HOGERE ZEEVARTSCHOOL ANTWERPEN

Logiciel Webapp pour la gestion du logbook et de la maintenance en salle des machines

par Alban PIRARD

Mémoire présenté pour l'obtention du titre de
Bachelor en Mécanique Navale

Promoteur : Willem MAES
Année académique 2019 - 2020

1 Avant-propos

Lors de mes deux embarquements en tant qu'élève, j'ai pu observer que les logiciels présents sur les navires marchands ne sont pas souvent aux standards informatiques actuels. Tandis que nous pouvons organiser nos agendas en parlant à nos smartphones, ou visiter des villes en réalité virtuelle depuis notre canapé, les logiciels à bord des navires sont archaïques, remplis de bugs, quand ils ne sont pas remplacés par des tableurs Excel.

Ayant des connaissances assez diverses en informatique et voulant m'améliorer dans ce domaine, je me suis donc décidé à créer de toute pièce un logiciel pour mes confrères mécaniciens.

Les retours négatifs principaux sont sur la collecte quotidienne des données (niveaux, pressions, filtres…) qui sont enregistrées dans un tableur où la moindre erreur commise crée un branle-bas de combat de tous les machinistes pour rectifier l'erreur, sous peine d'attendre le prochain port qu'un informaticien vienne régler le problème.

L'autre souci souvent relevé est la gestion des stocks de pièces. Les stocks sont mal gérés et ne sont pas alignés avec les maintenances prévues.

Les maintenances, elles, sont souvent gérées sur un logiciel ou un tableur qui annonce seulement la date limite ou le nombre de tours auquel doit être fait l'opération.

Il fallait donc faire un logiciel qui puisse enregistrer des données, prévoir des maintenances, prévoir le nombre de pièces nécessaires à l'opération et gérer le stock en même temps. Le tout en étant facilement modifiable et configurable par les ingénieurs à bord.

Le dernier point important pour la ligne à suivre lors du développement était la facilité d'installation et l'adaptabilité sur un navire déjà en service. Il serait presque inutile de le faire fonctionner uniquement sur les nouveaux navires. Ici, nous avons une application web rapide à mettre en place, utilisable depuis n'importe quel endroit du navire, avec n'importe quel support (ordinateur, tablette, smartphone) et qui viendra vite remplacer les logiciels obsolètes présents.

Comme en informatique rien n'est jamais fini, et tout peut toujours être amélioré, quelques fonctionnalités supplémentaires ont été ajoutées pour faciliter la vie des ingénieurs.

Je tiens à remercier l'école de navigation d'Anvers, l'équipe pédagogique ainsi que mon promoteur Willem MAES pour le soutient apporté durant ce projet.

2 Résumé (FR)

Le logiciel développé pour ce mémoire a pour but principal d’être adaptable sur les navires déjà en fonctionnement. Il se veut aussi être intuitif, facile d’utilisation et complet.

Comme pour tout projet, une construction solide commence avec de la recherche, de l’anticipation et du raisonnement. Le développement n’a donc pas commencé immédiatement. Une période importante a été consacrée au choix des directions à prendre, l’ordre des parties à mettre en place et la préparation de l’environnement de développement.

Le développement a été organisé en fonction des difficultés, pour pouvoir apprendre la programmation au fil du projet. Il a aussi fallu adapter la structure en fonction des liens de dépendances qui existent entre les différentes parties.

Après une brève familiarisation avec l’aspect graphique, le développement de la partie *logbook* a débuté. S’en est suivi la partie maintenance, pièces de rechanges et gestion des stocks qui sont interdépendantes.

Des parties périphériques comme les menus, la gestion des utilisateurs et des *cookies* ont été implémentées au fur et à mesure de la programmation. Le développement s’est terminé par la page index, qui dépend totalement de toutes les parties créées au préalable.

Le logiciel est donc fini, prêt à être installé et utilisé sur un navire.

3 Abstract (EN)

The main purpose of the software developed for this thesis is to be adaptable to ships already in operation. It also has to be intuitive, easy to use and complete.

As any project, a solid build begins with research, anticipation and reasoning. Therefore, the web developing did not start directly. A consequent lapse of time was spent for choices as the directions to be taken, the order of setting up different parts and planning.

The development was organized depending on the difficulties, in order to learn web programming along the project itself. Adaptation has also been necessary towards the dependencies that exist between the different parts.

After a brief familiarization with the graphic aspect, the development of the logbook part started. This was followed by the maintenance and spare parts / stock inventory parts which are interdependent.

Peripheral parts such as menus, users and cookie management have been implemented in parallel of the core build. The development was completed with the index page which entirely depends on all the parts created beforehand.

The software is thus finished, ready to be installed on a ship and to be used.

4 Table des matières

5 Index des figures

6 Liste des abréviations

BDD : Base de données, divisée en tables, regroupant toutes les données du logiciel.

CSS : (*Cascading Style Sheets*). En français, feuilles de style en cascade. Regroupés dans divers fichiers chargés dans le « *header* » de la page HTML, ils formatent cette dernière ainsi que tous ces composants. C'est en quelques sorte le maquillage de la page.

FTP / SFTP : FTP (*File Transport Protocole*) est un protocole de transport de données sur un réseau entre deux machines. SFTP (*Secure File Transport Protocole*) ajoute un cryptage entre les deux machines afin de sécuriser l'échange.

HTML : (*Hypertext Markup Language*). Ce n'est pas un vrai langage de programmation mais un plus un codage permettant de mettre en forme des textes à l'aide de *tags*. Les *tags* doivent être ouverts puis fermés. Par exemple le *tag* h1 précise que le texte est un titre de niveau 1 soit le plus élevé. On code ainsi :

```
<h1>Mon titre</h1>
```

HTTP / HTTPS : (*Hypertext Transfer Protocol*). C'est un protocole de transfert et d'affichage de contenus de type HTML. Un navigateur comme Chrome ou Firefox est nécessaire pour l'utiliser.

PHP : C'est le langage de programmation le plus utilisé pour concevoir des pages *Web* dynamiques. C'est un langage interprété par un serveur W*eb* comme apache par exemple et qui n'a pas besoin d'être compilé afin de s'exécuter. Il est libre et gratuit et possède un très grand nombre de librairies.

SQL : (*Structured Query Language*). C'est un langage qui permet d'exploiter une base de données relationnelles. On peut faire des recherches, des ajouts, des modifications et des suppressions de données.

URL : (*Uniform Resource Locator)*. C'est l'adresse d'une page ou d'un site Internet que l'on peut voir dans la barre d'adresse du navigateur. Elle commence par http:// ou https://.

WEBAPP : Abréviation pour une application *Web*. C'est une application accessible par un navigateur et qui est exécutée d'une part depuis un serveur *Web*, et d'autre part sur la machine cliente via le navigateur. La plus connue étant Gmail de Google.

7 Introduction

Les logiciels actuellement présents à bord des navires sont sous licence payante et généralement non-accessibles pour les particuliers. De plus, l'expérience d'utilisation de ces logiciels lors d'embarquements a mis leurs lacunes en évidence.

L'interrogation de départ a été de savoir s'il était possible de créer un logiciel mieux sur tous les points, en partant d'une feuille blanche. Et si oui, alors il fallait savoir par quels moyens, pour ensuite pouvoir le réaliser entièrement.

Au niveau informatique, la recherche a été consacrée aux différents systèmes et langages de programmation disponibles et possibles à maîtriser sur une courte période. Pour rester cohérent avec les principes de projet universitaire, l'accent a été mis sur l'utilisation de systèmes gratuits et libres.

Les orientations ont toutes été prises en fonction des expériences à bord et de documents techniques provenant de navires.

Le principal atout de ce projet, mis au point par un étudiant mécanicien, est la polyvalence qui permet une logique et une harmonie bien plus complète du logiciel. Il est très rare qu'un logiciel soit fait directement par la personne qui l'utilise. Le plus souvent, les incompréhensions des développeurs sur les besoins des utilisateurs nuisent à l'efficacité du système.

Chaque décision, page, ligne de code ou table SQL est ainsi pensée pour améliorer la qualité et l'efficacité du travail à bord. C'est le fil conducteur de ce mémoire.

La partie logbook a été faite pour simplifier la rentrée de données manuelles qui est quotidienne à bord (relevé des niveaux, des pressions, tout les capteurs non automatisés). Ce n'est pas révolutionnaire, mais juste une version améliorée de ce qui existe déjà. En revanche, ce qui est nouveau et possible grâce à ce système, c'est le suivi de ces information à bord ainsi que leur paramétrage.

Un autre aspect important de la marine est la sécurité. Les fautes, ou même les accidents, arrivent à cause d'erreurs qui peuvent être faites. Ces erreurs peuvent être rectifiées facilement avec un bon suivi. Mais pour détecter les erreurs involontaires inaperçues, voire

même les fraudes, une traçabilité a été mise en place. Chaque action importante faite sur le logiciel est alors enregistrée. À l'image d'une boîte noire, ces détails seront visibles seulement en cas de problème, par un administrateur, sur autorisation de la compagnie.

Des logiciels pour les maintenances prévues, ou les pièces de rechange, existent aussi. Seulement ils sont propriétaires à une certaine marque, ou ne comprennent pas toutes les informations. Le système développé ici ne remplace pas certains logiciels propriétaires comme ceux pour commander des pièces. Par contre, il centralise les informations dans un cadre clair, concis et simple. Mais surtout il apporte, encore une fois, un suivi et une accessibilité au niveau du navire.

L'accès aux données à bord est le fer de lance de ce logiciel. À l'exemple de la page index qui pourra être affichée en permanence sur l'ordinateur du Chef mécanicien, du Capitaine ou d'un officier. Pour un travail efficace, les informations doivent circuler vite et clairement.

8 Texte

8.1 Choix des directions à prendre

8.1.1 Choix de l’environnement d’exécution

Pour commencer, il a fallu choisir entre un logiciel installé directement sur les ordinateurs, ou un logiciel en ligne type *Webapp* dans un navigateur.

Un logiciel conventionnel fonctionnerait, mais avec des limites d’adaptation et de polyvalence difficilement contournables. En effet, il faudrait l’installer sur chaque ordinateur à bord, ce qui prendrait un temps considérable pour chaque navire et de même pour les mises à jours. De plus, il ne pourrait pas être installé en toute sécurité sur les ordinateurs personnels des officiers. Aussi, il ne fonctionnerait que sur les systèmes possédant le même *os* (*Windows*) avec ses problèmes que les autres logiciels connaissent déjà.

La seconde solution a été retenue pour avoir un logiciel pouvant être utilisé partout à bord d’un navire.

Un autre point important est l’échange des données. Dans une configuration classique, le logiciel installé sur l’ordinateur est un logiciel client. Il faut un logiciel serveur, installé sur un serveur local (à bord), ou sur un serveur distant nécessitant une connexion internet permanente.

L’avantage principal des logiciels conventionnels est la puissance de calcul fournie par la machine client et non par le serveur. Hors, dans notre cas, nous n’avons absolument pas besoin de puissance de calcul ,vu que nous allons seulement gérer des données.

Les logiciels type *Webapp* sont de plus en plus répandus et pour de bonnes raisons.

Premièrement, un seul logiciel doit être programmé. La partie client se fait à travers le navigateur W*eb*. C’est un avantage considérable à bord d’un navire. Il suffit d’installer le logiciel sur un serveur à bord. Tous les périphériques connectés au réseau local pourront y accéder quelques soient leurs systèmes d’exploitation ou taille d’écran (ordinateurs, tablettes, *smartphones*).

Deuxièmement, l'autre avantage d'une *Webapp* est la facilité de mise à jour. Sur un réseau internet normal, les mises à jours faites par les développeurs sont instantanées pour les utilisateurs. Sur un réseau local comme à bord, il suffit d'avoir un accès internet au serveur (par exemple dans un port), pour qu'un développeur fasse la mise à jour du serveur à distance. Les utilisateurs n'auront jamais à faire de manipulations pour faire fonctionner le logiciel et seront toujours tous en train de travailler sur la même version.

Le cahier des charges voulait un logiciel simple. Avec tous ces paramètres il paraissait donc évident de partir sur un développement de logiciel en ligne.

8.1.2 Choix du langage de programmation

Une fois le choix de la *Webapp* en ligne, une autre décision devait être prise. Il y a plusieurs langages possibles. Le choix n'étant pas un seul langage mais une combinaison de plusieurs.

Comme environnements de développement connus on trouve :

- Microsoft .NET. C'est une technologie répandue mais qui est propriétaire et payante.
- JAVA qui est assez simple et orienté objet, développé par Oracle. Il est moins rapide et utilise plus de ressources que les langages basés sur le langage C.
- PHP/JAVASCRIPT qui a été choisi pour multiples raisons :
    - basé sur le langage C (ayant des connaissances de base en Python l'apprentissage serait plus rapide),
    - gratuit et *open source*,
    - une importante communauté de développement, forums et tutoriels.

8.1.3 Choix du système de gestion de base de données

Il y a plusieurs types de bases de données (BDD)[9]. Le choix ici s'est porté sur une base de données relationnelles[11].

Plusieurs avantages se sont montrés convaincants :

- ce type de BDD fonctionne parfaitement pour les données bien structurées et sécurisées,
- la communication entre le serveur et la BDD est rendu facile par les requêtes SQL (qui seront faites ici via un connecteur PHP),
- une BDD scalable : le traitement des données ne sera pas ralenti par l'augmentation de celles-ci,
- la bonne gestion de la sécurité et des droits d'accès à plusieurs niveaux.

Il existe plusieurs systèmes de gestions de BDD. MySQL, Oracle Database, PostegreSQL, et Microsoft SQL Server en sont les principaux.

Avec mon hébergement est fourni MySQL que j'ai choisi. C'est un système libre qui fonctionne bien sur toutes les plateformes et qui a déjà fait ses preuves au niveau des *webapps* (Facebook, Twitter, Youtube…).

