RAPPORT DE STAGE

Réalisation d’un portail Intranet et de son outil d’administration.

Réalisé par
Eliott BASSIER

Projet encadré par
Gaëtan POHIER

Tutoré par
Zouhaira AYADI

Pour l’obtention du BUT 2

ANNÉE UNIVERSITAIRE 2023-2024

Remerciements

En premier lieu, il me semble indispensable de remercier Gaëtan POHIER, mon maître de stage, pour son encadrement, sa bienveillance et son aide constante durant toute la période du stage. Il est la personne qui m’a offert l’opportunité de réaliser ce stage au sein du Groupe MAURIN et m’a donné toutes les clés pour que je mène à bien la mission présentée en me donnant des directives concernant l'architecture souhaitée du projet.

Ensuite, la deuxième personne qu’il me tient à remercier est François CURRIEZ, le directeur informatique du groupe. François a su croire en moi et en l’évolution du projet tout au long de ma période en entreprise et m’a assisté dans l’analyse des besoins métiers au fil des réunions concernant le projet de portail Intranet.

Je souhaite également saluer la présence et le soutien apportés par Véronique GUIGUE, deuxième membre de l’équipe R&D* qui a toujours répondu à mes questionnements et a suivi l’évolution de mon projet alors qu’elle n’y était pas obligée, n’étant pas ma maîtresse de stage officielle.

Enfin, il me semble important et nécessaire de mentionner l’aide précieuse fournie par Madame Zouhaira AYADI, ma tutrice de l’IUT durant ce stage qui a su répondre à mes interrogations concernant les différents rendus écrits nécessaires et s’est rendue disponible afin de faire une visite virtuelle de mon lieu de stage.

Résumé

Sommaire

1. Table des figures

2. Glossaire

R&D : Acronyme de "Recherche & Développement"

I**ntranet** : Est dit d'une ressource (web ou non) accessible à un ensemble de personnes internes à une organisation uniquement (ici le Groupe MAURIN).

AGL : Atelier de génie logiciel. C'est un logiciel très complet permettant la réalisation de différentes actions afin de produire un logiciel/site web en essayant de gérer un maximum de tâches automatiquement, rendant la tâche de l'utilisateur plus simple.

Cloudflare : Service web ayant pour objectif d'augmenter la sécurité des différents accès à un site ou même d'augmenter ses performances.

UI/UX : Acronymes voulant respectivement dire Interface Utilisateur et Expérience Utilisateur. Ces deux facteurs sont très importants dans l'implémentation d'un site web tout public notamment: plus le UI et UX seront optimisées, plus l'utilisateur sera satisfait de son expérience de navigation sur la page web.

Symfony : Framework web composé de différentes librairies PHP permettant de faciliter certaines actions en automatisant certains processus qu'il faut mettre en place manuellement dans un projet PHP pur.

ORM : Acronyme pour "Object-relational mapping", c'est un outil permettant de créer des entités en base de données directement liées à une classe correspondante dans le code cela permet de gagner énormément de temps dans la mise en communication des informations rentrées par programmation et celles stockées en base puisque celles-ci sont liées dès leur création par l'outil.

AssetMapper : Librairie PHP fonctionnant de pair avec le moteur de vue Twig servant à ajouter une clé de hachage à la fin du nom d'un fichier afin de contourner les problèmes relatifs à la mise en cache de ce fichier, permettant aussi d'accéder à ce fichier en utilisant son chemin original.

Webpack Encore : Outil de Symfony permettant d'importer des fichiers en tant que modules dits "assets" (expliquant son utilisation couplée avec l'AssetMapper cité au-dessus), de leur appliquer un certain traitement, puis de les regrouper en un fichier de sortie.

Vanilla : Est dit d'un projet ne faisant appel à aucun framework.

XmlHTTPRequest : API permettant d'envoyer des requêtes http à une url voulue en choisissant sa méthode et décidant si celle-ci sera asynchrone ou non, permettant

aussi de réaliser une action lors du changement de son état via l’ajout d’un écouteur d'événement rattaché à elle.

.csv : Acronyme pour “Comma-Separated Value”, format de texte utilisé pour des données tabulaires avec lequel les différentes valeurs sont séparées par des virgules.

FormData : API Javascript permettant la sérialisation de données au format clé valeur pour leur envoi dans des requêtes HTTP de type POST.

RFC 9110 : Normes définies dans le cadre de l’encadrement des différentes sémantiques liées au protocole HTTP, explicitant notamment les statuts/codes de réponses de requêtes HTTP comme le célèbre 404 NOT FOUND.

toast : Élément apparaissant sur une page web donnant des informations sur le déroulement d’un processus en fonction de sa couleur ou du texte qu’il affiche.

Storage : API WEB permettant de stocker des informations dans le navigateur sous forme de paires clef / valeur. L’objet localeStorage en est une implémentation.

3. Introduction

Dans un monde où l'avancée technologique se fait chaque jour de plus en plus rapide, notamment avec l'émergence de technologies génératives comme nous avons pu le voir récemment. Il est nécessaire aux entreprises de disposer d'outils adaptés aux besoins des salariés en constante évolution. L'utilisation d'outils informatiques a effectivement depuis plus d'une vingtaine d'années facilité de nombreux processus métiers partout dans le monde mais ces outils ont très souvent du mal à rester pertinents dans la durée de par l'évolution des technologies entraînant aussi une évolution des besoins.

Pour contrer ce cycle de l'obsolescence que rencontre une solution informatisée pensée pour une entreprise, il faut adopter une vision globale tournée vers le futur et sur ce que l'on pourrait apporter afin de simplifier la vie des employés sur le long terme. Cette manière de penser responsabilise le développeur et le pousse à réaliser un produit qui sera non seulement accessible à la compréhension de tous mais aussi ouvert aux extensions afin de pouvoir faire évoluer ce produit en même temps que les attentes des utilisateurs.

Il y a une quinzaine d'années de cela, l'entreprise Groupe MAURIN avait demandé la création d'un portail INTRANET* permettant le référencement de différents liens techniques utilisables pour les employés au cours de leurs tâches au travail. Ces concepts étant encore naissants à cette époque, l'architecture de cette application n'était pas vraiment ouverte aux modifications et à l'ajout de contenu postérieur, ce qui a au fil de plusieurs années causé des problèmes au niveau de sa maintenance, de son esthétique et de sa clarté, entraînant une baisse de son utilisation chez les employés.

Ma mission au sein de l'entreprise est alors de trouver une solution permettant de remettre au goût du jour ce portail, le rendant ainsi adapté aux nouveaux besoins des collaborateurs du groupe et permettant sa maintenance facile tout en laissant la possibilité de l'améliorer dans le temps.

Suite à une brève présentation du Groupe MAURIN et de son secteur d'activité, je présenterai la solution pensée pour rendre à nouveau pertinente l'utilisation de la page recensant tous les liens métiers utilisés dans l'entreprise et la manière dont j'ai implémenté cette solution au cours de mes 12 semaines de stage. Cette présentation du développement de cette solution aura pour point de départ l'analyse de l'existant et des problèmes que celui-ci pose en mettant en avant la rédaction du cahier des charges que doit suivre le projet. Nous continuerons notre compte rendu

en nous attardant sur les spécificités techniques du portail Intranet et notre implémentation de la réponse aux besoins métiers formulés dans la partie analytique. Enfin, nous nous attarderons sur la question de l’organisation et de la gestion du projet ainsi que de la mise en place de l’environnement technique dans lequel j’ai pu travailler durant le stage.

4. Contexte du stage

Groupe MAURIN est une firme française spécialisée dans l’import de véhicules étrangers depuis 1982 pour leur revente en France, Belgique et Espagne. Aujourd’hui, le groupe est considéré comme l’un des cinq premiers distributeurs automobiles de France. Celui-ci compte aujourd’hui 201 concessions et plus de 3500 collaborateurs dans les trois pays dans lesquels il est implanté.

