



RAPPORT DE STAGE CHEZ UPLIX

Effectué par Arnaud Mirocha dans le cadre du BTS SIO option
SLAM du lycée St-Adjutor de Vernon



Agence de SEO

09/01/2023 – 17/02/2023

UPLIX

20 Rue de Madrid Paris

REMERCIEMENTS

Je remercie sincèrement Emmanuel de Vauxmoret (CEO de Uplix) qui m'a accueilli dans son agence. Je tiens également à remercier chaleureusement Julien Beugras, qui m'a encadré et conseillé avec bienveillance tout au long de cette expérience professionnelle.

Enfin, je suis reconnaissant envers toute l'équipe d'Uplix pour leur accueil chaleureux, leur bienveillance et leur bonne humeur constante, qui ont grandement contribué à rendre mon apprentissage et mon travail dans les meilleures conditions.

SOMMAIRE

- **Présentation de l'entreprise (Pages 3)**
- **Présentation du service d'accueil et des moyens informatiques (Page 4)**
- **Présentation des projets réalisés (Pages 5 à 15)**
- **Conclusion (Page 16)**

PRÉSENTATION DE L'ENTREPRISE



Uplix[↑] est une agence de SEO (Search Engine Optimization) nouvelle génération. Elle est spécialisée dans le SEO (référencement naturel), à différencier du SEA pour Search Engine Advertising (référencement par publicité).

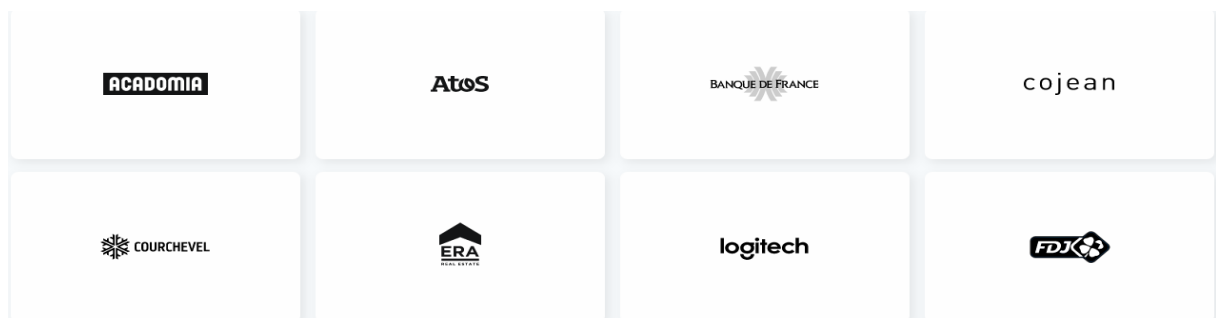
Le but ici est d'aider à augmenter la visibilité d'un site internet en l'aidant à être mieux référencé par les moteurs de recherches.

Uplix[↑] a été fondée en décembre 2020 par Emmanuel de Vauxmoret, son CEO, et David Benhamou, CEO de Wizzmedia.

L'idée était d'aborder le référencement naturel avec une vision nouvelle et innovante pour proposer à ses clients une meilleure offre.

Les clients d'Uplix sont divers et variés.

Le référencement naturel et le gain de visibilité que cela entraîne, intéresse tout type d'entreprises quel que soit la taille ou le domaine d'exercice :