8.1.4 Choix de l'environnement graphique

Le logiciel a pour but d'être agréable à utiliser, ce qui est rarement le cas dans les logiciels spécialisés.
Les critères choisis sont donc d'avoir un logiciel :

- moderne,
- fluide,
- intuitif,
- personnalisable.

Actuellement, ce qu'il se fait de mieux au niveau des interfaces logiciels *webapp* est le *framework Bootstrap*.

Après quelques recherches, un modèle (*template*) libre (gratuit) basé sur *bootstrap* 4 et sous licence MIT a montré posséder toutes les qualités requises.
Ce modèle est le sb-admin-2 [4].
Il remplit les critères graphiques souhaités, et en plus présente des outils de base et une présentation qui convient bien au projet.

8.1.5 L'ordre des parties à développer

Ne connaissant pas ces langages de programmation, il a fallu décider un ordre. Commencer du plus facile et aller vers le plus complexe est une bonne méthode pour avoir un développement fluide en accord avec l'apprentissage.

Le plus facile étant le HTML et ensuite le PHP, le MySQL et finalement le JAVASCRIPT.

Les bases de données étant toutes connectées entre elles via PHP et MySQL, il a aussi fallu discerner l'ordre pour les dépendances. Par exemple il n'est pas judicieux de faire la partie « maintenance » avant la partie « *spareparts* ». Sinon, il faudrait revenir plus tard sur la première pour ajouter ce qui n'existait pas au moment de la programmation.

Les longs fichiers se multiplient vite lors du développement. Pour ne pas se perdre et commettre des grosses erreurs, il faut bien planifier à l'avance. Les problèmes ne disparaîtront pas mais les risques peuvent être diminués.

Cette réflexion est très importante. Une aberration dans le plan peut prendre des jours de travail à rectifier, voire même tout recommencer depuis le début.

La ligne à suivre choisie a donc été :

1. La partie *logbook*
2. La partie *spareparts*
3. La partie maintenance
4. L'index

Certains aspects complexes aurait du être faits en premier pour une meilleure fluidité dans le développement. La gestion des utilisateurs et la sécurité des données sont de bons exemples. Ils sont présents dans toutes les parties. Malheureusement, n'ayant pas les connaissances requises au début, il a fallu les faire plus tard.

8.1.6 Organisation des BDD dans le logiciel

Après plusieurs schémas pour organiser les différentes parties qui vont communiquer entre elles, un système de base s'est rapidement mis en place. Au fur et à mesure du développement, des parties se sont ajoutées pour améliorer le système de base.

Pour une meilleure compréhension des différentes parties et leurs connections, un organigramme a été créé (figure 1).

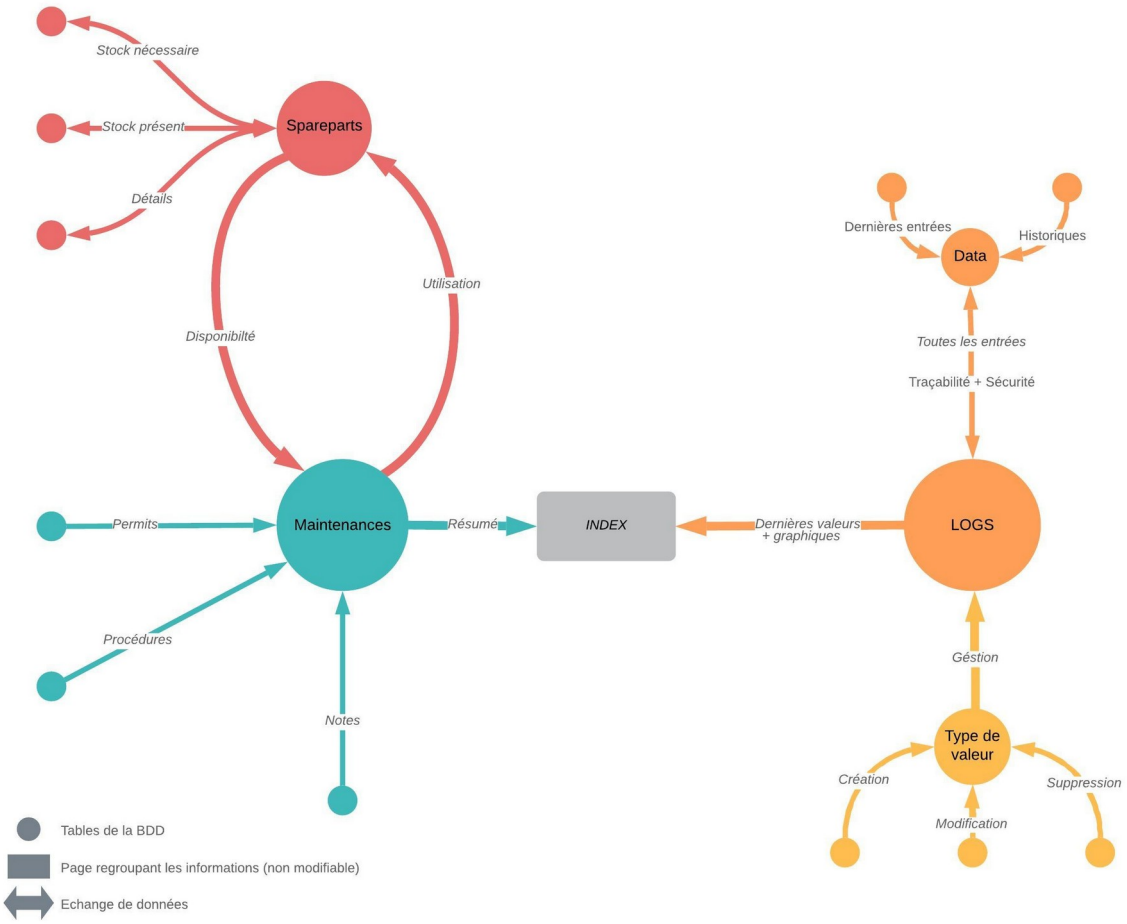


Figure 1 : Organigramme général du logiciel et BDD

Source : fait sur Lucidchart [10]

Certaines bases de données et pages W*eb* ne sont pas présentes dans l'organigramme car elles sont principalement fonctionnelles et moins représentatives du fonctionnement global.

8.2 Préparation

8.2.1 Mise en place du serveur *Web*

Plutôt que de développer le logiciel en local (fichiers sur mon ordinateur), j'ai préféré prendre un hébergement gratuit auprès d'une association locale [1] qui me fournit :

- un espace W*eb* de 5Gb,
- un serveur apache avec PHP et toutes les librairies nécessaires,
- un serveur MySQL avec une base de données,
- les outils PHP/MySQL dont phpMyAdmin [6].

8.2.2 Mise en place des outils

Tout d'abord l'ordinateur pour le développement a été réinstallé sous Zorin OS 15, un système d'exploitation Linux libre basé sur Debian.

Pour les échanges de fichiers avec le serveur distant, Filezilla (figure 2) est un logiciel libre tout à fait adapté. Il gère très bien le SFTP(6).

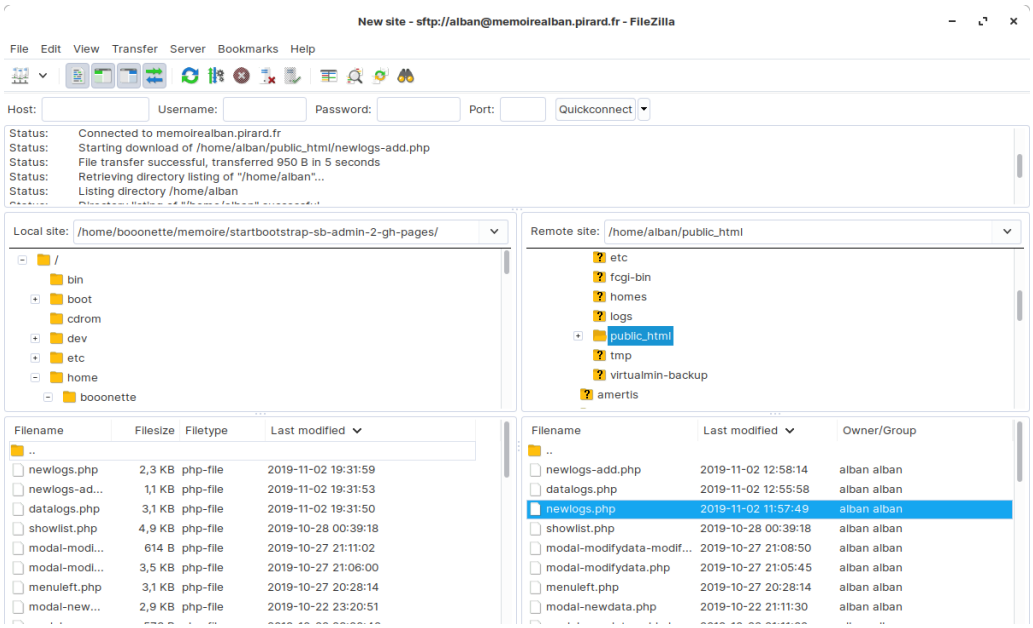


Figure 2 : Logiciel FileZilla [8]

Source : capture d'écran

Pour visualiser, écrire et modifier les fichiers, VSCodium (figure 3) a été utilisé. C'est une version libre du logiciel de programmation VSCode de Microsoft.

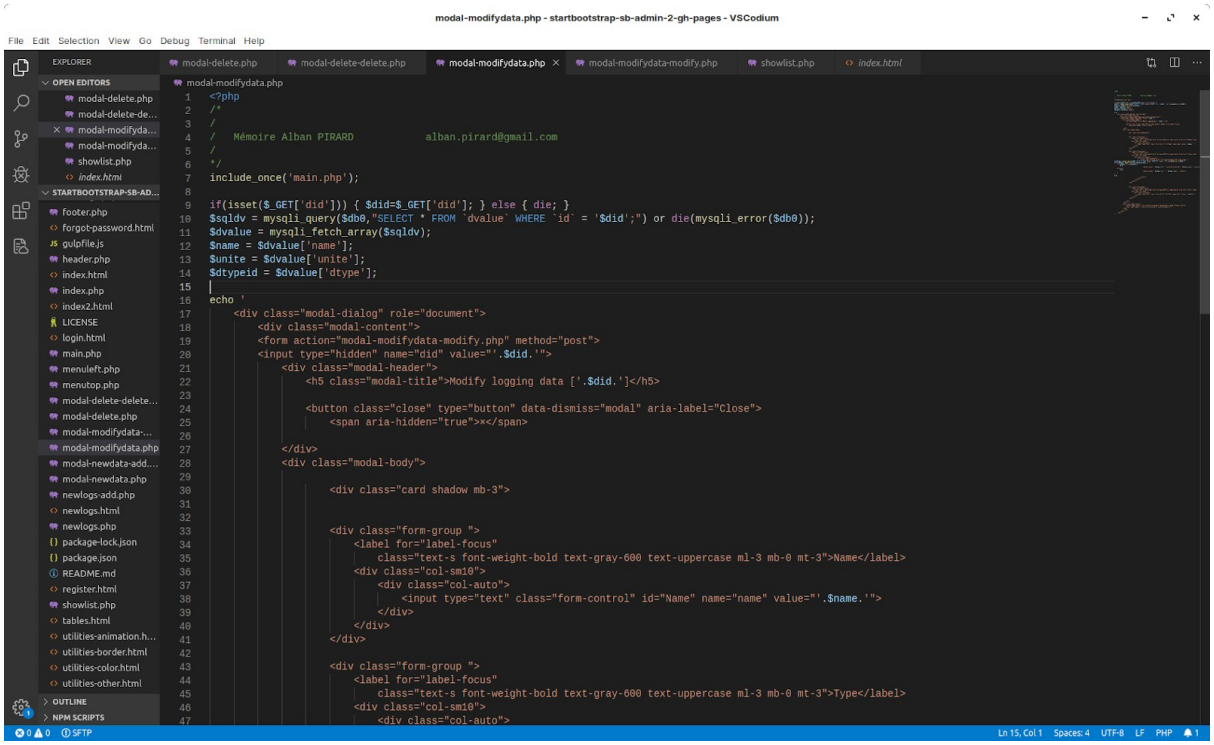


Figure 3 : Logiciel VSCodium [12]

Source : capture d'écran

8.2.3 Mise en place de la partie graphique : HTML

Une fois les différents fichiers HTML mis en ligne sur le serveur, on peut commencer à les modeler pour adapter les pages à nos futurs besoins.
La page « index.html » (figure 4) a servi d'apprentissage du langage HTML.
La bonne surprise a été le formatage HTML d'origine, bien organisé et avec des commentaires qui aident à se repérer dans le code.

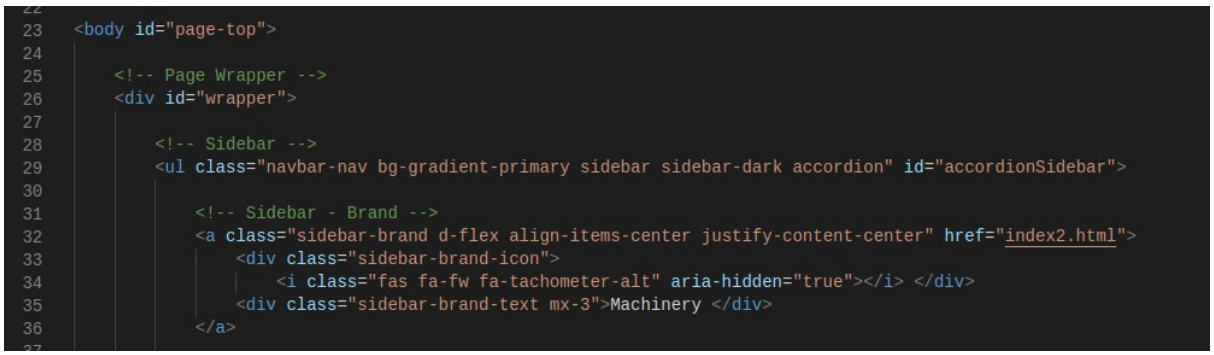


Figure 4 : Code de la page « index.html »

Source : capture d'écran

De plus, le HTML s'est révélé être un langage très intuitif. Une fois qu'on a compris le format des lignes, la syntaxe et la structure des pages, il est très facile de mettre en place des pages à son goût.