Étant acteur autant dans le marché de véhicules neufs que de véhicules d’occasion pour plus d’une vingtaine de marques diverses, on ne peut définir de client type du groupe. Nous pourrions prendre en guise d’exemple la concession SODIRA Mercedes-Benz Nîmes dans laquelle l’équipe R&D trouve ses bureaux, celle-ci reçoit plutôt des personnes assez aisées pour acheter un véhicule dit « Haut-de-gamme ». En opposition les concessions Dacia comme celles de Corbeil-Essones ont plus pour vocation d’accueillir des clients aux moyens moins élevés pour investir dans une voiture.

Cette entreprise cherche aujourd’hui à moderniser son parc informatique et ses différents outils internes. J’ai la chance d’avoir pu intégrer l’entreprise en tant que stagiaire le 8 avril 2024 sous la tutelle de Monsieur Gaëtan POHIER, Analyste Projets Applicatif et leader de l’équipe de Recherche et Développement afin de remplir la mission confiée par Monsieur François CURRIEZ, Directeur des pôles Informatique et Stratégie DATA .

Cette équipe est composée de M. POHIER cité plus-tôt et de Véronique GUIGUE, Analyste Programmeuse. Leur mission s’articule autour de la création ou mise à niveau de logiciels internes de l’entreprise afin de répondre aux besoins des différents corps de métier présents dans le groupe. Ils sont en étroite collaboration avec d’autres services comme ceux liés à l’infrastructure réseau et sont dotés de tâches annexes comme répondre à des tickets remontant des problèmes de collaborateurs ou la vérification des écritures comptables dans l'un de leur outil interne.

En plus de toutes ces tâches, M. POHIER doit s’occuper de la maintenance du portail Intranet, sujet de mon stage et doit ajouter les liens que les collaborateurs veulent y voir apparaître.

TODO : faire partie organisation travail interne cf rapport dodo

5. Analyse

Pour revenir sur le sujet de mon stage cité précédemment, il semble indispensable pour comprendre les défis et les opportunités liés à cette mission, une analyse approfondie du portail Intranet. Celle-ci permettra de mettre en lumière les forces et les faiblesses actuelles du portail, d'identifier les besoins des utilisateurs et de proposer des améliorations pertinentes.

5.1. Analyse du Sujet

Les buts du portail énoncé précédemment seraient de pouvoir recenser de manière efficace et lisible les différents liens afin que celui-ci ait une réelle utilité et que les différents collaborateurs n'aient plus à passer par la sauvegarde de liens directement intégrée dans les navigateurs de nos jours.

Aussi, dans le but de rendre la maintenance de ce portail facile, Monsieur CURRIEZ m'a exprimé sa volonté de pouvoir réaliser des statistiques sur les différents liens affichés et la fréquence à laquelle ils ont été cliqués par les utilisateurs afin de savoir quels liens sont utilisés et lesquels sont désuets pour ainsi pouvoir enlever ceux n'ayant pas d'importance, car cela n'est aujourd'hui pas possible.

Enfin, ce portail aujourd'hui utilisé seulement par les branches francophones du groupe aurait pour objectif d'être utilisé par tous les employés, y compris ceux travaillant en Espagne et en Belgique. Ceci implique de prévoir plusieurs traductions du contenu accessible.

Au cours de ce rapport, nous allons identifier les problèmes posés par le portail actuel en mettant en avant les différents besoins d'amélioration présents. Ainsi, nous passerons en revue les différentes fonctionnalités de ce portail en pointant du doigt les processus défectueux/non-optimisés et en y proposant des solutions et améliorations.

5.2. Analyse de l'environnement technique

La création d'une solution adaptée aux besoins de l'entreprise passe d'abord par la compréhension de son environnement technique afin de fournir un produit adapté à ce qu'ils attendent et qu'ils seront capables d'utiliser et maintenir sans souci.

L'entreprise Groupe MAURIN possède en son sein un département Informatique composé d'une quinzaine de personnes travaillant dans toute la France sur différentes tâches. Parmi eux nous trouvons l'équipe R&D citée précédemment à laquelle j'ai été intégré en tant que stagiaire. Par souci de productivité, celle-ci utilise la suite logicielle de PCSOFT nommée WINDEV/WEBDEV afin de produire leurs

applications. Cet AGL* leur permet de gagner énormément de temps sur la mise en relation des parties FrontEnd et BackEnd des applications qu'ils développent ce qui leur permet de mettre en production environ 3 à 4 logiciels internes à l'entreprise par an.

L'entreprise utilise différentes base de données MySQL afin de stocker les informations nécessaires au bon fonctionnement de ses processus métiers. L'interface permettant la gestion de ces différentes bases de données utilisée par l'équipe R&D est un logiciel libre de droit nommé HeidiSQL permettant d'administrer les différentes bases mais aussi d'en faire des sauvegardes.

Enfin tous les collaborateurs sont munis d'un pc portable sous système d'exploitation Windows 11 afin de ne pas dépayser les utilisateurs mais certains serveurs utilisés pour héberger des logiciels web de l'entreprise sont sous distribution Linux et plus précisément Ubuntu.

5.3. Analyse de l'existant

Dans cette partie du compte-rendu traçant l'analyse du logiciel actuel, nous allons passer différents points clés d'une architecture applicative en revue afin de dresser un portrait du produit préexistant et de trouver de possibles voies d'amélioration.

5.3.1. Sécurité

Lorsqu'il faut réfléchir à un outil informatique disponible sur le web, l'une des premières occupations des développeurs et de savoir si le produit sera disponible au grand public, s'il stocke des données et si oui sous quel format et dans quelles conditions le fait-il, ou bien même s'il comportera des modules qu'il faudra doublement sécuriser afin d'éviter les infractions de potentiels utilisateurs malveillants.

Dans notre cas précis, l'implémentation actuelle du portail pose certains problèmes de sécurité et de confidentialité ce qui pourrait endommager l'image de la firme si quelqu'un de mal intentionné s'en rendait compte.

Effectivement, la version actuelle du portail pose la question de la confidentialité des activités du groupe sachant que ce site censé être « intranet » est totalement accessible par navigateur à n'importe quel personne ayant son lien. L'accès à ce portail est seulement protégée par un accès Cloudflare* qui demande à l'utilisateur de cocher une case pour prouver qu'il n'est pas un robot puis l'accès au site est directement donné ensuite. Cette approche est intéressante pour éviter certaines attaques automatisées mais ne protège pas l'accès d'un potentiel concurrent par exemple.

Pour remédier à ceci, en discussion avec Messieurs POHIER et CURRIEZ, nous

avons pensé à deux implémentations d'un processus permettant la sécurisation de l'accès au portail.

En premier, s'il est important de garder accessible par tout utilisateur pour des raisons pratiques ou légales qui nous sont inconnues et n'ayant pas eu plus d'informations durant les différentes réunions que nous avons pu faire à ce sujet, nous avons pensé à garder un accès public aux informations non confidentielles du portail comme par exemple les liens permettant l'accès aux sites professionnels des différents constructeurs qui nécessitent dès l'arrivée sur leur page de s'authentifier. Cependant les informations plus sensibles telles que les accès aux logiciels internes comme la bibliothèque de documents du groupe serait cloisonnée avec une authentification via formulaire. Il peut être faisable de récupérer les identifiants des sessions Windows de chaque employé grâce à l'annuaire LDAP de l'entreprise par exemple. L'objectif de faciliter la navigation des employés n'est pas à perdre de vue, ainsi il est envisageable de stocker les informations de session de chaque personne connectée durant une durée entre 2 semaines et 1 mois afin de ne pas rendre lourde l'utilisation de cet outil pour autant.

