

Licence Professionnelle
Systèmes informatiques et Logiciels
METINET

Du 3 octobre au 15 septembre

RAPPORT D'ALTERNANCE

Hugo Alliaume

Promotion 2016/2017
\$

NinaCMS

SAILING Communication & Technologies
58, rue des Clercs 38200 Vienne

Maître d'Alternance
M. Philippe BOIREAUD

Tuteur d'Alternance
M. Fabrice JAILLET

Université Claude Bernard – Lyon 1



Institut Universitaire de Technologie

Institut Universitaire de Technologie

Département Informatique
Site de Bourg-en-Bresse

Département Informatique – IUT Lyon 1
71 rue Peter Fink – 01000 BOURG-EN-BRESSE
Tél. : 04 74 45 50 59 – Fax : 04 74 45 50 51
Mail : iut.lp.metinet@univ-lyon1.fr
Site web : <http://iut.univ-lyon1.fr>

Remerciements

Je remercie la société Sailing Communication & Technologies, pour m'avoir accepté en alternance et considéré tel un véritable alternant (notamment pour le respect des périodes de travail, rien ne m'était demandé en dehors des heures de travail), où j'ai pu me rendre utile pendant la période de développement du projet NinaCMS.

Je remercie M. Philippe BOIREAUD, mon maître d'apprentissage, qui a su me guider et me ré-orienter pendant la période de développement du projet NinaCMS, avec lequel j'ai pu beaucoup apprendre sur le monde professionnel.

Je remercie Mme. Céline RUELLAN, Directrice des Ressources Humaines, pour avoir répondu à diverses questions par mail ainsi que pour m'avoir envoyé mes bulletins de paie au format PDF pendant toute cette année.

Je remercie M. Fabrice JAILLET, mon tuteur d'alternance, pour m'avoir suivi et encadré durant la période de l'alternance, ainsi que pour ses visites en entreprise pour voir si tout se passait bien et observer l'avancement du projet.

Enfin, je remercie les enseignants et surtout les intervenants (plus particulièrement M. Boris GUÉRY) qui nous ont accompagnés pendant cette année d'alternance à l'IUT de Bourg-en-Bresse, afin de nous préparer pour le monde professionnel de l'entreprise notamment grâce à leurs multiples années d'expérience.

Table des matières

Remerciements.....	3
1. Présentation de l'entreprise.....	7
1.1. Activités principales.....	7
1.2. Présentation de l'effectif.....	7
1.3. Partenaires.....	8
2. Présentation de l'environnement de travail.....	9
2.1. Environnement humain.....	9
2.2. Matériels et logiciels utilisés.....	9
3. Travail effectué.....	11
3.1. Présentation d'un composant.....	11
3.2. Présentation des Field.....	13
3.3. Présentation des Fragment.....	13
3.4. Partie publique.....	13
3.5. Partie administration.....	13
3.1. Présentation de l'architecture du projet.....	14
4. Conclusion.....	17
4.1. Expérience acquise.....	17
4.2. Avenir du projet.....	17
5. Glossaire.....	19
6. Bibliographie.....	22

1. Présentation de l'entreprise

SAILING est une agence de communication Digitale & Print. Actuellement composée de 6 collaborateurs, elle a pu dégager un chiffre d'affaire de 450.000€ pendant l'année 2016.

Le siège de la société se trouve à Saint-Maur-des-Fossées (94) dans la région parisienne, et le siège secondaire se situe à Vienne (38) au sud de Lyon. C'est dans ce dernier que j'ai pu effectuer ma formation en alternance pendant toute cette année.

L'agence a vu le jour en 1996 et s'est d'abord concentré autour des métiers de l'édition et de la conception graphique de supports imprimés. Plus tard, la démocratisation du Web et l'émergence de nouveaux supports de communication ont naturellement conduit l'équipe à se positionner sur de nouveaux métiers.

Après 10 années d'existence, la structure est devenue « **SAILING Communication & Technologies** », et s'est définitivement organisée en trois pôles :

- Conseil en communication & Marketing,
- Création & Édition,
- Développement web & Intégration technique / R&D.

1.1. Activités principales

Les activités principales de SAILING sont les suivantes :

- C'est un studio de création graphique (création de maquette de site web, de logo, ...),
- C'est un prestataire internet (fournit une connexion Internet à des entreprises et/ou particuliers),
- Elle s'occupe du développement et de l'intégration de site web,
- Exerce de la R&D, et auteur du moteur d'applications web WysiUp.