Commencer par travailler sur une page index en pur HTML a donc été une bonne idée. Cela a permis d'apprendre la base qui était nécessaire pour la suite.

8.2.4 La segmentation

Dans le modèle de base, chaque page (URL) est composée d'un seul fichier HTML.

Le code est donc trop long et trop brouillon pour être utilisé tel quel. Le contenu des pages va être travaillé pour chaque fichier, tandis que certaines parties sont toujours les mêmes (comme le menu gauche, la barre en haut et le pied de page (figure 7)). Il suffit de les mettre dans un fichier PHP séparé, que l'on appelle (récupère) dans nos fichiers PHP principaux.

Ces fichiers sont « menuleft.php », « menutop.php » (figure 5) et « footer.php » (figure 6). (voir chapitre 8.6)

files-procedure.php

```php
<?php
/*
/
/   Mémoire Alban PIRARD            alban.pirard@gmail.com
/
*/
include_once('main.php');
include('header.php');
include('menuleft.php');
include('menutop.php');

```

Figure 5 : Début de page PHP

Source : capture d'écran

```php
include('footer.php');
?>
```

Figure 6 : Fin de page PHP

Source : capture d'écran

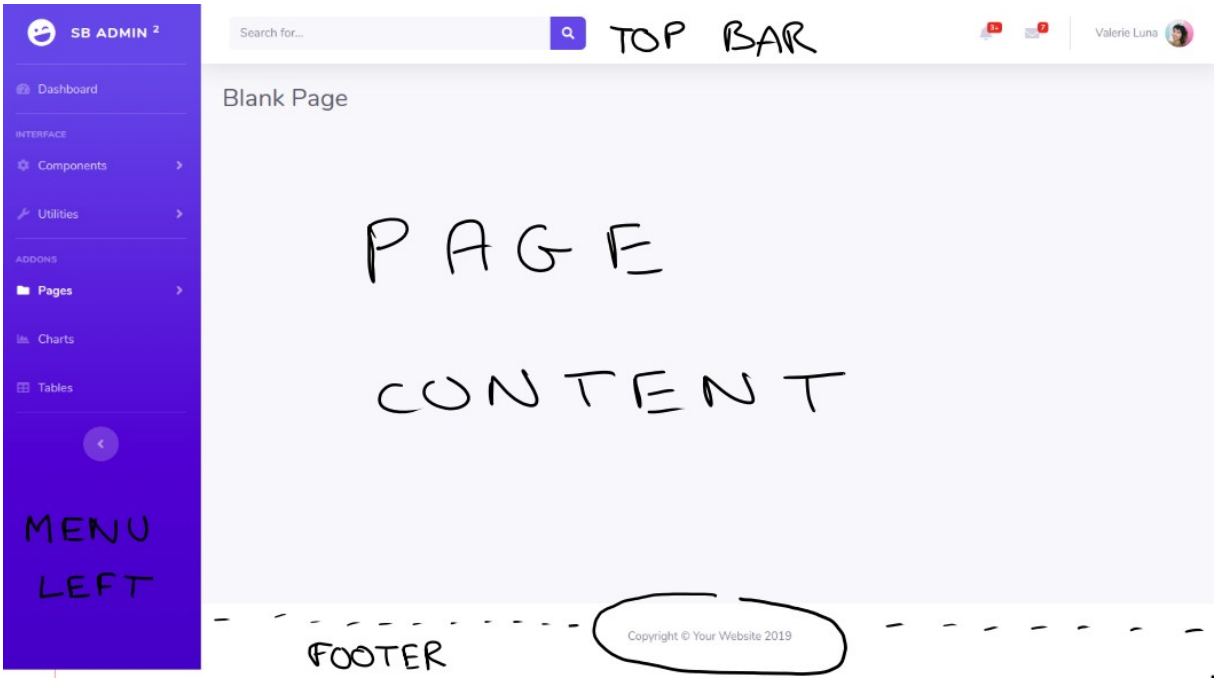


Figure 7 : Segments récupérés du modèle HTML

Source : capture d'écran

8.2.5 Les paramètres

Pour chaque script PHP, il est nécessaire d'avoir :

- un certain nombre de paramètres comme par exemple le connecteur MySQL pour la BDD avec le *login* et mot de passe,
- diverses données comme l'ID de l'utilisateur ou son IP,
- des fonctions récurrentes.

Tout cela est rassemblé dans un unique fichier appelé ***main.php*** (chapitre 8.6.5) et qui est appelé au début de chaque script PHP.

8.3 Partie *logbook*

8.3.1 Base de données

Pour gérer les données, il faut mettre en place une base de données en MySQL. Pour ce faire, j'ai utilisé phpMyAdmin, une application préinstallée qui permet de gérer la base de données (créer des tables, les paramétrer et les gérer).

Tout d'abord, il faut créer une table des différents logs qui seront entrés (Niveau d'huile moteur, pression dans le FW *generator*, etc.).

Dans cette table (*dvalues*), sont répertoriés :

- un ID (pour y associer une valeur),
- un nom,
- une unité (cm, kPa…),
- un type de valeur (pression, température, niveau…),
- un min et un max des valeurs possibles,
- la personne qui l'a créée et la dernière à l'avoir modifiée.

J'ai voulu organiser les *logs* par type de valeurs, comme c'est souvent le cas à bord, lors de la routine du *junior officer*. Donc, la table des types se compose d'un ID, un nom et des informations sur la création et modification.

Ces deux tables peuvent donc être remplies ou modifiées à bord, mais ce sera assez rare une fois que tout est opérationnel. L'ajout se fera, par exemple, si on ajoute un nouveau capteur de pression et qu'on veut un suivi supplémentaire à cet endroit.

Finalement, la table qui nous intéresse le plus sera celle des *logs*, composée de :

- un ID,
- une date et heure,
- l'ID de la personne qui l'a rentré,
- le type de *logs*,
- l'ID du nom (de la première table pour créer la liaison),
- et la valeur rentrée,

A partir de ces tables, la création des pages du logiciel consacrées aux logs est possible.

La première page a été « datalogs.php ». C'est un tableau qui se met à jour tous les jours et qui affiche, pour chaque valeur, le dernier log de la journée. C'est important de pouvoir entrer des valeurs différentes au cours de la journée, et d'avoir une trace de toutes les *logs* pour apercevoir s'il y a un problème.

Les *logs* s'ajoutent et sont modifiables ou supprimables par les officiers machines. C'est important qu'ils soient modifiables, car une erreur de mesure peut être commise, ou une erreur lors de la saisie des données et cela peut fausser le système.

8.3.2 La sécurité informatique

Lors du paragraphe précédent, il a été remarqué que l'accès et la modification des données étaient très importants, mais aussi dangereux concernant la responsabilité des officiers.

La situation que l'on veut éviter est par exemple : le 4eme fait sa tournée, remarque trois jours consécutifs avec une baisse anormale du niveau d'huile du moteur principal. Le second lui dit que ce n'est pas possible, et modifie les valeurs du 4eme. Une semaine plus tard, le manque d'huile crée une surchauffe et une casse ou un incendie. La faute va être mise sur le 4eme car les données rentrées étaient fausses.

Pour palier ce problème, un système de *logs* a été mis en place qui permettent d'enregistrer toutes les créations/modifications/suppressions en fonction de :

- la date,
- l'IP de la machine,
- le nom d'utilisateur,
- la donnée,
- et l'action faite.

Ainsi, en cas de problème, ou conflit, il faudra juste contacter une personne tierce pouvant retracer les faits.

8.3.3 Les difficultés rencontrées

8.3.3.1 L'organisation

Pour un tableau comme celui ci, il faut faire plusieurs requêtes SQL. Ce système de requêtes avec des boucles « *while* » est utilisé dans tout le logiciel. La partie *logbook* a été la première partie à utiliser ces systèmes, donc il est important d'en expliquer le fonctionnement.

Déjà il y a plusieurs cartes. Les cartes sont des blocs indépendants dans les pages W*eb* du logiciel. Chaque carte possède un tableau, trié en fonction du type de données :

- niveau
- pression
- filtre
- température
- etc.

Dans le code, on le retrouve sous la forme

id	name
1	level
2	pressure
3	temperature
4	filter
5	running hours
6	water

Figure 8 : Base de donnée MySQL « dtype »

Source : capture d'écran

```
23  $sqltype = mysqli_query($db0,"SELECT * FROM `dtype`;")or die(mysqli_error($db0));
24
25  while($type = mysqli_fetch_array($sqltype)) {
26      $typeid = $type['id'];
27      $typename = $type['name'];
28
29      $sql = mysqli_query($db0,"SELECT * FROM `dvalue` WHERE dtype = '$typeid';")or die(mysqli_error($db0));
30
31      echo '
32                  <div class="card shadow mb-4">
33                      <div class="card-header py-3">
34                          <h6 class="m-0 font-weight-bold text-primary">'.strtoupper($typename).'</h6>
35                      </div>
```

Figure 9 : Boucle des cartes en fonction des « dtype »

Source : capture d'écran

La variable « $sqltype » récupère les données de la base de données « dtype » (ID et nom).

Avec la boucle « *while* », on permet au logiciel de pouvoir tout afficher dans des cartes différentes. Pour chaque type de données (pression, température…) il va créer une carte. Dans cette carte on fait une autre requête SQL pour appeler les « dvalues » qui ont comme type de données l'ID du « dtype ».

	id	name	unite	dtype	min	max	usercreate	usermodify
Edit Copy Delete	3	Diesel Oil	cm	1	35	480	1	4
Edit Copy Delete	4	FWT Left	m3	6	0	0	1	1
Edit Copy Delete	5	FWT Center	m3	6	0	0	1	1

Figure 10 : Tableau MySQL "dvalues"

source : capture d'écran

Cela peut paraître confus, mais justement, la création de boucles rend le système plus simple et plus clair.

Par exemple :

1. « sqltype » récupère les données de « dtype » (figure 8),
2. la boucle « *while* » commence (figure 9),
3. elle affiche la première carte, celle des *levels* (dans le tableau de la figure 9, la carte s'affiche avec le titre dans la figure 19),
4. elle va chercher dans « dvalues » toutes les lignes qui ont un « dtype » = 1 (figure 10).

Et, à partir de ce point, la création de tableaux devient plus simple et mieux organisée.

8.3.3.2 Le tableau

Dans le « *card body* » on crée un tableau, et on annonce les différentes colonnes qui seront présentes.

Ici le tableau a été choisi avec des titres de colonnes en haut et en bas pour une meilleur visibilité.

Si les tableaux sont longs, il sera plus facile de se référer au bas du tableau que de remonter en haut pour voir quelle case correspond à quelle colonne.

Le tableau étant mis en place, il faut créer une autre boucle qui va afficher chaque ligne de la base de données.

Certains champs de nos « dvalues » font référence à d'autre tables MSQL. Il faut donc une requête SQL pour chaque valeur dont le nom ou l'info n'est pas directement dans le tableau « dvalues ».

La boucle est la suite de la demande SQL de rechercher toutes les valeurs pour un « dtype ».

```
    <thead>
        <tr align="center">
            <th>Name</th>
            <th>Min</th>
            <th>Max</th>
            <th>Unit</th>
            <th>Last Log Date</th>
            <th>Last Log User</th>
            <th>User Create</th>
            <th>User Modify</th>
            <th>Actions</th>
            <th>ID</th>
        </tr>
    </thead>
    <tfoot>
        <tr>
        <th>Name</th>
        <th>Min</th>
        <th>Max</th>
        <th>Unit</th>
        <th>Last Log date</th>
        <th>Last Log User</th>
        <th>User Create</th>
        <th>User Modify</th>
        <th>Actions</th>
        <th>ID</th>
</tfoot>
    <tbody>';
```

Figure 11 : Titres de colonnes du tableau

source : capture d'écran

```
while($data = mysqli_fetch_array($sql)) {
    $did = $data['id'];
    $min = $data['min'];
    $max = $data['max'];

    $useridcreate = $data['usercreate'];
    $sqluser = mysqli_query($db0,"SELECT * FROM `users` WHERE id = '$useridcreate';")or die(mysqli_error($db0));
    $user = mysqli_fetch_array($sqluser);
    $username = $user['login'];

    $useridmodify = $data['usermodify'];
    $sqluser2 = mysqli_query($db0,"SELECT * FROM `users` WHERE id = '$useridmodify';")or die(mysqli_error($db0));
    $user2 = mysqli_fetch_array($sqluser2);
    $username2 = $user2['login'];

    $sqllastlog = mysqli_query($db0,"SELECT * FROM `dlogs` WHERE `dtype` = '$typeid' AND `dname` = '$did' ORDER BY `date` DESC L
    $date = mysqli_fetch_array($sqllastlog);
    $lastdate = $date['date'];


    $sqllastuser = mysqli_query($db0,"SELECT * FROM `dlogs` WHERE `dtype` = '$typeid' AND `dname` = '$did' ORDER BY `date` DESC
    $lastuser = mysqli_fetch_array($sqllastuser);
    $lastuserlog = $lastuser['user'];
    $sqllastuser2 = mysqli_query($db0,"SELECT * FROM `users` WHERE `id` = '$lastuserlog';")or die(mysqli_error($db0));
    $lastuser2 = mysqli_fetch_array($sqllastuser2);
    $lastuserlog2 = $lastuser2['login'];
```

Figure 12 : Boucle + requêtes SQL tableau

Source : capture d'écran

Ici, on peut voir que les valeurs pour le minimum et le maximum sont directement dans la BDD, alors il suffit de les définir.