La seconde option serait de fermer entièrement l'accès au site aux visiteurs et le rendre accessible seulement par le réseau interne Maurin. Cette partie n'est pas de mon ressort et sera implémentée par les personnes chargées de l'infrastructure de l'entreprise. Ceci-dit cette option étant tout aussi intéressante que la première aurait pour avantage de ne pas alourdir l'utilisation du portail avec un système de connexion et d'être plus intéressante d'un point de vue de la sécurité car il est plus sage de miser sur un accès impossible par l'extérieur sachant que tout le monde n'est pas au courant de l'importance de disposer de mots de passes robustes aujourd'hui. Il est certain qu'au moins un employé parmi les 3500 collaborateurs possède un mot de passe simple tel que « 1234 » ou « 0123456789 ».

Ici, le problème est une question d'expérience utilisateur et n'a pas de solution plus valable que l'autre, il faut seulement savoir quelles concessions sont prêtes à faire les utilisateurs au nom de la sécurité.

5.3.2. Maintenance du produit

À l'heure actuelle, la maintenance du portail dans sa version actuelle pose un réel problème d'efficacité, étant un poids dans la productivité de l'équipe R&D.

L'intégration actuelle des liens se fait en brut dans le code HTML de la page en affichant des balises pour chaque lien. Cependant, le groupe ayant grandement évolué depuis la création de ce portail, le nombre de liens à intégrer prévu de base est aujourd'hui largement dépassé. Ainsi, lorsque l'ajout d'une nouvelle balise se fait, celle-ci est ajoutée à l'endroit où l'on trouve la place de la mettre et nécessite tout de

même de faire des modifications à la feuille de style du portail, ayant pour répercussions de coûter du temps de travail à Monsieur POHIER qui doit très souvent prendre sur son temps de travail pour ajouter de nouveaux accès et en gérer l'affichage dans un langage et une syntaxe vieillissante.

Comme énoncé plus tôt, il est aussi difficile de savoir quels liens sont encore utilisés aujourd'hui ce qui rend une grande partie de l'affichage obsolète. Cette partie utilisée par des accès non-utiles pourrait être occupée par des accès à des outils ayant plus d'importance de nos jours.

Pour éviter tous ces problèmes d'administration, il a été convenu en accord avec Messieurs POHIER et CURRIEZ qu'il serait dans tous les cas nécessaire de monter une base de données recensant les différents liens afin de faire les statistiques demandées sur les utilisations de chaque lien. De ce fait, les différents accès seraient récupérés dynamiquement et affichés en fonction de ce qui figure dans la base de données par rapport à leur utilisation. Cela éviterait les problèmes actuels liés à leur affichage. Pour aller plus loin, il semble impératif l'implémentation d'une interface d'administration des différents accès créés afin de pouvoir gérer la population de la base de données sans avoir à passer par un logiciel tel que phpmyadmin ou HeidiSQL, rendant centralisée et fonctionnelle l'utilisation du portail et évitant des heures de maintenance inutiles aux membres de l'équipe de développement.

On peut même envisager de cacher de manière automatique les liens inutilisés depuis 1 an avec l'implémentation de la base de données recensant les visites de chaque accès. Ceci pourrait-être réalisé grâce à l'aide de Triggers actifs sur la base de données déployée.

Enfin, comme énoncé plus tôt, les développeurs ayant l'habitude de programmer avec l'AGL WINDEV/WEBDEV, ils n'ont ni l'habitude, ni l'envie de maintenir du code dans d'autres langages compliqués. De ce fait, il semble important de comprendre quels sont les langages qui leur semblent facile à administrer en cas de bugs sur le produit final après la fin du stage. Il va donc de soi que l'utilisation de frameworks PHP tels que Symfony ou Laravel leur compliquerait la tâche en plus d'être inappropriée au vu de la taille du projet qui n'a pas vocation d'utiliser un framework entier mais peut-être seulement certains composants facilitant les processus de l'application, sous condition d'être bien commentés afin que leur maintenance soit facile pour l'équipe R&D.

5.3.3. Utilisation du Produit

L'utilisation actuelle du produit est actuellement très simple : on trouve une page affichant tous les liens disponibles d'accès, en cliquant sur l'un d'entre eux la

page nous redirige vers le site associé.

Cette simplicité est un point fort du portail actuel mais crée d'autres problèmes liés à l'UI/UX*.

TODO : insérer figure le design actuel du portail.

Concernant l'interface utilisateur, on peut constater que les différents liens ne sont plus exactement délimités par les catégories auxquelles ils doivent appartenir et que certains d'entre eux sont mal classés. Ceci crée de la confusion chez les utilisateurs qui ne savent pas spécialement ou chercher les informations dont ils ont besoin (à moins d'avoir l'habitude de naviguer parmi les catégories mélangées). L'objectif du projet de refonte serait d'y voir plus clair au niveau des classifications des liens et de permettre aux différents employés de retrouver une certaine cohérence au niveau de leurs recherches.

Aussi, il semble important de souligner que certains accès comme les liens constructeurs ne sont pas stylisés et sont affichés de manière assez brute sous forme de liste à puce. La dimension visuelle de ce choix d'implémentation ne permet pas à l'utilisateur de s'y retrouver facilement parmi les options disponibles et peut parfois rebuter à passer par le portail pour trouver une information souhaitée.

L'expérience utilisateur est elle aussi dégradée par certains détails qui n'ont à première vue pas d'importance mais qui à force d'utilisation rendent l'usage du portail inefficace.

Le vrai problème que rencontrent les différents collaborateurs vis-à-vis de cet outil est qu'un clic sur un lien remplace la page du portail en la page souhaitée. Cela signifie que si nous souhaitons retourner sur le portail il faut soit quitter la page actuelle soit ouvrir une nouvelle page dans le navigateur afin de retourner sur le portail sans fermer le premier accès ouvert.
Réutiliser l'intermédiaire du portail pour accéder à un nouveau site voulu semble redondant au niveau du processus de pages.

Une manière envisageable afin de corriger ceci serait de faire en sorte que chaque lien cliqué ouvre automatiquement un nouvel onglet du navigateur tout en conservant l'accès au portail. Cela permettrait d'utiliser cet outil comme la page de favoris du navigateur qui a aujourd'hui remplacé l'utilisation de la version actuelle du portail. Ainsi, un collaborateur pourrait ouvrir la page de l'outil au début d'une journée de travail et la refermer à la fin de celle-ci après avoir eu accès aux liens souhaités.

Afin de dresser un portrait du futur portail, il est nécessaire de noter tous les éléments cités dans cette partie réservée à l'analyse. Ceci faisant, la phase de

conception de l'outil sera facilitée par la mise en avant des spécificités techniques et permettra de faire correspondre le produit fini aux besoins des utilisateurs.

5.4. Cahier des charges

Comme vient de l'être vient de l'être expliqué, il est souvent nécessaire d'apporter une dimension théorique au projet afin d'encadrer celui-ci et d'y définir les spécifications techniques. La création d'un cahier des charges est une pratique tendant vers cet objectif et permettant de respecter des consignes claires et précises tout au long du processus de développement jusqu'à la mise en production du produit fini.

La phase d'analyse du projet a permis d'arriver à une idée précise des différents attendus du produit fini. Tout le processus organisationnel autour du portail Intranet sera détaillé dans les parties du cahier des charges qui vont suivre.

