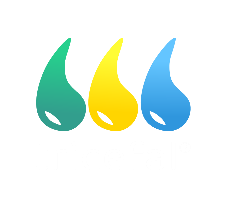


|  |
| --- |
| BACKEND  Compte-rendu de stage :  API REST Back-office  mars-mai 2021 |
|  |
| 25 mai  Tricefal  Créé par : Florent SANIKA |



Remerciements :

Je remercie Tricefal pour l’opportunité de stage et pour nous avoir appris que programmer avec les bonnes pratiques, c’est mieux.

*« Always code as if the person who ends up maintaining your code is a violent psychopath who knows where you live. »*

Je remercie nos formateurs du CMFP Jean-Pierre BOTO et Stéphane [] qui m’ont apporté les connaissances pour un apprentissage constant et autonome.

Table des matières

1. Présentation

* Entreprise
* Compétences couvertes par le projet :
* Résumé

1. Projet

* Expression des besoins
* Spécification technique
  + Les données
  + Le côté utilisateur
  + L’authentification
  + L’architecture
  + Les tests
* Jeu d’essai
* Point particulier

# Présentation

## Compétences couvertes par le projet

Le projet suivant couvre les compétences suivantes :

* Développer les composants d’accès aux données.
* Développer la partie backend d’une application web ou web mobile.

## Résumé

Pour ma période en entreprise, j’effectue un stage 10 semaines en distanciel dans une entreprise de portage salarial, TRICEFAL.

Contexte :

Logée à Puteaux, Tricefal apporte une solution pour contractualiser un freelance avec ses clients.

Le but : conserver la liberté du consultant, et acquérir les avantages des salariés.

Sur leur site, quelqu’un d’intéressé ou qui souhaite en savoir plus peut s’inscrire.

La demande :

Faire un écran d’inscription pour la partie back-office du site. Cet écran permet la connexion d’un employé pour avoir accès à la liste des inscrits, de mettre à jour ces derniers avec les différentes informations, de les activer ou les désactiver.

Une nouvelle base de données sera créée à terme. Il faut donc afficher les inscrits de la première base de données, mais en cas de modifications, une copie de l’inscrit doit être faite dans la nouvelle base.

En effet, un écran d’inscription front-office est déjà disponible et relié à leur base de données via une api.

Ma mission consistait à développer une API REST en Python contenant un CRUD ainsi que les méthodes activer/désactiver pour la partie back-office de leur site. Je devais ensuite la relier à une nouvelle base de données d’une part, puis effectuer l’intégration au front-end développée par une autre équipe d’autre part.

Mon API est reliée avec la base de données (PostgreSQL en production et SQLite en développement) en utilisant la librairie SQLAlchemy et avec la partie front-end en utilisant le micro-framework Flask avec l’extension Flask-RESTFUL. La sérialisation des données avant l’envoi vers le front-end se fait via la librairie Marshmallow.

La partie test de mon application se fait avec le framework Pytest.

Cette application a été réalisée sous une architecture hexagonale.

# Projet

## Expression des besoins

Faire un écran d’inscription pour la partie back-office du site. Cet écran permet à un employé de TRICEFAL de se connecter pour avoir accès à la liste des inscrits, de mettre à jour ces derniers avec les différentes informations, de les activer ou les désactiver :

* L’authentification se fait coté front-end via OIDC. En revanche, le token est nécessaire à chaque requête vers l’API distant et est donc transmis à chaque requête.
* Une nouvelle base de données sera créée à terme. Il faut donc afficher les inscrits de la première base de données, mais en cas de modifications, une copie de l’inscrit doit être faite dans la nouvelle base.
* En effet, un écran d’inscription front-office est déjà disponible et relié à leur base de données via une api.
* Le code principal doit cependant être indépendant du système de stockage utilisé, ou de l’interface utilisé coté client.
* Cette application devra être effectué avec une architecture hexagonale. Cette architecture permet de rendre les composants de l’application interchangeable et indépendant

## Spécification technique

Nous avons choisi Python par rapport à PHP pour ce projet :

* La courbe d’apprentissage de Python est plus rapide que PHP pour des débutants.
* La gestion de l’environnement est plus solide et efficace grâce aux environnements virtuels (venv) qui permet de définir les versions utilisées pour Python et ses modules et nous donne la possibilité de faire cohabiter plusieurs versions de Python sur une machine en fonction des projets.
* La modularité du langage, avec Pip (package installer for Python) - en place depuis plus longtemps que Composer qui manque apparemment encore de fiabilité - qui permet l’installation et la gestion facile des bibliothèques.
* Cette modularité amène à une meilleure flexibilité et évolutivité avec l’émergence de nouvelle tendances (IA/ML) pour de future fonctionnalité du site.

NB : J’utilise pipenv qui remplace à la fois pip et venv

Je dois donc développer une API REST en Python contenant un CRUD ainsi que les méthodes activer/désactiver pour l’écran d’inscriptions de la partie back-office de leur site.

### Les données :

Mon API est reliée avec la base de données (PostgreSQL en production et SQLite en développement) en utilisant la librairie SQLAlchemy :

Le but en utilisant cette librairie est de rendre l’application indépendante de la base de données utilisée. Leur base actuelle est sous PostgreSQL, ils peuvent demain décider de migrer vers MySQL ou Oracle sans avoir besoin de modifier tout le code principal. L’utilisation de SQLite en développement s’explique par le fait qu’il est naturellement supporter par Python.

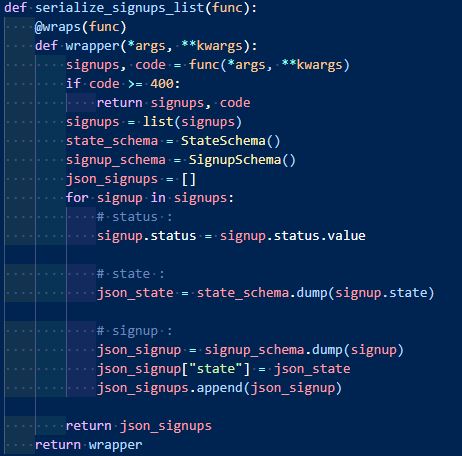
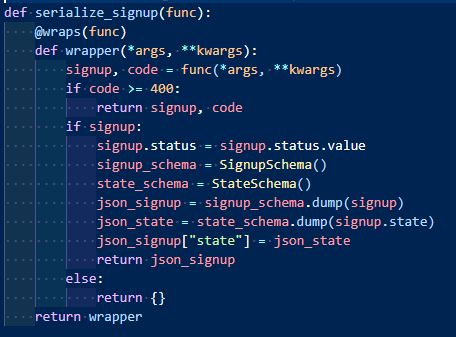
### Le côté utilisateur :

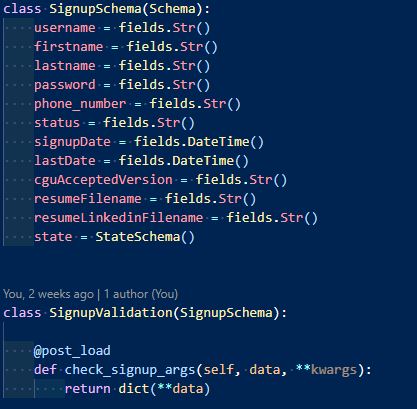
Il a fallu décider du framework à utiliser pour ce projet. Et notre chargé de stage nous a demandé de choisir entre Flask et Django car il ne connaissait pas Python.