En revanche, le nom de la personne qui a créé la ligne n'est pas affiché, seulement son ID est présent.

On fait donc une requête SQL vers la BDD des « *users* » pour récupérer le nom en fonction de l'ID.

Pour certains champs, il faut naviguer entre plusieurs BDD afin de récupérer un nom ou une valeur (comme pour « $lastuserlog2 » dans la figure 12 qui affiche le nom de la dernière personne à avoir inséré un *log* pour ce champs).

La difficulté principale est donc d'avoir un code organisé avec des noms définis clairement.

Finalement, une fois toutes les données rassemblées, on peut les insérer dans le tableau.

Dans la figure 13, on peut voir que chaque ligne est une boucle qui correspond à un ID (qui n'est pas affiché).

Pour chaque ID, le code va chercher les noms, valeurs automatiquement.

```
<tr>
    <td>'.$data['name'].'</td>
    <td align="center">'.$min.'</td>
    <td align="center">'.$max.'</td>
    <td align="center">'.$data['unite'].'</td>
    <td align="center">'.$lastdate.'</td>
    <td align="center">'.$lastuserlog2.'</td>
    <td align="center">'.$user['login'].'</td>
    <td align="center">'.$user2['login'].'</td>
    <td align="center"><button type="button" class="btn btn-warning" data-dismiss="modal" onClick="
        $(\'#mymodal\').load(\'modal-modifydata.php?did='.$did.'\');
        $(\'#mymodal\').modal(\'show\');
        return false;">Modify</button>
    <button type="button" class="btn btn-danger" onClick="
        $(\'#mymodal\').load(\'modal-delete.php?did='.$did.'\');
        $(\'#mymodal\').modal(\'show\');
        return false;">Delete</button></td>
    <td align="center">'.$data['id'].'</td>
</tr>';
```

Figure 13 : Création automatique des tableaux avec les données MySQL

Source : capture d'écran

Dans la dernière colonne apparaissent les boutons « modifier » et « supprimer ».

Ces boutons ouvrent une fenêtre modale tout en envoyant l'ID de la ligne du tableau ($did) sur laquelle on a cliqué (qui est aussi l'ID de la ligne dans notre BDD).

8.3.3.3 Les modales

Les modales sont des petites fenêtres qui apparaissent en superposition de la fenêtre principale. Elles sont appelées par un code en Javascript. Elles permettent d'interagir sur la fenêtre en cours. Par exemple, pour modifier une donnée, ou pour demander une confirmation avant une action irrémédiable.

8.3.4 L'aspect visuel

8.3.4.1 Les tableaux

Chaque tableau (figure 14) est dans une carte en fonction du type de valeur. On y retrouve les valeurs des sept derniers jours. Si une valeur a été rentrée plusieurs fois, seulement la dernière rentrée sera affichée. En revanche, toutes les données rentrées sont accessibles par le Chef (ou toute personne ayant les droits d'accès équivalents). Il suffit d'être connecté avec un compte autorisé et cliquer sur la ligne souhaitée.

WATER

Name	22/05/2020	21/05/2020	20/05/2020	19/05/2020	18/05/2020	17/05/2020	16/05/2020
FWT Left	40.00 m3	40.00 m3	40.00 m3	38.00 m3	36.00 m3	36.00 m3	42.00 m3
FWT Center	25.00 m3	30.00 m3	35.00 m3	42.00 m3	42.00 m3	42.00 m3	40.00 m3
FWT Right	40.00 m3	40.00 m3	40.00 m3	35.00 m3	30.00 m3	30.00 m3	41.00 m3
Name	22/05/2020	21/05/2020	20/05/2020	19/05/2020	18/05/2020	17/05/2020	16/05/2020

Figure 14 : Tableau des logs

Source : Capture d'écran

8.3.4.2 Le détail des données

Une fois la ligne d'une mesure cliquée, la page de détail s'ouvre. On y retrouve toutes les valeurs rentrées, ainsi que la date et la personne concernée.

Si une erreur, lors de l'ajout, a été commise, le Chef peut venir supprimer ou modifier une donnée. Ces actions sont bien sur enregistrées pour éviter les problèmes de sécurité cités précédemment.

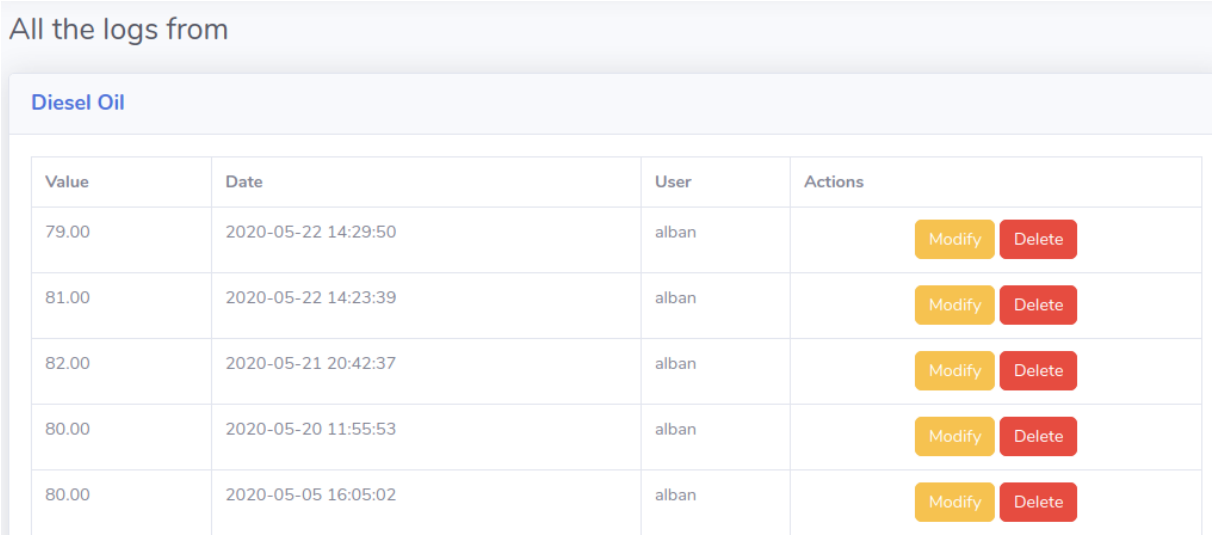


Figure 15 : Page de détail d'une ligne des logs

Source : Capture d'écran

Si aucune mesure n'a été rentrée un jour, alors il n'est pas possible de la rentrer à posteriori. C'est un choix qui a été fait pour, de nouveau, éviter la falsification de données.

La modification et suppression se font à travers des modales (figures 16 et 17) pour éviter les erreurs liées à un clic accidentel.

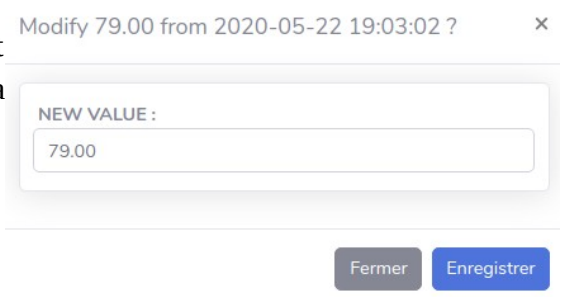


Figure 16 : Modale de modification d'une mesure enregistrée

Source : Capture d'écran

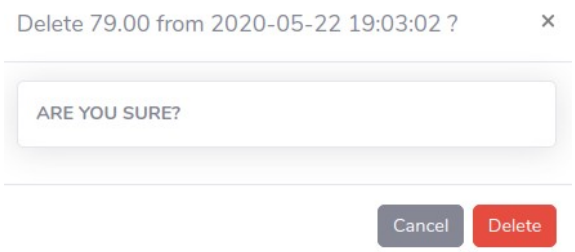


Figure 17 : Modale de suppression d'une mesure enregistrée

Source : Capture d'écran

8.3.4.3 Le formulaire

Le formulaire pour entrer les mesures est accessible seulement si un compte est connecté.

La page se compose de plusieurs cartes, toujours en fonction du type de valeur. On peut remplir le nombre de lignes que l'on veut. Cela permet, par exemple, à un officier qui remarque une mesure différente au cours de la journée, de venir l'ajouter sans devoir tout remplir.

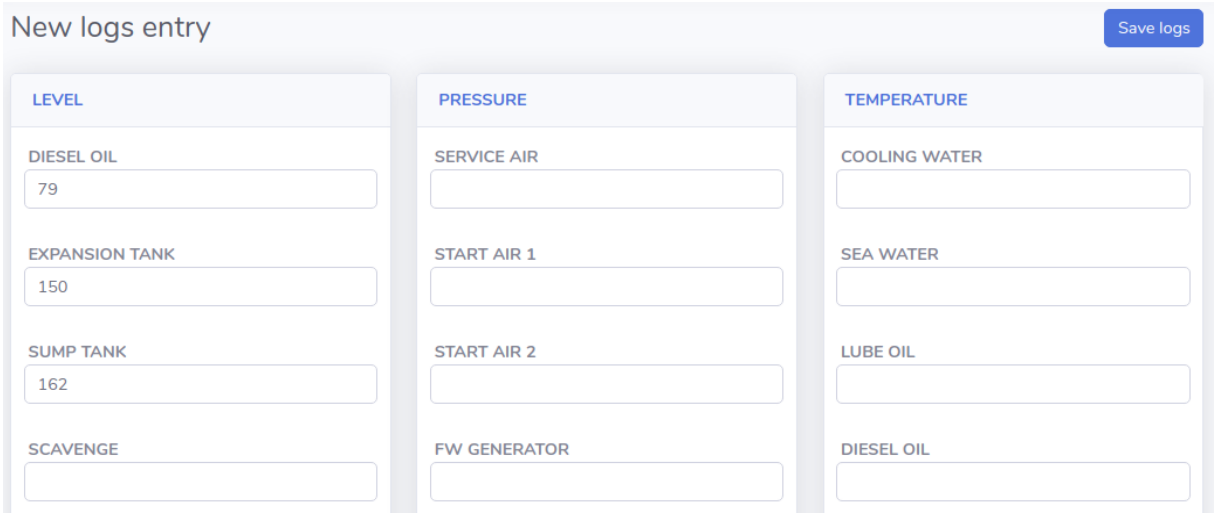


Figure 18 : Page formulaire d'ajout de logs

Source : Capture d'écran

Quand des mesures sont écrites dans les champs, il suffit d'appuyer la touche entrée ou cliquer sur le bouton « *Save logs* ». Les données sont alors enregistrées et une redirection est faite vers la page des logs.

8.3.4.4 Les paramètres des différentes mesures

Cette partie n'est visible que pour les officiers ayant un droit de modification sur le logiciel.

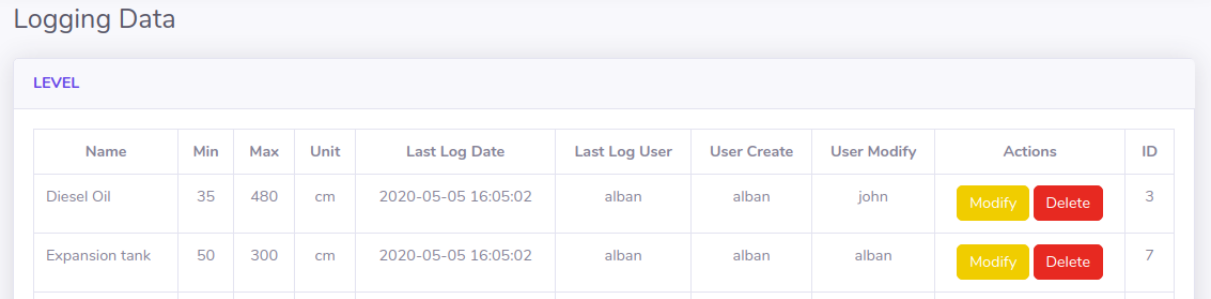


Figure 19 : Tableau récapitulatif des différents logs

Source : capture d'écran

Dans ce tableau on peut voir :

- les valeurs max et min de chaque mesure ainsi que l'unité,
- la date de la dernière mesure rentrée ainsi que la personne concernée,
- les noms des officiers qui ont créé et modifié ces paramètres,
- un bouton de modification et suppression.

L'ajout et la modification d'une mesure se font à travers une modale identique.

La modale de modification s'affiche avec les valeurs actuelles déjà présentes dans les champs (figure 20).

La modale de suppression est juste une confirmation comme précédemment dans la partie des *logs* (figure 17).