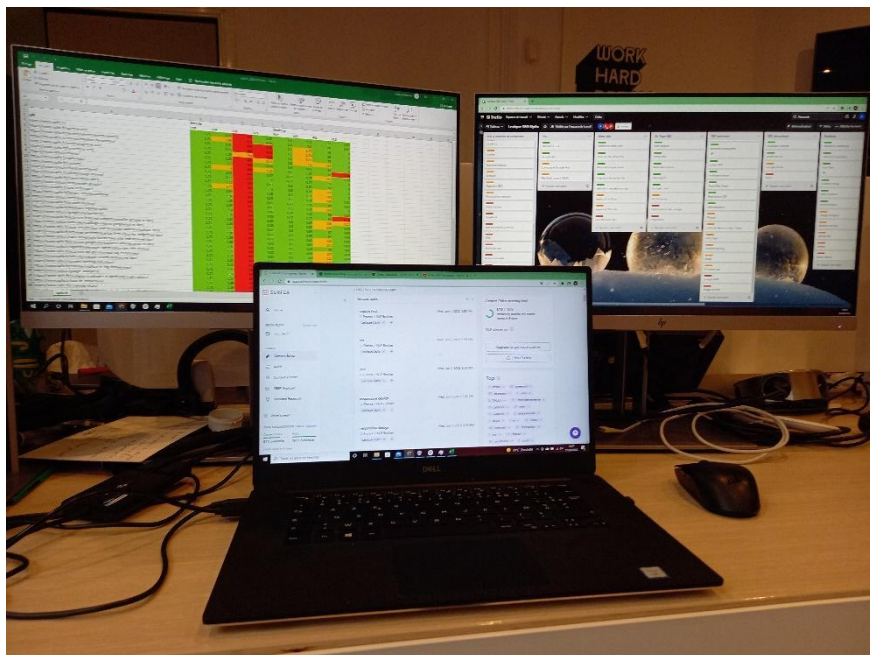


Présentation du service d'accueil et des moyens informatiques

Service d'accueil : J'ai été intégré au service informatique, aux côtés de consultants SEO

Les moyens informatiques sont les mêmes pour chacun :

- Un bureau en open space
- Un pc portable
- 1 ou 2 écrans supplémentaires



Présentation du projets réalisé

Lors de mon stage je n'ai été affecté qu'à un seul projet, le développement d'un site client. Ce site avait pour but d'être utilisé en interne comme un outil. Il devait avoir les fonctionnalités suivantes :

- Permettre la création et la connexion à un compte utilisateur.
- Faire en sorte que les données ne soient accessibles qu'aux utilisateurs connectés.
- La création, la modification et la consultation de projets, pages web, offre de backlink, commande de backlink...
- Permettre le suivi des backlinks/articles.

Pour ce projet je suis parti de 0, il a donc fallu que je mette en place plusieurs choses avant de m'attaquer au développement du site client.

Plan des tâches effectuées en 2 parties :

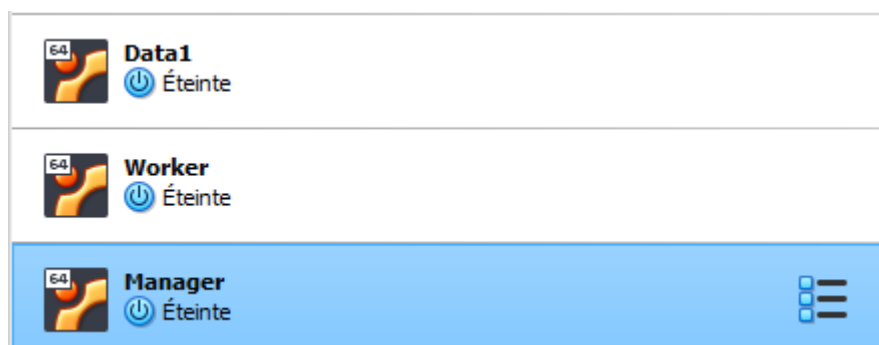
- 1) Mise en place d'un espace de développement :
 - Mise en place de Docker
 - Mise en place Portainer
 - Mise en place de Traefik
 - Mise en place de la Bdd (postgresql et pgadmin)
 - Mise en place de Strapi (construction bdd et api)
 - Mise en place de appsmith
- 2) Développement du site client et de ces fonctionnalités.

Mise en place d'un espace de développement

Pour pouvoir développer en local et permettre le déploiement sur des serveurs j'ai dû créer trois machines virtuelles. Disposant de ressources limitées, j'ai d'abord dû effectuer quelques tests pour connaître la consommation des machines virtuelles.

VM	RAM	CPU	ESPACE
Manager	3 Go	3 Coeurs	25 Go
Worker	6 Go	3 Coeurs	25 Go
Data	2 Go	2 Coeurs	25 Go

Il s'agit de machines ubuntu servers 22.04 avec chacune leur fonctionnalité.



Une machine virtuelle “Manager” qui héberge Docker, Portainer, Traefik et PgAdmin.

Docker est une plateforme de **conteneurisation** qui permet de créer, de déployer et de gérer des applications dans des conteneurs. Cela facilite la création d'environnements de développement reproductibles, de systèmes de test automatisés, de déploiements de production simplifiés et de gestion d'infrastructures à grande échelle.

En utilisant des images de conteneurs, Docker permet une transition fluide et sans accroc entre l'espace de développement et l'espace de production, en garantissant que les services fonctionnent de manière cohérente, quelles que soient les configurations de l'hôte sur lequel ils sont exécutés.