N’ayant pas assez d’expérience, nous lui avons donc fait une présentation sur les principales caractéristiques et différences de chacun pour qu’il puisse faire son choix.

Flask est un micro-framework qui permet une plus grande liberté pour l’architecture, plus pratique pour une architecture hexagonale. Il nous permet de n’installer que les composants nécessaires, et la prise en main est plus rapide. Flask a donc été choisi pour le coté utilisateur. L’extension Flask-RESTFUL s’occupe du routage de l’URL vers les classes concernées.

La sérialisation des données avant l’envoi vers le front-end se fait via la librairie Marshmallow.



Marshmallow s’occupe aussi de la vérification et de la validation des données envoyées depuis le front.

### L’authentification :

L’utilisateur se connecte depuis le front-end et l’authentification se fait via OIDC gérée par Angular.

*OIDC : Il s’agit d’une couche supplémentaire sur le protocole OAuth 2.0.*

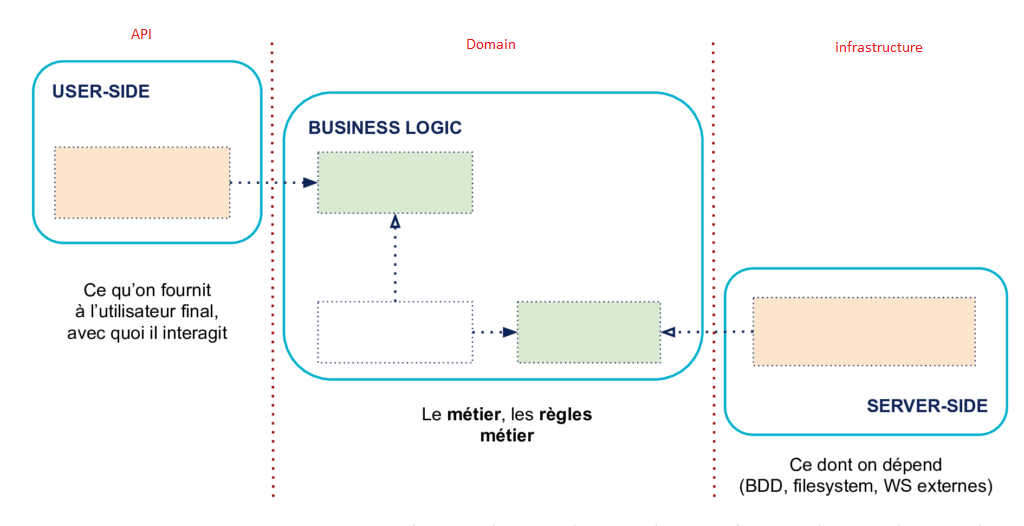
*Il permet à une application ou un site de demander et obtenir des informations sur l’utilisateur finale au format REST/JSON. (Qui est-il, quel droit a-t-il…) Il utilise un système de jeton (token) d’authentification transmis dans la requête*

En revanche, le token est nécessaire à chaque requête vers l’API distant et est donc transmis à chaque requête.

De cette manière, le backend peut vérifier que l’utilisateur est bien connecté sans avoir besoin de stocker son mot de passe.

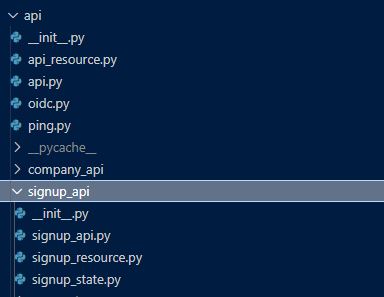
La partie test de mon application se fait avec le framework Pytest, qui permet une plus grande flexibilité pour les tests unitaires.

### L’architecture :

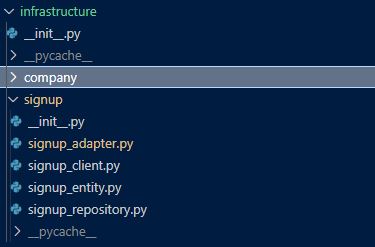


L’architecture hexagonale permet l’inversion de contrôle et l’injection de dépendance. Seul est important la « logique métier ».

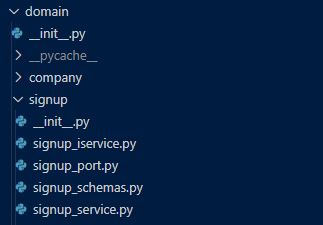
L’user-side et le server-side peuvent être modifiés ou remplacés sans impacter le reste du code. Dans l’application, l’user-side correspond au package API, gérer par Flask-RESTFUL,



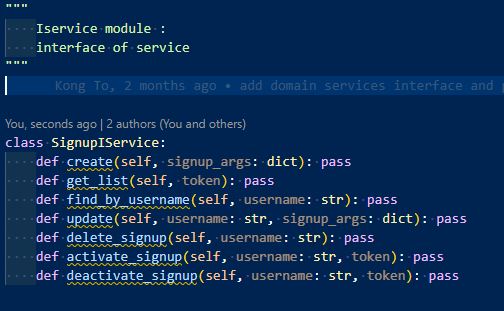
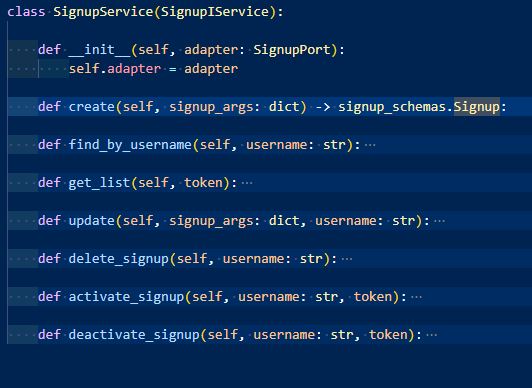
l’infrastructure est le server-side avec SQLAlchemy,



