour en retirer un **rapport de statistiques**, il faut donc faire un rapport écrit en HTML dans le plugin et le renvoyer comme résultat.

LA FONCTION D'ANALYSE

C'est cette fonction qui fait tout le travail de l'application, elle reçoit les fichiers à traiter et une liste des options que l'utilisateur a choisi.

Elle doit ensuite utiliser un module Node.js externe pour les analyser. Finalement, il faut formater les résultats pour qu'ils correspondent à ce qu'Analiz exige comme structure des données et les renvoyer après chaque analyse de fichier.

CONCLUSION

Comme je l'ai répété tout au long de ce rapport, la réalisation d'Analiz fut une expérience d'apprentissage de nouveaux outils de développement. Il est donc évident que je ne maîtrise pas encore parfaitement ces outils et que l'application peut encore être améliorée tant au niveau de ses fonctionnalités qu'au niveau de la qualité de son code.

J'ai énormément appris durant ce développement, je me suis d'ailleurs rendu compte de certaines "erreurs" de débutant une fois le développement fini. C'est pourquoi je compte procéder à une refactorisation de certaines parties d'Analiz afin d'offrir une application propre et efficace à mes futurs utilisateurs.

Mon objectif premier était bien entendu de réaliser une version de l'application permettant de réaliser quelques analyses mais surtout d'avoir un système de plugins fonctionnels. Pour l'instant il n'est possible que de modifier les plugins installés dans Analiz qu'au moment de la compilation de l'application. L'ajout d'un centre de gestion des plugins pour pouvoir en installer directement depuis l'application est prévu très prochainement.

De plus, j'ai une liste de fonctionnalités assez grande à ajouter à Analiz. J'espère par ces idées permettre à l'utilisateur de

gagner du temps et de la simplicité dans la réalisation de ses tâches d'analyse. Voici un aperçu des améliorations les plus utiles auxquelles j'ai pensé :

- Permettre d'importer et d'exporter une configuration d'options pour les plugins.
- Pouvoir exporter les résultats d'une analyse dans un fichier PDF.
- Actualiser les résultats d'un plugin grâce à un bouton d'actualisation. Cela permet de ne pas refaire toute les analyses.
- Sauvegarder les analyses effectuées dans un historique permettant à l'utilisateur de revenir sur une analyse précédemment réalisée.

Au final, je suis assez content de cette première application. Cette expérience m'a montré une nouvelle facette du développement web, je n'avais jusque là principalement réalisé que des sites Internet avec comme technologie principale le PHP. La plupart était des sites à but informatifs, tels que des blogs ou des sites personnels ou de petites applications de gestion de données. Je suis maintenant très intéressé dans le développement d'applications en Node.js, c'est une technologie en plein essor et avec une communauté très active.

Ce projet m'a aussi permis de me rendre compte de certaines erreurs que j'ai pu commettre dans l'organisation de mon travail. Je me suis lancé un peu trop tête baissée dans la réalisation de certains points que j'ai dû recommencer peu après car ils ne prenaient pas en compte toutes les considérations que l'application demandait au final.