Une fois Docker installé et initialisé, il faut créer un **docker swarm** qui permet d'harmoniser la gestion des conteneurs entre les trois machines. Une fois les trois machines connectées et dans le même swarm, on peut commencer à installer et initialiser les différents services qui seront utilisés. cependant "l'interface" de docker se présente sous cette forme :

```
root@manager:/# docker service ls
```

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
7183n0a4r35r	appsmith_appsmith-ce	replicated	1/1	appsmith/appsmith-ce:v1.8.11	
sg6q9rkn2ch6	pgadmin_app	replicated	1/1	dpage/pgadmin4:latest	
1iv1kp1y18gp	portainer_agent	global	3/3	portainer/agent:latest	
6o7kighr5j5bv	portainer_portainer	replicated	1/1	portainer/portainer-ce:2.15.1	*:9000->9000/tcp
wtvas98n8se5	postgres_postgres	replicated	1/1	postgres:latest	*:5432->5432/tcp
bxzt8240z42e	registry_registry	replicated	1/1	registry:2	
iozn6njke4a	strapi_strapi	replicated	1/1	arnaudmirocha/strapi-worker:0.3	
rcq3es1insrq	traefik_app	replicated	1/1	traefik:v2.9.1	*:8080->8080/tcp
eb1v1auoce41	whoami_whoami	replicated	1/1	traefik/whoami:latest	

Et son fonctionnement se fait uniquement par ligne de commande, ce qui sur un serveur ubuntu n'est pas pratique de plus les fichiers de configuration font environ une quarantaine de lignes chacun. C'est pourquoi j'ai utilisé une extension de Visual Studio Code (Remote-SSH) pour mettre en place une connexion SSH entre mon pc et mes serveurs afin de rendre les manipulations plus simples.

Une fois docker en fonctionnement, j'ai mis en place un service Portainer. Il s'agit d'une interface graphique qui permet de gérer les conteneurs/images/services de Docker. Et son apparence est bien plus attractive que celle de Docker.

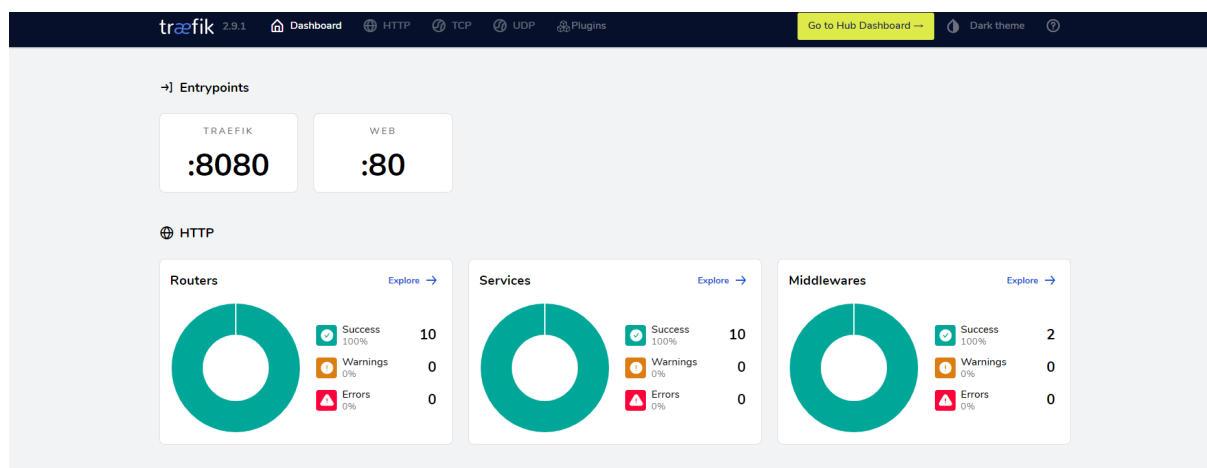
8 Stacks	9 Services
17 Containers 10 running 1 healthy 0 stopped 0 unhealthy	24 Images 11.4 GB
20 Volumes	22 Networks
- GPUs	

Name	State	Quick Actions	Stack	Image	Created	IP Address	Host	Published Ports	Ownership
appsmith_appsmith-ce.1.vipif...	healthy		appsmith	appsmith/appsmith-ce:v1.8.11	2023-03-05 17:17:11	10.0.2.9	worker	-	administrators
portainer_agent.w3kpl224k4bjd...	running		portainer	portainer/agent:latest	2023-03-05 18:47:28	10.0.3.15	data	-	administrators
postgres_postgres.1.lzpk9r5e...	running		postgres	postgres:latest	2023-03-05 17:17:09	10.0.0.7	data	-	administrators
portainer_agent.jl68ro971w09...	running		portainer	portainer/agent:latest	2023-03-05 18:47:27	10.0.3.13	worker	-	administrators
strapi_strapi.1.b7rkh7hpykh...	running		strapi	arnaudmirocha/strapi-worker:0.3	2023-03-05 17:17:11	10.0.2.7	worker	-	administrators
whoami_whoami.1.o4rziH9nznj9...	running		whoami	traefik/whoami:latest	2023-03-05 17:17:11	10.0.2.8	worker	-	administrators
portainer_portainer.1.ky7i2um...	running		portainer	portainer/portainer-ce:2.15.1	2023-03-05 18:47:30	10.0.0.13	manager	-	administrators

Portainer permet de manipuler Docker et ses services sans passer par les commandes et en ayant une meilleure vision d'ensemble, ce qui simplifie grandement les manipulations. De plus il permet d'accéder très rapidement et

facilement aux consoles des conteneurs et donc il permet en cas de problème de rapidement localiser et identifier les erreurs.