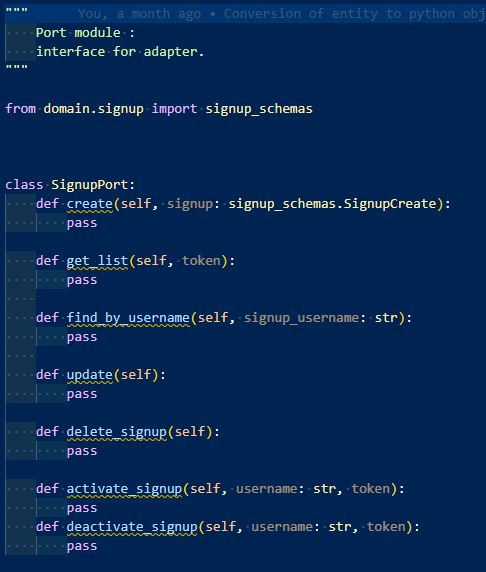
et le code métier est dans le package domain.



L’inversion de contrôle est effectuée à l’aide d’interfaces. Une interface intérieure – SignupIService dont le service héritera, les dépendances user-side utiliseront cette interface. C’est le seul cas où le code métier est visible depuis l’extérieure :



Interface intérieur



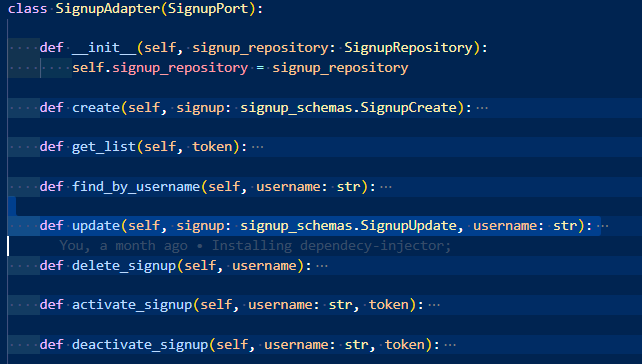
Interface extérieure



Implémente

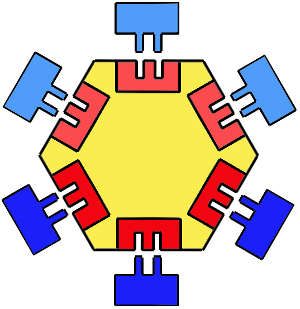
Utilise

et une interface extérieure dont les dépendances server-side hériteront.



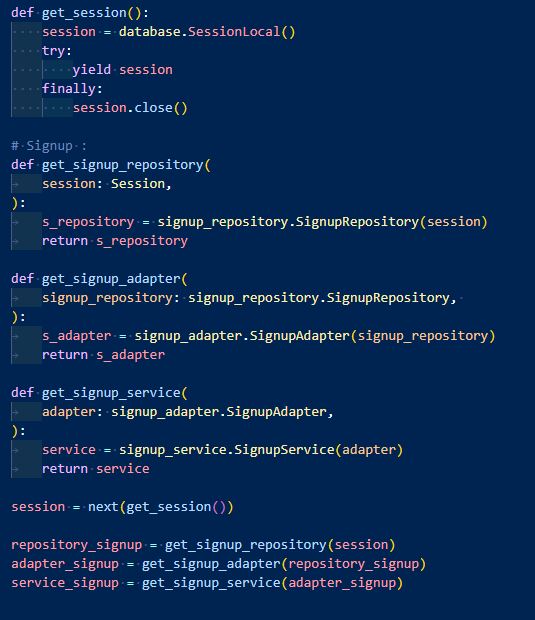
Ainsi, le code métier ne dépendra pas d’une dépendance extérieure mais des interfaces.

L’autre nom de l’architecture hexagonale et l’architecture port adapteurs. Les interfaces de domain sont les ports sur lesquelles viennent se brancher les classes filles des dépendances qui sont les adapteurs.

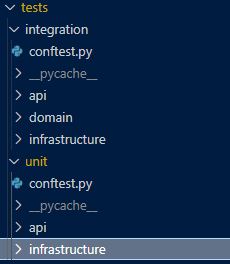


L’injection de dépendance se fait au niveau du container à la racine du package.

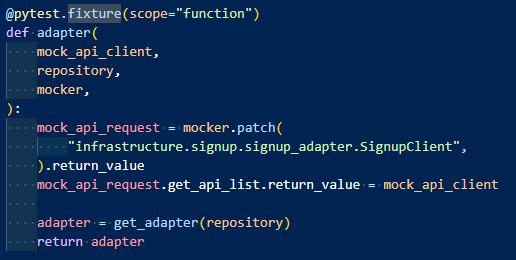
On instancie les différentes classes dans le container et on injecte les instances dans les constructeurs.



### Les tests :

Les configurations nécessaires aux tests se font dans les fichiers conftests.py  
Le fichier commun à tous les tests se situe à la racine du projet. Un deuxième fichier spécifique se trouve dans les dossiers de tests ‘integration’ et ‘unit’.

Ces fichiers de configurations contiennent par exemple les ‘fixtures’ qui vont remplacer les comportements des éléments extérieurs le temps du test.

Par exemple :

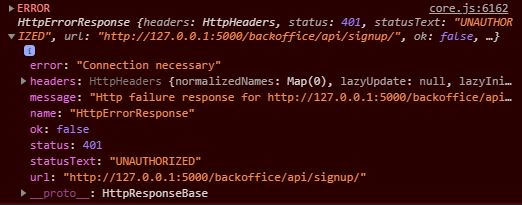
La fixture ci-dessus me permet de mocker le comportement du client appelé pendant l’exécution. Lors du test, SignupClient ne sera pas appeler, le mock\_api\_request sera appelé à la place.

De cette manière, si l’API distant est en maintenance ou ne fonctionne plus, le test fonctionne toujours.