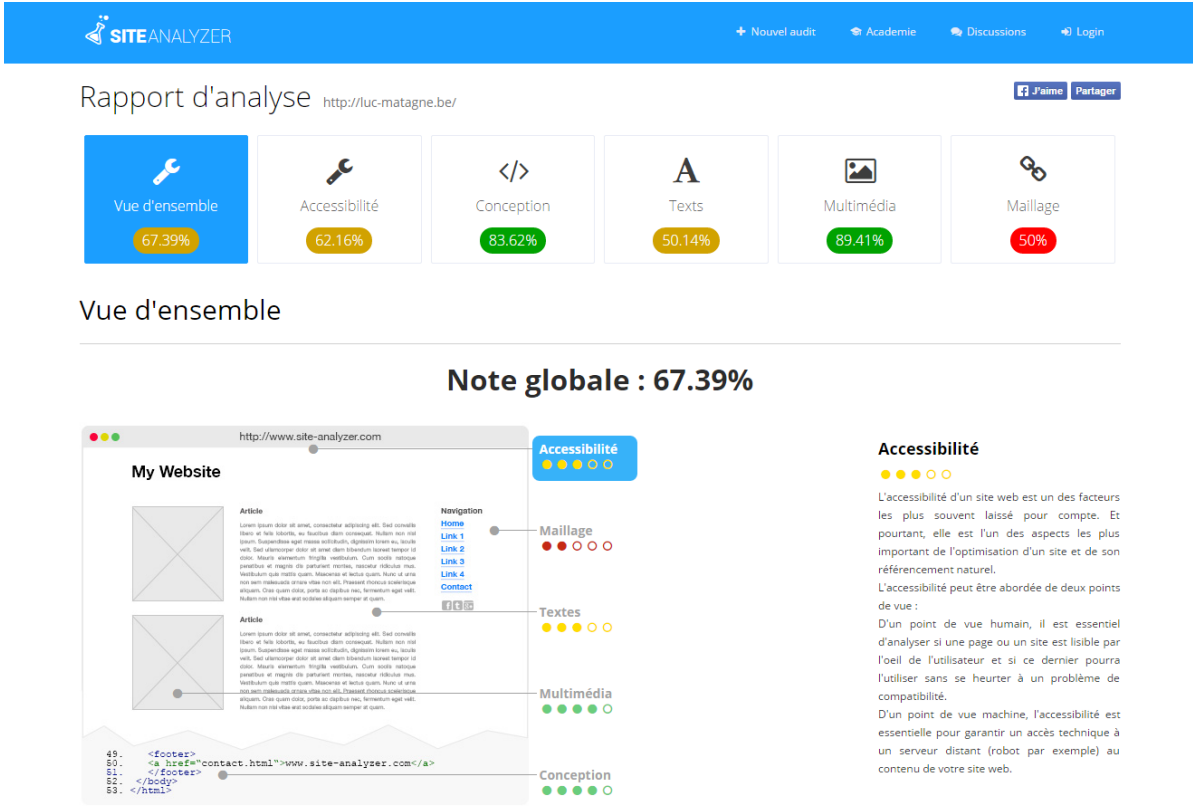
En conclusion, j'espère que la lecture de ce rapport vous a permis de mieux comprendre les différentes étapes du développement de mon projet de fin d'études et de découvrir, si vous ne les connaissiez pas déjà, les technologies des Web Components et Polymer. Ils furent une très bonne découverte pour moi et j'espère avoir réussi à vous partager cette passion naissante. Je me réjouis d'entendre vos retours sur Analiz.

ANNEXE

LIENS UTILES

- Les dépots Github : https://github.com/analiz-app
- Le site de l'application : https://analiz-app.github.io
- Le pdf : https://analiz-app.github.io/rapport.pdf

