

# DOSSIER DE PROJET

Conception et réalisation  
d'une **Application Web**  
**ANGULAR® + SYMFONY®**

---

Dossier de projet présenté le 2 février 2024 en vue de l'obtention du titre RNCP n°36042

**Concepteur Développeur d'Applications**

# SOMMAIRE

I. Liste des compétences du référentiel couvertes par le projet .....	p.4
II. Résumé du projet (anglais).....	p.5
III. Expression du Besoin.....	p.6 à II
III.I Définition.....	p.6
III.I.1 Scénario du produit.....	p.6
III.I.2 Objet de l'application .....	p.7
III.I.3 Évolutions envisagées.....	p.7
III.I.4 Utilisateurs.....	p.7
Cibles .....	p.7
Contexte utilisateur.....	p.7
III.2 Contraintes du projet.....	p.8 à II
III.2.1 Fonctionnelles .....	p.8
III.2.2 Techniques .....	p.8 à 9
III.2.3 Réglementaires et légales.....	p.10
III.2.4 Temporelles, logistiques et humaines .....	p.10
III.2.5 Budgétaires .....	p.11
III.3 Identité visuelle et charte graphique.....	p.11
IV. Gestion de Projet .....	p.12 à 15
IV.1 Définition.....	p.12
IV.2 Entreprise et organisation du travail.....	p.12
IV.3 Suivi de projet.....	p.12 à 14
IV.4 Planification et objectifs .....	p.14
IV.5 Plan de qualité.....	p.14 à 15

<b>V. Spécifications fonctionnelles du projet.....</b>	<b>p.16 à 23</b>
V.I Définition .....	p.16
V.2 Diagrammes UML.....	p.16 à 17
V.3 Scénario nominal et alternatif .....	p.18
V.4 Histoires utilisateur (user stories).....	p.19
IV.3.1 Product backlog.....	p.19 à 20
V.5 Expérience utilisateur (UX) .....	p.20 à 21
V.6 Parcours utilisateur.....	p.21 à 22
V.7 Interface utilisateur (UI).....	p.22 à 23
<b>VI. Spécifications techniques du projet .....</b>	<b>p.24-35</b>
VI.I Définition .....	p.24
VI.2 Choix technologiques et argumentaire .....	p.24 à 26
VI.I.I Front-end avec Angular® .....	p.25
VI.I.2 Back-end avec Symfony® .....	p.25
VI.I.3 API.....	p.26
VI.3 Architecture et environnement de développement .....	p.26 à 29
VI.2.1 Modèles MVC et MVVM.....	p.26 à 27
VI.2.2 Base de données .....	p.27 à 28
VI.2.3 Environnement de développement.....	p.28 à 29
VI.4 Domaines, hébergement et argumentaire.....	p.29
VI.5 Exigences de programmation.....	p.29 à 32
VI.4.1 Convention de nommage et documentation .....	p.30
VI.4.2 POO et principes SOLID .....	p.30 à 31
VI.4.3 Design patterns .....	p.32
VI.6 Accessibilité .....	p.32 à 33
VI.7 Sécurité .....	p.33 à 34
VI.8 Maintenance et évolutivité .....	p.35
<b>VII. Réalisations du candidat.....</b>	<b>p.36 à 55</b>
<b>VIII. Présentation du jeu d'essais .....</b>	<b>p.56</b>
<b>IX. Description d'un cas de veille sécuritaire : accès aux données via Doctrine .....</b>	<b>p.57 à 59</b>
<b>X. Description d'une situation de travail : sécurisation d'une App Angular® .....</b>	<b>p.60 à 63</b>
<b>XI. Annexes .....</b>	<b>p.64</b>