5.4.1. Contexte du projet

Groupe MAURIN est une entreprise spécialisée dans l'import de véhicules étrangers en France, Espagne et Belgique. Les différents collaborateurs du groupe ont accès à une page web dite "Intranet" sur laquelle ils ont accès à des liens leur permettant d'effectuer des tâches techniques en relation avec leur métier. La dernière mise à jour architecturale de cet outil date de 2008-2010 ce qui fait qu'il est aujourd'hui obsolète. Une refonte de celui-ci est donc plus que nécessaire afin de le rendre à nouveau utilisable, lui ajouter des fonctionnalités et l'adapter à l'esthétique des sites contemporains et à la nouvelle charte graphique du groupe.

5.4.2. Besoins

Tout d'abord, il est impératif de savoir quels liens sont obsolètes et lesquels sont à l'heure actuelle toujours utilisés par les différents employés. Il sera ainsi nécessaire de stocker les informations relatives aux clics sur chacun des liens affichés sur la page en y explicitant l'utilisateur qui y a cliqué, et la date à laquelle cette interaction est survenue. Pour aller plus loin et prendre en considération les visiteurs du portail qui ne seraient pas authentifiés, il est possible que l'on doive stocker les adresses IP des personnes interagissant avec les différents accès visibles sur la page d'accueil.

Ensuite, afin de ne pas déboussoler les différents utilisateurs, il sera nécessaire de définir des différents droits d'accès concernant les droits d'accès de chaque identifiant de connexion et cloisonner les fonctionnalités en fonction de quel type d'utilisateur visite la page. Cela est aussi nécessaire dans une optique de sécuriser la partie administration de l'outil.

Concernant ladite partie réservée à l’administration du portail, elle doit se présenter sous forme d’une interface d’accès à la base de données intégrée à la page web et permettant de manière asynchrone l’ajout, la lecture,la modification et la suppression de liens et de groupements de liens en permettant de gérer si les différents liens en base seront affichés ou non sur la page visible par tous les utilisateurs. Celle- ci devra aussi afficher des boutons réservés aux statistiques, fonctionnalité qui sera explicitée ci-dessous.

Effectivement, il sera nécessaire de pouvoir récupérer des statistiques globales d’utilisation de la page mais aussi des statistiques individuelles concernant chaque lien affichable en fonction d’une période définie avec une certaine valeur par défaut. L’objectif de cette fonctionnalité est de maintenir la cohérence de l’affichage des différents accès dans le temps et faire en sorte de pouvoir enlever ceux inutilisés.

Dans la même optique il sera nécessaire de stocker les dates correspondant à la dernière fois qu’un utilisateur s’est connecté afin de savoir si celui-ci utilise souvent l’outil mis à disposition et dans le cas contraire de lui demander de faire des retours sur le produit afin de pouvoir en faire des améliorations futures.

5.4.3. Résultats attendus

En prenant en compte les spécificités cités juste au-dessus nous devrions arriver à un archétype de page précis et assez épuré :

- Une page d’accueil recensant tous les liens classifiés par catégories accessibles via une barre de navigation, avec l’appel aux différents liens sans avoir à recharger la page. Seules les catégories ne comportant aucune information confidentielle doivent-être affichées aux utilisateurs non authentifiés, le reste serait accessible après connexion. Cette page possèderait aussi un bouton permettant l’affichage d’un formulaire réservé à la connexion et d’un autre permettant l’accès à la page d’administration dans le cas où l'utilisateur possède les permissions nécessaires.

- Une interface d’administration permettant la gestion des entités présentes en base de données seulement pour celles ayant besoin d’être éditées : celles relatives aux liens et à leurs groupements. Un affichage des accès au format de liste triable par nom et autres attributs comme le statut affiché ou non doit y être possible. Des opérations de filtrage par nom de catégorie ou de sous-catégorie de lien doivent y être possible pour faciliter l’expérience utilisateur. Dans le même objectif, il doit être possible de trouver directement un lien souhaité en le recherchant dans une entrée dédiée à cette action Les opérations relatives à un lien existant doivent figurer sur la ligne ou celui-ci est affiché sous forme de bouton qui affiche un formulaire

réservé à cette action. La présence de boutons d'ajout de nouvelles entités en base de données est essentielle.

- La page précédente doit aussi permettre l'affichage de statistiques ce qui est un peu plus précis, d'où le besoin d'expliciter le résultat dans un point différent. Effectivement, deux types d'affichages devraient être prévus à cet effet. Le premier concernant les statistiques relatives à chaque accès doit être accessible de la même manière que les actions exclusives aux liens en passant par les boutons présents sur la ligne de celui-ci, ce qui aurait pour effet d'afficher un formulaire demandant de rentrer les intervalles de la période à prendre en compte pour les statistiques puis affichant le nombre de clics total sur celui-ci ainsi que le nombre de clics d'utilisateurs non connectés puis un classement des cinq utilisateurs ayant le plus accédé à celui-ci durant la période donnée. Le second est supposé s'afficher suite à un clic sur le bouton prévu dans la barre de navigation et proposerait de la même manière que le premier de choisir une période sur laquelle il pourrait se baser pour recenser le nombre de clics totaux en comparant le nombre de clics par catégories sous forme de diagramme "camembert" par exemple. Il doit aussi y figurer un bouton permettant l'export des données dans la base de données sous format Excel afin de pouvoir faire des opérations sur les tableurs en plus de celles disponibles sur la page.

5.4.4. Contraintes

Le projet présente plusieurs contraintes qu'elles soient au niveau de l'architecture ou bien concernant les interactions des différentes personnes avec l'outil. Celles-ci auront une importance plus ou moins importante dans le processus de développement compliquant parfois la tâche de conception.

La première d'entre elles est l'utilisation de langages de programmation simples comme le HTML/CSS, Javascript et PHP afin que la maintenance du produit soit la plus simple possible en cas de dysfonctionnement ou autre. Il faudra donc éviter au maximum d'utiliser des bibliothèques externes sans commenter le fonctionnement de celles-ci. L'usage de framework tels que Symfony n'est donc pas adapté au développement de cet outil même si celui-ci facilite grandement la tâche des développeurs, car il faut un certain temps afin de s'acclimater à son utilisation.

Il est aussi nécessaire de réaliser des sondages afin de pouvoir peupler correctement la base de données et vérifier quels liens sont affichés sur la version actuelle du portail et sont toujours utilisés par les différents collaborateurs et aussi de savoir si ces derniers ont besoin de voir apparaître d'autres accès à d'autres services n'existant pas déjà.

Aussi il faudra rendre fluide le processus de connexion puisque les utilisateurs n'ont habituellement pas à réaliser cette tâche. Ainsi, il faudra récupérer

les identifiants des sessions Windows via l’annuaire LDAP du groupe pour pouvoir mettre en place une connexion SSO* afin d’alléger la charge mentale des collaborateurs. Enfin la date d’expiration de la session utilisateur devra être fixée environ trois semaines après la date de dernière authentification afin que l’utilisation de la page web Intranet ne diffère quasiment pas de celle actuelle.

5.4.5. Ressources à mobiliser

Ce cahier des charges fournit plusieurs ressources permettant d’adopter une architecture applicative optimisée et de disposer d’informations nécessaires au bon développement.

Pour garantir l'efficacité et la pertinence de notre base de données, il a été impératif de procéder à une analyse approfondie des besoins. Cette étape cruciale nous a permis d'identifier les exigences spécifiques et de structurer les informations de manière logique et optimisée. En effectuant cette analyse, nous avons pu établir une base de données la plus cohérente possible, minimisant les redondances et assurant une gestion fluide et efficiente des données.

TODO : voir schéma MCD en annexe.

Cette approche nous permet non seulement de maintenir l'intégrité des informations, mais également d'améliorer la performance et la facilité d'utilisation de notre système.