Suite à ça, j'ai mis en place le service Traefik. C'est un reverse proxy et un contrôleur de trafic pour les applications basées sur des conteneurs, offrant une gestion de mise en réseau, de routage et de sécurité avancée et facile à configurer pour assurer la fiabilité et la sécurité des applications web et des microservices. Traefik peut facilement être intégré dans des environnements Docker et Kubernetes, offrant une gestion simplifiée du réseau pour les conteneurs.



Traefik peut détecter automatiquement les services Docker et Kubernetes (un autre service de conteneurisation), ce qui permet une configuration automatique des règles de routage et de l'équilibrage de charge. Traefik utilise des "labels" Docker pour détecter les services et fournir des informations supplémentaires

```
labels:
- "traefik.enable=true"
- "traefik.docker.network=traefik_public"
- "traefik.constraint-label=traefik_public"
- "traefik.http.routers.strapi.rule=Host(`strapi.local`)"
- "traefik.http.routers.strapi.entrypoints=web"
- "traefik.http.services.strapi.loadbalancer.server.port=1337"
```

Traefik peut automatiquement générer et gérer les certificats SSL pour les domaines configurés, en utilisant Let's Encrypt. Ce qui permet de mettre en place un protocole HTTPS facilement.

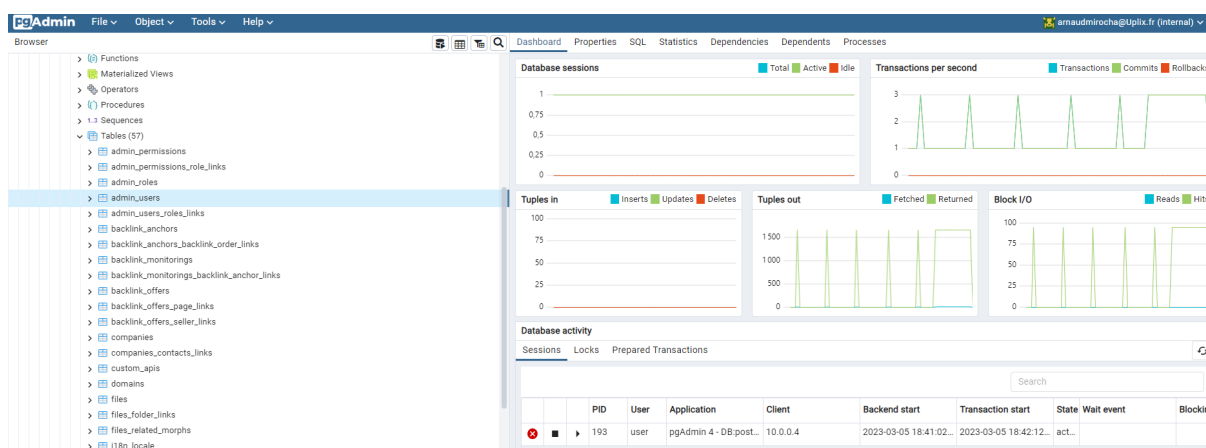
Pour accéder aux différentes interfaces (Portainer, Traefik, Appsmith...) il faut passer par un navigateur, cependant pour des raisons de sécurité, les machines "worker" et "data" ne sont accessible qu'en passant par la machine "manager" donc pour y accéder il faut entrer l'adresse ip de la machine manager dans la barre d'URL. Mais comme l'accès de tous les services se fait par la même adresse ip, comment les différencier ? Et bien c'est encore un fois grâce à Traefik qui va permettre de différencier chaque service grâce à la ligne suivante :

"traefik.http.routers.strapi.rule=Host(`strapi.local`)" qui va permettre d'attribuer un URL à un service. Il suffit d'également modifier le fichier hosts de son ordinateur (qui a la même fonction qu'un serveur DHCP) pour rediriger l'ip de la machine "manager" vers l'url des service comme ceci :

```
C: > Windows > System32 > drivers > etc > hosts

1  #
2  127.0.0.1 localhost
3  ::1 localhost
4
5  192.168.56.101 traefik.local
6  192.168.56.101 appsmith.local
7  192.168.56.101 portainer.local
8  192.168.56.101 whoami.local
9  192.168.56.101 pgadmin.local
10 192.168.56.101 strapi.local
11 192.168.56.101 registry.local
```