# I. COMPÉTENCES RÉFÉRENTIEL

## I. Liste des compétences du référentiel couvertes par le projet

01 Activité-Type	02 Activité-Type	03 Activité-Type
<p><b>Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité</b></p> <ul style="list-style-type: none"><li>• Maquetter une application.</li><li>• Développer une interface utilisateur de type desktop.</li><li>• Développer des composants d'accès aux données.</li><li>• Développer la partie front-end d'une interface utilisateur web.</li><li>• Développer la partie back-end d'une interface utilisateur web.</li></ul>	<p><b>Concevoir et développer la persistance des données en intégrant les recommandations de sécurité</b></p> <ul style="list-style-type: none"><li>• Concevoir une base de données.</li><li>• Mettre en place une base de données.</li><li>• Développer des composants dans le langage d'une base de données.</li></ul>	<p><b>Concevoir et développer une application multi-couche répartie en intégrant les recommandations de sécurité</b></p> <ul style="list-style-type: none"><li>• Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement.</li><li>• Concevoir une application.</li><li>• Développer des composants métier.</li><li>• Construire une application organisée en couches.</li><li>• Préparer et exécuter les plans de tests d'une application.</li></ul>

### Observation

Les compétences du référentiel non couvertes par ce projet sont les suivantes : développer une application mobile et exécuter le déploiement. Concernant l'application mobile, l'entreprise n'avait aucun besoin dans ce domaine. Quant au déploiement, celui-ci n'a pu être réalisé avant la fin de la période de stage. Ces tâches sont traitées dans le dossier professionnel.

## II. RÉSUMÉ DU PROJET

**This project aimed to develop a web application that faced constraints in terms of technology, time, and design. SCRUM methodology and tools such as Figma®, Angular® and Symfony® were employed to design and develop this application.**

The primary goal was to create a blog-type application. The objectives of this platform were to share information about an ongoing video game production and to unite the future community of players.

Ensuring compliance with essential accessibility, eco-design and security standards were requested as possible. Application scalability was ensured through adherence to key design patterns and a modular architecture.

Use cases delineated user actions and their corresponding application responses. Then, these were translated into user stories representing various functionalities. User experience considerations involved user research, proto-persona and user flow to optimize interface design.

Technology choices centered on specific frameworks known for their security and modularity. Database design leveraged MySQL WorkBench®, and concerning hosting, a shared hosting option was chosen.

Coding practices concentrated on adhering to specific good practices and security protocols. The project resulted in a robust and educational development methodology and a user-centered web application.

# III. EXPRESSION DU BESOIN

## III.I Définition

Le besoin concernant l'application présentée dans ce dossier de projet a été formulé par Vitalis Studio sous la forme d'un cahier des charges MVP (Minimum Valuable Product). Cette partie présente la définition de ce besoin ainsi que les principaux éléments du cahier des charges qui m'ont été nécessaires pour bien appréhender les attentes du "Client" et les enjeux sous-jacents.

### III.I.I scénario du produit

Le Studio Vitalis développe actuellement un jeu vidéo dans l'univers d'une Web Fiction bien connue des internautes. Afin de communiquer au plus près de son cœur de cible, les représentants de Vitalis Studio, M. Éric SCHEIBER et M. Adrien CONSTANS ont souhaité la création d'un site Internet dédié à la communication de leur produit.

Ce support devait permettre de faire connaître le jeu en diffusant des informations relatives à l'avancement du développement. Cette communication devait cependant se faire de façon "indirecte", c'est-à-dire de manière narrative et immersive pour le futur utilisateur, plutôt que par du marketing traditionnel.

L'objectif était de créer un engouement autour du produit, sans en révéler pour autant sa nature de jeu vidéo. En effet, le lecteur devait être maintenu dans l'ignorance jusqu'au jour de la mise sur le marché.

Le concept de cette communication indirecte était d'attiser la curiosité des acheteurs

potentiels par l'activation du biais cognitif dit d'"heuristique catégorielle". C'est-à-dire apporter des informations clés sur le produit de façon à conforter l'acheteur dans son choix, et de ce fait, créer les conditions propices à l'acte d'achat.

Cet engouement créé autour d'un pseudo-mystère doit permettre de fédérer une communauté. C'est cette communauté "de niche" qui formera le gros des acheteurs du produit.