## Jeu d’essai :

Si on essaie d’afficher la liste des inscrits sans être connecté, la liste est vide :

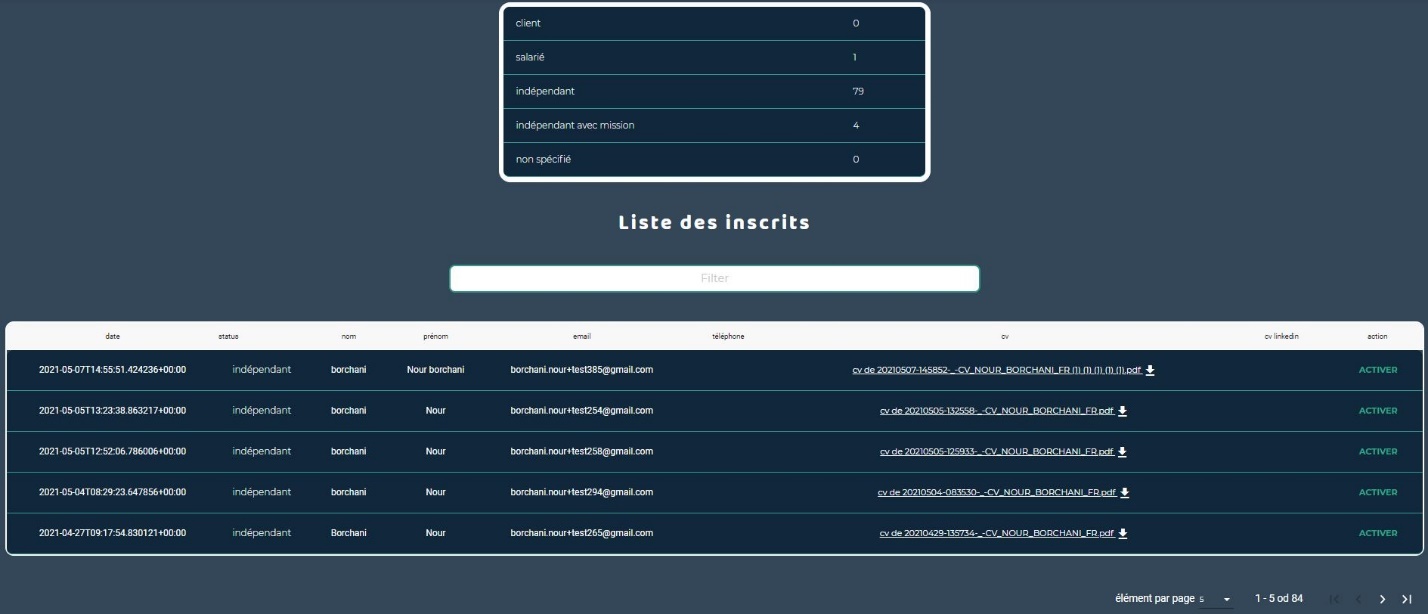
Et la réponse de l’api :

Coté serveur : Coté navigateur :

On se connecte :



On réessaie connecté :

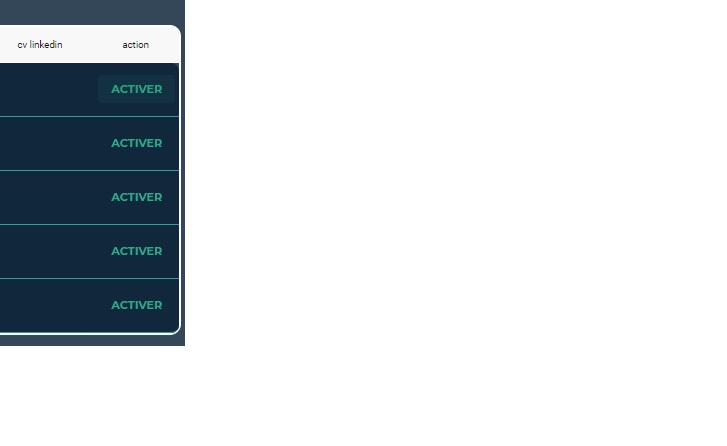


La réponse du serveur :



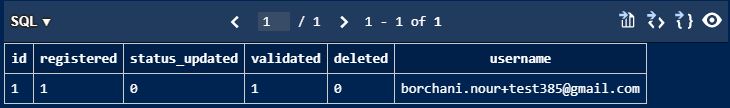
Lors du lancement du serveur, une base de données vide est créée.

En appuyant sur le bouton activer :



Une copie de la ligne correspondante est faite dans la base locale :

Son statut validated passe à true :



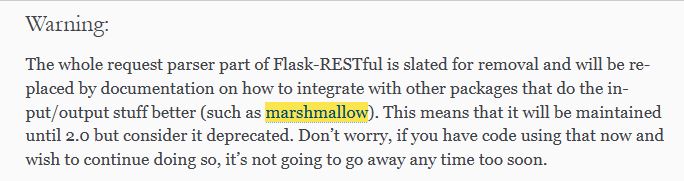
La page d’inscription est rechargée mais cette fois-ci elle affiche la ligne locale : celle qui est activée.



## Point particulier :

Flask-RESTFUL possède actuellement ‘reqparse’ qui s’occupe de la validation des données entrantes.

Problème :



Au début du projet, reqparse a été mis en place. Il peut être garder encore pour quelques années, mais pour être sûr d’avoir une meilleure évolutivité, il devait être remplacé. Marshmallow pouvant gérer la validation (+ désérialisation) et la sérialisation, il a été choisi pour le remplacer.

**Marshmallow** is an ORM/ODM/framework-agnostic library for converting complex datatypes, such as objects, to and from native Python datatypes. In short, marshmallow schemas can be used to:

* Validate input data.
* Deserialize input data to app-level objects.
* Serialize app-level objects to primitive Python types. The serialized objects can then be rendered to standard formats such as JSON for use in an HTTP API.

The Python ecosystem has many great libraries for data formatting and schema validation. In fact, marshmallow was influenced by a number of these libraries. Marshmallow is inspired by [Django REST Framework](https://www.django-rest-framework.org/), [Flask-RESTful](https://flask-restful.readthedocs.io/), and [colander](https://docs.pylonsproject.org/projects/colander/en/latest/). It borrows a number of implementation and design ideas from these libraries to create a flexible and productive solution for marshalling, unmarshalling, and validating data.

TRADUCTION

Marshmallow est une librairie ORM/ODM/framework-agnostique pour la conversion de type de données complexe, telle que les objets, de et vers les types de données Python natifs. En bref, les schémas marshmallow peuvent être utilisé pour :

* Valider les données en entrée.
* Désérialiser les données d’entrée en objets au niveau de l’application.
* Sérialiser les objets de l’application en types primitifs de Python. Les objets sérialisés peuvent ensuite être utilisés dans des formats standards telle que JSON pour une utilisation dans une API http.