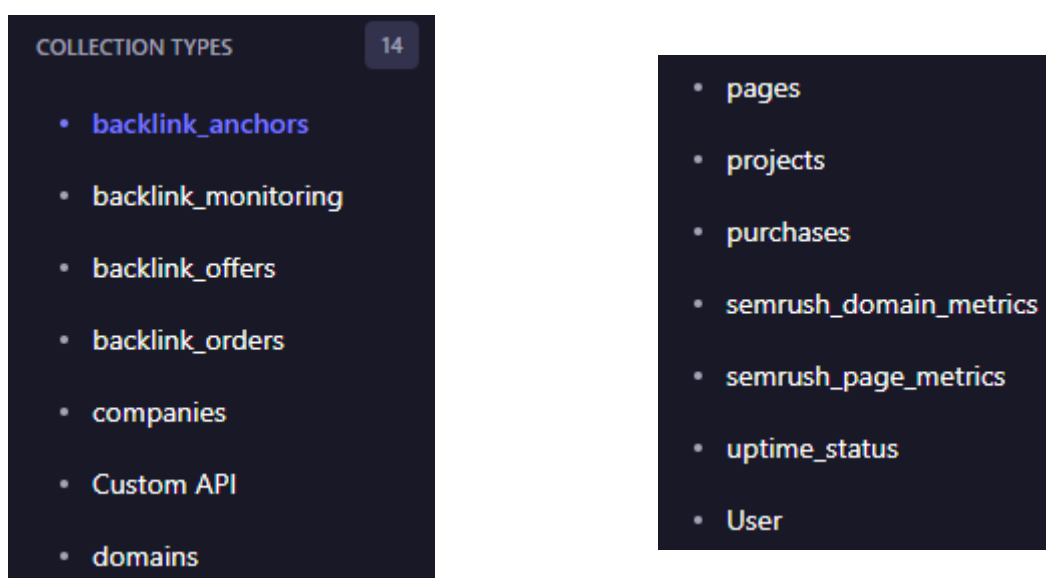
Une fois Traefik installer et configurer, j'ai mis en place une base de données Postgresql ainsi qu'un service Pgadmin qui est une plateforme de gestion open-source pour PostgreSQL qui permet d'avoir un vue d'ensemble sur la base de données.



Une machine virtuelle “Data” qui héberge une base de données Postgresql.

Pour la gestion des données, il m’a été demandé de mettre en place une base de données PostgreSQL, il s’agit d’un système de gestion de base de données reconnu pour sa fiabilité, sa sécurité, ses performances élevées et son support de **transactions ACID**. Il est extensible et peut être facilement adapté aux besoins des utilisateurs. Les transactions ACID (Atomicité, Cohérence, Isolation, Durabilité) sont un ensemble de propriétés qui garantissent la fiabilité des opérations de traitement de données dans une base de données relationnelle.

La base de données mise en place compte à ce jours 14 tables



La table User, permet le stockage des utilisateurs ainsi que de leur JWT (JSON Web Token) qui permet leur authentification et rend donc possible leur accès aux données. Cette table va être sollicitée à chaque connexion, chaque création de compte et chaque requête d’API demandant d’être authentifié.

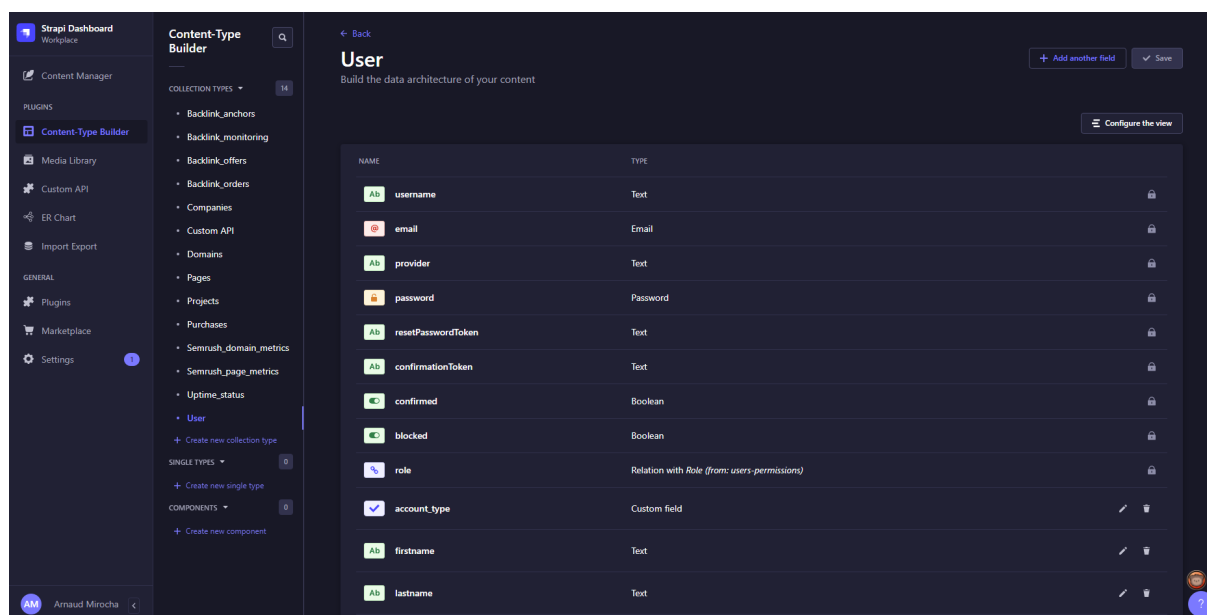
La table Custom API est utilisée par Stapi pour stocker des requêtes d’API personnalisées.