SOMMAIRE DE L'ANNEXE

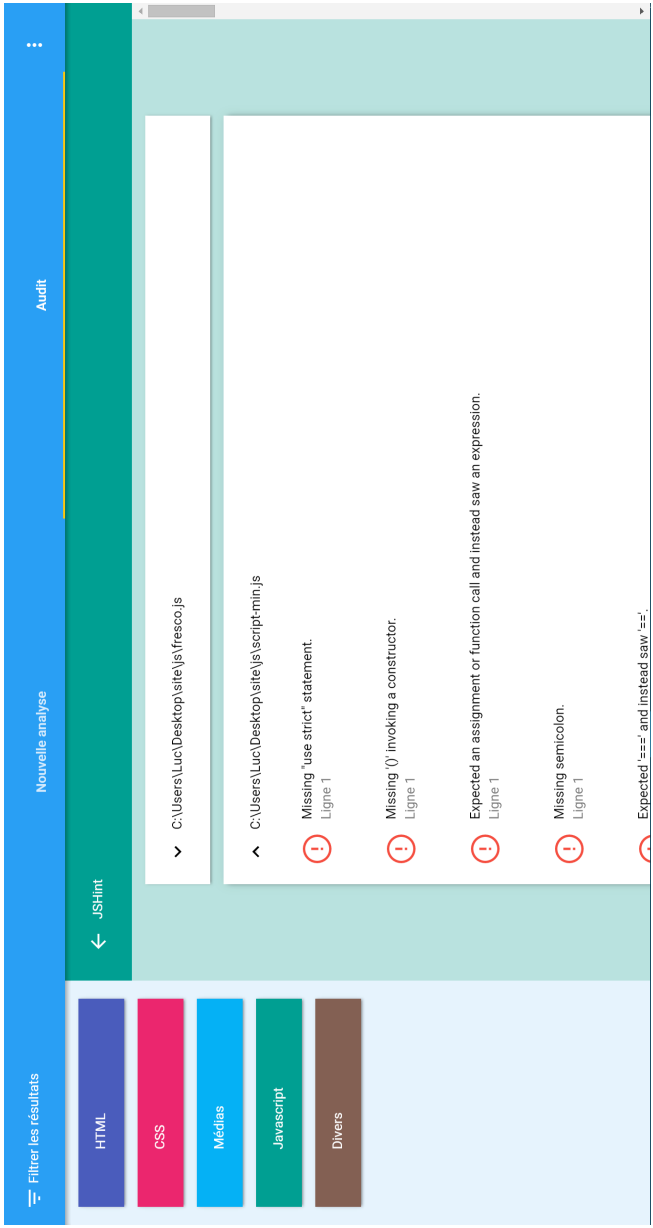

SITEANALYZER
Nouvel audit
Academie
Discussions
Login
Rapport d'analyse http://luc-matagne.be/
J'aime
Partager
Vue d'ensemble
67.39%
Accessibilité
62.16%
Conception
83.62%
Texts
50.14%
Multimédia
89.41%
Maillage
50%
Vue d'ensemble
Note globale : 67.39%
http://www.site-analyzer.com
My Website
Article
Navigation
Home
Link 1
Link 2
Link 3
Link 4
Contact
Article
Accessibilité
Maillage
Textes
Multimédia
Conception
49. <footer>
50. www.site-analyzer.com
51. </footer>
52. </body>
53. </html>
Accessibilité
L'accessibilité d'un site web est un des facteurs les plus souvent laissé pour compte. Et pourtant, elle est l'un des aspects les plus important de l'optimisation d'un site et de son référencement naturel.
L'accessibilité peut être abordée de deux points de vue :
D'un point de vue humain, il est essentiel d'analyser si une page ou un site est lisible par l'oeil de l'utilisateur et si ce dernier pourra l'utiliser sans se heurter à un problème de compatibilité.
D'un point de vue machine, l'accessibilité est essentielle pour garantir un accès technique à un serveur distant (robot par exemple) au contenu de votre site web.

ANNEXE 2

Écran de résultats pour une analyse de validation

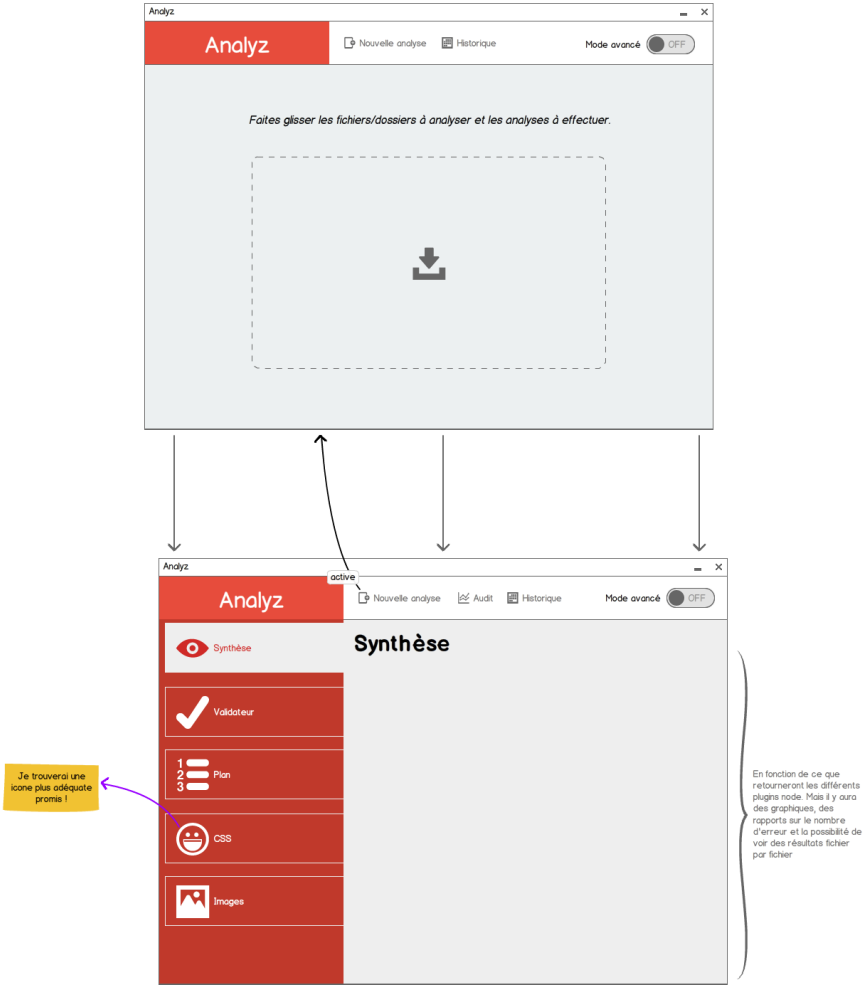
ANNEXE 2 (SUITE)

Écran de résultats pour une un rapport de statistiques



ANNEXE 3

Wireframe du mode simple d'Analiz



ANNEXE 3 (SUITE)

Wireframe du mode avancé d'Analiz