Ainsi, pour que cette stratégie fonctionne, le faisceau de "preuves" qu'il s'agit bien d'un jeu vidéo devait converger vers le lecteur de façon qu'il parvienne lui-même à cette conclusion. Ce processus "d'auto-récompense" ("J'en étais sûr !") donne l'illusion au lecteur d'avoir fait partie du projet depuis ses prémisses. Ce sentiment facilite d'une part l'appropriation du produit et crée d'autre part un sentiment d'appartenance à la communauté. Ces sentiments peuvent alors être exploités dans une campagne de financement participatif sur Kickstarter notamment. Les fonds ainsi récoltés (en échange de compensations cohérentes

pour les backers<sup>1.</sup>, appelé “paliens”) permettront de financer la suite du développement du produit.

### III.I.2 objet de l'application

Baptisée “I Found That - Blog” (abrégé IFT dans le présent dossier), est une application web d'échange entre utilisateurs d'une même communauté. De type “Blog”, le site IFT devait apporter des informations textuelles et visuelles à l'utilisateur concernant le produit. Pour cela, des articles fictifs devaient alimenter le site et ainsi offrir au lecteur une immersion totale avec le scénario du produit (voir III.I.1 page précédente « Scénario du produit »).

L'application devait être propice à l'échange entre utilisateurs au moyen de commentaires, possiblement longs.

En outre, le site IFT devait permettre la visualisation en “temps-réel” de l'avancement du développement du jeu. Ce dernier devait être matérialisé par une barre de chargement, graphiquement similaire à une barre de défragmentation. Cette jauge est visible en permanence sur le site.

### III.I.3 évolutions envisagées

Le site IFT sera retiré au profit d'un site promotionnel classique de type Landing Page dès que le produit sera révélé au grand public. Ce nouveau site présentera les démos et images du jeu, ainsi que la campagne de financement participatif prévue sur Kickstarter.

Pour cela, une redirection de l'URL du Blog (“www.ifoundthat.blog”) vers l'URL principale du jeu est prévue. Est également prévu d'afficher un message fictif avant la migration pour entretenir le mystère et l'immersion, sous la forme d'un compte à rebours jusqu'au jour du lancement du produit.

### III.I.4 utilisateurs

#### Cibles

Les utilisateurs de cette application sont issus de deux cibles distinctes :

- ♦ Cible primaire : les membres de la communauté.
- ♦ Cible secondaire : les joueurs de jeux vidéo horrifiques.

Une première tranche d'âge des utilisateurs est estimée entre 18 et 24 ans, et une seconde entre 35 et 44 ans.

#### Contexte utilisateur

L'univers fictionnel du jeu est alimenté depuis 2006 sous forme d'écriture collaborative par un public de passionnés dont l'objectif est de faire grandir la base de données du site principal. Ainsi, en rédigeant de plus en plus de récits au sein de cet univers, cette communauté cherche à construire une véritable “légende urbaine”.

Domaine de niche, les attentes des utilisateurs fans de cet univers sont donc très précises. Aussi, l'application devait se faire le reflet de l'ambiance régnant dans cette œuvre, notamment en termes de design graphique et d'immersion narrative.

---

<sup>1.</sup>. Backers : dans le financement participatif, les personnes qui soutiennent financièrement un projet.

## III.2 Contraintes du projet

L'application IFT répond à des contraintes spécifiques tant techniques, légales que temporelles ou budgétaires. Ces contraintes sont détaillées dans cette partie.

### III.2.1 Fonctionnelles

Dans le but d'optimiser l'UX du site et éviter tout point de frustration chez l'utilisateur final, un soin particulier devait être apporté aux points suivants :

- ◆ Sign-in/Sign-up : la connexion et l'inscription au site devaient être facilitées au maximum, notamment par la réduction du nombre d'étapes et de formulaires.
- ◆ Évolutivité et mise à jour : la charge de la base de données ainsi que l'hébergement devaient pouvoir évoluer si le trafic du site le nécessitait. Par ailleurs, les mises à jour devaient être réalisées dès que nécessaire, surtout celles relatives à la sécurité de l'application.
- ◆ Traçabilité des connexions : les connexions utilisateurs devaient être retracées dans un Journal des Logs. Ce journal devait être chiffré pour garantir la confidentialité des utilisateurs.
- ◆ Backup : les sauvegardes de la base de données et de la "base de code" (ou code base) devaient être assurées par le biais d'un hébergement adéquat, mais également *via* des sauvegardes manuelles régulières sur un support amovible. La fréquence des sauvegardes manuelles a été laissée à la discrétion de la DevOps ou du prestataire de maintenance informatique.

