Programmation Web

*Plane Shop*





Berdah Kevin

Ghenai Younes

4SI2 ESIEA 2018

Table des matières

I. Introduction……………………………………………….3

1. Qu’est-ce que Plane Shop ? ……………………3

2. Cahier des charges………………………….3

II. Conception du projet…………………………..

1. Réalisation………………………….4

2. Déploiement………………………………4 à 6

III. Planification du projet………………………………………

1. Planning………………………………….7

2. Difficultés rencontrées…………………………….7

IV. Conclusion……………………………………………….8

V. Annexe……………………………………………………8

I. Introduction

1. Qu’est-ce que Plane Shop ?

Plane Shop est un site web dédié au secteur de l’aviation.

Il est possible sur Plane Shop, pour les plus riches d’entre vous, d’acheter divers avions selon vos envies.

Sur notre site il est possible de choisir entre plusieurs marques d’avions :

* Airbus
* Boeing
* Dassault Aviation

L’affichage de Plane Shop est adaptatif selon vos écrans et le type de support, c’est à dire responsive, il s’affiche d’autant bien sur un téléphone que sur une tablette ou bien un écran d’ordinateur quelle que soit sa résolution.

2. Cahier des charges

Pour ce projet web, nous avions reçu plusieurs requêtes de la part du client, nous devions développer notre projet en HTML5, CSS3 et JavaScript. L’utilisation de Node.js et de Vue.js a été une des contraintes du projet.

Une fois le site créé, Node et Vue implémentés, nous devons déposer notre projet sur https://github.com, puis grâce à notre dépôt, héberger notre site sur <https://glitch.com/> à fin qu’il soit visible sur internet.

II. Conception du projet

**1. Structure**

Dès le commencement, nous avons réalisé une maquette sur ce à quoi notre site pourrait ressembler, nous nous sommes tout de suite mis d’accord pour cette forme :

* Une barre de navigation : fixée en haut de notre page.
* Un bloc unique central : affichage des produits.
* Un bas de page : centré en bas du site contenant quelques informations.

Nous avons rapidement avancé sur la partie minimale du contenu HTML, c’est-à-dire une barre de navigation, un bas de page ainsi qu’un bloc pour les articles, nous ne voulions pas nous attarder là-dessus sachant l’ampleur du travail que nous avions à fournir. De même pour le CSS nous l’avons terminé pratiquement en un jour, et en rajoutant quelques éléments au fur et à mesure que nous ajoutions du contenu HTML. Nous nous sommes ensuite renseignés d’une part sur le framework Vue.js puis sur Node.js lorsque le site était viable avant de mettre en place le serveur.

Concernant le Vue.js, il nous était alors inconnu au départ. Mais en lisant la documentation et en l’implémentant petit à petit, nous avons saisi la grande utilité de ce framework. En effet, une fois le Vue.js implémenté, nous évitions ainsi d’avoir à réaliser plusieurs fichiers HTML correspondant au nombre de pages du site. Toutes ces pages étaient donc factorisées dans un seul et même fichier HTML. Enfin, une fois l’environnement Node.js implémenté, notre serveur étant ainsi fonctionnel, nous avons pu intégrer une plateforme de connexion pour le client.

**2. Le code**

Pour ce qui est du détail de la partie technique, elle se décompose de la manière suivante. Nous disposons de deux tableaux d’objets, présents côté client et côté serveur (quatre au total). L’un concerne les produits mis en vente, avec son nom, son prix, son stock, son image ainsi qu’une description. L’autre sert de référence au panier, contenant également des objets parallèlement à la liste des produits.

Le bloc central présent sur la page de chaque marque d’avion est constitué de cadres. Ces cadres font référence à chacun des items de la liste des produits. Il y a autant de cadres qu’il n’y a de produits. Nous indiquons également au site de différencier et classer chaque produit dans une page différente en fonction de sa marque. Ainsi, les Airbus se retrouvent sur une même page, les Boeing sur une autre, et les Dassault sur une autre encore.