Ayant donné une architecture solide pour stocker nos données, il est important que le contenu de la base de données soit aussi pertinent de son côté. Pour aller dans ce sens, l’utilisation d’une feuille de calcul partagée permettant de recenser tous les accès utiles à été mise en place. L’utilisation de ce système en parallèle du processus de développement permettra de gagner du temps sur la mise en production du site web intranet en évitant à chaque personne ayant une responsabilité sur la feuille de style de donner des informations cruciales au dernier moment.

TODO : voir screen feuille de calcul liens portail maurin

Enfin, après avoir passé en revue les différentes ressources à utiliser pour améliorer ce qui n’est pas visible par l’utilisateur, il faut prendre en compte la direction esthétique suivie par l’entreprise. La phase d’analyse a permis de dresser des maquettes en accord avec la vision de M. CURRIEZ dans l’optique de rendre la page la plus plaisante à naviguer.

TODO : voir maquettes du portail

5.4.6. Délais à respecter

La période de stage dans le groupe est à découper en plusieurs parties. Ce cahier des charges ayant pris deux semaines à être rédigé en passant par les étapes de maquettage et de conception de la base de données, il ne reste que dix semaines de travail afin de présenter un produit fini. Vient ensuite la période de réalisation, correspondant à celle ou le stagiaire développe une solution afin de répondre aux besoins métiers exprimés plus tôt. Celle-ci doit durer 8 semaines du 22 avril 2024 au 14 juin 2024. Les deux semaines restantes devront servir à la mise en production de la nouvelle page web Intranet afin de corriger d’éventuels problèmes qui n’existaient pas lors du développement en local. Cette période permettra aussi en cas de validation complète de compléter la documentation du produit afin de faciliter la future maintenance de celui-ci par les membres de l’équipe R&D.

6. Rapport Technique

Ayant rendu compte dans la partie précédente des différentes étapes de recherches s'étant succédées durant l'analyse du projet, nous allons maintenant pouvoir rendre compte des phases de conception, réalisation et validation du portail Intranet. Nous aborderons le sujet de ces phases séparément en s'attardant notamment sur celle de la réalisation, la scindant en trois parties correspondant aux trois grands besoins auxquels le projet doit répondre: le recensement de liens métiers, l'administration d'une base de données intégrée au site puis enfin l'analyse statistique de données dans un but de maintenance.

Cette approche permettra de relever les différents atouts de chaque aspect de l'application développée, permettant aussi la meilleure compréhension de celle-ci. Cette même compréhension passe d'abord par une explication claire de la base même d'un projet, soit sa conception.

6.1. Conception

Comme tout juste énoncé, il est impossible de comprendre le fonctionnement d'un programme sans en connaître sa base ou son architecture. C'est pourquoi il faut s'impliquer tout autant dans la phase de conception d'un produit que dans son développement sous peine de rencontrer des problèmes au cours de ce dernier.

Cette phase renvoie notamment énormément au cahier des charges puisqu'elle relève de la compréhension des besoins clients afin de prévoir une réponse adaptée à formuler avant la phase de réalisation afin de faciliter cette dernière.

Par ailleurs, pour revenir à ce cahier des charges, nous pouvons associer sa conception avec des compétences et apprentissages critiques du BUT Informatique. Effectivement, l'établissement d'un cahier des charges au préalable avant de s'attaquer à la programmation pourrait être mis en association avec l'apprentissage critique 21.01 "Adopter de bonnes pratiques de conception et de programmation" puisque c'est une pratique vraiment récurrente dans le monde professionnel de par le gain d'efficacité que celle-ci permet. De plus, le schéma MCD réalisé au cours de cahier des charges à été possible grâce aux cours reçus tout au long de mes deux années de BUT Informatique notamment en suivant l'apprentissage critique 14.03 "Concevoir une base de données relationnelle à partir d'un cahier des charges".

Je ne m'attarderai pas plus sur cette phase, son travail étant répertorié dans le cahier des charges, nous avons assez de matière pour comprendre la suite du rapport technique et sa partie dédiée à toute la réalisation du portail.

6.2. Réalisation

Disposant maintenant des clés nécessaires à la compréhension de l'architecture applicative souhaitée, nous pouvons aborder la description de la phase de développement du portail Intranet.

Nous avons d'abord fait le choix de ne pas reprendre le code existant car étant obsolète et ne correspondant plus aux besoins actuels, il aurait été plus un poids de devoir le redéfinir pour l'adapter à nos besoins que de directement en recréer une architecture de base. Effectivement celui ne contenait que de simples balises <a> permettant d'accéder à l'URL voulue au clic de celle-ci ou alors des liste déroulantes d'éléments qui avaient le même effet. Les feuilles de style ne sont pas non plus récupérables.

En partant de zéro, nous avons fait le choix d'utiliser une architecture Modèle - Vue - Contrôleur ce qui a rendu les opérations sur la base de données bien plus simples en faisant en sorte que les actions sur la page n'appellent que les fichiers relatifs aux Contrôleurs, améliorant la lisibilité des intéractions et rendant le code plus facile à maintenir. L'un des autres choix réalisés est de n'avoir utilisé que du PHP et Javascript sans utiliser de framework. Au commencement de la période de développement, nous avions réfléchi en accord avec M. Pohier à utiliser Symfony* afin de faciliter le processus de développement grâce aux nombreux outils qu'il propose : ORM* intégré, système de routage automatique, moteur de template simple d'utilisation… Cela impliquait cependant que je sache utiliser cette solution. N'étant pas le cas j'ai essayé de me former afin de comprendre son utilisation mais un cas de figure précis m'a forcé à abandonner l'utilisation de ce framework.

J'ai effectivement rencontré quelques problèmes lors de l'utilisation de Symfony lorsque j'ai voulu coupler son utilisation avec du Javascript, ce qui dans la plupart des cas fonctionne mais qui dans mon implémentation actuelle de certaines fonctionnalités posait problème.

L'un des avantages du JS est qu'il permet de modifier dynamiquement des ressources présentes sur la page que cela soit juste une balise, une image chargée, ou bien toute la page elle-même et son style associé. Il est possible de prendre une balise <img> censée afficher une image lorsqu'on lui donne une url appelée "src". Cependant l'un des composant de Symfony nommé Twig possède une fonctionnalité nommée AssetMapper* qui change à chaque lancement de la page le chemin d'une ressource donnée pour y rajouter un clé de hachage entre la fin du nom du fichier et son extension. Cette opération se faisant au chargement de la page et générant une clé au hasard à chaque fois, il est impossible de récupérer le nom exact du nom de fichier généré. De ce fait la fonctionnalité js que j'essayais d'implémenter ayant pour but de changer l'url de l'image source de ma balise n'arrivait tout bonnement pas à retrouver l'image souhaitée étant donnée que son nom s'est vu rajouter une clé de

hachage à son nom de fichier.

Ce problème aurait pu être résolu avec l’utilisation d’une autre librairie proposée par Symfony nommée Webpack Encore*. Cependant, afin de garder pour objectif une maintenance facile du produit, j’ai préféré ne pas utiliser cette nouvelle librairie qui m’était présentée comme solution temporaire mais qui n’aurait été qu’un fardeau dans le cadre de la maintenance du produit puisque sa syntaxe nécessite une bonne connaissance du framework ce que n’est pas le cas de M. POHIER et Mme GUIGUE.

La résolution à ce problème a été de re-basculer sur une architecture Vanilla* avec nos vues qui important de manière classique des scripts javascripts qui récupèrent les images de manière normale sans qu’aucune clé de hachage n'empêche cette opération.

Pour la suite des choix opérés sur l’architecture, nous expliqueront dans les parties réservées aux trois grandes fonctionnalités de la page pourquoi nous avons fait ces choix d’implémentation et pourquoi cela nous semble être l’une des meilleures solutions.