1.2. Présentation de l'effectif

Voici une liste non exhaustive des personnes travaillant actuellement chez SAILING :

- Philippe Boireaud : À la tête du pôle de compétence de la **direction technique et R&D**,
- Yann Ruellan : À la tête du pôle de compétence des **conseils en communication et le pilotage de projet**,
- Éric Ruellan : À la tête du pôle de compétence de la **direction artistique**,

- Céline Ruellan : Directrice des Ressources Humaines,
- Alexis Lachaux : Commercial.

1.3. Partenaires

SAILING est partenaire de FEEL EUROPE Groupe, une SSII conseil en systèmes d'information, en mesure de soutenir des projets nécessitant de faire appel à des compétences techniques additionnelles.

2. Présentation de l'environnement de travail





2.1. Environnement humain

Mon année d'alternance s'est déroulée dans l'agence de Vienne. Uniquement composée de développeurs, j'ai donc travaillé dans le service de développement et réalisation de projets webs avec Philippe Boireaud et Thibault Laperrière, lui aussi alternant (Philippe étant situé dans son bureau, et Thibault et moi dans l'open-space).







Pendant cette année d'alternance, j'ai eu pour tâche de travailler uniquement sur le projet NinaCMS, le tout attentivement surveillé et orchestré par Philippe. J'ai aussi eu l'occasion d'aider à plusieurs reprises lorsqu'il avait besoin d'aide pendant son travail.

2.2. Matériels et logiciels utilisés


L'ordinateur de travail était directement fourni par l'entreprise, le système d'exploitation Ubuntu étant déjà pré-installé, j'ai juste eu à installer mon environnement de développement.

	Système d'exploitation : Ubuntu 16.04
	Environnement de Développement Intégré (EDI) : PhpStorm (2017.2)
	Navigateurs Web : Google Chrome (XX), Mozilla Firefox (XX), et Opera (XX)
	Traitement de texte : LibreOffice Writer (5.2)

J'ai utilisé les langages de programmation, outils de développement, et les frameworks suivants :

	PHP (5.6)
	Des composants de Laravel (routage, DI, ORM, validation, ...)
	Node.js (6.11)
	VueJS (2.4)
	Webpack (3.5)
	HTML , JavaScript (ES6) et SCSS .

J'ai aussi utilisé divers outils externes pour le bon déroulement du projet :

	Git , GitKraken et Gogs
	JSDoc et un plugin « jsdoc-vuejs » créé à l'occasion pour documenter des composants VueJS
	Sami

3. Travail effectué

J'ai donc travaillé en autonomie sur NinaCMS tout le long de l'alternance, en étant encadré par mon Maître d'Alternance, Philippe Boireaud. Je me suis basé sur son prototype afin de continuer le projet.

Le projet se base sur le principe de composants. Dans NinaCMS, **tout** est un composant. Cela peut être un composant « **Article** », un composant « **Conteneur de colonnes** », ou alors un média « **Image** », « **Video** », etc. Un composant peut être inclus dans un autre, on arrive au final à un arbre de composants.

NinaCMS étant un éditeur visuel de site web, nous avons donc une **partie publique** qui sera affichée aux visiteurs du site, et nous avons une partie administration – basée sur la partie publique - qui sera réservée à l'administrateur.

La partie administration s'accroche à la partie publique en y ajoutant plusieurs menus pour modifier, déplacer, copier, coller des composants, ainsi qu'un explorateur de composants, un explorateur de fichiers, et divers outils.

3.1. Présentation d'un composant

Un composant doit être au minimum composé d'une **classe PHP** et d'un **template (modèle) Twig**. Il peut aussi être composé de styles CSS, de scripts JS, et d'images.

Il est situé dans le dossier « **app/components** », dans un sous-dossier portant le nom du composant (ex : le composant « **Article** » se situe dans le dossier « **app/components/Article** », et le nom de sa classe PHP est « **ArticleComponent** », déclarée dans le fichier « **ArticleComponent.php** »).













La **classe PHP** du composant hérite d'une classe nommée « **BaseComponent** », cette dernière permet de rajouter plusieurs variables de configuration ainsi que plusieurs fonctions pour gérer ce composant.