Pour chaque produit, le client peut incrémenter sa quantité dans le panier, en cliquant sur le bouton « Commander ». Le stock est alors mis à jour côté client et côté serveur. A chaque clic sur le bouton, une requête est envoyée au serveur pour vérifier que le stock est toujours positif. Lorsque le stock de l’article est épuisé, la page envoie une alerte au client lui indiquant que tel ou tel produit est en rupture.

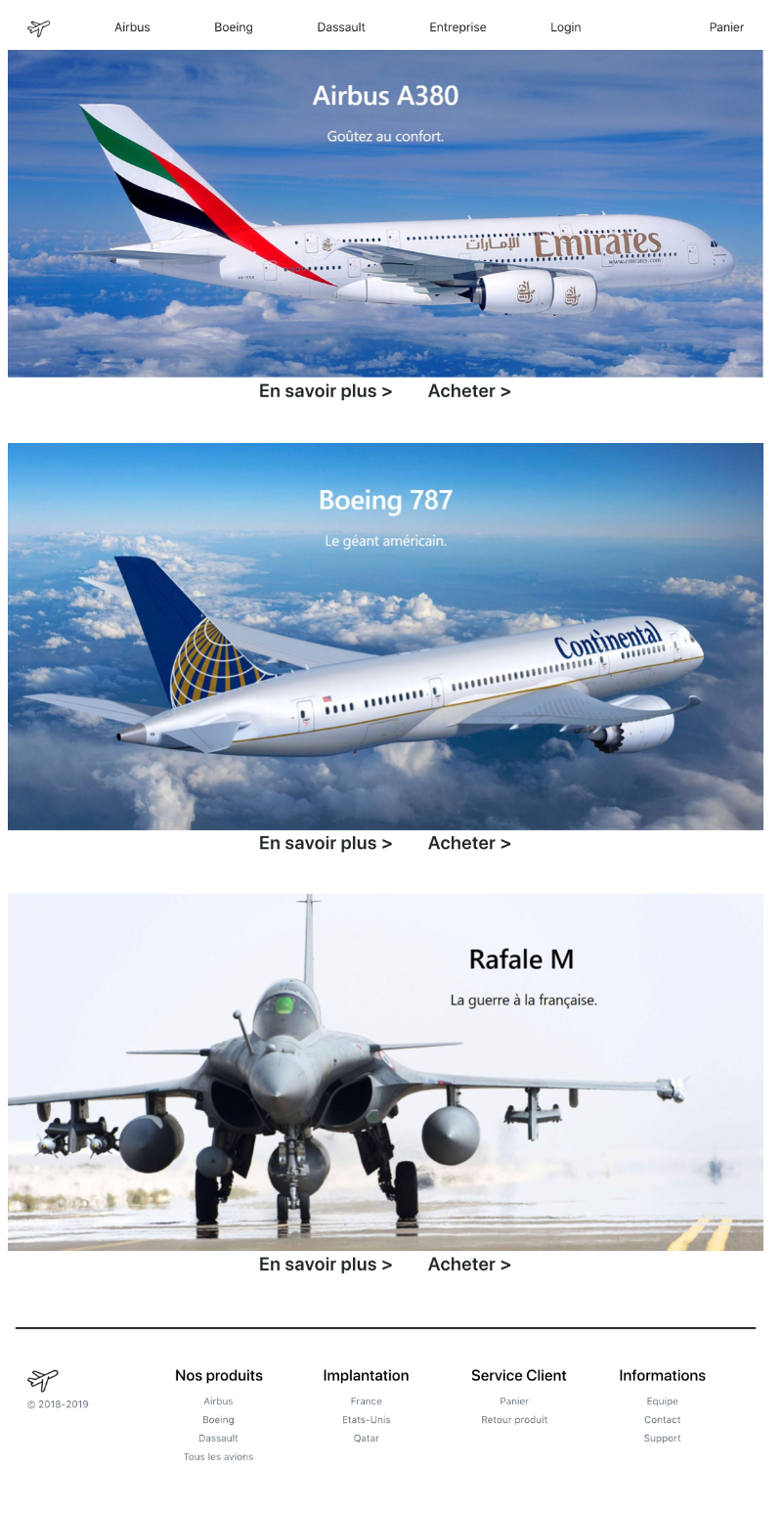
Une fois ses achats terminés, le client souhaite valider son panier pour payer. Il est alors impératif pour lui de se connecter s’il a déjà un espace personnel, ou de s’inscrire en tant que nouveau client. Ainsi, à chaque tentative de connexion, le serveur vérifie dans sa liste d’utilisateurs (dans un fichier JSON) si le mail ou le mot de passe correspondent à un utilisateur connu, auquel cas il aura accès à un bouton « Valider et payer » dans le panier pour finaliser sa commande.