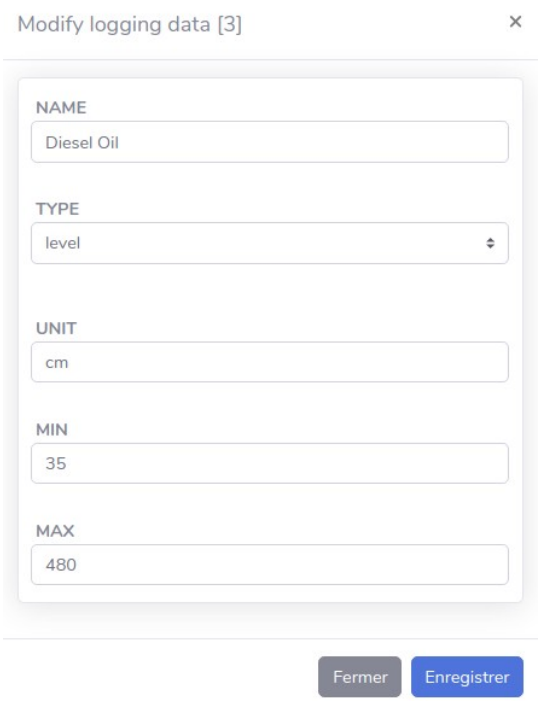


Figure 20 : Modale de création et modification d'une mesure

Source : Capture d'écran

8.4 La partie maintenance

8.4.1 Base de données

La table principale pour les maintenances est composée de :

- un ID de la maintenance (seulement utile pour la BDD),
- le numéro de maintenance que l'on retrouve dans les manuels,
- le type de maintenance (numéroté de 1 à 3) :
  - 1 = réparation d'une casse (*emergency*),
  - 2 = réparation d'une future casse (*preventive*),
  - 3 = maintenance prévue par le constructeur (*scheduled*),
- le numéro de la machine concernée (uniquement utile pour la BDD),
- le numéro du manuel dans lequel la maintenance est décrite,
- la date de création de la maintenance,
- la date limite pour l'effectuer (si prévue par le constructeur),
- le nombre d'heures de fonctionnement limite pour l'effectuer (comme précédemment),
- la date à laquelle la maintenance a été archivée,
- les notes.

Autour de cette table principale viennent se greffer plusieurs autres telles que :

- les pièces de rechange (*spareparts*),
- le lien entre les pièces de rechange et le numéro de maintenance,

- les différentes machines,
- les numéros de maintenances,
- les types de maintenances,
- les permis de travail enregistrés,
- les permis vierges,
- les procédures des maintenances.

L'organisation de base est la même que pour la partie *logbook*. La différence ici est le nombre de tables périphériques.

Une des difficultés a été de différencier une maintenance en particulier (par exemple changer les segments du MP à 40 000 heures) de la maintenance en générale (changement des segments du MP).

Ainsi, une fois qu'une maintenance est créée, certaines informations comme les notes, les pièces à prévoir et les instructions sont liées à la maintenance en général. La prochaine fois que cette maintenance sera faite, il n'y aura presque rien à faire au niveau du logiciel. Il y aura les notes informatives des derniers officiers à l'avoir faite, les pièces à prévoir et les instructions. Il suffira alors de remplir un permis de travail si nécessaire et faire le job.

Il était très important de pouvoir faire cela car le but du logiciel est de simplifier au maximum le travail des ingénieurs. Pour l'instant, à bord, chaque fois qu'une maintenance doit être faite, l'officier doit aller chercher dans les manuels les instructions, puis les pièces et le prochain officier qui fera la même maintenance devra tout rechercher également. Maintenant, il suffira de le faire qu'une seule fois (donc pas plus que d'habitude) jusqu'à la fin de vie du navire.

Il a fallu faire attention aux doublons. Les doublons sont les mêmes valeurs que l'on retrouve dans plusieurs tables. Chaque valeur ne doit être présente qu'une seule fois pour éviter les *bugs*.

8.4.2 L’aspect visuel

8.4.2.1 Listes des maintenances

Comme la partie maintenance est plus grande que la partie *logbook*, il a fallu diviser et organiser pour ne pas avoir trop d’informations sur la page en même temps.

Il y a donc deux pages d’accès aux maintenances, une pour celles prévues, et une pour celles déjà archivées. La page est donc composée d’un tableau récapitulatif (figure 21).

Name	Maintenance N°	Type	Machinery	Created	Limit	Running Hours
injector replacement bis	45674	Scheduled Maintenance	Auxiliary generators	2020-04-21	0000-00-00	456333
valves replacement	456723	Breakdown repair	Auxiliary generators	2020-04-21	0000-00-00	0
purge	7767	Preventive Repair	boiler	2020-04-21	0000-00-00	0
piston check 3	5668556	Scheduled Maintenance	Auxiliary generators	2020-04-16	2020-04-29	54000
Lube oil	635329333	Scheduled Maintenance	Auxiliary generators	2020-04-16	2020-04-30	37000
injector check	1234	Scheduled Maintenance	Auxiliary generators	2020-04-16	2020-04-30	51000

Figure 21 : Tableau, liste des maintenances

Source : capture d’écran

La liste est triée par date limite. Chaque ligne est cliquable pour accéder à la page de la maintenance en question.

8.4.2.2 Onglet « Summary »

L’onglet ouvert par défaut est le sommaire de la maintenance (figure 22). C’est un aperçu des différentes parties. La barre de progression représente les heures de fonctionnement du moteur, les données sont prises directement dans les données du *logbook*.

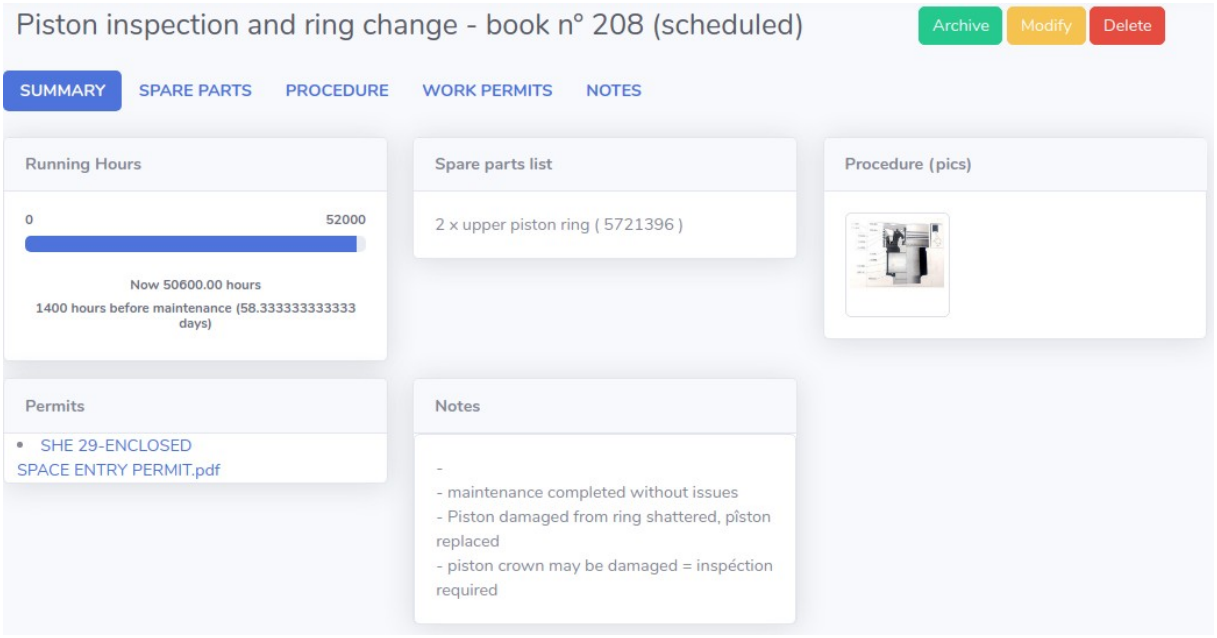


Figure 22 : Onglet sommaire d'une page d'une maintenance

Source : capture d'écran

8.4.2.3 Onglet « Spare parts »

Dans l'onglet des pièces de rechange (figure 23), on peut sélectionner parmi les pièces disponibles pour cette machine. Le stock est affiché directement dans la liste.

Il suffit, pour chaque pièce, de sélectionner la pièce et le nombre d'exemplaires souhaité.

Une fois la maintenance archivée, les pièces sélectionnées seront soustraites au stock actuel.

De cette manière, les provisions du stock seront toujours à jour.

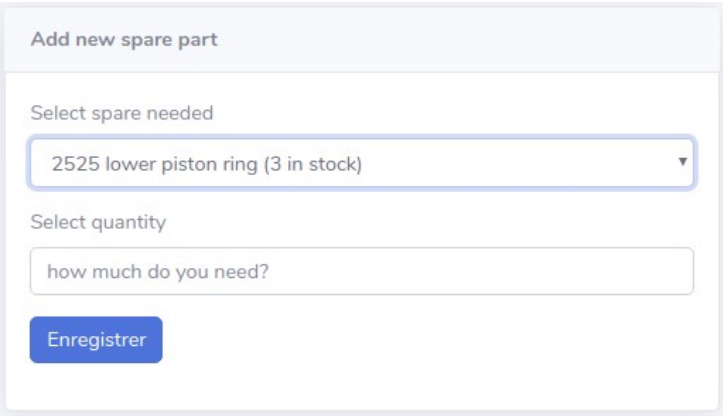


Figure 23 : Sélection des pièces de rechanges

Source : Capture d'écran

8.4.2.4 Onglet « Procedure »

Dans cet onglet, on peut retrouver des captures d'écran, des photos ou des *scans* des manuels contenant les instructions (figure 24). Les fichiers supportés sont multiples (jpeg, png, pdf...).

Il suffit de faire glisser un fichier dans le cadre ou cliquer directement dessus pour l'importer. Le code de cette partie est en JavaScript pour ne pas à avoir à recharger la page [7]. Tout se fait instantanément.

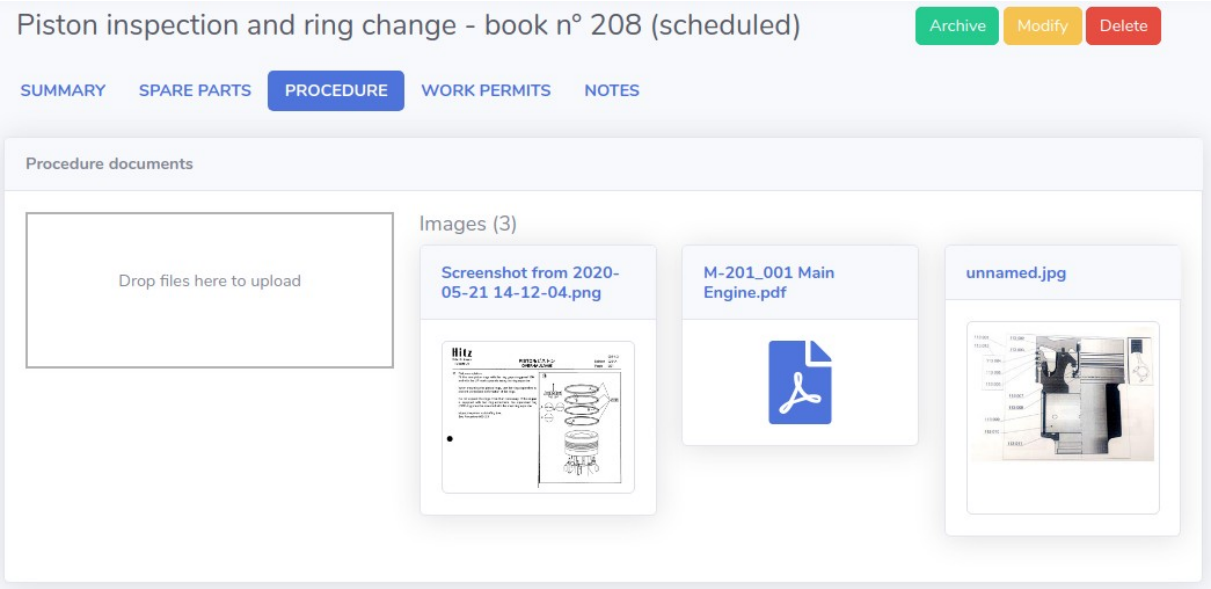


Figure 24 : Onglet des instructions de la maintenance

Source : Capture d'écran

Ces fichiers sont donc liés à la maintenance de telle manière qu'ils seront présents la prochaine fois qu'elle devra être faite.

8.4.2.5 Onglet « Work permits »

Dans cet onglet, le système est globalement le même que dans celui des instructions. La différence principale se situe dans le lien entre les données et la maintenance. Comme un permis est valable seulement pour une journée ou un quart, il faut le refaire à chaque maintenance, voire même plusieurs fois pour la même pour le même job. Les permis de travail sont enregistrés en fonction d'une maintenance en particulier.

Il y a en haut à gauche le bouton pour sélectionner les permis vierges (figure 25). Lors d'un clic, une modale s'ouvre avec les différents permis vierges de la compagnie.

On peut les ouvrir dans un nouvel onglet ou dans le lecteur de fichier de l'ordinateur pour les imprimer.

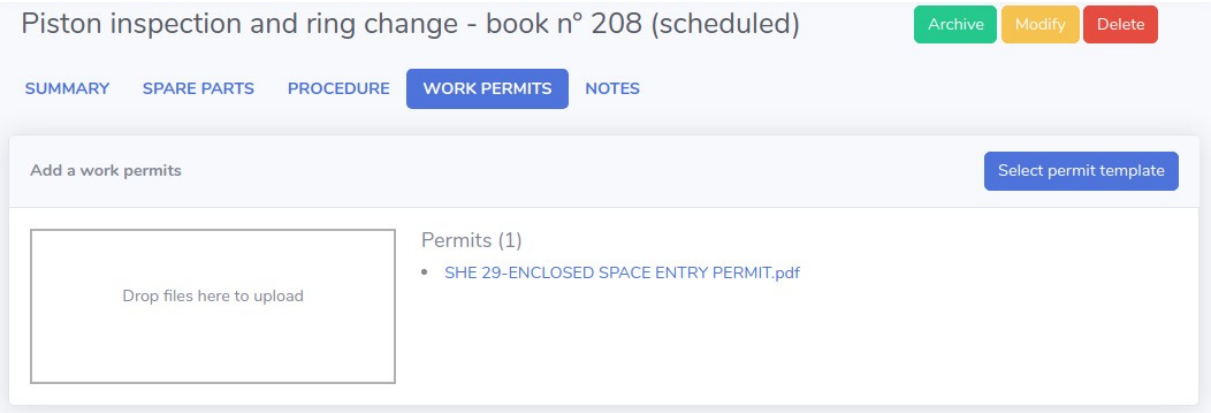


Figure 25 : Onglet des permis de travail

Source : Capture d'écran

Une fois le permis rempli par l'officier responsable, il suffit de le scanner et importer le fichier de la même façon que pour les instructions. Le permis est alors définitivement enregistré.

