

Dossier de projet

M. Maxime Cottin

Refonte du site web de la société
Retz Tactical Games
avec interface administrable



Période de formation : du 02 août 2021 au 15 avril 2022

Formateur : M. Yohann Moy

Mme Mangé Cannarozo Géraldine & Mme Moukala Flores Graziela

8 rue Mayenne 02200 Soissons

03 23 54 70 95

Sommaire :

Sommaire :	3
Remerciements :	5
Résumé du projet :	6
Cahier des charges :	7
Présentation de l'entreprise :	7
Présentation du projet :	7
Objectifs du site :	7
Méthodologie du suivi :	7
Liste des compétences couvertes par le projet :	8
Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité :	8
Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité :	9
Spécifications techniques du projet :	10
Framework utilisés :	10
Librairies utilisées :	10
Outil back-end :	10
Outils utilisés :	11
Réalisations :	12
Extraits de code front-end commentés :	12
Extraits de requêtes back-end commentées :	14
Jeu d'essai :	15
Description d'une situation de travail ayant nécessité une recherche à partir d'un site anglophone :	16
Suite du projet - Réservation de partie :	18
Suite du projet - Créations customisées :	19
Annexe : Road map du projet :	20
Annexe : Structure d'un projet NextJS :	21
Annexe : Interfaces Prismic :	22
Annexe : Maquette mobile du site :	24
Annexe : Visuels du site :	25

Remerciements :

Je tiens à remercier toutes les personnes qui ont contribué au succès de ma formation et qui m'ont aidé lors de la rédaction de mon dossier de projet.

Tout d'abord, j'adresse mes remerciements à mon formateur, Mr MOY Yohann qui m'a beaucoup aidé dans la création de ce projet afin de cibler les priorités et convenir aux compétences demandées.

Je tiens à remercier mon maître de stage durant les 3 périodes en entreprise Mr BOEDEC Gaetan pour m'avoir fait découvrir différentes façon de travailler notamment avec NextJS au travers de sa société. Du temps qu'il aura passé avec moi et du partage de son expérience et son expertise au quotidien.

Enfin je tiens à remercier toutes les personnes qui m'ont conseillé et relu lors de la rédaction de ce dossier de projet.

Résumé du projet :

La projet présenté dans ce dossier est aujourd'hui un projet factice. Il a pour but final d'être présenté à la société quand il atteindra son fonctionnement complet (voir la road-map en annexe en page 19).

La société "Retz Tactical Games" à besoin de mettre à jour sa présence sur le web suite à une croissance et une diversification des services proposés. J'ai placé la priorité sur la mise en place d'un catalogue des produits proposés. Il a été décidé de faire un site avec une interface utilisable par le client de manière à pouvoir mettre à jour facilement le contenu du site.

J'ai donc opté pour une utilisation d'un CMS Headless en back-end qui offrira une interface utilisateur accessible au client afin de pouvoir ajouter, supprimer ou modifier du contenu facilement. Et côté client l'utilisation d'un framework qui ne nécessitera pas de maintenance dès qu'il y aura une modification sur le CMS avec un code qui s'adapte au contenu présent sur le système de gestion de contenu.

Plusieurs autres spécificités du site sont en cours de conception tel que la possibilité de commander ou réserver des produits sur le site. La possibilité de réserver des places pour une partie avec une vue sur un calendrier. La possibilité de créer des répliques customisées avec un visuel qui se met à jour ou de choisir des customisations pré-crées.

Project summary :

The project presented in this file is today a dummy project. Its final goal is to be presented to the company when it will reach its complete functioning (see the road-map in annex page 19).

The company "Retz Tactical Games" needs to update its presence on the web following a growth and a diversification of the proposed services. I placed the priority on the implementation of a catalog of the proposed products. It was decided to make a site with an interface usable by the customer so as to be able to easily update the content of the site. I therefore opted to use a Headless CMS on the back-end which will offer a user interface accessible to the client in order to add, delete or modify content easily. And on the client side the use of a framework that will not require maintenance as soon as there is a modification on the CMS with a code that will adapt to the content present on the content management system.

Several other specificities of the site are being designed such as the possibility to order or reserve products on the site. The ability to reserve seats for a game with a calendar view. The ability to create custom replicas with a visual that updates or to choose pre-created customizations.

Cahier des charges :

Présentation de l'entreprise :

La société "Retz Tactical Games" est une société de vente et location de matériel d'airsoft et de paintball. Ils proposent également un service de personnalisation de répliques sur mesure. Ils organisent aussi des parties quasiment chaque week-end sur leur terrain.