Les autres tables sont liées entre elles et représentent les données qui seront manipulées sur le site.

On peut voir sur les captures d’écrans ci-dessus qu’il y a 14 tables, or si on se fie à l’interface de PgAdmin on remarque la présence de 57 tables. Ceci est dû à la manière dont Strapi gère les relations entre les collections.

Une machine virtuelle “Worker” qui héberge Appsmith et Strapi.

Pour la machine “Worker” j’ai d’abord mis en place un service Strapi. Strapi est probablement la meilleure découverte que j’ai faite durant ce stage. Il s’agit d’une plateforme de gestion de contenu (CMS) Headless, conçue pour aider les développeurs à créer rapidement et facilement des API de contenu pour des applications web.



Il peut se mettre en place en une quinzaine de minutes et peut être utilisé avec plusieurs types de base de données (PostgreSQL, MongoDB, MySQL, etc.). Une fois connecté à une base de données, il permet l’ajout, la modification et la suppression des tables et le tout très simplement et très rapidement sans écrire la moindre ligne de SQL.

Strapi permet aussi de faire des relations entre les tables. Lorsqu’on crée une nouvelle relation entre deux tables, Strapi va stocker cette relation dans une autre table créée pour l’occasion et invisible depuis l’interface de Strapi, c’est pour cela qu’il y avait une différence entre le nombre de tables affichées par Strapi et par PgAdmin.

J’ai dit plus haut que Strapi pouvait se mettre en place en 15 minutes, cependant c’est dans un petit projet, sans prendre en compte Docker, Traefik etc... Pour ce projet-ci, j’ai perdu beaucoup de temps sur la configuration de Strapi et son intégration, surtout que au cours du projet, pour l’ajout de plugin (permettant la création de MCD, l’insertion de données depuis un fichier excel...), j’ai dû passer de la version 3.0 à la 4.0. Les différences entre Strapi v3 et Strapi v4 sont significatives concernant le fonctionnement de l’API et la configuration avec Docker.

Lors de la création d'un conteneur avec Docker on utilise une image (une image est un package autonome qui contient tout ce dont une application a besoin pour fonctionner, y compris le code source, les bibliothèques, les dépendances, les fichiers de configuration, etc.). Habituellement pour trouver une image, on peut se rendre sur docker hub, et suffit juste d'appeler l'image

```
services:
  appsmith-ce:
    image: appsmith/appsmith-ce:v1.8.11
```

Or, n'ayant pas trouvé d'image de Strapi v4 déjà fait, j'ai dû la créer moi même à partir d'un Dockerfile.