L’écosystème de Python possède plein de librairies pour le formatage et la validation de schéma. En fait, Marshmallow a été influencé par beaucoup de ses librairies. Marshmallow est influencé par [Django REST Framework](https://www.django-rest-framework.org/), [Flask-RESTful](https://flask-restful.readthedocs.io/), et [colander](https://docs.pylonsproject.org/projects/colander/en/latest/). Il empreinte un certain nombre d’idées d’implémentation et de design de ces librairies et apporte une solution productive et fexible de marshalling, démarshalling et de validation de données.

Marshalling *: le fait de rassembler des données et de les transformer en un format standard avant l’envoie vers un réseau (webopedia : data marshalling)*

Front-End

TP GROUPE :

VENDEECOM



|  |
| --- |
|  |
|  |
| Créé par : Florent SANIKA |

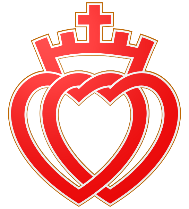


Table des matières

1. **Présentation**

* Résumé
* Cahier des charges
* Compétences couvertes par le projet :

1. **Spécifications**

* Maquette
* Partie visuelle
* Partie code

1. Conclusion

# Présentation

## Résumé

Effectuer le site web responsive d’une agence de marketing digital qui vise comme clients les TPE (Très petite entreprise) et PME (petite et moyenne entreprise).

Le but de ce site est d’augmenter la visibilité de l’entreprise et ainsi de pouvoir augmenter sa clientèle.

Il met en avant les compétences et expertises de chaque employé, mais aussi les différents articles et nouvelles de l’entreprise et les services disponibles. Il contient aussi les références des anciens clients.

A partir d’un cahier des charges, il faut effectuer un maquettage du site puis développer la partie front-end du site en HTML/CSS.

Ce site possèdera donc une page d’accueil contenant la présentation de l’entreprise et de l’équipes, une page blog contenant les articles de la boite, une page référence de client, une page contenant les services disponibles, et une page avec les informations de contact.

La maquette a été effectuée sous **Pencil**.

L’ensemble du développement est effectué sous Visual Studio Code.

Compétence couverte par le projet :

* + - Maquetter une application
    - Réaliser une interface utilisateur web statique et adaptable

## Cahier des charges

**Présentation de l’Entreprise.**

* VendéeCom est une jeune société crée par quatre développeurs full-stack. La vision de l'entreprise est de pouvoir créer le site internet de chaque client avec un suivi personnalisé. La maintenance et le référencement font partis des missions et services proposés ainsi que l'accompagnement des TPE, PME dans le digital quel que soit leur domaine d'activité.
* Le site internet de VendéeCom met en avant nos savoir-faire et nos compétences afin de donner une idée globale de ce que peut attendre un client à la recherche de la création et du référencement de son site internet.
* Date butoir pour la fin du projet : mise en ligne du site escomptée le 15/10/2020.
* Site web à créer de A à Z, nécessité de réaliser un maquettage complet et détaillé.

**Objectif** :

* Nombre de devis par mois
* Nombre de contrat par mois
* Nombre de partenaire
  + Publicité
  + Réseaux

**Cible à laquelle s’adresse le site.**

* Particuliers, auto-entrepreneur, TPE, PME etc... Les clients ne sont pas spécialement formés au web et doivent comprendre facilement nos spécificités.
* Où se trouve votre cible de manière générale :
* Internet : notre site doit être correctement référencé sur google mais aussi sur les réseaux sociaux ainsi que dans les canaux numériques départementale et régionale.
* On vise une clientèle régionale mais notre présence doit être nationale comme toute entreprise du numérique.

Les utilisateurs utilisent plus les ordinateurs que les mobiles.

**État des lieux de l’existant**

* Aucunes bases de travail existante.
* Ressources interne pour la conception, intégration.
* Récupération des mail newsletter flux rss
* Formulaire de contact : des utilisateurs via le formulaire de contact : base de données client.
* Cette page nécessite une interface graphique pour pouvoir être mise à jour par le responsable com
* Compteur de visite

Contenus présents sur le site

Optimisation pour les moteurs de recherche : optimiser le code source pour les moteurs de recherche pages par page en fonction des mots-clefs indiqués par le client = balises meta, title…

Par pages :

**A propos :**

Vidéo présentation équipe – compétence – service.

Mot clé des compétences de l’équipe SEO

**Expérience**

Mot clé différentes technologies maîtrisées

Liste partenaires/salariés

Temps dans la boite

**Actualité :**

Dernier partenariat publique, optimisation de la boite, nouvelle technologie maîtrisé…

Communication ciblé / mot clés SEO

**Référence :**

Liste - lien vers les sites des partenaires/clients

Liste des étapes de chaque projet

**Contactez-nous :**

Formulaire de communication

FAQ

Mail/téléphone

**Contraintes techniques**

* Il s’agit ici d’informer le prestataire sur le périmètre technique escompté :
* Difficulté : créer du dynamisme dans la page.
* Technique de rédaction, intégration : HTML, CSS, PHP, pas de CMS,
* Pas de transfert de compétences au client à la livraison

* Navigateur à cibler : Chrome
* Récupération des mails des utilisateurs via le formulaire de contact : base de données client.

Prestations attendues & modalités de sélection du prestataire

**Prestations attendues**

1. Domaine : [www.vendeecom.fr](http://www.vendeecom.fr/)
2. Le développement : **à charge de l’équipe de développement web.**
3. Le design, la recherche de logo : **à charge de l’équipe de développement web.**
4. L’optimisation au référencement naturel : **à charge de l’équipe de développement web.**
5. Base de donnes **: à charge de l’équipe de développement web**.
6. La maintenance à l’année du site : **à charge de l’équipe de développement web.**

**Planning & suivi du projet**

***15 octobre 2020 : fin du développement et mise en place du site en ligne.***

1. Maquette via **Pencil, gestion via Trello**

## Compétences couvertes par le projet

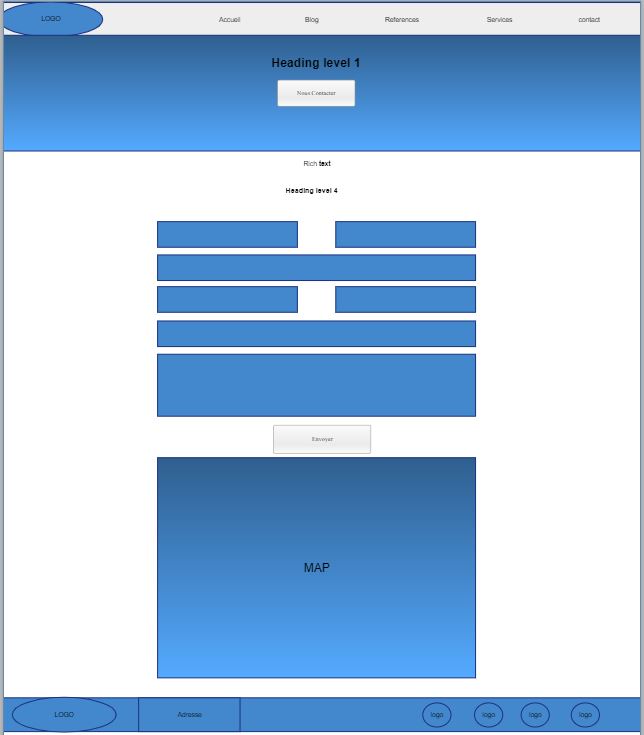
* Maquetter une application
* Réaliser une interface utilisateur web statique et adaptable