S’il s’agit de son premier achat sur Plane Shop, le client entre ses informations pour pouvoir s’inscrire. Le serveur va alors les récupérer et les écrire dans le fichier JSON, pour que le client soit reconnu à sa prochaine tentative de connexion. Une fois enregistré, le client est redirigé vers son panier pour valider son achat, puis une fois cela fait, il est encore une fois redirigé vers une page confirmant que son paiement a été accepté.

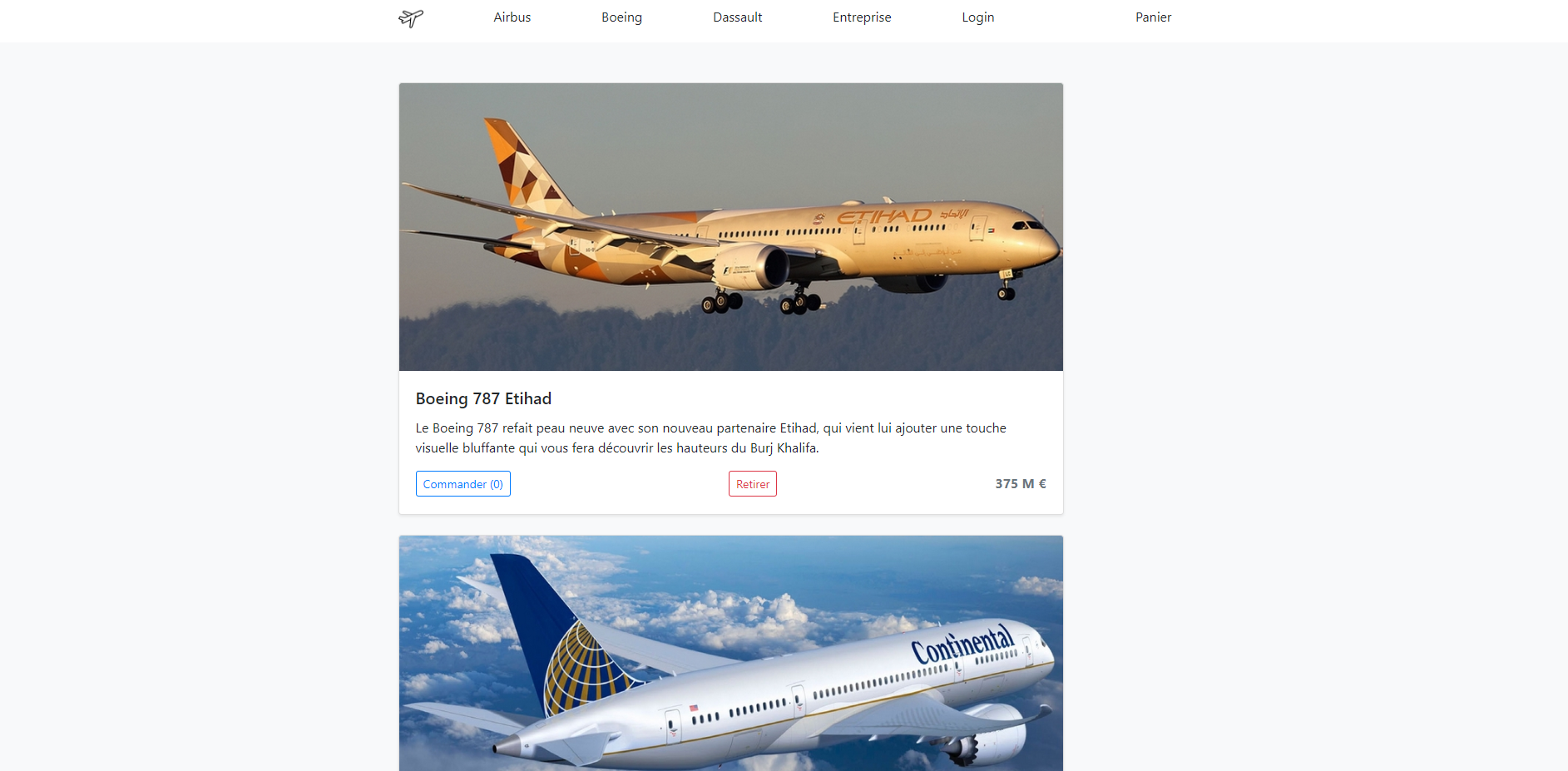
L’avantage d’écrire les utilisateurs dans un fichier JSON réside dans la sauvegarde des données dans le cas où notre serveur est éteint, chaque modification sera donc enregistrée.