Un composant peut ou non être composé de champs, appelés **Field**. Je présenterai les **Field** plus tard, mais concrètement, un **Field** a un certain type (texte, image, adresse URL, etc ...), et il est utilisé dans la partie administration pour pouvoir customiser certaines valeurs du composant. Par exemple, un composant « **Article** » pourra avoir des **Field** qui permettront de modifier le **titre**, le **chapeau**, le **contenu**, l'**image d'illustration**, etc.

En revanche, le **Field** est incapable de faire le rendu de ses propres valeurs. En effet, si dans un composant nous avons deux **Field** de type **texte**, il se peut que nous voulions afficher la valeur du 1^{er} **Field** au format **HTML**, et la valeur du 2^{ème} **Field** au format **texte**.

Cela reste un exemple simple et nous pourrions effectivement tricher pour arriver au résultat souhaité, mais ce n'est pas voulu. Pour avoir le rendu voulu sans tricher, nous utilisons des **Fragment** (j'en parlerai également plus tard). Ils sont utilisés pour le rendu des valeurs des **Field**, et ce sans s'occuper du type de **Field**. On aura un **Fragment** pour afficher du **contenu HTML**, un autre pour afficher du **contenu textuel**.

3.1.1. Représentation d'un composant en base de données


nina_objects	
 id	int(11)
 root_id	int(11) unsigned
 parent_id	int(11) unsigned
 order	int(11) unsigned
 category	varchar(40)
 class	varchar(80)
 label	varchar(150)
 fields_values	text
 styles	text
 user_id	varchar(45)
 group_id	varchar(45)
 rights	varchar(3)

























Représentation d'un composant en base de données

Pour en revenir au composant, voici comment il est stocké en base de données :

- **id** : l'identifiant du composant, typiquement un nombre unique ≥ 1 ,
- **root_id** : l'identifiant racine utilisé pour l'arbre de composants, correspond à un composant ayant pour **id** ce **root_id**,
- **parent_id** : l'identifiant du composant parent (cela implique que ce composant soit son enfant), ayant pour **id** ce **parent_id**,
- **order** : la position du composant avec selon ses frères, un nombre ≥ 0
- **category** : la catégorie, peut être un composant, un fichier, ...
- **class** : sa classe, si la catégorie est un composant, alors sa valeur doit être l'un des composants qui se situent dans le dossier « **app/components** »,
- **label** : le nom à afficher dans l'explorateur de composants ou dans l'explorateur de fichier, selon la catégorie,
- **fields_values** : toutes les valeurs des **Field** liés au composants,
- **styles** : des styles CSS s'appliquant au composant,
- **user_id**, **group_id**, et **rights** : permettent la gestion des permissions, lire [Permissions UNIX](#).

3.1.2. Représentation d'un composant dans le code PHP

▼  BaseComponent

-   fields:Nina\Field\BaseField[]
-   fieldsRules:array
-   fieldsFileRules:array
-   contains:array
-   icon:string
-   name:string
-   template:null|string
-   styles:array|null
-   scripts:array|null
-   adminStyles:array|null
-   adminScripts:array|null
-   group:string

3.2. Présentation des Field

3.3. Présentation des Fragment

3.4. Partie publique

3.5. Partie administration

3.5.1. Présentation du manager de composant (ObjectManager)































3.5.2. Présentation du manager des composants colonnes (BlocManager)

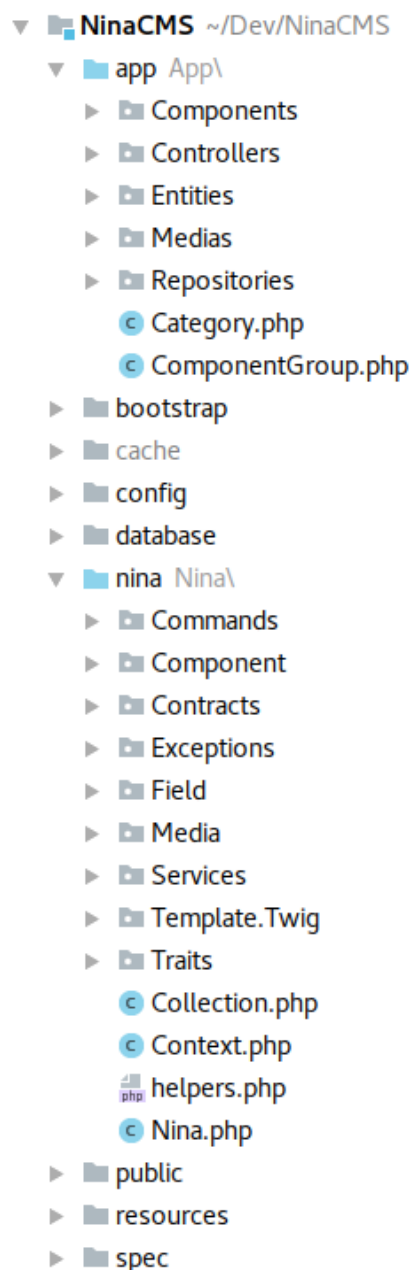
3.5.3. Présentation de l'explorateur de composants