Ce système améliore grandement la gestion et la sécurité des documents officiels à bord.

8.4.2.6 L'onglet des notes

Le dernier onglet est consacré aux notes et indications des officiers sur la maintenance.

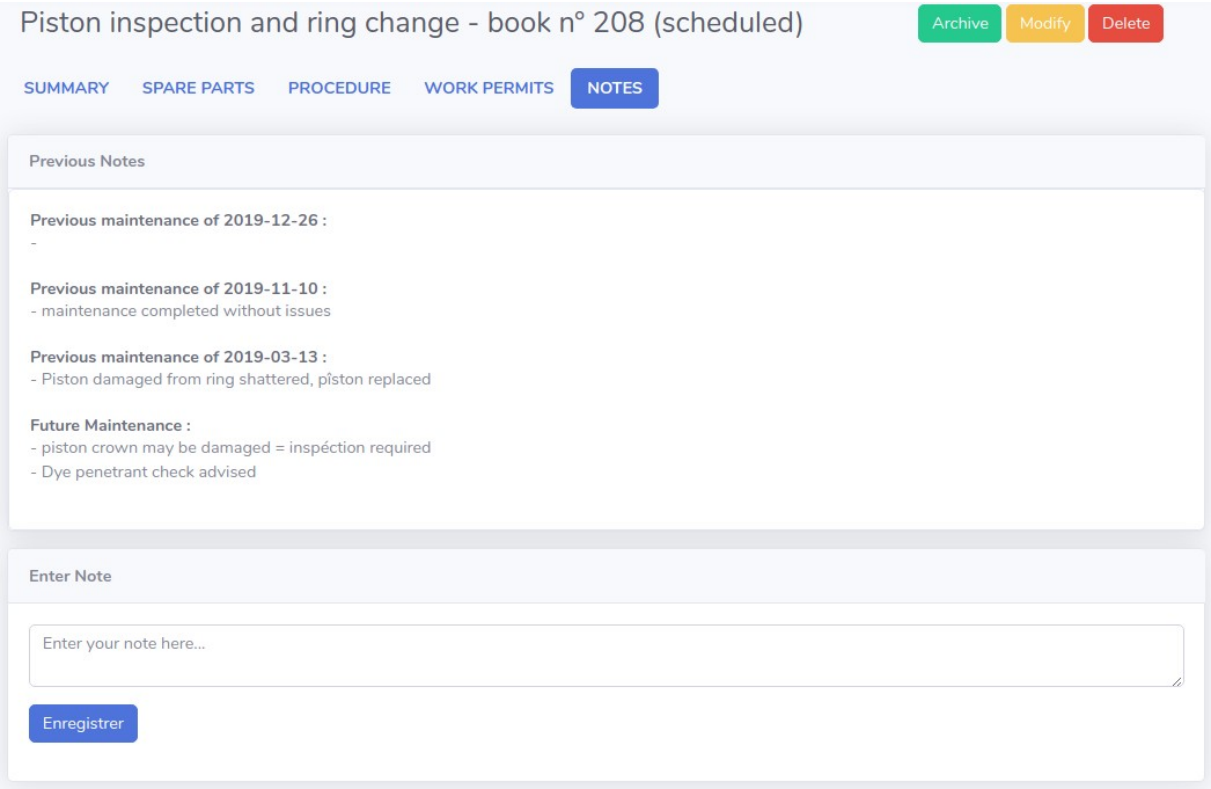


Figure 26 : Onglet des notes de la maintenance

Source : Capture d'écran

Toutes les anciennes notes sont présentes. C'est aussi un grand avantage à bord car cela permet de savoir si un problème avait déjà était remarqué ou non. Cela évite aussi aux officiers de faire face aux mêmes problèmes tout seuls, alors qu'une solution avait déjà été trouvée.

8.4.2.7 Création, modification, suppression et archivage

La création et la modification d'une maintenance se font à travers deux modales similaires.

Dans la modale de modification (figure 27), les champs sont déjà pré-rentrés pour faciliter la tâche.

Bien sur, les actions effectuées sont enregistrées pour garder une traçabilité.

Finalement, l'archivage ainsi que la modification se font à travers des modales de confirmation ouvertes par des boutons (figure 28).

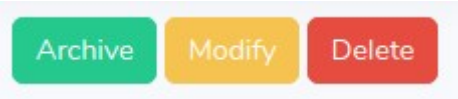


Figure 28 : Boutons archivage, modification et suppression de maintenance

Source : Capture d'écran

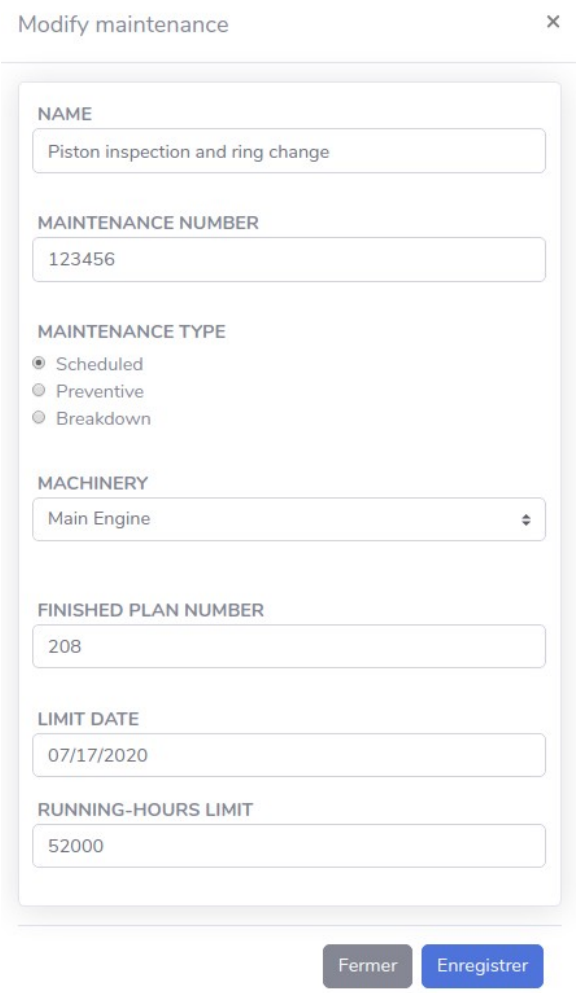


Figure 27 : Modales création/modification maintenance

Source : Capture d'écran

8.4.3 Les pièces de rechange, « *Spare parts* »

La mise en place de cette partie a été similaire à celle du *logbook*. Les différences principales sont la liaison avec la partie maintenance et la gestion des stocks.

La plus grande difficulté n'a pas été au niveau du code mais au niveau des manuels fournis par les constructeurs.

En effet, tout le logiciel est conçu pour pouvoir être installé sur des navires déjà en fonctionnement, aussi bien que sur des nouveaux. Il est donc entièrement basé sur des manuels techniques récupérés à bord (dont ceux d'un VLCC battant pavillon Belge).

Ces manuels, que l'on retrouve dans le poste de contrôle des machines, sont divisés en sous parties correspondant à chaque machine. Il y a par exemple 7 livres de HITACHI-MAN B&W pour le moteur principal, 3 livres de YANMAR pour les générateurs, 5 livres de MITSUBISHI pour les chaudières et ainsi de suite. Or, chaque constructeur n'organise pas de la même manière ses documents. Les références sont différentes ; les informations aussi.

Le système a donc été créé afin de permettre aux officiers de se retrouver dans cette montagne incohérente d'informations.

8.4.3.1 Les tableaux

Les pièces sont donc regroupées dans des tableaux en fonction de la machine concernée. Ainsi, si deux pièces différentes, appartement à deux machines distinctes, ont le même numéro, il n'y aura pas de problème. Au niveau de la BDD, les pièces sont liées à la table des machines au même titre que les maintenances. Cela permet lors d'une maintenance de n'avoir le choix que parmi les pièces de la machine en question.

Le tableau (figure 29) est trié par numéro croissant.

Les spécificités des pièces sont :

- un nom,
- un numéro,

- le livre dans lequel on peut la retrouver,
- le nombre en stock,
- le nombre en stock recommandé par les manuels,
- le nombre actuellement en utilisation,
- le(s) matériau(x) dont elle est composée,
- le poids et son unité,
- les remarques faites par les autres officiers,
- des informations sur la création et modification,
- un bouton suppression et un bouton modification.

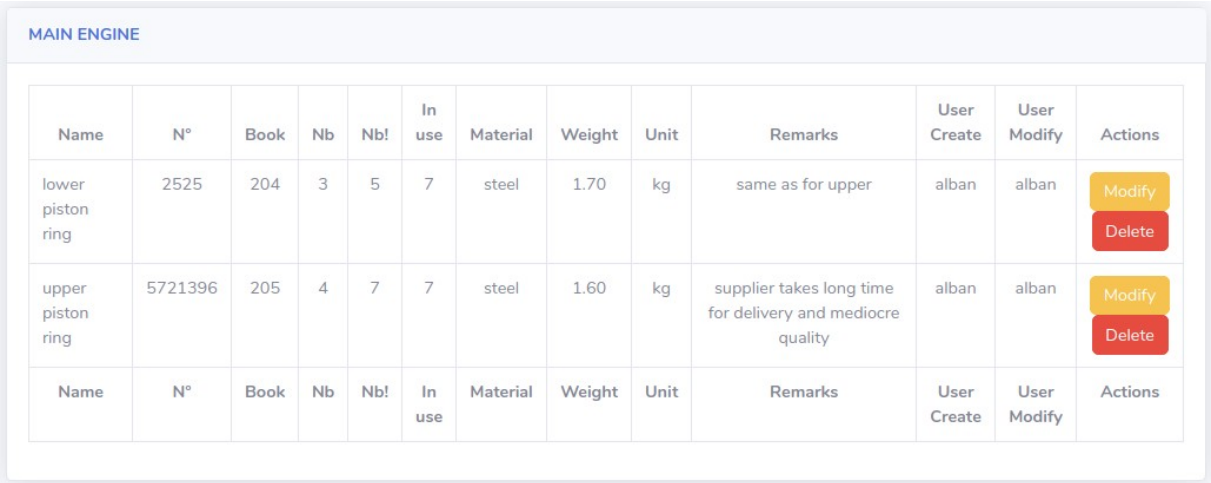


Figure 29 : Tableau des pièces de rechange

Source : Capture d'écran

La création et modification d'une pièce se fait à travers des modales similaires à celles que l'on retrouve dans la partie maintenance.

8.4.4 La gestion des fichiers

La gestion des fichiers est présente dans les paramètres qui ont été créés pour le Chef (comme pour les paramètres du *logbook*). Ils sont accessibles par lui seul ou par les personnes ayant les mêmes droits d'accès à bord.

8.4.4.1 Les fichiers des procédures

Les fichiers des procédures sont enregistrés sur le serveur en fonction du numéro de maintenance. Le choix a été fait ici de les afficher sous forme de tableaux dans des cartes. Chaque carte regroupe alors les fichiers d'une machine en particulier pour garder la même simplicité dans tout le logiciel. Avec une base de donnée plus conséquente, le choix judicieux serait de trier les documents en fonction du numéro de maintenance.

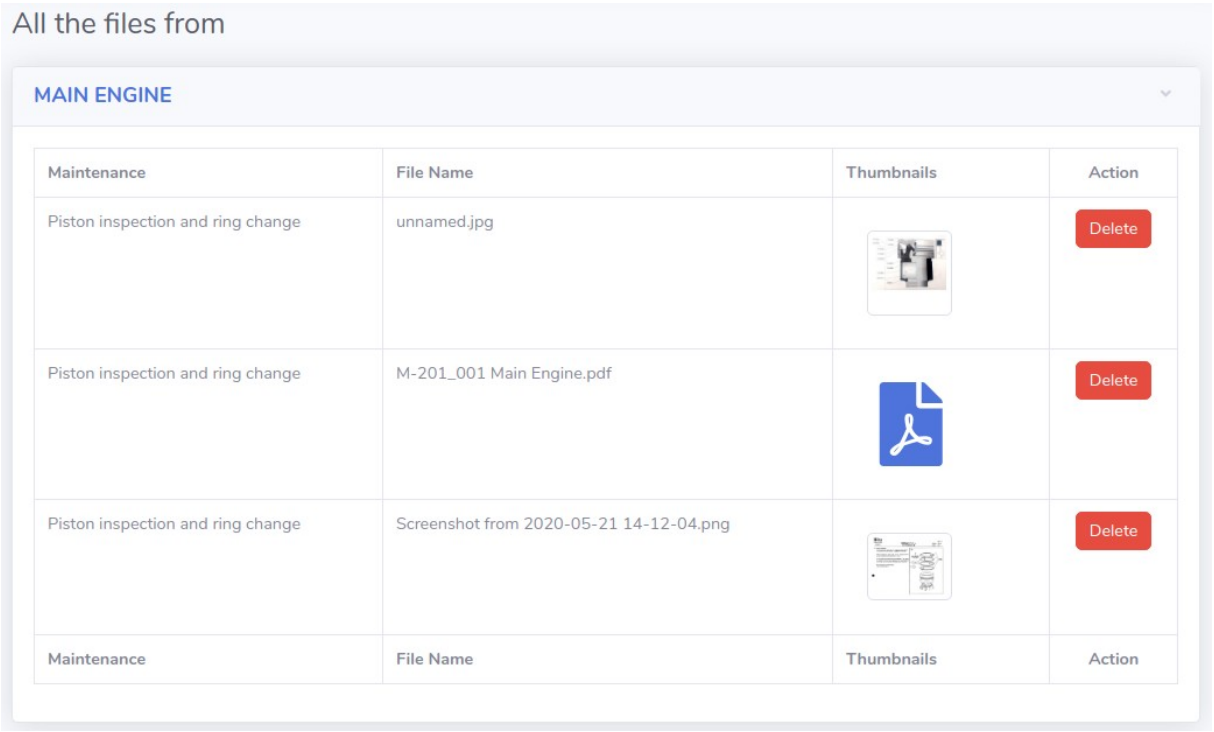


Figure 30 : Page regroupant les fichiers des procédures

Source : Capture d'écran

Dans ce tableau, on retrouve donc le nom de la maintenance, le nom (et l'extension) du fichier, une miniature cliquable et un bouton pour le supprimer. Le bouton « *Delete* » ouvre une modale de confirmation.