Présentation du projet :

Ce projet est mis en place afin de renforcer la présence sur le web de la boutique "Retz Tactical Games" grâce. Dans un premier temps ce sera un site vitrine proposant toutes les répliques disponibles en magasin. Ainsi qu'une autre catégorie permettant de voir les services proposés lors des organisations de parties.

Dans un dernier temps, il y aura un outil conçu pour créer en temps réel des customisations de répliques et ainsi pouvoir faire des demandes personnalisées.

Objectifs du site :

La mise en avant du catalogue avec, sur le long terme, la possibilité d'acheter ou réserver des articles sur le site.

La possibilité de voir sur un calendrier les dates des parties futures et y réserver un emplacement avec la formule désirée.

La création d'un outil permettant la customisation de matériel avec rendu en temps réel ainsi qu'un aperçu du prix final (à titre indicatif, une customisation demandant plus ou moins de travail suivant le matériel choisi).

Méthodologie du suivi :

Le projet étant un projet que je réalise seul et n'ayant pas de date limite, j'ai créé un fichier simple qui répertorie les différentes fonctionnalités du site ainsi que les technologies qui sont prévues pour chacune de ces fonctionnalités.

Liste des compétences couvertes par le projet :

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité :

Maquetter une application :

- La maquette prend en compte les spécificités fonctionnelles décrites dans les cas d'utilisations ou les scénarios utilisateurs.
- La maquette est conforme à l'expérience utilisateur et à l'équipement ciblé.
- La maquette respecte les principes de sécurisation d'une interface utilisateur.
- Le contenu de la maquette, pour la partie visible, est rédigé, en français ou en anglais, de façon adaptée à l'interlocuteur et sans faute.

Réaliser une interface utilisateur web statique et adaptable :

- L'interface est conforme à la maquette de l'application.
- Les bonnes pratiques de structuration sont respectées y compris pour le web mobile.
- Les pages web s'adaptent à la taille de l'écran.
- Le site respecte les règles de référencement naturel.
- La démarche de recherche permet de résoudre un problème technique ou de mettre en œuvre une nouvelle fonctionnalité.
- La documentation technique liée aux technologies associées, rédigée en langue anglaise, est comprise (sans contre-sens, ...).

Développer une interface utilisateur web dynamique :

- Les pages web sont conformes à l'expérience utilisateur y compris pour l'expérience mobile.
- L'architecture de l'application répond aux bonnes pratiques de développement et de sécurisation d'application web.
- L'application web est optimisée pour les équipements mobiles.
- La démarche de recherche permet de résoudre un problème technique ou de mettre en œuvre une nouvelle fonctionnalité.
- La veille sur les vulnérabilités connues permet d'identifier des failles potentielles.
- La documentation technique liée aux technologies associées, rédigée en langue anglaise, est comprise (sans contre-sens, ...).

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité :

Élaborer et mettre en oeuvre des composants dans une application de gestion de contenu ou e-commerce :

- Les composants complémentaires ou réalisés s'intègrent dans l'environnement de l'application.
- Les composants serveur contribuent à la sécurité de l'application.
- Le code source des composants est documenté ou auto-documenté.
- Les tests garantissent que les traitements serveurs répondent aux fonctionnalités décrites dans le cahier des charges.
- La démarche de recherche permet de résoudre un problème technique ou de mettre en oeuvre une nouvelle fonctionnalité.
- La veille sur les vulnérabilités connues permet d'identifier des failles potentielles.

La documentation technique liée aux technologies associées, rédigée en langue anglaise, est comprise (sans contre-sens, ...).

Spécifications techniques du projet :

Framework utilisés :

NextJS :

NextJS est un framework basé sur ReactJS orienté Server Side Rendering (création de page web par le serveur qui les envoie compilées au client). Ce framework rassemble le côté front-end et back-end au même endroit. Son principal inconvénient se situera au niveau du référencement. Dans le cas présent, le projet est destiné à rester local et la visibilité se fait premièrement au travers des réseaux sociaux donc cela ne gêne pas.
(voir la structure d'un projet NextJS en annexe page 20)

Librairies utilisées :

Apollo GraphQL :

Apollo GraphQL est une suite d'outils open-source permettant de créer des requêtes et interpréter le résultat de ces dernières au format JSON. On peut ensuite utiliser ce résultat dans notre code. Cet outil est conçu pour fonctionner avec les frameworks les plus populaires tels que React, Vue et Angular.