### III.2.2 Techniques

#### Accessibilité

Bien que n'étant pas légalement concerné par le Référentiel Général d'Amélioration de l'accessibilité (RGAA), Vitalis Studio ayant à cœur de rendre ses supports accessibles au plus grand nombre, les principes généraux du RGAA ont fait partie intégrante du développement de l'application.

En outre, le site IFT s'adressant principalement à une cible internationale, le respect de la norme WCAG 2.2 (W3C Accessibility Guidelines) devait également être appliquée. Le niveau double A (AA) était requis *a minima* pour une accessibilité dite "de base".

Les principes d'accessibilité web que respecte l'application sont les suivants :

- ◆ Perceptibilité : le design graphique minimaliste facilite la perception visuelle (et auditive en cas de contenu audio) du contenu du site, notamment en proposant un contraste de couleur supérieur à 3.1 ou en proposant des équivalents textuels aux éléments visuels du site (photo, illustration, pictogramme, etc). L'application dispose d'une structure identique en version Desktop ou Mobile avec une mise en page simplifiée (uniquement textuelle), grâce au responsive design.
- ◆ Utilisabilité : l'application fournit à l'utilisateur un contexte simple pour faciliter son orientation lors de la navigation sur le site. Toutes les fonctionnalités du site devaient être accessibles au clavier et le rôle de chaque fonctionnalité devait être précisé avec la norme ARIA à destination des logiciels d'assistance.

- ♦ Compréhension : l'application devait faire en sorte que les fonctionnalités se comportent comme attendu (ex : un clic sur un bouton déclenche une action). De plus, l'utilisateur devait pouvoir corriger ses erreurs de saisie le cas échéant.
  - ♦ Robustesse : la compatibilité du code source avec les technologies d'assistance (norme ARIA) devait être également assurée pour une meilleure pérennité du site.
- ♦ Limitation des services tiers : les services tiers devaient être limités au maximum.
- De plus, les référentiels suivants devaient être suivis au maximum pour garantir les bonnes pratiques en matière d'éco-conception, tout spécialement sur les parties back-end et front-end :
- ♦ Référentiel Général d'éco-conception de Services Numériques (RGESN).
  - ♦ Guide de Référence de conception responsable de services numériques (GR49I).

## Éco-conception

L'éco-conception web vise à optimiser la gestion des ressources serveur. Ainsi, l'application devait être conçue de façon responsable et devait notamment respecter les points suivants :

- ♦ Lazy loading : charger les modules, composants ou fichiers uniquement au moment de leur utilisation.
- ♦ Compression et minification : réduire la taille des images sans perdre de qualité et réduire la taille des fichiers CSS, JavaScript® pour diminuer le temps de chargement.
- ♦ Regroupement de requêtes : combiner les appels HTTP pour limiter le nombre de requêtes au serveur et mettre en cache les requêtes fréquemment utilisées pour éviter des appels répétitifs à la base de données.
- ♦ Tree shaking : éliminer le code inutilisé lors de la construction de l'application.
- ♦ Mise en cache des données : stocker localement des données statiques pour éviter des requêtes redondantes.

## Sécurité

De façon générale, tout devait être mis en œuvre en matière de sécurité informatique pour minimiser les attaques. Les principales failles de sécurité tant Back que front-end ou relatives aux frameworks et aux langages utilisés pour le développement, devaient être comblées.