**2. Déploiement**

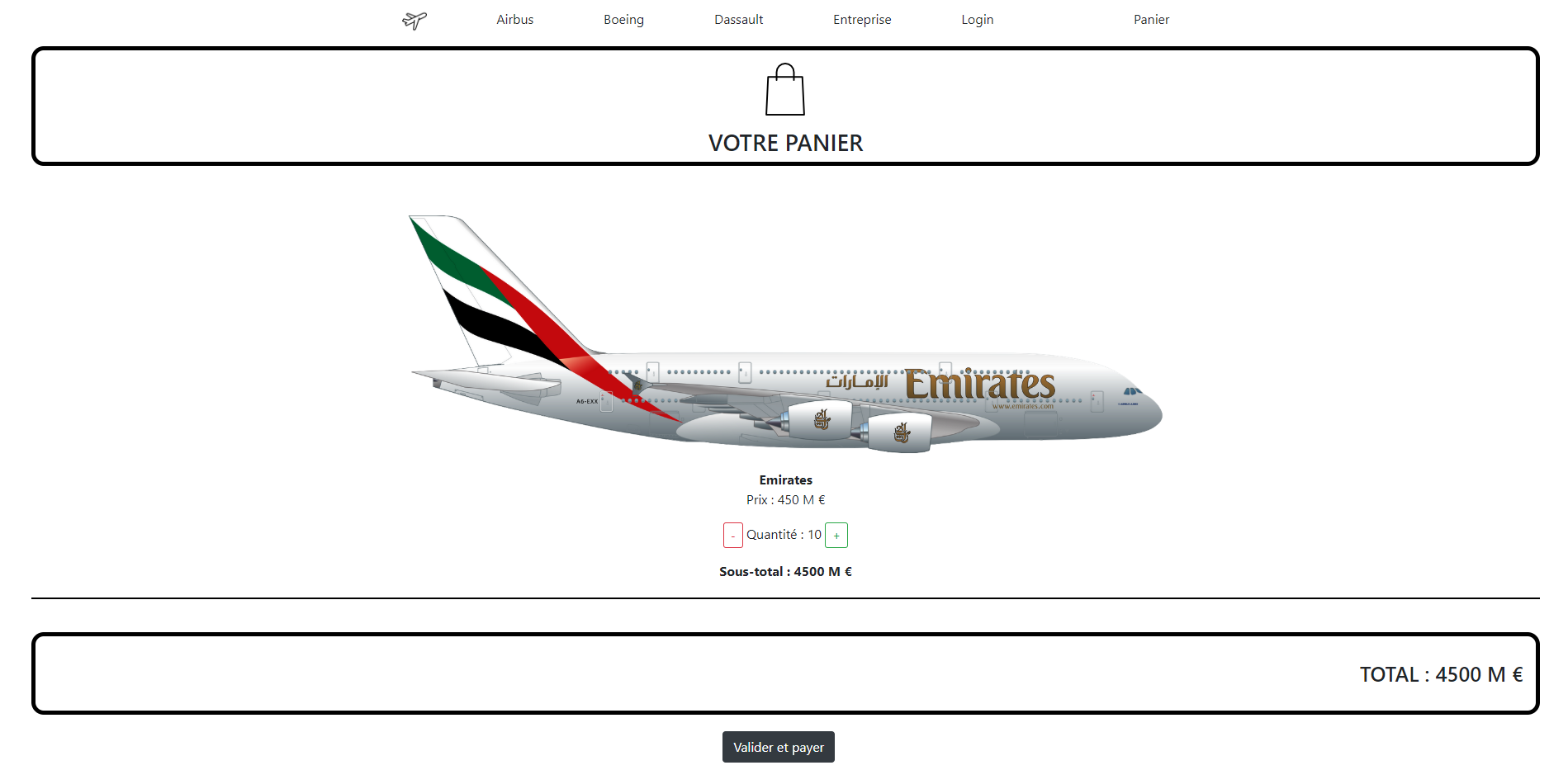
Au fur et à mesure que le projet se dessinait et se complexifiait, nous mettions régulièrement à jour notre projet sur Github à fin d’être synchronisés. Cette façon de faire nous était utile, puisque nous pouvions faire une copie pour la modifier, sans pour autant écraser la version antérieure. Ainsi, si un problème majeur devait survenir, telle qu’une perte des fichiers ou une modification trop conséquente (et donc pratiquement irréversible) du code, nous aurions toujours une version sûre.