# Spécifications :

Le site est constitué de cinq pages : une page d’accueil, une page blog, une page références, une page services et une page contact.

### Maquette

La maquette a été réalisé sous Pencil. Le travail de maquettage a été séparé entre nous quatre. Je me suis occupé de la partie contact que voici :



Ce sont les différents champs à entrer pour le formulaire de contact.

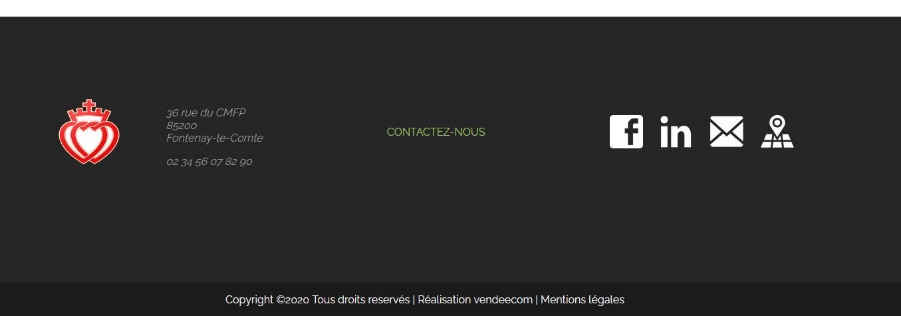
### Partie visuelle

Partie commune à toutes les pa ges :

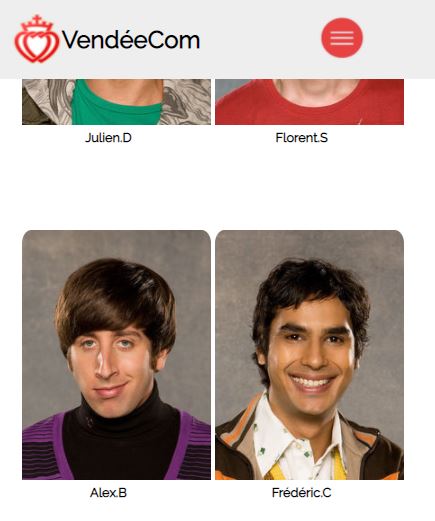
Header



footer

Page d’accueil :



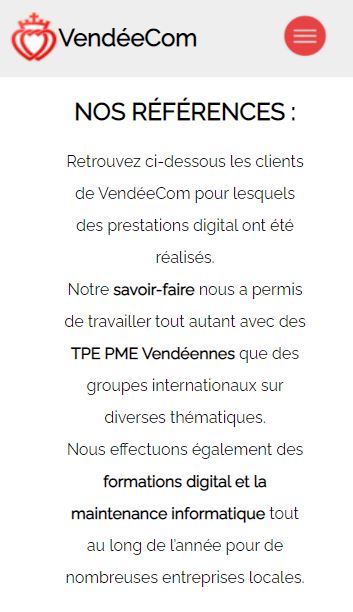
Page blog :