SwiperJS :

SwiperJS est une librairie permettant la création de carrousel pour les applications web et mobiles. Cette librairie donne la possibilité de créer rapidement et simplement des galeries d'images avec de nombreuses fonctionnalités telles que le choix de slide par vue, le type de navigation désiré, et d'autres options.

Outil back-end :

Prismic :

Prismic est un CMS dit Headless (uniquement sur la partie back-end) qui permet la création de modèle de données. Il offre également la possibilité d'optimiser les images suivant les différents périphériques d'affichages. Il dispose également de son propre serveur GraphQL qui déploie une documentation propre à chaque projet et permet la création et les tests de chaque requête.

(voir les interfaces Prismic en annexe page 20)

Outils utilisés :

Visual Studio Code :

Pour programmer mon projet j'ai utilisé l'éditeur de code Visual Studio Code car il offre un interface très accessible avec de nombreuses fonctionnalités qui améliore l'ergonomie de travail.

De plus j'y ai ajouté quelques extensions afin de me faciliter la tâche sur le projet comme par exemple :

- Bracket Pair Colorizer : Permet de colorer les parenthèses, accolades et crochets ensembles afin de ne pas confondre ces dernières quand on a de longues portions de codes imbriqués.
- Git Lens : Permet de voir le nom du dernier commit sur un fichier
- Prettier : Permet de formater le code proprement pour éviter les erreurs d'indentations et garder les mêmes espacements ou logiques de présentations au travers de tous les fichiers.

Figma :

Pour créer les maquettes du site j'utilise Figma car il est utilisable sur toutes les plateformes. Il est capable de lire les fichiers venant des plateformes concurrentes et également il est facile de faire des rendus avec les animations et les interactions que l'on aura sur le site final. Ce qui permet de présenter un rendu plus proche de la réalité au client.
(voir la maquette en annexe page 23)

Notion :

Pour faire le suivi de mon projet, y annoter toutes les idées plus ou moins claires ou encore recenser tous les liens utiles à la création de ce dernier j'utilise Notion. Avec tous les templates et les possibilités de présentations des ressources ça en fait un outil très puissant. De plus il est facilement accessible sur smartphone via un lien, ce qui permet de l'alimenter facilement en toute circonstances.

Réalisations :

Extraits de code front-end commentés :

```
interface PageArticleDetailsProps {
  replique: any;
  uid: any;
}

const PageArticleDetails = ({ replique }: PageArticleDetailsProps) => {
  return (
    <main>
      <Head>
        <script
src="https://kit.fontawesome.com/ec5d791fc6.js"></script>
        </Head>
        <Header isHomePage={false} />
        <TabBar />
        <section>
          <h1>Nos répliques</h1>
          <div>
            <ItemFull
              itemGallery={replique.node.gallery}
              itemName={replique.node.name}
              itemPrice={replique.node.price}
              itemDescription={replique.node.product_description}
              itemContent={replique.node.product_content}
            />
          </div>
        </section>

        <Footer isScroll={true} />
      </main>
    );
  };
};
```

Extrait de code gérant l’affichage des pages de chacune des répliques présentes sur le site.

On commence par typer les différentes variables que l’on utilise dans l’interface.

Ici l’objet ‘replique’ qui fait référence à une instance d’article de type *Replique* dans le CMS et l’objet ‘uid’ qui sert ici dans une fonction hors de cet extrait créant l’url de la page.

La fonction *PageArticleDetails* est le début de la structure de création de page qu'utilise NextJS dans laquelle on commence par insérer la ou les variables externes (présentement on appelle la variable 'replique'). Entre accolades et avant le *return* on peut créer des fonctions qui feront varier le rendu de la page. Dans mon cas j'ai choisi de gérer ces éventualités directement dans les composants concernés comme pour les composants *Header* et *ItemFull*.

Chacune des pages fait référence à des composants que l'on importe en début de fichier et est appelé sous forme de balise avec comme attributs toutes les variables les concernant et qui affectent l'affichage final.

L'exemple le plus simple ici est le *Header* avec la variable *isHomePage*. J'ai utilisé l'affichage conditionnel que propose NextJS afin de créer un seul header qui affichera ou non la bannière avec le slogan avec un simple booléen.