3.5.4. Présentation de l'explorateur de fichiers

3.5.5. Présentation des divers outils

3.1. Présentation de l'architecture du projet

- ▼  **NinaCMS** ~/Dev/NinaCMS L'architecture de NinaCMS reprend celle de certains frameworks PHP avec lesquels j'ai pu travailler, notamment Laravel et Symfony :
- ▼  **app** App\
 - ▶  **Components**
 - ▶  **Controllers**
 - ▶  **Entities**
 - ▶  **Medias**
 - ▶  **Repositories**
 -  **Category.php**
 -  **ComponentGroup.php**
 - ▶  **bootstrap**
 - ▶  **cache**
 - ▶  **config**
 - ▶  **database**
 - ▼  **nina** Nina\
 - ▶  **Commands**
 - ▶  **Component**
 - ▶  **Contracts**
 - ▶  **Exceptions**
 - ▶  **Field**
 - ▶  **Media**
 - ▶  **Services**
 - ▶  **Template.Twig**
 - ▶  **Traits**
 -  **Collection.php**
 -  **Context.php**
 -  **helpers.php**
 -  **Nina.php**
 - ▶  **public**
 - ▶  **resources**
 - ▶  **spec**
 - Le dossier « **app** » est le dossier spécifique au client, on y trouve les **composants Nina** (ex : « **ArticleComponent** »), les **controllers** (retournent une réponse en fonction d'une requête HTTP, via un système de routes), les **entities** (représentent les tables en base de données), les **repositories** (séparent la couche d'accès au données de la couche métier), et deux classes contenant uniquement des constantes,
 - Le dossier « **bootstrap** » contient les fichiers qui seront exécutés au démarrage de l'application, le dossier « **config** » contient des fichiers de configuration, le dossier « **database** » contient les différentes **migrations** (une sorte de contrôle de version de la base de données, permet de créer/supprimer des tables, ajouter/enlever des colonnes, ...) et les **seeders** (alimentent la base de données avec des données),
 - Le dossier « **nina** », le cœur de NinaCMS, contient les bases fondamentales du projet :
 - Le dossier « **Commands** » contient des commandes pouvant être exécutées dans un invité de commandes,
 - Les dossiers « **Component** », « **Field** » et « **Media** » contiennent des classes de bases (« **BaseComponent** », « **BaseField** » et « **BaseMedia** ») qui sont héritées par les composants, champs, et médias situés dans le dossier « **app** »,
 - Le dossier « **Contracts** » contient des contrats, des interfaces PHP afin de garantir qu'une classe « remplisse son contrat »,
 - Le dossier « **Exceptions** » contient une grande liste des exceptions (classe PHP pour lancer une erreur) spécifiques à Nina,