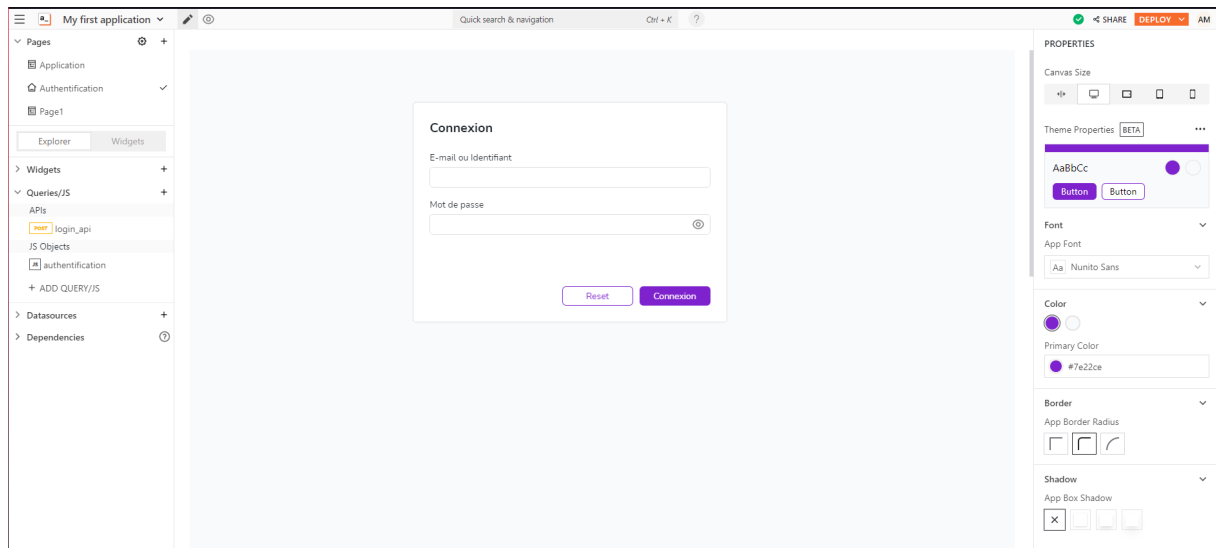
```
FROM node:16-alpine
# Installing libvips-dev for sharp Compatibility
RUN apk update && apk add --no-cache build-base gcc autoconf automake zlib-dev libpng-dev nasm bash vips-dev
ARG NODE_ENV=development
ENV NODE_ENV=${NODE_ENV}
WORKDIR /opt/
COPY ./package.json ./package-lock.json ./
ENV PATH /opt/node_modules/.bin:$PATH
RUN npm install
WORKDIR /opt/app
COPY ./ .
RUN npm run build
EXPOSE 1337
CMD ["npm", "run", "develop"]
```

Après l'avoir créé il fallait la stocker pour pouvoir l'utiliser pour cela j'avais deux options, soit je la stockais sur docker hub (en public), soit je la stockais grâce à registry (en local). J'ai pris la première option car elle était plus simple et rapide à mettre en place.

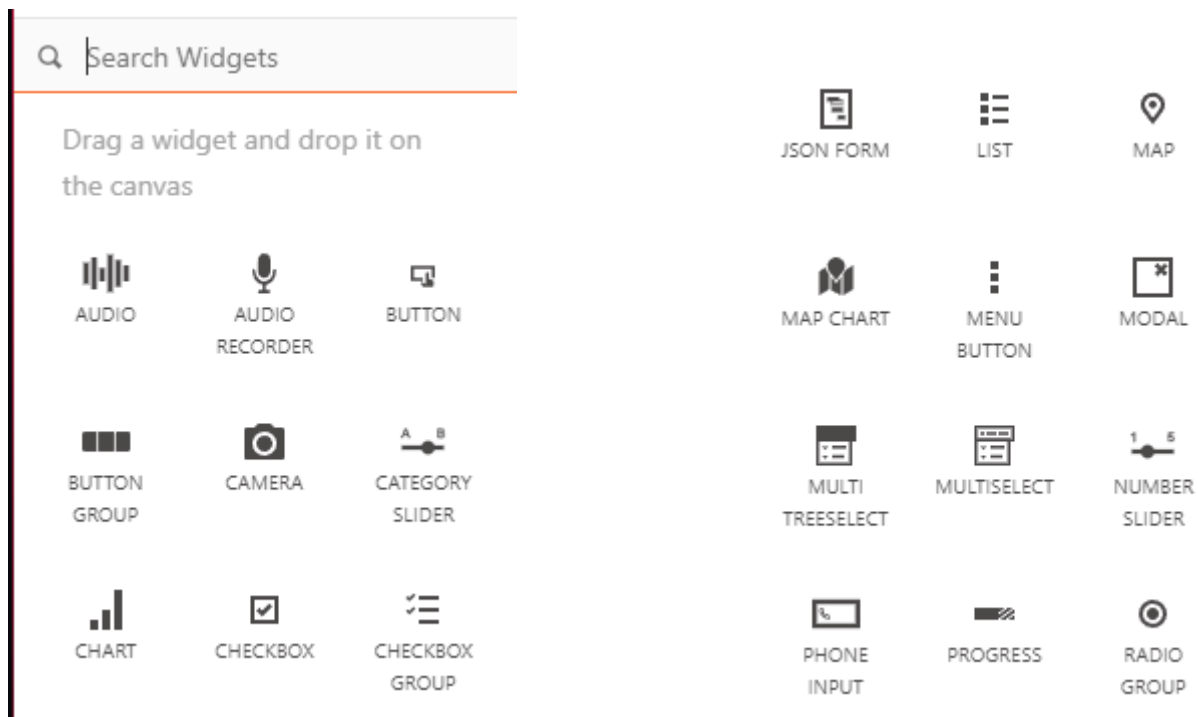
Une fois ce dernier service configuré, l'espace de développement est enfin prêt et le développement du site client peut commencer.