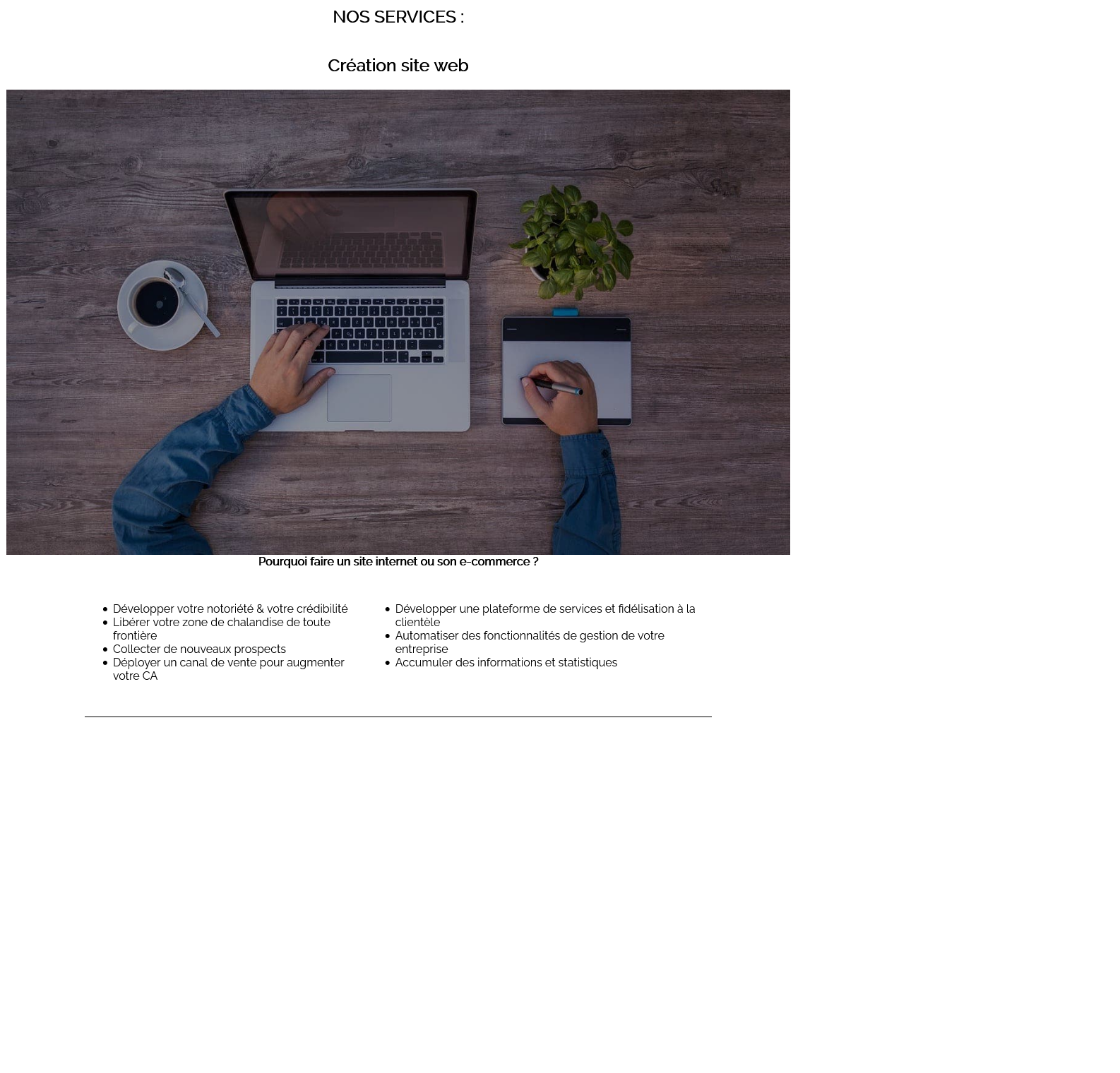


Une page références :



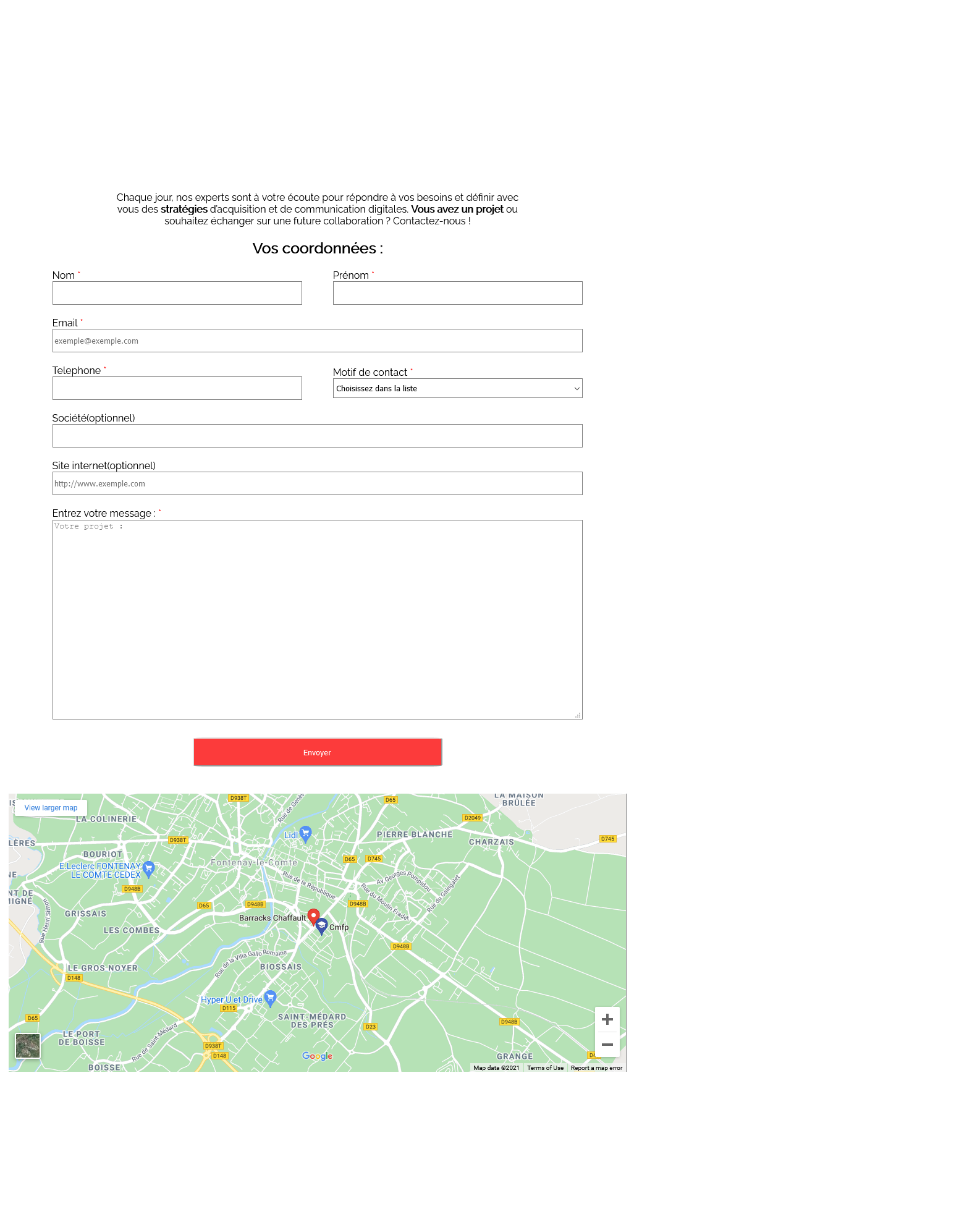
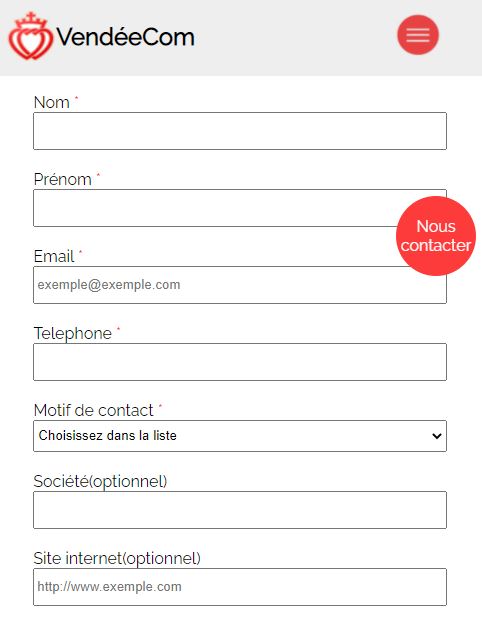


Une page services :





Une page contact :