Extraits de requêtes back-end commentées :

```
import { PrismicLink } from "apollo-link-prismic";
import { ApolloClient, InMemoryCache } from "@apollo/client";

export const clientGraphQL = new ApolloClient({
  link: PrismicLink({
    uri: "https://ras-website.prismic.io/graphql",
  }),
  cache: new InMemoryCache(),
});
```

```
export const queryHomePage = gql`
  query HomePage {
    allHomepages {
      edges {
        node {
          cover
          presentation
          video_link
        }
      }
    }
  }
`;
```

Extrait de code de la requête GraphQL appelant le contenu de la page d'accueil

Ce code est en deux parties mais ces deux parties sont complémentaires. Simplement elles sont divisées en deux fichiers différents pour des raisons d'organisation des fichiers.

Le premier fichier utilise la librairie Apollo uniquement la partie client car mon serveur GraphQL est déjà hébergé sur le CMS.

J'importe donc 2 dépendances, *ApolloClient* et également *PrismicLink* qui sert spécialement avec le CMS afin d'extraire les données du serveur GraphQL. La dernière dépendance *InMemoryCache* n'est pas obligatoire, elle sert pour ajouter des options spécifiques à la réception de la requête mais il est préconisé de l'ajouter même si on n'y ajoute pas d'options comme c'est le cas ici. Pour terminer, *PrismicLink* prends simplement en paramètre le lien du projet hébergé sur le CMS et vient s'inclure dans la structure de la fonction *ApolloClient*.

La seconde partie représente la structure d'une requête GraphQL ici importée avec l'outil *gql*. La structure est similaire à ce que l'on pourrait voir pour une recherche dans un objet en JSON. GraphQL possède une documentation qui fait référence au projet sur lequel on est et schématise ce dernier afin de nous aider à créer les requêtes.

Jeu d'essai :

Présentation du jeu d'essai :

Le jeu d'essai consiste à créer, modifier et supprimer du contenu sur le CMS mis à disposition du client. Ainsi le client peut mettre à jour ce qui est affiché sur le site web en fonction des stocks disponibles en boutique sans que cela ne demande de compétences particulières ou une intervention quelconque sur le code.

Veille sur les vulnérabilités de sécurité :

N'ayant pas de possibilité d'injecter du contenu autre qu'au travers du CMS, il n'y a pas de failles de sécurité majeures. Chaque entrée dans la création de données est typée, requise ou non. Grâce à GraphQL la requête retire uniquement les données désirées et rien de plus contrairement à une API ce qui évite tout risque.

La seule erreur possible vient du lien dans la vidéo en page d'accueil. Si le lien ne viens pas du site *Youtube*, le lecteur ne pourra pas afficher de vidéo et sera par conséquent grisé.

Recherches :

NextJS étant un framework qui charge toutes ses pages au préalable (Dit en Server Side Rendering ou Rendu du Côté du Serveur), le client ne peut avoir d'effet sur ces dernières. De plus tout étant typé en amont sur le CMS avec le modèle de données et en aval avec TypeScript cela restreint encore les risques possibles.

Description d'une situation de travail ayant nécessité une recherche à partir d'un site anglophone :

Contexte :

Afin d'optimiser le code et l'automatisation de mon site web, j'ai voulu utiliser le routage dynamique afin de ne créer qu'une page qui sera capable d'afficher chacun des articles. Je me suis donc penché simplement sur la documentation de NextJS.

Extrait du site anglophone :

Defining routes by using predefined paths is not always enough for complex applications. In Next.js you can add brackets to a page (`[param]`) to create a dynamic route (a.k.a. url slugs, pretty urls, and others).

Consider the following page `pages/post/[pid].js`:

```
import { useRouter } from 'next/router'

const Post = () => {
  const router = useRouter()
  const { pid } = router.query

  return <p>Post: {pid}</p>
}

export default Post
```

Any route like `/post/1`, `/post/abc`, etc. will be matched by `pages/post/[pid].js`. The matched path parameter will be sent as a query parameter to the page, and it will be merged with the other query parameters.

For example, the route `/post/abc` will have the following `query` object:

```
{ "pid": "abc" }
```


Traduction de l'extrait en français :

Définir des routes en utilisant des chemins prédéfinis n'est pas toujours suffisant pour des applications complexes. Avec NextJs on peut ajouter des crochets pour créer des route dynamiques (Connu sous le nom d'URL slug, ...).

Considérons la page '*pages/post/[pid].js*' :

```
import { useRouter } from 'next/router'

const Post = () => {
  const router = useRouter()
  const { pid } = router.query

  return <p>Post: {pid}</p>
}

export default Post
```

Chaque routes comme '*post/1*' ou '*post/abc*' sera associée à '*pages/post/[pid].js*'. Le paramètre de chemin associé sera envoyé via une requête à la page et sera fusionné avec les autres paramètres de la requête.