Enfin, la publication de notre site sur l’hébergeur a été rapide et intuitive, et nous pouvons en voir le résultat final :

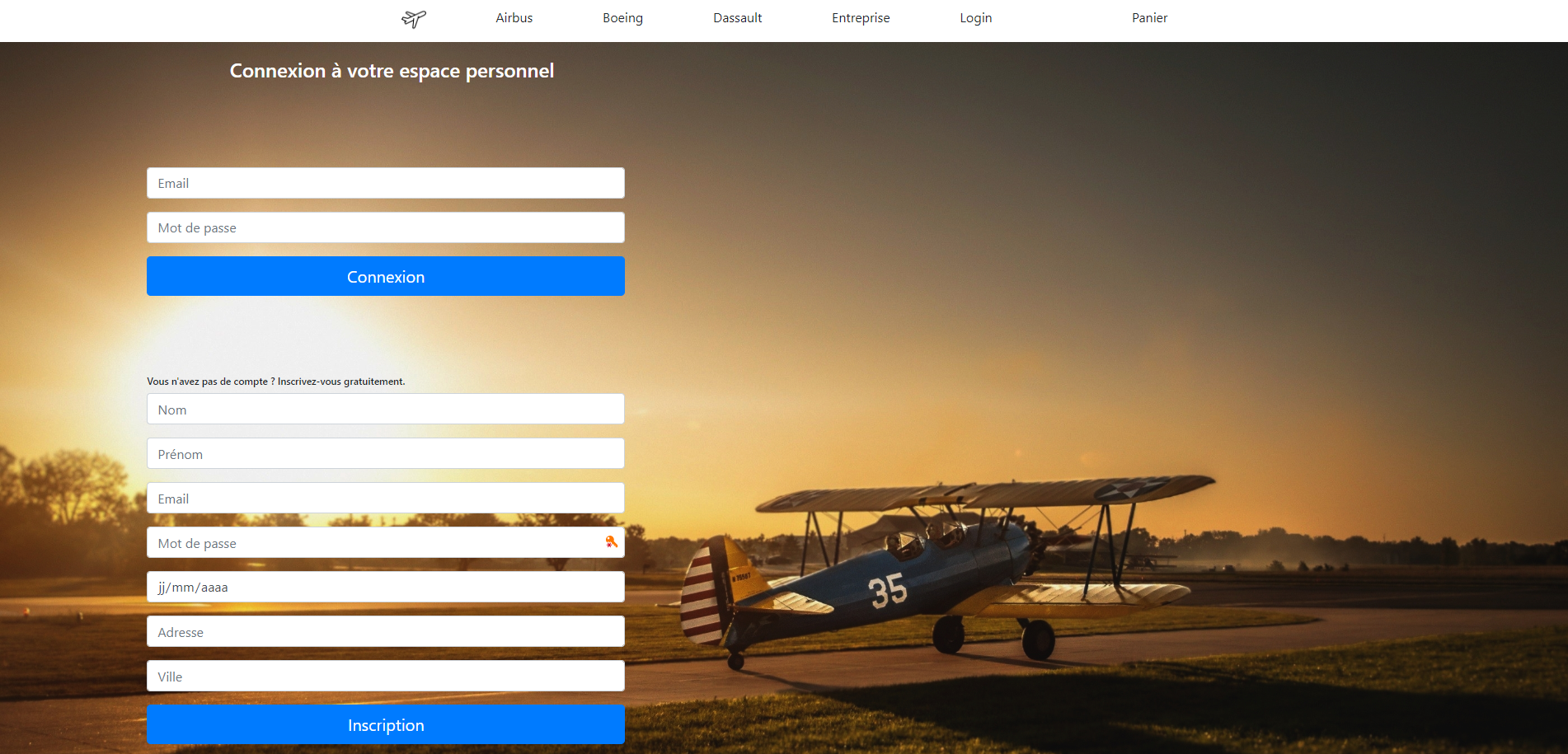
**Page d’accueil Plane Shop**



**Page des produits**



**Panier de commandes**



**Connexion à l’espace personnel**

III. Difficultés rencontrées

nous avions un peu plus d’une dizaine de pages html. Toutes les pages html étaient reliées entre elles via le menu de notre site, mais avec l’obligation d’utiliser Vue.js, cela nous a permis de réduire ce nombre de page à une seule unité. En effet grâce à **Vue.js**, une seule page index.html existe, aucune autre.

En comparaison, sans l’utilisation de Vue.js, chaque page est chargée lorsque l’on clique sur celle-ci, alors qu’avec l’utilisation de Vue.js, lors du chargement de la page d’accueil, cela prend un peu plus de temps, mais il n’y a plus de chargement de pages durant toute la durée de notre navigation.

L’apprentissage de Vue.js a été long, en effet nous ne savions pas par où commencer, nous avons suivi de nombreux tutoriels, qui souvent ne menaient pas à la fonctionnalité souhaitée, mais à force de persévérance nous avions réussi. Avec un peu de recul ce n’est vraiment pas compliqué, surtout qu’on savait ce qui n’allait pas tarder : Le panier, Node.js, puis le login.

Concernant **Node.js**, on appréhendait beaucoup son implémentation, ce fut le cas. Mais après plusieurs jours d’apprentissage, de lecture de documentation, de tutoriels, de vidéos sur Youtube, et l’aide guidée du professeur, nous sommes parvenus à mettre notre projet sur un serveur local.