6.2.1. Affichage de liens métiers en fonction de leur classification

La première mission que devait remplir ce portail est celle dont le résultat touche le plus de monde puisqu' elle est directement en lien avec le besoin primaire du portail, c'est-à-dire faire gagner du temps aux collaborateurs en répertoriant tous les accès auxquels ils puissent avoir besoin lors de leur labeur.

En réponse à ce besoin, nous avons décidé de réaliser une première page dédiée à cette fonction présentant une entête à l'effigie du groupe, un barre de navigation recensant toutes les catégories de liens disponibles et leur sous catégories respectives. Le corps de la page affiche le résultat des liens disponibles dans la catégorie dite “sélectionnée” sous forme de cartes/groupe montrant les différents accès disponibles.

Le concept de Carte a été pensé pour améliorer l’UI et rendre plus intuitive la navigation par rapport à l’ancien portail. Effectivement, l’ancienne solution proposait dans une seule catégorie plusieurs lignes correspondant à un même constructeur automobile, par exemple FORD bénéficiait d’un accès normal, un accès “PTS” et un accès “EASY” étalés sur trois lignes différentes sans image permettant de reconnaître qu’ils avaient un lien entre eux. L’objectif des cartes était d’afficher au recto le logo relatif à la marque ou au service auquel on souhaite accéder et au verso les différents liens techniques qui y sont associés, un peu à la manière d’une

carte de visite. Pour les marques / services ne nécessitant qu’un seul accès, il nous a semblé inutile de surcharger l’utilisateur avec l’animation de carte qui se retourne pour accéder à une seule information. Ainsi, lorsque celle-ci est survolée, nous l’avons seulement rendue directement cliquable et ramenant vers l’accès recensé, l’indiquant grâce à un effet visuel.

L’appel de ces cartes se fait de manière asynchrone afin de ne pas surcharger visuellement l’utilisateur, cependant c’est la seule requête asynchrone au serveur qui fonctionne d’une manière bien spécifique.

Effectivement, dans une optique d’optimisation de l’application, nous avons essayé de garder un maximum de cohérence dans notre arborescence de fichiers et par le chemin qu'empruntent nos différentes requêtes. Celle-ci fait entorse à la règle de par son utilisation qui est bien plus spécifique par rapport aux autres. Dans chaque requête effectuée nous avons essayé de juger si il était important de d’abord appeler une seule fois la récupération de toutes les entités d’un type prévu depuis la base MySQL et seulement parcourir un tableau d’entités lorsque nous voulons récupérer des données précises ou s’il est plus judicieux de faire plusieurs requêtes différentes sans avoir à rafraîchir la page pour obtenir à chaque fois ces données depuis la base, ce cas est l’un des seuls où cette solution a été trouvée plus intéressante. Celle-ci a été jugée plus judicieuse puisqu’il existe plusieurs cas où il vaut mieux récupérer petit à petit les données qui nous intéressent. Cela permet d’abord de récupérer des quantités de données raisonnables et de ne pas prendre celles qui n’intéressent pas les utilisateurs, par exemple, un carrossier sera certainement intéressé par les liens constructeurs et logiciels internes de l’entreprise mais ne souhaitera sûrement pas ralentir sa navigation car sa requête cherche tous les liens relatifs aux métiers de l’administratif en plus de ceux souhaités. Ensuite certains types de liens sont des sujets de requêtes “ponctuelles” pour chaque utilisateur, les accès aux différents avantages collaborateurs seront sans doute bien moins utilisés que ceux menant aux logiciels permettant de réaliser des tâches techniques, c’est pourquoi on ne souhaite pas les récupérer à chaque requête mais uniquement dans des cas bien spécifiques.

Ainsi pour récupérer les différentes cartes et accès associés à elles, nous avons créé un fichier spécifique dans l’arborescence liée au code source, dans le dossier ‘Lib’ destiné à accueillir les librairies de fonctionnalités . le fichier créé, requetesCartes.php permet de récupérer l’appel d’une méthode GET avec pour paramètre le nom de la catégorie souhaitée ou de la sous-catégorie souhaitée et d’appeler les méthodes relatives aux Cartes permettant la mise en forme au format HTML des données récupérées directement depuis la base de données.

TODO : afficher ANNEXE : requetesCartes.php

L'une des particularités des méthode appelées est qu'elles vérifient les informations récupérées en base pour savoir s' il faut afficher les entités et sous quel format le faire le cas échéant. Par exemple, un lien dont l'attribut statut possède la valeur '0' en base (correspondant au booléen 'false' du côté du de PHP) ne doit pas être affiché car il n'est pas jugé présentable à l'utilisateur, soit parce qu'il est en cours d'édition, soit parce qu'il ne fonctionne pas ou autre.

Cette fonction fait appel aux objets de type Repository de chaque Entité afin de récupérer en base les informations souhaitées, puis on utilise différentes fonctions pour savoir quel type d'affichage utiliser en fonction des informations relatives à chaque Carte et au nombre de liens censés s'y trouver. La seule exception à cette règle est les cartes de sous catégorie "Avantages Collaborateurs" qui ont un design unique de par le besoin de leur afficher le descriptif de prévisualisation de l'unique lien mis en avant par celles-ci !

TODO : mettre visuel Carte Avantage Collaborateur.

Nous avons fait le choix de créer une fonction différente à celle des autres cartes tellement ce cas est particulier. Cette dernière est appelée si les conditions sont réunies à la réception de la requête par requeteCartes.php et son fonctionnement ne diffère quasiment pas de la fonction relative aux autres types de carte.

Le dernier avantage de cette implémentation est qu'il n'est pas nécessaire de recharger la page pour avoir toutes les informations sur les liens à jour. Si une URL vient à être modifiée via l'interface d'administration, il suffit de se rendre sur la catégorie de cette carte pour la voir mise à jour au niveau de son lien, il est possible que la mise en cache du navigateur ne change pas l'image associée à cette carte ne change pas automatiquement, mais les informations cruciales comme l'URL des liens seront bien à jour. Le seul exemple où ceci ne marche pas est si le lien modifié se trouve dans la catégorie dans laquelle nous sommes actuellement, comme nous n'envoyons pas de requête dans ce cas, nous n'obtenons pas les informations à jour, mais ce cas de figure est très rare.

Le choix du fichier séparé seulement pour cette solution sert à apporter de la clarté dans le code source afin de différencier ce besoin technique des autres. Étant celui élémentaire au fonctionnement de la page, il est nécessaire de trouver rapidement les informations y étant associées et de pouvoir le corriger en cas de problème sans que cela n'impacte le reste du portail. On peut aussi se demander pourquoi ne pas avoir adopté cette solution au reste du projet. Tout simplement parce que voir une dizaine de fichiers associés chacun à un cas d'appel asynchrone n'aide en rien dans l'objectif de simplicité que nous essayons de mettre en avant.

La dernière fonctionnalité technique présente sur cette page est la possibilité de l'utilisateur de se connecter via un formulaire d'authentification afin d'accéder à

tous les liens ou même à l'interface d'administration du portail si il possède les droits nécessaires. L'affichage du formulaire à cet effet se fait en cliquant sur un bouton présent sur la droite dans la balise du header.

Il faut prévoir l'implémentation de plusieurs détails afin d'optimiser au mieux le processus de connexion pour les utilisateurs. D'abord il faut que l'utilisateur soit mis au courant si il ne remplit pas tous les champs du formulaire ou s'il se trompe dans ses identifiants en affichant à l'écran un toast* de couleur rouge lui faisant comprendre que la connexion n'a pas été faite puisqu'une information est erronée ou non-remplie. Aussi il faut que les deux champs du formulaire soient remplis afin de pouvoir envoyer la requête de connexion, sinon ça ne sert à rien car on sait déjà que les conditions du couple clef valeur qu'est l'identifiant lié au mot de passe de compte ne sera pas remplie. Enfin il faut que lors d'une connexion réalisée avec succès, un toast vert s'affiche à l'écran une seule fois et qu'il ne s'affiche pas à nouveau au rafraîchissement de la page.