Par exemple la route '*post/abc*' aura l'objet de requête suivant :

```
{ "pid": "abc" }
```

Suite du projet - Réservation de partie :

Contexte :

La réservation de partie sera sous forme d'une boutique à part et comportera tout ce que l'on peut acheter ou louer pour participer à un événement.

Actuellement une page est réservée sur le site web répertoriant toutes les informations et les forfaits possibles avec leurs descriptions et leurs prix. Tout est actuellement créé dans le CMS et la présentation sous forme d'un tableau se met à jour en fonction du contenu présent sur le CMS afin que le client puisse gérer facilement ce qu'il affiche ou non dans ce tableau et mettre à jour le contenu au besoin.

Amélioration à apporter :

Pour améliorer cette section, j'ai prévu de la repenser sous forme d'une boutique où l'on pourra acheter ses forfaits et réserver ses places. Le nombre de consommables sera des articles sans limites tandis que les forfaits pour une place avec ou sans matériel seront des articles avec limitations (Le terrain ne pouvant pas accueillir une infinité de personnes par partie).

Outils utilisés :

Afin de mettre en œuvre cette partie du projet, j'ai prévu de me tourner simplement vers le starter kit que propose NextJS pour une boutique e-commerce. N'ayant pas de demande extravagante sur cette partie, je choisis donc cette stratégie qui me permettra d'avoir un ensemble des plus optimisé.

De plus la documentation NextJS étant extrêmement complète et claire, cela me permettra de ne pas avoir à mener de nombreuses recherches sur des sites externes.

Suite du projet - Créations personnalisées :

Contexte :

Dans l'optique de créer un espace dédié à la création de réplique personnalisée avec un maximum d'options (dans les limites du possible afin de ne pas nuire à la performance du site).

L'idée est de créer un rendu 3D où l'on pourra accessoriser les répliques que l'on choisit tout en ayant un rendu en temps réel de ce que l'on fait.

Optionnellement il faudrait pouvoir y ajouter des thèmes comme les vieillissements ou patinages qui sont des choses qui reviennent régulièrement.

Il faudrait aussi proposer des customisations clés en main comme la boutique le fait déjà actuellement.

L'outil ne sera pas une boutique, les customisations ayant des prix variables et demandant d'un modèle à l'autre plus ou moins de temps de travail, le final sera envoyé par mail ou dans une interface en back-end au client (la société d'airsoft) afin que celui ci puisse valider en personne son projet et qu'un prix puisse être fixé.


Outils utilisés :

Afin de concevoir cet outil, j'ai pensé à utiliser ThreeJS pour les visuels en 3D car il ne nuit pas aux performances et permet un rendu de qualité.

L'intégralité des éléments comme les articles et les accessoires seront stockés dans le CMS pour une facilité de mise à jour par le client en fonction de son catalogue actuel.

Le projet étant encore à son état de conception, il y a peu d'infos pour le moment. Le but étant d'allier performance et praticité.

Annexe : Road map du projet :