- Le dossier « **Services** » contient des services, c'est-à-dire des **morceaux de code** qui ont pour but d'être **ré-utilisables** partout dans le projet, ou alors pour séparer ce code des **controllers** par soucis d'organisation et de clarté,
- Le dossier « **Template** » contient des abstractions pour les systèmes de **template** (modèle), uniquement **Twig** est supporté pour l'instant,
- Le dossier « **Traits** » contient différents **traits** PHP, ils sont utiles pour la composition de **classes PHP**, c'est-à-dire de ré-utiliser des fonctions et variables dans une classe sans pour autant faire d'héritage.
- Le dossier « **public** » contient les ressources publiques (les **assets**, c'est-à-dire des **styles CSS**, des **scripts JS**, et des **images minifiées** et **compressés**, le code de la **partie Administration**, et les fichiers de la Médiathèque). Ce dossier jouera le rôle de racine pour le serveur web **Apache** ou **nginx**, cela permet d'isoler le reste du projet, notamment des informations sensibles (comme le dossier « **config** »),
- Le dossier « **ressources** » contient lui aussi des ressources, mais privées. C'est dans ce dossier que se trouvent les fichiers **CSS**, **JS** et **images originaux** (sans avoir été minifiées et compressés). Le code de la **partie Administration** s'y trouve, quasiment entièrement composée de **composants VueJS** ! On y trouve aussi les différents **templates** Twig (ceux pour les **Field** et **Fragment**), ainsi que des fichiers de **traductions** (français et anglais) afin de traduire l'administration et les exceptions.
- Le dossier « **spec** » contient des tests unitaires, c'est-à-dire des tests permettant de tester .

4. Conclusion

4.1. Expérience acquise

4.2. Avenir du projet

5. Glossaire

→ **CSS** : Cascading StyleSheet, permet de définir le style visuel d'une page HTML (placement et taille des éléments, marges, couleurs, ...).

→ **Framework** : Structure logicielle regroupant plusieurs composants afin de fournir des bases pour le développement d'un projet informatique – <https://fr.wikipedia.org/wiki/Framework>.

→ **Git** : Logiciel de gestion de version – <https://git-scm.com>.

→ **GitKraken** : Interface graphique dédiée à la gestion de projets versionnés grâce au programme git – <https://www.gitkraken.com>.

→ **Gogs** : Service web auto-hébergé et de gestion de développement de logiciels, utilisant le logiciel de gestion de version Git – <https://gogs.io>.

→ **HTML** : HyperText Markup Language, un format de données utilisant des balises pour décrire le contenu des pages web – <https://fr.wikipedia.org/wiki/HTML>.

→ **JavaScript** : Langage de programmation de scripts initialement utilisé côté client sur des pages web dynamiques – <https://fr.wikipedia.org/wiki/JavaScript>.

→ **JSDoc** : Programme permettant de générer une documentation en récupérant et en interprétant des commentaires dans du code JavaScript – <http://usejsdoc.org>.

→ **JSON** : Format d'échange de données s'inspirant de JavaScript, notamment utilisé pour des communications réseau dû à sa syntaxe légère et épurée – <https://fr.wikipedia.org/wiki/JSON>.

→ **Laravel** : Framework PHP suivant le principe du Modèle/Vue/Contrôleur, et permet au développeur de développer très rapidement un projet – <https://laravel.com>.

- **Node.js** : Environnement permettant d'exécuter du JavaScript côté serveur – <https://nodejs.org>.
- **PHP** : **PHP** **H**yperText **P**reprocessor, langage utilisé pour générer de l'HTML – <http://php.net>.
- **Sami** : Un générateur de documentation de code PHP – <https://github.com/FriendsOfPHP/Sami>.
- **SCSS (Sass)** : Pré-processeur CSS qui permet une écriture plus simple et plus organisée de code CSS, et qui sera ensuite compilé en CSS – <http://sass-lang.com>.
- **VueJS** : Framework JavaScript permettant de créer des interfaces web interactives en utilisant le principe de composants webs – <https://vuejs.org>.
- **Webpack** : Empaqueur de code JavaScript reposant sur le principe des modules JavaScript – <https://webpack.js.org>.

6. Bibliographie

- <http://www.sailing-up.com>
- <https://developer.mozilla.org/en-US>
- <https://vuejs.org>
- <http://www.php.net>
- <https://laravel.com>
- <https://github.com>

Résumé en français :

Mots clés :

Éditeur visuel de site web, WYSIWYG, interactivité, CMS, Composants, drag & drop, explorateur de fichiers, AJAX, JSON, PHP, Laravel, Twig, Node.js, Vue, Vuex, Webpack, Gulp, ESLint, git.

Matériel / logiciels / méthodes utilisé(e)s :

- Ordinateur de l'entreprise configuré sous Ubuntu, deux écrans, une souris et un clavier,
- PhpStorm, Apache, PHP, MySQL, Node.js, git, GitKraken, Google Chrome, Mozilla Firefox, Opera,
- Méthode agile avec pour client mon maître d'apprentissage, car je développais un produit pour le compte de l'entreprise.