## Palette de couleurs

La palette de couleurs à été obtenue avec l’aide de l’outil Kuler de Adobe. Nous somme parti des couleurs du sites ainsi que d’un logo trouvé sur internet pour définir la palette ci-contre.

|  |  |
| --- | --- |
| Item | Couleur |
| Header |  |
| Background | #a5bacd |
| Title text | #363d43 |
| Menu |  |
| Text | #000000 |
| Background | Transparent |
| Current item Text | #000000 |
| Selected item background |  |
| Contact |  |
| Text | #000000 |
| Body |  |
| Background | #363d44 |
|  |  |
| Tile |  |
| Background | #e2e7f4 |
| Text color | #486e90 |
| Text color alt | #363d43 |
| Area highlight | #dadfeb |
|  |  |
| Button |  |
| Background | #547fa7 |
| Text | #f1f1f1 |
|  |  |
| Text Area |  |
| Background | #f1f1f1 |
| Text | #000000 |
| Border | #000000 |
|  |  |
|  |  |

## Navigation avec niveau unique

Le site web est fait selon un modèl simple de navigation ne contenant qu’un seul niveau. L’utilisateur peut naviguer à travers toute les pages via les éléments de l’en-tête : le menu pour accéder aux différentes pages ainsi que le logo pour revenir à la page d’accueil.

|  |
| --- |
| Macintosh HD:Users:dannyt:INF777:Documents:images:Web Site.png |

Lorsque l’utilisateur se situe sur une page autre que la page d’accueil, l’item de menu demeure sélectionné afin d’indiquer où l’utilisateur se trouve.

|  |
| --- |
| Macintosh HD:Users:dannyt:INF777:Documents:images:header-with-selection.png |

## Structure d’une page

|  |
| --- |
| Macintosh HD:Users:dannyt:INF777:Documents:images:page structure.png |

Chaque page est constituée de blocs distincts pouvant être réorganisés au besoin. Les bloc principaux sont l’en-tête et le contenu. L’en-tête contient le logo, le titre, le menu ainsi que les coordonnées principales.

Le contenu contient les différentes « tiles » représentant du contenu spécifique. Deux type de tiles ont été créées pour le site : la première d’une taille large pouvant contenir plus d’un sous-contenu. Elle est utilisée dans la majorité des pages ayant un contenu associé avec une photo.

La seconde tile est de format plus petit et est utilisée pour un contenu moins volumineux.

Les tailles ont été ajustés de façon à ce qu’une tile large utilise la même largeur que deux petites tiles incluant l’espace entre celles-ci. La largeur des tiles est fixe pour le site, mais chaque page peut jouer sur leur hauteur.

Le but de la tile plus large avec deux sous-contenu était de pouvoir réajuster le contenu verticallement lorsque sur un écran de petite taille comme un téléphone intelligent. Malheureusement, par manque de temps nous n’avons pas pu implanter cette version.

L’organisation du contenu s’adapte en fonction des différentes tailles d’écrans. Pour notre besoin, nous avons focusé sur trois principaux types d’écrans :

* Écran d’ordinateur portable de 1366 x 768 pixels
* Écran d’iPad en format paysage de 1024 x 768 pixels
* Écran d’iPad en format portrait de 768 x 1024 pixels

|  |
| --- |
| Macintosh HD:Users:dannyt:Pictures:iPhoto Library:Masters:2015:04:03:20150403-194957:IMG_0017.PNG |

iPad Air format paysage (haut de la page)

|  |
| --- |
| Macintosh HD:Users:dannyt:Pictures:iPhoto Library:Masters:2015:04:03:20150403-194957:IMG_0018.PNG |

iPad Air format paysage (bas de la page)

|  |
| --- |
| Macintosh HD:Users:dannyt:Pictures:iPhoto Library:Masters:2015:04:03:20150403-194957:IMG_0020.PNG |

iPad Air format portrait (haut de la page)

|  |
| --- |
| Macintosh HD:Users:dannyt:Pictures:iPhoto Library:Masters:2015:04:03:20150403-194957:IMG_0019.PNG |

iPad Air format portrait (bas de la page)

|  |
| --- |
|  |

iMac avec safari

Nous avions l’intention de faire une version pour téléphone intelligent, malheureusement le manque de temps nous en a empêché. Le problème est, que même si les tiles pourraient s’adaptées, l’en-tête et le menu quant-à eux demandent un redesign complet. Le menu serait problablement de type « popup » avec le contenu glissant vers la droite pour laisser la place au menu à gauche comme beaucoup d’applications mobiles font déjà.

## Site ave intelligence du côté client

Le choix technologique à été de n’utiliser que les technologies HTML5 standards et de mettre l’intelligence du côté client. C’est à dire qu’il n’y a aucune composition de fichiers faites par le serveur lors du chargement d’une page. Si une composition est nécessaire, c’est le client qui va chercher les éléments nécessaires et construit la page.

Par conséquent, aucun serveur spécifique n’est nécessaire pour installé la page. L’installation finale est faite sur un serveur IIS avec Microsoft Windows, mais les tests ont été effectués avec un simple serveur NodeJS local.

### NodeJS

La technologie NodeJS nous a permis d’avoir un serveur web simple et portable pouvant exister au même endroit que le code. C’est-à-dire qu’en allant chercher le code de la page web sur GitHub, nous avions automatiquement le serveur pouvant faire fonctionner ce site localement.