Retz Tactical Games

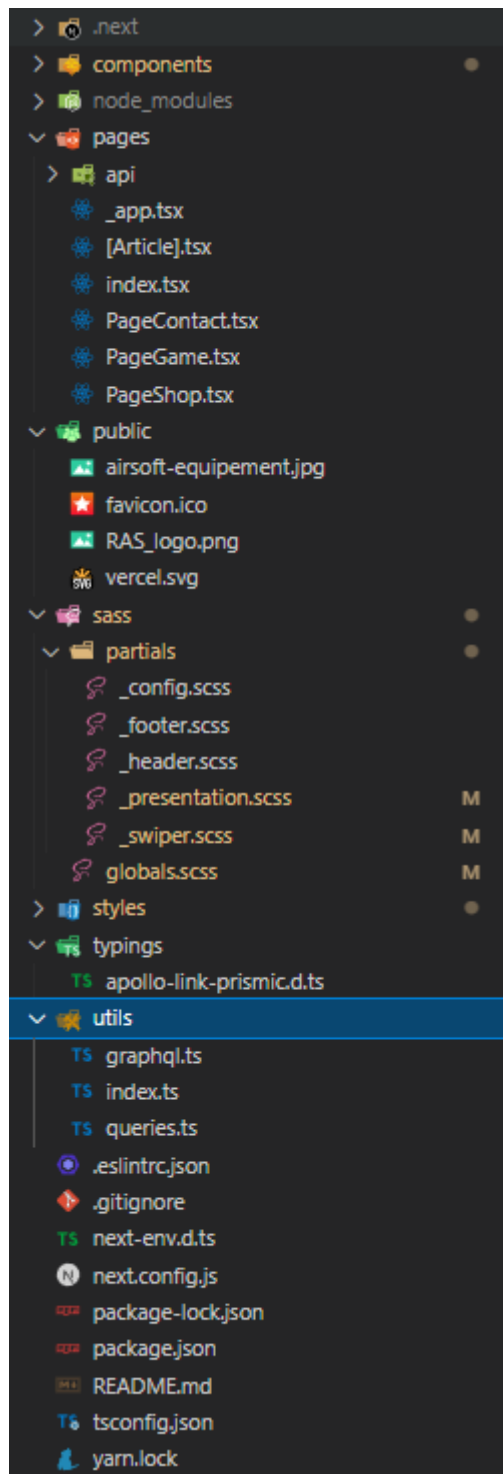
Roadmap du projet Retz Tactical Games

Table

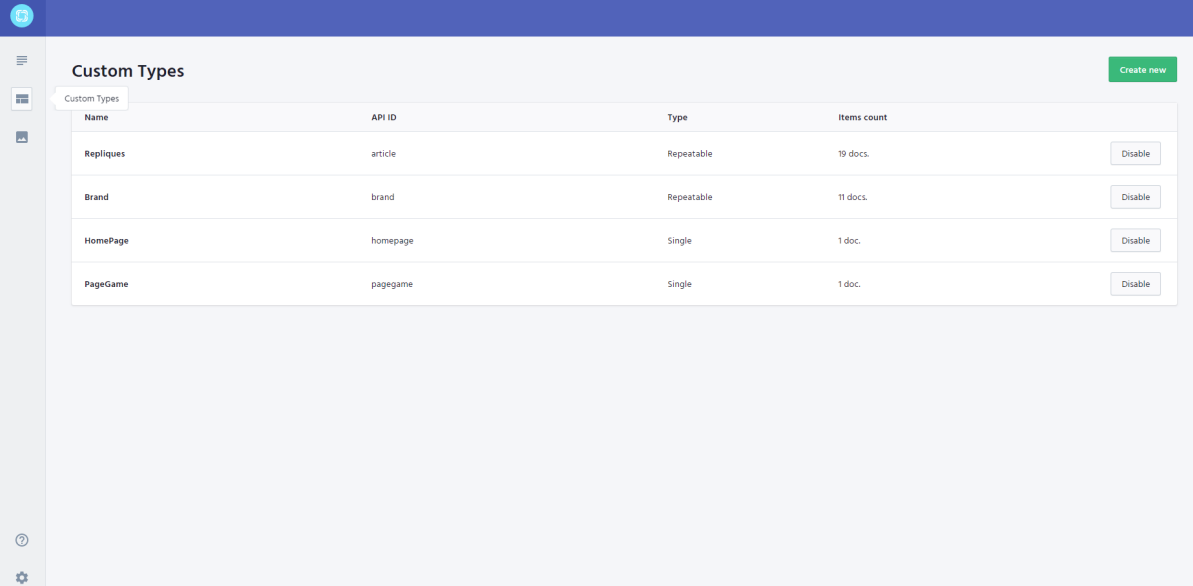
Untitled

Nom de la tâche	Statut	Notes supplémentaires
Maquette	En cours	La version desktop du site se base sur la maquette mobile dans un premier temps
Choix des technologies	Terminé	NextJS avec CMS headless (Prismic)
Création des modèles de données	Terminé	
Documentation du projet	En cours	À terminer en même temps que le projet
Création des composants NextJS	Terminé	
Mise en place du système de requêtes GraphQL	Terminé	
Choix des technologies de la customisation des répliques	En cours	ThreeJS si rendu 3D
Mise en place du système de réservation / paiement	Non commencé	

Annexe : Structure d'un projet NextJS :