A l'heure de la rédaction de ce rapport, la connexion par protocole SSO n'est pas encore implémentée et la manière de se connecter utilisée est simplement par rapport à des identifiants rentrés en base de données. Ces identifiants sont assez classiques et consistent en un login et un mot de passe haché grâce à la fonction PHP password_hash($chaineDeCaractèresAHacher). Afin de vérifier que l'utilisateur a rempli les bons identifiants, la fonction JS envoie une requête au serveur (les spécifications de ce type de requêtes seront explicitées dans la partie prochaine relative à l'interface d'administration car les requêtes de ce type y sont plus fréquentes) afin de récupérer en base de données l'utilisateur correspondant au login entré dans le formulaire d'authentification et en vérifiant si le mot de passe associé à celui-ci dans son entité correspond à celui saisi dans le formulaire. Cette opération est réalisée grâce à la méthode PHP password_verify(chaine1, chaine2) qui vérifie si la première chaîne non-hachée correspond à la deuxième qui elle est hachée. Si c'est le cas, alors on établit la connexion en démarrant une Session PHP à laquelle on passe des informations concernant l'utilisateur en variables de session. La dernière action à réaliser est de mettre à jour l'objet utilisateur récupéré dans la base de données afin de changer sa date de dernière connexion à la date actuelle.

TODO : afficher fonction contrôleur utilisateur connexion()

La dernière difficulté rencontrée correspond à l'affichage unique du toast vert indiquant que l'utilisateur est bien connecté. Pour réussir à atteindre cet objectif, l'utilisation de l'API WEB Storage* m'a permis de stocker des informations locales sous la forme de variables au format clé-valeur.

La manière de procéder a été d'instancier une variable relative à la possibilité de l'affichage du toast de connexion avec comme valeur un booléen true lorsque la requête de connexion renvoie un code de retour positif. Ensuite nous avons ajouté à

la fenêtre de la page d'accueil du portail un gestionnaire d'événement qui indique qu'à son chargement, on va fouiller dans la variable localeStorage qui implémente l'API Storage et que si la clef concernant le toast de connexion est trouvée et que la valeur booléenne associée est vraie alors on peut afficher le toast et directement changer cette valeur en fausse afin que l'action ne se produise qu'une seule et unique fois comme souhaité.

TODO : afficher annexe écouteur d'évenement animations.js window.onload

L'une des améliorations qui pourrait être pensée pour le futur de ce portail et améliorer l'UX par la même occasion serait de donner un ordre de récupération des différentes cartes en fonction de l'utilité que chacune a aux yeux des différents utilisateurs. Par exemple, pour un collaborateur de concession Mercedes-Benz, il serait intéressant de voir apparaître en premier les cartes liées à son lieu de travail plutôt que celles en rapport avec Citroën. Ceci pourrait aisément prendre forme avec le troisième besoin fonctionnel qui correspond aux statistiques effectuées pour chaque lien et de trier les cartes récupérées par la requête à la base de données par nombre de clics décroissant du collaborateur sur chacune d'elles.

Ayant passé en revue l'affichage des différents accès techniques, il faut maintenant expliciter le choix réalisé au niveau de l'interface d'administration de ces derniers.

6.2.2. Administration intégrée de la base de données

L'affichage des différents liens comme expliqué précédemment implique que ceux-ci soient à jour et présentables. Il est possible que certains liens deviennent obsolètes dans certains cas précis comme lorsque l'hébergeur du service décide de changer de nom de domaine, l'accès menant à celui-ci figurant sur le portail ne prévoyant pas ce changement n'est alors plus fonctionnel. Pour contrer ce type de problème il est nécessaire de pouvoir gérer les différents accès en passant par le portail afin de faciliter les démarches de chacun. Dans les faits, même une personne ne sachant pas développer devrait grâce à cette interface pouvoir être capable d'éditer les liens et cartes apparaissant sur le portail. La récupération de ces différents lien se fait via l'appel d'une requête envoyée via l'API XmlHTTPRequest* qui appelle de manière asynchrone le Contrôleur de l'entité souhaitée qui celui-ci renvoie les informations récupérées sous format d'une chaîne d'information avec le même mode de fonctionnement qu'un fichier au format .csv* permettant de stocker dans un tableau d'objets Javascript les différentes entités créées à partir de la chaîne renvoyée pour chacune d'entre elles

Todo : insérer annexe code constructeur objet JS plus affectation tab objet

Cette solution a été pensée pour éviter de récupérer plusieurs fois des informations non-susceptibles de changer afin de gagner du temps d'exécution sur la page.

Cette page d’administration se présente sous la forme de grand tableau dont l’entête permet de récupérer seulement les résultats de certaines catégories , et de trier par ordre alphabétique ou de statut en affichant en premier soit les liens affichables soit ceux qui ne le sont point (ceux dont le statut en base de données correspond respectivement à ‘0’ ou ‘1’).

Todo : insérer figure du visuel de la page d’administration

Il y est aussi possible de rechercher par le nom de lien dans une barre de recherche pour accélérer le processus de navigation.

Ces différentes options d’affichage, toutes différentes les unes des autres, ont lors du processus de développement causé beaucoup de conflits entre elles, l’un des problèmes récurrents était que lorsqu’on sélectionnait une catégorie de liens à afficher dans le tableau de résultats, qu’ensuite nous faisions un recherche dans la barre prévue à cette utilisation et que nous voulions filtrer par statut, nous obtenions un résultat assez précis correspondant à la recherche. Cependant en effaçant les caractères de la barre de recherche le résultat affiché ne correspondait plus du tout à la recherche : cela s’expliquait par les différentes opérations effectuées sur le tableau des liens affichables mettant en déroute tout le système et l’ordre d'exécution des traitements, rendant un résultat correspondant à des informations antérieures à celles voulues par les options de filtrage sélectionnées.

Dans ce cas de figure le programme Javascript faisait à chaque appel d’une des fonctionnalité une sauvegarde des liens affichés dans l’état antérieur au tri/filtrage puis affichait les données sous la forme voulue, cependant cette implémentation ne prenait pas en compte le fait que les options possédaient des états cumulatifs: dans un scénario où était appelé en premier la recherche des liens contenant le mot clé “mail” l’état antérieur à cette recherche est celui affichant tous les liens désordonnés, en ajoutant un filtre sur les noms de liens affichés par ordre alphabétique, l’état antérieur de ce filtre représente les liens de la recherche désordonnés. A ce moment du processus si on désactivait la recherche pour revenir à l’état antérieur à celle-ci nous retournions sur tous les liens en désordre alors qu’il était encore spécifié que nous les voulions dans l’ordre alphabétique!

Ces défauts d’implémentation ont poussé à une phase de test et de recherche permettant de visualiser un système permettant d’éviter tout type de conflit. Les cours de première année de BUT Informatique sur les opérations ensemblistes m’ont permis d’imaginer les différents tableaux stockant les information sur chaque opération de tri/filtrage comme des ensembles indépendant dont on fait l’intersection au moment de la recherche afin de n’obtenir que les résultats relatifs à notre recherche. Aucune option spécifiée revient à demander d’afficher tous les liens ce

qui implique que la valeur de base de chaque tableau correspondant à une action d'affichage soit égale à tous les accès récupérés via la base de données.

TODO : afficher annexe code `afficherLiensAffichables();`

La méthode array.slice() a permis de faire des copies de tableau de manière la plus optimisée possible, et la méthode array.filter(value=> array2.includes(value)) permet de faire l'opération ensembliste d'intersection de deux tableaux désignés. Le filtrage complet fait, il est possible d'enfin afficher les liens obtenus de ce calcul. Cela diffère complètement de la solution précédente puisqu'elle faisait toutes ses opérations dans un tableau réservé à l'affichage ce qui n'était clairement pas optimisé avec du recul.