8.4.4.2 Les permis de travail signés

Les permis de travail remplis et signés sont eux enregistrées pour chaque maintenance.

L'organisation de la page est similaire à celle des procédures (figure 31). Une colonne qui affiche la date de création de la maintenance a été ajoutée pour éviter les confusions et les erreurs. La date a aussi été ajoutée dans la modale de confirmation de suppression.

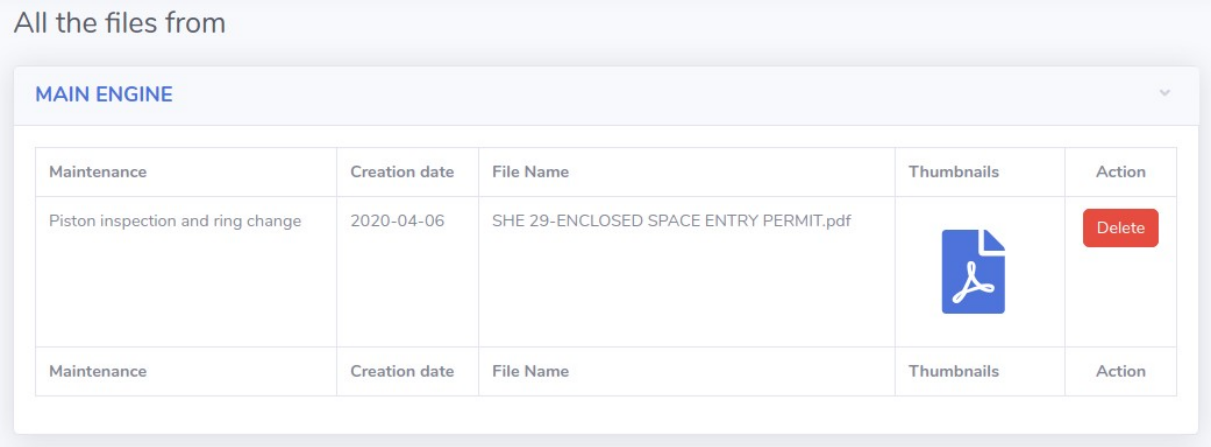


Figure 31 : Page regroupant les permis de travail signés

Source : Capture d'écran

8.4.4.3 Les permis de travail vierges

Les permis de travail vierges sont sur une page des paramètres assez simple. On peut voir les permis présents et les supprimer. On peut aussi en importer des nouveaux.

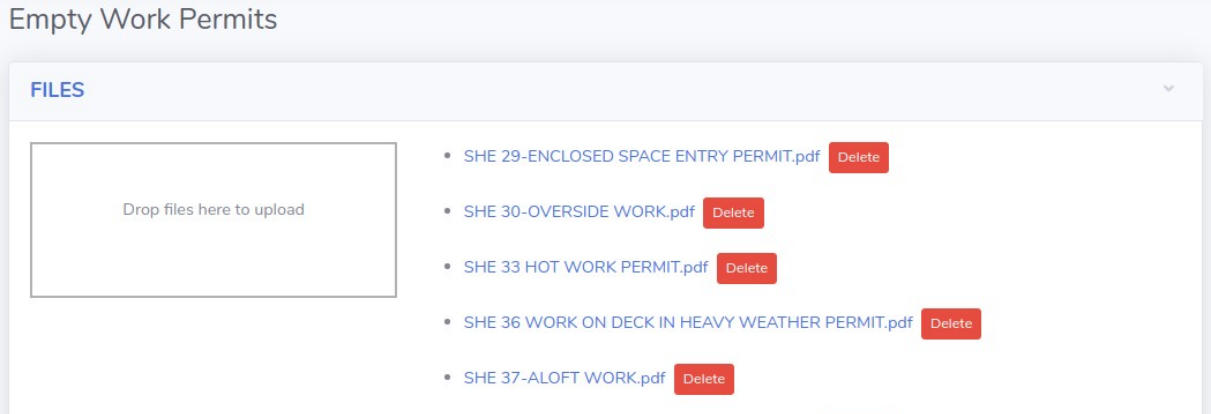


Figure 32: Page de gestion de permis de travail vierges

Source : Capture d'écran

8.5 La page index

La page index est un résumé des différentes informations utiles que l'ont retrouve dans le logiciel. Elle a été créée pour rester ouverte tout le temps sur l'écran du Chef, du Capitaine ou de l'ordinateur dans la salle des machines. D'ailleurs elle se rafraîchit automatiquement toutes les minutes (délai choisi arbitrairement et facilement modifiable).

L'organisation, ainsi que toutes les cartes qui composent cette page, ont été faites à titre d'exemple. Les possibilités pour afficher et organiser ces données sont trop grandes pour tout montrer.

8.5.1 Niveaux

La carte des niveaux (figure 33) est basée sur une applet du modèle *bootstrap* de base. Il a été paramétré avec php pour que les barres changent de couleur en fonction du niveau.

Comme les *logs* ont une valeur maximum et un minimum, les limites sont faites en fonction d'un pourcentage (X) :

- si X < 10 % ou X > 90 % la couleur sera rouge
- si X < 25% ou X > 75% la couleur sera orange
- sinon elle sera grise

À l'intérieur des barres, on retrouve le niveau exact et au dessus le pourcentage. Le pourcentage est en fonction du niveau. En réalité, les tanks à bord d'un navire ne sont pas forcément carrés, et le volume ne correspond pas linéairement au niveau mesuré. Une correction est faite par les officiers à bord, ce qui pourrait facilement être mis en place ici.
On pourrait même avoir l'information du volume réel pour mieux anticiper les variations.

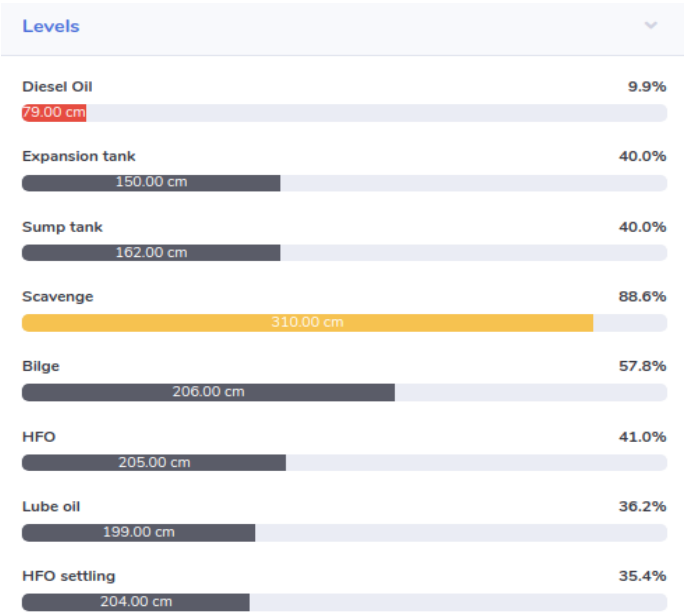


Figure 33 : Carte des niveaux de la page index

Source : Capture d'écran

8.5.2 Filtres

Pour les filtres, il est important d'avoir un suivi du nombre de nettoyages à contre courant par jour. Il faut pouvoir apercevoir l'évolution pour anticiper un nettoyage complet des filtres.

Les outils du modèle *bootstrap* n'étant pas suffisants, il a fallu se tourner vers d'autres solutions. Parmi celles-ci, des graphiques proposés librement par des développeurs ont paru tout à fait adaptés [2,3,5]. Ces graphiques sont codés en JavaScript.

Lors du passage de la souris sur chaque barre, on peut voir s'afficher des informations comme le nombre exact de « *backflushs* ».

Ce type d'outil interactif est dit « *responsive* ».

Les données qui s'affichent sont récupérées dans les derniers logs.

Figure 34 : Carte des compteurs de filtres de la page index

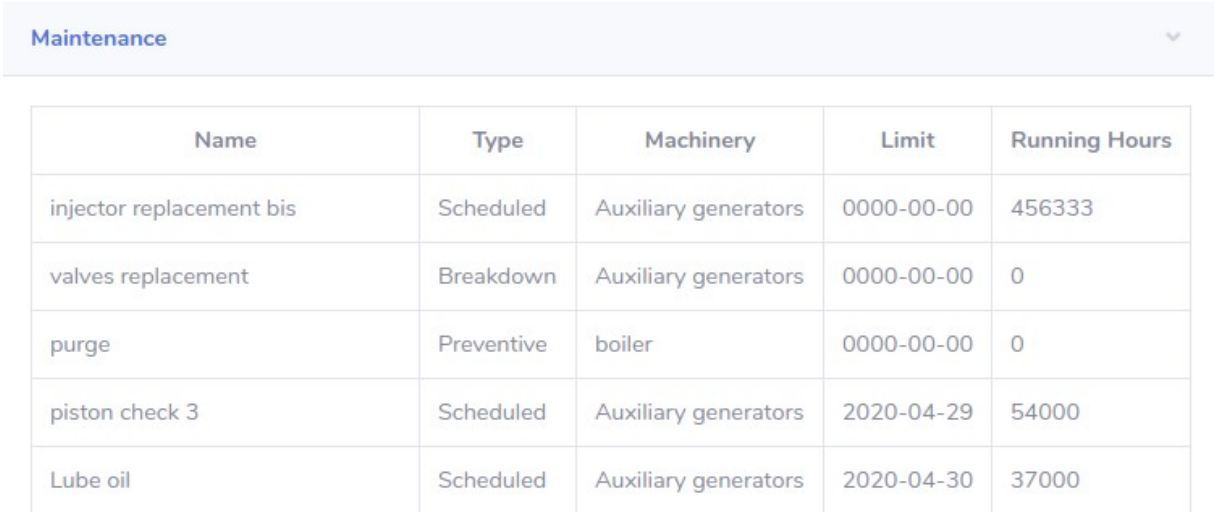
Source : Capture d'écran

A bord, cette mesure est remise à zéro au niveau du compteur chaque fois qu'elle est notée. Le choix a été fait d'afficher chaque mesure rentrée, même s'il y en a eu plusieurs le même jour. Ce système laisse peut être plus de possibilités d'analyse aux officiers que d'avoir une valeur quotidienne.

8.5.3 Maintenances

Une carte avec un tableau résumé des maintenances est présente (figure 35). Elle est repliée d'origine. Il faut cliquer dessus pour que le tableau s'affiche. Ce n'était pas spécialement nécessaire, mais c'est pour montrer qu'on peut le faire.

C’est une liste des maintenances à venir, mais avec moins d’informations que dans celui de la page « Maintenance ». Par contre, les lignes restent cliquables pour avoir un accès rapide à la maintenance voulue.



Name	Type	Machinery	Limit	Running Hours
injector replacement bis	Scheduled	Auxiliary generators	0000-00-00	456333
valves replacement	Breakdown	Auxiliary generators	0000-00-00	0
purge	Preventive	boiler	0000-00-00	0
piston check 3	Scheduled	Auxiliary generators	2020-04-29	54000
Lube oil	Scheduled	Auxiliary generators	2020-04-30	37000

Figure 35 : Carte des maintenances de la page index

Source : Capture d’écran

8.5.4 Suivi de l’eau douce

Les *tanks* d’eau douce fluctuent énormément de jours en jours. Ce sont des valeurs très importantes à bord car il faut faire des prévisions en fonction de la consommation, la production, la zone géographique et les opérations à venir.

Ce graphique a été choisi car il représente bien les fluctuations des niveaux d’eau dans les *tanks*. La possibilité de pouvoir colorer la partie inférieure de la courbe est aussi visuellement satisfaisante.

La source du script est la même que pour les filtres. En revanche, le paramétrage est différent. Ici les données affichées sont prises sur le dernier *log* de chaque journée. Ainsi, l’échelle est linéaire.

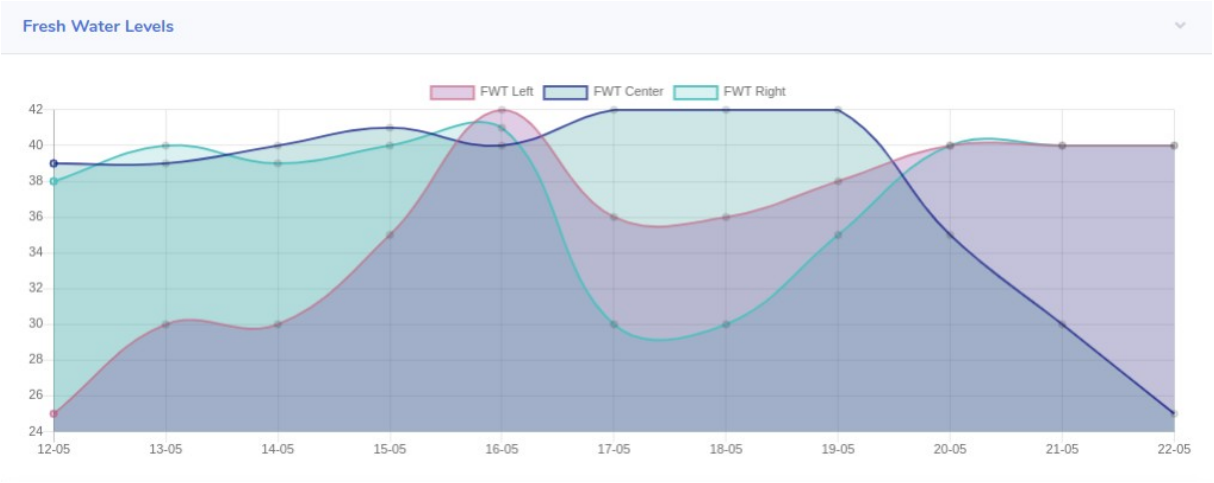


Figure 36 : Carte du suivi de l'eau douce de la page index

Source : Capture d'écran

Encore une fois, ce graphique est réactif au passage de la souris. On peut donc passer le curseur sur les petits points qui donneront des informations sur la valeur utilisée.