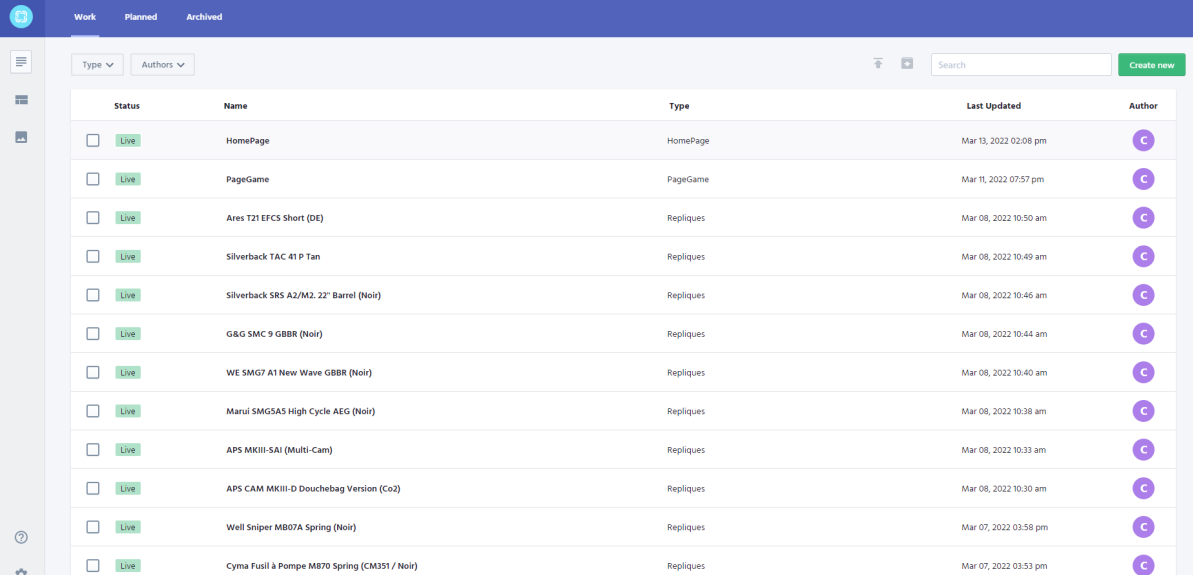


Annexe : Interfaces Prismic :



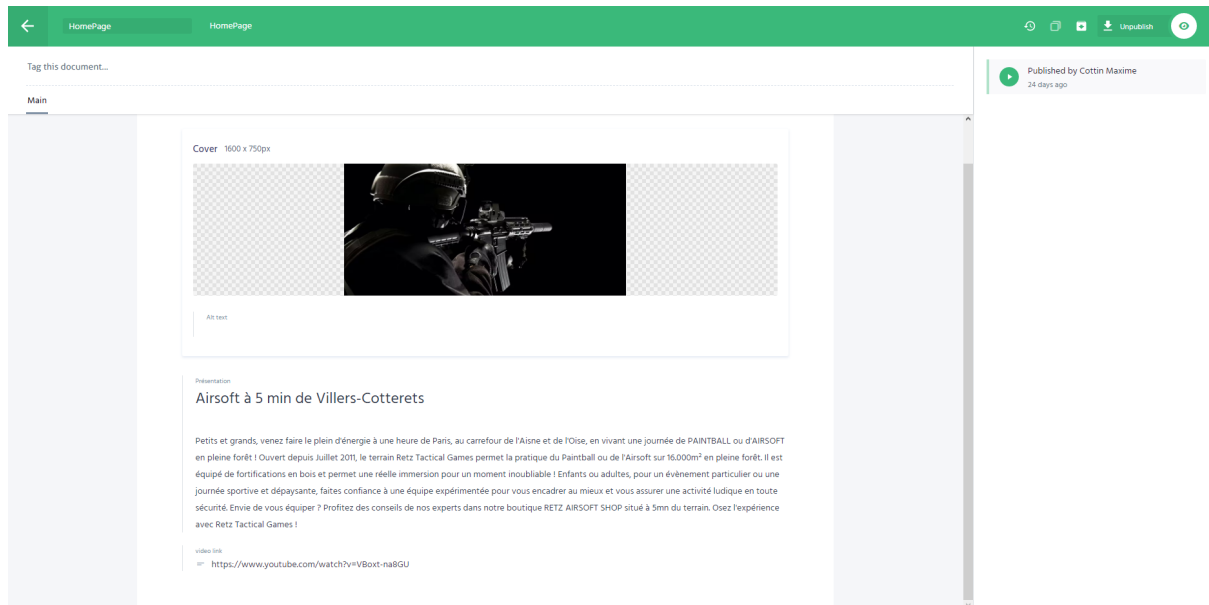
Custom Types Create new

Name	API ID	Type	Items count	
Repliques	article	Repeatable	19 docs.	Disable
Brand	brand	Repeatable	11 docs.	Disable
HomePage	homepage	Single	1 doc.	Disable
PageGame	pagegame	Single	1 doc.	Disable