Pour revenir au rôle premier de cette interface, plusieurs boutons ont été implémentés dans l'objectif d'afficher des formulaires dédiés à chaque action possible sur les différentes entités. Chaque clic sur l'un de ces boutons fonctionne de la manière: si c'est pour l'affichage d'un formulaire d'ajout, on créera une balise <form> avec les entrées souhaitées, si c'est pour un formulaire d'édition on fera de même et pré remplira les informations des différents champs avec les informations correspondant à l'entité cliquée, si c'est un formulaire de suppression on créera les boutons permettant de faire le choix de si on supprime ou non l'entité choisie. Après avoir fait ces actions la même action va être appelée, évitant la duplication de code en passant en paramètre à la fonction afficherForm() le formulaire créé afin qu'il soit affiché à l'écran.

Todo : afficher annexe code afficherform() de admin.js

Lors de la soumission des différents formulaires, la méthode de requête choisie lors de l'ajout et l'édition de données est POST car elle permet l'utilisation d'une API intégrée à Javascript permettant de réunir sous forme de tableau clé valeur les valeurs étant ajouté à l'objet voulu. Cette API se nomme FormData* et a pour avantage de parfaitement retranscrire le contenu des balises <input type="file"> au travers des requêtes envoyées ce qui m'a tourné vers son utilisation puisqu'il fallait stocker les images relatives aux cartes et les potentiels fichiers vers lesquels peuvent mener un lien affiché sur le portail. La fonction append() de ce type d'objet permet d'ajouter directement le contenu souhaité à la liste de données à envoyer dans la requête ce qui est bien pratique pour récolter les différentes données des formulaires

todo : afficher annexe code récupération depuis une form peu importe avec formdata

Une fois la récupération terminée, si toutes les infos nécessaires sont remplies, on envoie la requête à une url dont la construction est souvent la même : "../../web/nomDeL'ActionARealiser"; Ce type d'URL peut faire penser à

l'implémentation des routeurs PHP vue au S4 mais n'est pas exactement réalisée de la même manière. Après comparaison avec les besoins des projets vus en cours, je me suis rendu compte qu'un routeur PHP n'avait pas d'utilité au niveau des pages mêmes-visitées par les utilisateurs mais surtout au niveau des différents appels asynchrones. De ce fait, j'ai décidé de n'utiliser un processus de routage uniquement au niveau des appels asynchrones en mettant en place un contrôleur Asynchrone présent dans le dossier web à la racine du projet. J'ai aussi joué avec les propriétés du serveur Apache sur lequel était hébergé mon projet grâce à un fichier nommé .htaccess afin de rediriger automatiquement les URLS contenant "dev/web/" (dev étant le répertoire de mon projet qui sera soumis à changements lors de la phase de mise en production du portail) vers ledit contrôleur asynchrone qui traitera lui même quelle action il doit faire en fonction de l'URL envoyée, présente dans le tableau de paramètres $_SERVER à son attribut $_SERVER['REDIRECT_URL'].

todo : afficher annexe .htaccess permettant redirection
todo : afficher annexe code index.php dans dev/web/

Les actions ensuite exécutées par le code PHP sont relativement similaires et consistent toutes en un action sur la base de données en fonction des informations données. Les seuls cas échappant à cette règle sont ceux impliquant le dépôt de fichiers dans le répertoire d'images ou de documents du portail.

Todo : afficher annexe code fichier dans ajout carte ou ajout lien

La procédure indique qu'il faut transformer le nom du fichier en le nom de l'entité associée sans ses caractères spéciaux. On vérifie ensuite si à l'URL ou il doit être téléchargé il existe déjà un fichier à écraser ou non et essaie de télécharger le fichier souhaité dans l'un des sous-dossiers des ressources du portail. l'écrasement d'un fichier existant nécessite d'avoir les permission de supprimer ce dernier, c'est pourquoi utiliser la fonction chmod() est une sécurité afin de ne pas avoir de mauvaise surprise. Le code de retour 500 ne devrait normalement jamais être renvoyé mais est quand même prévu afin que l'application n'ait pas de comportements inattendus.

Concernant les codes de retour retournés par les différentes opérations, ils ont été pensés pour répondre au mieux aux normes RFC 9110*. Un ajout avec succès dans la base de données renvoie un code HTTP 201 CREATED, dans le cas contraire si le problème vient de la requête qui est mal formée on renvoie 400 BAD REQUEST ou alors si le problème vient du serveur on renvoie 500 INTERNAL SERVER ERROR. Lors d'une requête d'édition avec succès on renvoie 200 OK, dans le cas inverse le retour est le même que pour les opérations d'ajout. Enfin, lors de la suppression, si celle-ci s'est effectuée, le programme retourne un code 204 NO CONTENT et retourne sinon la même réponse que dans les cas précédents.

En fonction du code reçu par le gestionnaire d'événement lié à l’état de la requête, la page va afficher un toast de couleur rouge ou verte en fonction de si l’entité souhaitée a été créée.

TODO : insérer figure toast vert entité créée.

L’objectif est tout de même de ne jamais voir s’afficher de toast rouge sauf lors de la vérification des entrées saisies si jamais l’utilisateur oublie un certain champ d’un formulaire, l'empêchant d’envoyer sa requête à la base de données.

TODO : insérer figure toast rouge info oubliée dans form

Chaque action d’ajout, édition, suppression dans la base de données appelle à nouveau la récupération des différentes entités afin que l’interface soit à jour sans avoir à rafraîchir la page d’administration. Dans la plupart des cas, il suffit de faire ceci afin d’obtenir toutes les informations des liens du portail à jour. Cependant durant le processus de développement j’ai dû faire face à un cas d’utilisation précis ou une partie des informations récupérées ne voulait pas se mettre à jour: en éditant une Carte sans changer son nom mais en changeant l’image associée à celle-ci n’affichait pas directement la nouvelle image sélectionnée. Cela était dû à la mise en cache des informations : le navigateur stocke des informations relatives à une URL et lors de l’appel de celle-ci il récupère la donnée mise dans sa mémoire en cache et non celle présente dans le dossier souhaité. c’est pour ca que ce problème ne survenait pas lors du changement du nom de la carte puisqu’il n’existait pas d’image chargée dans la mémoire à ce chemin.

La correction trouvée à ce problème pour contourner la mise en cache des images à été de différencier les nouvelles et anciennes images en cache en ajoutant un horodateur au format JS Date().getTime() à la queryString récupérant l’image de sorte à ce que chaque appel mette à jour chaque chemin d’images affichant les liens avec les images correspondant à leurs cartes mises à jour.

L’une des améliorations qui serait intéressante en terme d’expérience utilisateur serait de faire en sorte que la recherche de liens implémentée via la barre de recherche présente dans la navbar permette aussi de vérifier si il y a une correspondance avec le nom d’une des cartes existantes, auquel cas il faudrait afficher dans les liens trouvés tous ceux affiliés à cette carte. Pour le moment si on tape “Ford” dans la barre de recherche, la recherche fera seulement apparaître le lien éponyme et non ceux liés à la carte possédant un autre nom de lien comme “Easy” par exemple ce qui peut poser des problèmes si l’administrateur ne connaît pas le nom du lien.

6.2.3. Affichage de statistiques correspondant à la fréquentation des liens

Afin de satisfaire le dernier besoin formulé par M. CURRIEZ et son équipe, il faudrait pouvoir dresser des statistiques sur l’utilisation personnelle de chaque accès présent sur le portail Intranet du groupe.