Nous avons rencontrés pas mal de difficultés quant au panier, en effet quand on ajoutait des produits au panier à partir de la page produit, le panier s’incrémentait bien, le stock décrémentait bien, mais dès lors qu’on modifiait la quantité voulue à partir du panier, le stock revenait à son origine. Nous avions réussi à régler ce problème, en effet on utilisait une seule variable pour deux fonctionnalités (stock, quantité).

**Login => lorsque le client s’inscrit, il est bien connecté, il peut valider son panier. Mais lorsqu’il tente de se reconnecter, la fonction login() ne lit pas le fichier JSON.**

IV. Conclusion

Nous sommes très satisfaits de ce projet, en effet, sachant coder en html et en css, nous pensions que réaliser un site web était si simple et rapide, mais nous ne connaissions pas toutes les fonctionnalités qui sont « cachées », tels que le framework Vue.js.

Nous avions eu carte blanche sur le résultat final de notre projet, nous avons réalisé ce site en binôme, de manière autonome, ce qui nous a particulièrement plu.

Nous n’avons pas baissé les bras du début à la fin, et nous sommes plutôt fiers du résultat final de notre site.

V. Annexes

**Informations complémentaires :**

Un client a déjà passé commande, il a accepté de nous laisser ses identifiants pour faciliter la connexion :

* Email : younes@gmail.com
* Mot de passe : younes

**Bibliographie :**

- Github de l’énoncé : https://github.com/Strift/esiea-web-programming

* Github du projet : https://github.com/Younes-ghenai/Younes\_Kevin\_PlaneShop
* Glitch du projet : https://plane-shop.glitch.me

- Etherpad : https://etherpad.net/p/esiea-4a-web-fsi2

- Guide VueJS : https://vuejs.org/v2/guide/

- Guide NodeJS : https://nodejs.org/en/docs/