### Partie code



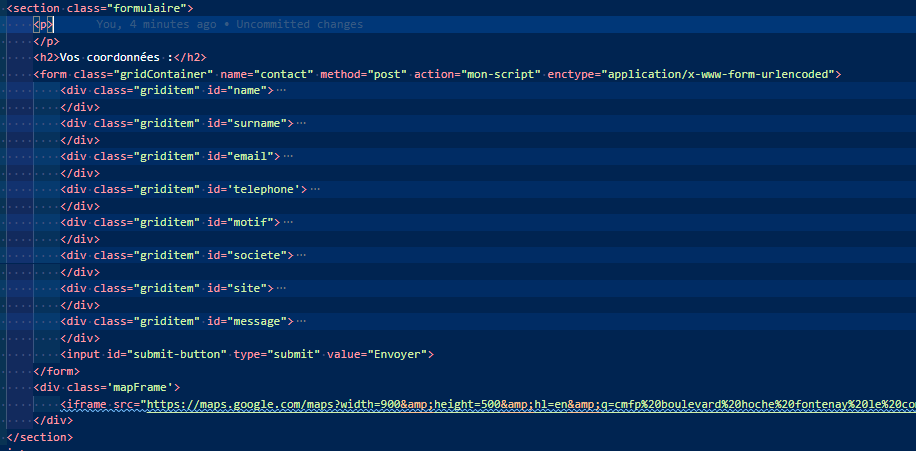
Voici l’architecture générale du code de projet. Les fichiers html sont à la racine. Un seul fichier CSS est commun pour toutes les pages. Chacun d’entre nous s’occupait du style pour sa page et je me suis occupé de l’intégration des différentes pages du projet.

Les pages ont été passés sur le site w3 validator.

Nous avons commencé par faire un header et un footer commun à toutes les pages.

Ensuite nous nous sommes partagés les tâches pour la réalisation du site.

J’étais en charge de la page de contact.



Pour le style responsif, j’ai choisi d’utiliser les grids layouts plutôt que le flex qu’on nous avait appris jusque-là.

Pourquoi :

J’ai donc fait des recherches pour savoir dans quels cas utiliser des flexbox et dans quels cas utiliser des grids.

Source : https://css-tricks.com/things-ive-learned-css-grid-layout/

Grid is designed to be used with flexbox, not instead of it

Grid and flexbox can act in similar ways. You may have seen people using flexbox to construct grid systems but that’s not what flexbox was designed for. It’s worth reading Jake Archibald’s blog post [Don’t use flexbox for overall page layout](https://jakearchibald.com/2014/dont-use-flexbox-for-page-layout/) :

1. Flexbox is for one dimensional layout (row or column).
2. CSS grid is for two dimensional layout.

Or as Rachel Andrews [put it](https://rachelandrew.co.uk/archives/2016/03/30/should-i-use-grid-or-flexbox/):

« *Flexbox is essentially for laying out items in a single dimension – in a row OR a column. Grid is for layout of items in two dimensions – rows AND columns*. »

They can be combined as well. You can turn a grid item into a flex container. You can turn a flex item into a grid.

Let’s take one useful example. We want to vertically center the text inside a grid item but we want the background (whether a color, a gradient or an image) to cover the items entire grid area. We can use the align-items property with a value of center but now the background doesn’t fill the whole area of the item. The default for align-items is stretch — as soon as you set it to any other value, it no longer fills the space. Let’s instead leave it set to stretch and turn our grid item into a flex container.

**Traduction** :

Grid et flexbox peuvent agir similairement. Vous avez peut-être vu des gens utiliser des flexbox pour construire des systèmes de grids mais ce n’est pas ce pour quoi les flexbox ont été créés. Ul peut être intéressant de lire le blog publié par Jake Archibald [*Don’t use flexbox for overall page layout*](https://jakearchibald.com/2014/dont-use-flexbox-for-page-layout/) :

1. Les flexboxs sont pour les dispositions uni-dimensionnelles (lignes ou colonnes)
2. Les grids sont pour les dispositions en 2 dimensions.

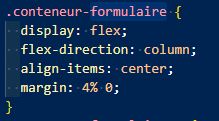
Ou comme l’a dit Rachelle Andrews :

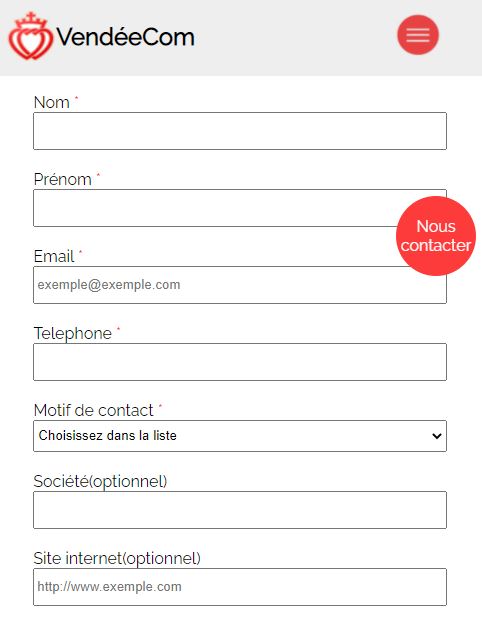
« *Le flexbox sert essentiellement à disposer les éléments sur une seule dimension – sur une ligne OU sur une colonne. Le grid sert à la disposition d’éléments en 2 dimensions – lignes ET colonnes.* »

Ils peuvent aussi être combinés. Vous pouvez convertir un élément grid en conteneur flex. Vous pouvez convertir un élément flex en [conteneur] grid.

Prenons un exemple utile. Nous voulons centrer verticalement le texte à l’intérieure d’un élément grid mais nous voulons que l’arrière-plan (que ce soit une couleur, un dégradé ou une image) couvre tout l’aire de l’élément grid. On peut utiliser la propriété align-items avec la valeur center mais maintenant l’arrière-plan ne remplit plus tout l’air de l’élément. La valeur par défaut d’align-items est stretch – dès qu’on le règle sur une autre valeur, il ne remplit plus l’espace. On peut, à la place, le laisser sur stretch et convertir notre élément grid en conteneur flex.

Dans notre cas, le conteneur du formulaire est une flexbox, et le formulaire est une grid layout :





Associer aux requête @media, les grids permettent

à la page de s’adapter à la taille de l’écran.

# Conclusion :

Durant ces projets j’ai réalisé :

* Une API REST sous Python en utilisant le micro-framework Flask, l’outil SQLAlchemy pour la base de données. Cette application contient un CRUD + 2 méthodes supplémentaires pour modifier le statut des inscrits.
* Un site statique et adaptable en HTML/CSS.

Ce stage m’a permis de me rendre compte que je n’étais pas prêt pour le monde de l’entreprise mais que je voulais continuer dans cette branche. Je souhaite de plus me spécialiser en Python. Je compte donc continuer mes études vers une formation en alternance.