Développement du site client et de ces fonctionnalités

Pour le développement du site client, j'ai utilisé Appsmith. Appsmith est une plateforme open-source qui permet aux développeurs de créer rapidement des applications web. Elle prend en charge plusieurs types de bases de données.



Appsmith permet un développement très facile du front grâce à son système de widget.



Pour la mise en forme du site j'ai principalement utilisé des JSON form, qui permettent une manipulation simplifiée d'objet JSON.

The screenshot shows a web form titled "Modifier une commande". It contains several input fields: "Statut" with a dropdown menu showing "Terminée"; "Prix" with a currency selector (€) and a text input showing "52552"; "Projet" with a dropdown menu showing "project 25"; "Vendeur" with a dropdown menu showing "Vendeur2"; "Page Référente" with a text input showing "https://www.uplix.fr/clients/"; and "Anchors" with a dropdown menu showing "1" and a sub-section labeled "Anchors Text". At the bottom right of the form are two buttons: "Reset" and "Submit". The form is styled with a light gray background and rounded corners.

Un formulaire peut être lié à un objet JSON pour que celui-ci s'adapte dès qu'il y a une modification sans avoir à recharger la page ou le formulaire.

Source Data

```
{{appsmith.store.selectedProject}}
```

Ici le formulaire responsable de la modification d'un projet est lié à un objet JSON nommé "selectedProject" que j'ai stocké dans le store de Appsmith (ce qui correspond à un local storage).

On peut aussi mettre des tableaux et les lier à des objets JSON pour qu'ils créent automatiquement les bonnes colonnes, qu'ils les remplissent etc...




Q Search...

Filters

Download

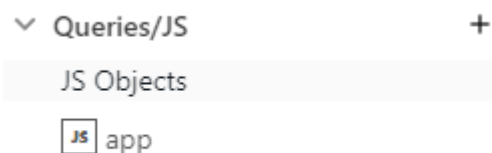
3 Records

Page 1 of 1

URL	Prix	Statut	Modifier
https://www.uplix.fr/agence-seo/	52552	Terminée	
https://www.google.fr/imghp?hl=fr&authuser=0&ogbl	500	En attente de rédaction	
https://www.google.fr/imghp?hl=fr&authuser=0&ogbl	458	En cours	

Avec Appsmith on peut directement connecter son application web à sa base de données pour pouvoir effectuer des requêtes directement depuis l'interface de Appsmith cependant, on fait le choix d'utiliser uniquement l'API mise en place par Strapi pour ajouter, modifier et supprimer les données.

Pour gérer la partie back de l'application web (donc la gestion des données etc...), Appsmith permet de créer des fonctions en javascript qui sont appelables depuis n'importe quel widget.



Et c'est ici qu'est gérée la manipulation des données, ainsi que leur stockage dans le store de Appsmith.

```
getUserData : async () =>{
  const getUser = await fetch(
    "http://strapi.local/api/users/"+appsmith.store.userId+"?populate=*",
    {
      method: "GET",
      headers: {
        'Authorization': "Bearer " + appsmith.store.jwt,
        Accept: "application/json",
        "Content-Type": "application/json",
      },
    }
  );
  const user = await getUser.json();
  await storeValue("user", user, false)
  await storeValue("projects", user.projects, false)
  await storeValue("purchases", user.purchases, false)
  await storeValue("backlink_offers", user.backlink_offers, false)
}
```


Conclusion

Je tire un bilan très positif de ce stage, qui fut une expérience très enrichissante tant sur le plan professionnel que personnel.

J'ai pu appréhender toutes les facettes du monde de l'entreprise et du développement web. J'ai pu découvrir des outils et des technologies modernes et j'ai pu profiter d'une expérience de développeur full-stack en touchant à un peu tout les domaine

Je suis reconnaissant envers l'équipe d'Uplix pour cette formidable expérience, qui m'a permis de travailler dans un environnement positif et collaboratif, et de bénéficier d'un encadrement de qualité. J'ai pu apprendre de mes collègues et de mes responsables, et j'ai été encouragé à poser des questions et à proposer des idées pour améliorer mes compétences et contribuer aux projets de l'entreprise.

Uplix possède également un site et des réseaux sociaux :

- Site internet : uplix.fr 
- LinkedIn : [Uplix](#) 
- Compte Instagram : [uplix_live](#) 
- Newsletter : uplix.fr
- Twitter : [uplix_live](#)