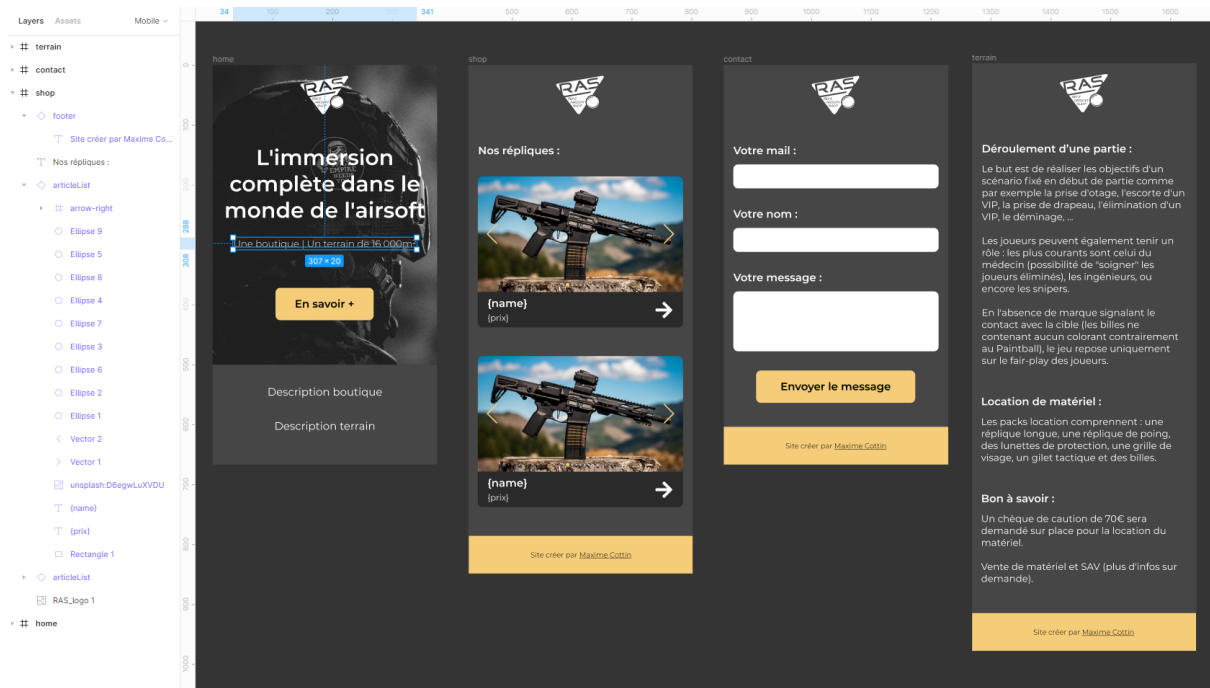


Work Planned Archived Type Authors Search Create new

Status	Name	Type	Last Updated	Author
<input type="checkbox"/> Live	HomePage	HomePage	Mar 13, 2022 02:08 pm	C
<input type="checkbox"/> Live	PageGame	PageGame	Mar 11, 2022 07:57 pm	C
<input type="checkbox"/> Live	Ares T21 EFCS Short (DE)	Repliques	Mar 08, 2022 10:50 am	C
<input type="checkbox"/> Live	Silverback TAC 41 P Tan	Repliques	Mar 08, 2022 10:49 am	C
<input type="checkbox"/> Live	Silverback SRS A2/M2. 22" Barrel (Noir)	Repliques	Mar 08, 2022 10:46 am	C
<input type="checkbox"/> Live	G&G SMC 9 GBBR (Noir)	Repliques	Mar 08, 2022 10:44 am	C
<input type="checkbox"/> Live	WE SMG7 A1 New Wave GBBR (Noir)	Repliques	Mar 08, 2022 10:40 am	C
<input type="checkbox"/> Live	Marul SMG5A5 High Cycle AEG (Noir)	Repliques	Mar 08, 2022 10:38 am	C
<input type="checkbox"/> Live	APS MKIII-SAJ (Multi-Cam)	Repliques	Mar 08, 2022 10:33 am	C
<input type="checkbox"/> Live	APS CAM MKIII-D Douchebag Version (Co2)	Repliques	Mar 08, 2022 10:30 am	C
<input type="checkbox"/> Live	Well Sniper MB07A Spring (Noir)	Repliques	Mar 07, 2022 03:58 pm	C
<input type="checkbox"/> Live	Cyma Fusil à Pompe M870 Spring (CM351 / Noir)	Repliques	Mar 07, 2022 03:53 pm	C



Annexe : Maquette mobile du site :



Annexe : Visuels du site :