Cette façon de faire simplifie grandement le développement et les tests du site web par chacune des personnes travaillant sur le site.

### JQuery

Afin de simplifier les opérations javascript sur le document, nous avons utiliser la librairie javascript Jquery. La librairie est obtenue dynamiquement via un url sur le site des librairies ajax de google « *http://ajax.googleapis.com/ajax/libs* »

### Variation pour différents appareils et différentes résolutions

L’utilisation des instructions « @media » a été utilisé pour permettre de faire varier la présentation selon les différentes résolutions supportées pour les différents appareils.

Par exemple, le contenu des informations pour la journée courante n’est pas affiché au même endroit pour un ordinateur ayant une résolution de 1366 pixels de largeur que pour une tablette de type iPad ayant 1024 pixels de largeur.

Du code javascript a été nécessaire afin de détecter le changement d’orientation et ajuster l’élément pour l’affichage de la carte avec « GoogleMap ». Sans ce code, l’élément gardait la taille qu’il avait lors de son chargement sans aucune mise-à-jour.

### Fragments

Nous avons utiliser le concept de fragments dans le but de ne pas charger tout le contenu au premier affichage mais tout de même permettre un site dynamique. L’utilisation de fragments est un concept utilisé par AngularJS pour permettre de charger uniquement les contenus nécessaires d’une application web au cours de la navigation par l’utilisateur.

Pour ce faire, nous devons utiliser des éléments « placeholders » qui serviront à savoir où insérer le nouvel élément lors de son chargement. L’instruction « load » de Jquery est utilisé pour faire le chargement dynamique et l’insertion de l’élément au bon endroit. Cet instruction nous permet aussi de faire une action lorsque l’élément est obtenu avec succès.

### Structure avec en-tête unique

Les technologies et méthodes indiquées précédemment nous ont aidé à structurer le site avec un seul en-tête plutôt que d’avoir à recopier l’en-tête dans chacune des pages. Le but d’avoir un en-tête unique est de faciliter la maintenance et d’assurer une meilleure cohésion du site.

Étant donné que nous n’utilisons pas de technologies spéciales pour la composition des pages du côté serveur, cette composition devait se faire du côté client. Un élément de type « *placeholder* » a été insérer dans chaque page web avec un type de classe spécifique « *header-location* ».

Pour faire fonctionner le tout, il suffit au développeur d’ajouter un fichier css et javascript dans sa page. Aucun appel n’est nécessaire, l’insertion de l’en-tête est faite automatiquement lorsque le document est prêt.

Comme chaque page est dans un sous-répertoire alors que la page d’accueil est directement à la racine, du code javascript à dû être développé afin d’ajuster les liens (URLs) inclus dans le header. Pour ce faire, le menu utilise un attribut « *page* » indiquant le lien relatif vers la page en question. Lors du chargement, c’est le code javascript qui est responsable de recréer le lien absolue en utilisant le contenu de l’attribut « page » et la valeur de l’url de la page contenu dans « *window.location* ».

Il en va de même pour le logo, pour lequel le lien doit être ajusté par le script afin de pointer au bon endroit. Pour le logo par contre, le lien est directement créé par le script. Une amélioration future serait d’utiliser un attribut permettant de le changer via le code HTML pour éviter d’avoir à modifier le code si le fichier de logo change.

## Contrôle pour le rating

Le site contient un contrôle permettant de donner une note à un médecin via un système de rating avec des étoiles. Ce contrôle est obtenu via une librairies web compatible avec jQuery et placée dans le sous-répertoire « libs » sur le serveur.

## Dépendence à Javascript

Pour les besoins de l’intelligence du côté client, javascript devient essentiel à notre site web et nous ne pouvons donc pas supporter qu’il soit désactivé. Ce qui, de toute façon, devient de plus en plus fréquent pour les nouveaux site web, plus particulièrement pour les applications web dynamiques.

## Affichage en utilisant un modèle de « tiles »

Un choix d’affichage basé sur des « tiles » permettant de changer le contenu affiché selon l’espace disponible est utilisé par le site. Pour l’instant cette méthode permet une optimisation du contenu, mais par la suite pourrait être augmenté en utilisant la librairie jQueryUI et le module « draggable » afin de permettre à l’utilisateur de choisir où il veut placer son contenu. (<https://jqueryui.com/draggable/>)

Bien sûr cette modification demandera des ajouts du côté client et serveur. Par exemple permettre à l’utilisateur de s’identifier et ajouter un module du côté serveur permettant de sauvegarder les préférences d’un utilisateur.

Le layout de base est défini dans un fichier css unique, lequel est réutilisé par la suite par les différentes pages. La largeur des tiles étant fixe, chaque page ne peut jouer que la hauteur de ces dernières pour s’ajuster au contenu.

## Résolutions

Optimisé pour les résolutions iPad Air Paysage et Portrait ainsi que desktop à 1366x768

## Difficultés avec outils Chrome et simulation de tablettes

Lors du développement, nous avons essayé d’utiliser les outils de développement Google Chrome afin de simuler l’affichage sur une tablette. Par contre, nous ignorons encore pourquoi, le résultat sur un iPad avec l’outil était différent du résultat sur un vrai iPad. Ce problème à ralenti le développement car il fallu refaire l’optimisation en utilisant un vrai iPad.