8.5.5 Températures et pressions

Les cartes des températures et pressions sont similaires. Elles affichent pour chaque composant, la dernière mesure rentrée dans le logiciel. Ce sont des cartes simples qui viennent directement du modèle *bootstrap*.

Les pictogrammes de températures et de pressions sont paramétrés comme les barres de niveaux. Quand les valeurs se rapprochent du minimum ou du maximum, la couleur change.

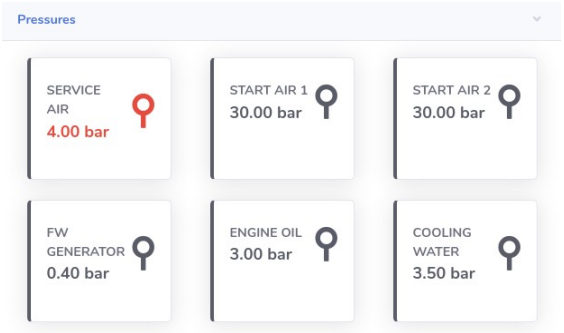


Figure 38 : Carte des pressions de la page index

Source : Capture d'écran

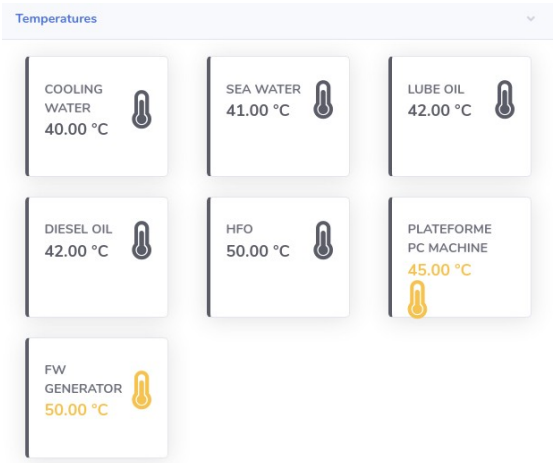


Figure 37 : Carte des températures de la page index

Source : Capture d'écran

8.6 Les parties additionnelles

Ces parties ont été développées en parallèle aux autres parties. Elles sont présentes dans chaque page du code. On y retrouve des informations utiles au niveau développeur comme utilisateur, ainsi que des actions pouvant être effectuées à tout moment.

On y retrouve les segments du menu, l'entête et le pied de page ainsi que les code « *main.php* » et « *header.php* ».

8.6.1 Le menu de navigation

Le menu est la colonne bleue que l'on retrouve sur la gauche de chaque page (figure 39). Il est composé de boutons d'accès aux pages dans un menu coulissant.

Le premier segment « *MACHINERY* », avec le symbole d'un capteur, renvoie à la page d'accueil.

Le deuxième segment est composé des pages utiles à l'utilisation quotidienne du logiciel. Quand l'utilisateur n'est pas connecté, ou n'a pas les droits suffisants pour accéder à certaines pages, seules les pages d'informations sont disponibles.

La troisième partie comporte l'accès aux paramétrages des données. Elle est seulement accessible aux utilisateurs disposant de tous les droits comme le Chef.

La dernière partie est un bouton pour réduire la taille du menu et augmenter la surface du contenu de la page. Quand la taille en largeur diminue (comme sur les smartphones) ce bouton est remplacé par un autre dans l'entête.

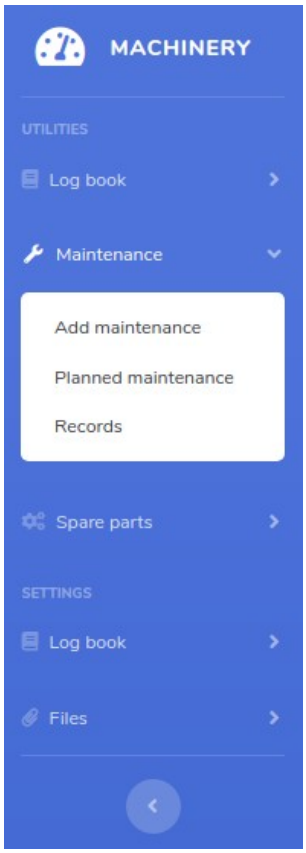


Figure 39 : Menu de navigation (« menuleft »)

Source : Capture d'écran

8.6.2 L’entête

L’entête (figure 40) affiche en temps réel la date et l’heure du serveur.

Sur la droite, un bouton permet d’accéder à la fenêtre de connexion.

Figure 40 : Entête (« topbar »)

Source : Capture d’écran

Cette fenêtre (figure 42) est dite de type « *dropdown* ». Elle apparaît sous le bouton et disparaît si l’on clique hors de sa zone (comme pour les modales).

Quand un utilisateur est connecté sur le logiciel, cela reste affiché (figure 41) pour éviter les malentendus.

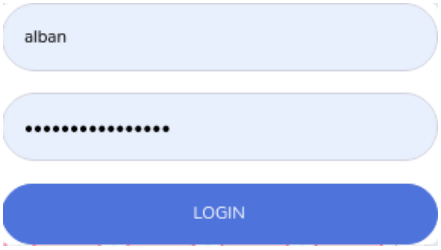


Figure 42 : Fenêtre de connexion de l'entête

Source : Capture d’écran

Figure 41 : Notification de la connexion

Source : Capture d’écran

8.6.3 Le pied de page

Contenu dans le fichier «*footer.php*», le pied de page affiche le copyright de l'application.

A l'instar du « *header.php* », il intègre également des parties invisibles, mais essentielles au fichier HTML généré, comme :

- le bouton de « *scroll* » en bas à droite des longues pages permettant de remonter instantanément en haut
- la « div » permettant d'appeler les modales
- les appels à tous les Javascripts nécessaires au fonctionnement du logiciel. Comme ils sont utilisés par le navigateur, il n'est pas nécessaire de les appeler au début de la page HTML dans le *header*. Ils sont donc à la fin, afin de ne pas ralentir l'affichage de la page.
- Enfin, un script qui rafraîchit toute les 2/10ème de secondes la date et l'heure dans l'entête, avant de fermer le fichier HTML.

Figure 43 : Pied de page (« footer »)

Source : Capture d'écran

8.6.4 Le « *header* »

Le fichier «*header.php*» initialise le fichier HTML généré en :

- précisant la langue par défaut, ici l'anglais
- définissant les *meta-tags* et le titre de la page,
- appelant les feuilles de styles en cascade (CSS).

8.6.5 Le « *main* »

Le fichier « *main.php* » est appelé sur toutes les pages, et certains script, comme les fichiers vus précédemment. Par contre, celui ci n'est appelé qu'une seule fois avec la ligne :

include_once('main.php');

Il se compose de plusieurs parties :

- en premier, l'ouverture d'une connexion permanente à la base de données MySQL, via l'initialisation du connecteur : « mysqli_connect ». Cette ouverture identifiée par la variable « $db0 » doit être unique. C'est l'une des raisons pour lesquelles le script « main.php » ne peut pas être appelé plusieurs fois
- ensuite, la récupération de variables du serveur dont l'adresse IP de l'utilisateur enregistrée dans la variable « $remoteip »
- Ensuite, la vérification du cookie :
  - s'il n'est pas bon ou s'il n'y en a pas, le visiteur n'est pas identifié,
  - s'il est identifié, le script récupère le niveau d'admin, dans la table « *users* », qui définit les droits de l'utilisateur :
    - niveau 0 = aucun droit d'ajout, de modification ou de suppression,
    - niveau 1 = Droits normaux pour les machinistes : ajout,

    - niveau 2 = Droits étendus pour les chefs : ajout, modification, suppression,

- enfin, la définition de fonctions qui peuvent ensuite être appelées par les autres scripts. C’est une autre des raisons pour lesquelles le script « *main.php* » ne peut pas être appelé plusieurs fois, car une fonction ne doit pas être définie plusieurs fois. Ici, on a une fonction *logthat*() qui enregistre une ligne d’informations dans le fichier des *logs* du logiciel.

9 Conclusion

Cette application, conçue en parallèle de l'apprentissage de la programmation, ne peut pas remplacer tout ce qui existe déjà à bord. Elle peut cependant montrer les lacunes de certains autres logiciels et en rendre beaucoup obsolètes.

Le développement est actuellement fini. Les fonctions souhaitées sont toutes opérationnelles ainsi que certains aspects facultatifs ajoutés pour le confort ou simplement par plaisir.

Ce logiciel est une base solide qui peut désormais être utilisée au quotidien. Il fonctionne, et aucun *bug* n'a persisté, ou n'est apparu depuis la fin du codage.

Il est fini, certes, mais c'est une base. Et comme cette base est solide, d'autres outils utiles et importants peuvent s'y greffer.

Peut être, pour l'occasion du mémoire de Master, pourrait-on voir une acquisition des données des capteurs automatisée en temps réel. Cela permettrait évidemment de regrouper et simplifier au maximum l'accès aux informations.

Il serait aussi intéressant de voir un logiciel complémentaire qui serait présent à terre au sein des compagnies. Les navires pourraient alors communiquer les informations par satellite. Les équipes à terre pourraient avoir un meilleur *monitoring* de ce qu'il se passe à bord et anticiper, par exemple, l'envoi de pièces pour les maintenances.

10 Bibliographie

[1] “ADN04 - Aide pour le développement des NTIC en Provence” (n.d.) https://www.adn04.org/ (Accessed 24 May 2020).

[2] “Bar · Chart.js documentation” (n.d.) https://www.chartjs.org/docs/latest/charts/bar.html (Accessed 22 May 2020).

[3] “Bootstrap Charts Guideline - examples & tutorial. Basic & advanced usage - Material Design for Bootstrap” (n.d.) https://mdbootstrap.com/docs/jquery/javascript/charts/ (Accessed 22 May 2020).

[4] Bootstrap, S. (n.d.) “SB Admin 2 - Free Bootstrap Admin Theme.” *Start Bootstrap*. https://startbootstrap.com/themes/sb-admin-2/ (Accessed 24 May 2020).

[5] “Chart.js | Open source HTML5 Charts for your website” (n.d.) https://www.chartjs.org/ (Accessed 22 May 2020).

[6] contributors, phpMyAdmin (n.d.) “phpMyAdmin.” *phpMyAdmin*. https://www.phpmyadmin.net/ (Accessed 24 May 2020).

[7] “Dropzone.js” (n.d.) https://www.dropzonejs.com/ (Accessed 31 May 2020).

[8] “FileZilla - The free FTP solution” (n.d.) https://filezilla-project.org/ (Accessed 24 May 2020).

[9] L, +Bastien (2019) “Base de données : qu’est-ce que c’est ? Définition et présentation.” *LeBigData.fr*. https://www.lebigdata.fr/base-de-donnees (Accessed 24 May 2020).

[10] “Logiciel de création de diagrammes et de supports visuels | Lucidchart” (n.d.) https://www.lucidchart.com/pages/fr/landing?utm_source=google&utm_medium=cpc&utm_campaign=fr_bucket_desktop_branded_x_exact_lucidchart&km_CPC_CampaignId=1535987490&km_CPC_AdGroupID=60295220122&km_CPC_Keyword=lucid%20chart&km_CPC_MatchType=e&km_CPC_ExtensionID=&km_CPC_Network=g&km_CPC_AdPosition=&km_CPC_Creative=309039280255&km_CPC_TargetID=aud-837074142685:kwd-55720648523&km_CPC_Country=1001015&km_CPC_Device=c&km_CPC_placement=&km_CPC_target=&mkwid=spboSGe5p_pcrid_309039280255_pkw_lucid%20chart_pmt_e_pdv_c_slid__pgrid_60295220122_ptaid_aud-837074142685:kwd-55720648523_&gclid=CjwKCAjwqpP2BRBTEiwAfpiD-x9Zl1LyCcSusQDUqHiye9GP1cqZ-kFI7DKbiFJ8RBohzVe3f13yxBoC2fwQAvD_BwE (Accessed 24 May 2020).

[11] “Qu’est-ce qu’une base de données relationnelle” (n.d.) *Oracle France*. https://www.oracle.com/fr/database/base-de-donnees-relationnelle-definition.html (Accessed 24 May 2020).

[12] “VSCodium - Open Source Binaries of VSCode” (n.d.) https://vscodium.com/ (Accessed 24 May 2020).

11 Annexes

Dans le dossier compressé « Annex-201402960.zip » se trouvent tous les fichiers PHP qui composent le logiciel.

Les noms ont été créés arbitrairement pour se repérer lors de la programmation. Il se peut qu'ils ne soient pas très compréhensibles.

Pour information, les fichiers qui finissent par :

- -add.php,
- -delete.php,
- -modify.php,

sont généralement des fichiers qui sont responsables de modifications dans la base de données SQL.