En outre, les bonnes pratiques de sécurité décrites ci-dessous devaient être suivies autant que possible. Ces bonnes pratiques ne sont pas exhaustives ! Cependant, elles ont constitué un minimum requis pour sécuriser l'application :

- ♦ Validation des entrées de données : toutes les entrées de données ("input") provenant des utilisateurs devaient être rigoureusement validées et filtrées pour vérifier la conformité des données et rejeter les entrées non autorisées.
- ♦ Utilisation de requêtes paramétrées ou de procédures stockées : lors de l'interaction avec une base de données, des requêtes para-

métrées ou des procédures stockées devaient être utilisées pour empêcher les attaques par injection SQL. De plus, les données ne devaient pas être concaténées directement dans des requêtes SQL.

- ◆ Échappement des caractères spéciaux : dans les requêtes SQL, les caractères spéciaux devaient être correctement échappés (utilisation du symbole antislash "\") pour éviter les attaques par injection SQL ou XSS (Cross Site Scripting).

- ◆ Mise à jour des composants et frameworks : les bibliothèques et les frameworks devaient être tenus à jour avec les dernières versions et correctifs de sécurité pour remédier aux vulnérabilités connues (voir IX p.57 "Description d'un cas de veille sécuritaire : accès aux données avec Doctrine").

- ◆ Gestion sécurisée des identifiants et des mots de passe : les mots de passe ne devaient jamais être stockés en clair. Des méthodes de hachage et de salage (salt) sécurisées devaient être mises en place pour renforcer la sécurité.

- ◆ Gestion des erreurs : Les messages d'erreur ne devaient pas divulguer d'informations sensibles ou des détails techniques. Les informations affichées aux utilisateurs devaient être restreintes et enregistrées de manière sécurisée dans des journaux (logs).

- ◆ Sécurisation des sessions et autorisations : les sessions utilisateurs devaient être sécurisées et les attributions d'accès appropriées aux ressources de l'application devaient être mises en place. Les cookies de session devaient également être protégés en utilisant

des mécanismes tels que le chiffrement et les Jetons CSRF (Cross-Site Request Forgery).

- ◆ Tests de sécurité : des tests de sécurité, y compris des tests d'intrusion et analyses de vulnérabilité devaient être réalisés pour identifier les failles de sécurité et les corriger.

### III.2.3 Réglementaires et légales

Pour fonctionner, l'application a besoin de recueillir certaines données personnelles de ses utilisateurs, notamment leur adresse email.

Ainsi, un point de vigilance tout particulier devait être apporté au respect du Règlement Général sur la Protection des Données (RGPD) ainsi qu'à l'Ordonnance n° 2018-II25 du 12 décembre 2018 précisant les conditions d'application du RGPD.

Ainsi, les sections présentes dans le footer du site IFT d'une part et la présence de mentions à cocher sur les formulaires devaient informer l'utilisateur des traitements informatiques appliqués aux données personnelles et la durée de conservation de celles-ci.

### III.2.4 Temporelles, logistiques et humaines

Le développement de l'application devait s'étendre sur deux mois et demi, conformément au calendrier établi (voir IV.2 p.I4 « Planification et objectifs »), soit de la semaine 45 à la semaine 3.

L'application devait être mise en ligne (dans sa version MVP) au plus tard le 24 janvier 2024.

Concernant les contraintes logistiques et humaines, les locaux de Vitalis Studio n'étant pas adaptés pour recevoir du personnel, le travail s'est effectué en distanciel exclusivement. Par ailleurs, le Studio ne comptant que deux développeurs spécialisés sur Unity®, personne n'a pu aider sur la partie développement web.

### III.2.5 Budgétaires

Le budget alloué à la partie infrastructure, c'est-à-dire à l'hébergement devait respecter une fourchette tarifaire entre 30 et 50€/HT

par mois maximum, soit 720€ TTC annuels (fourchette haute).

En conséquence, les technologies retenues pour la réalisation du site et l'hébergement devaient rester conformes à cette fourchette budgétaire.

### III.3 Charte graphique

La conception de cette application a nécessité la création d'une identité et d'une charte graphique dédiée. Ces éléments sont présentés ci-contre et détaillés en annexe n°1.

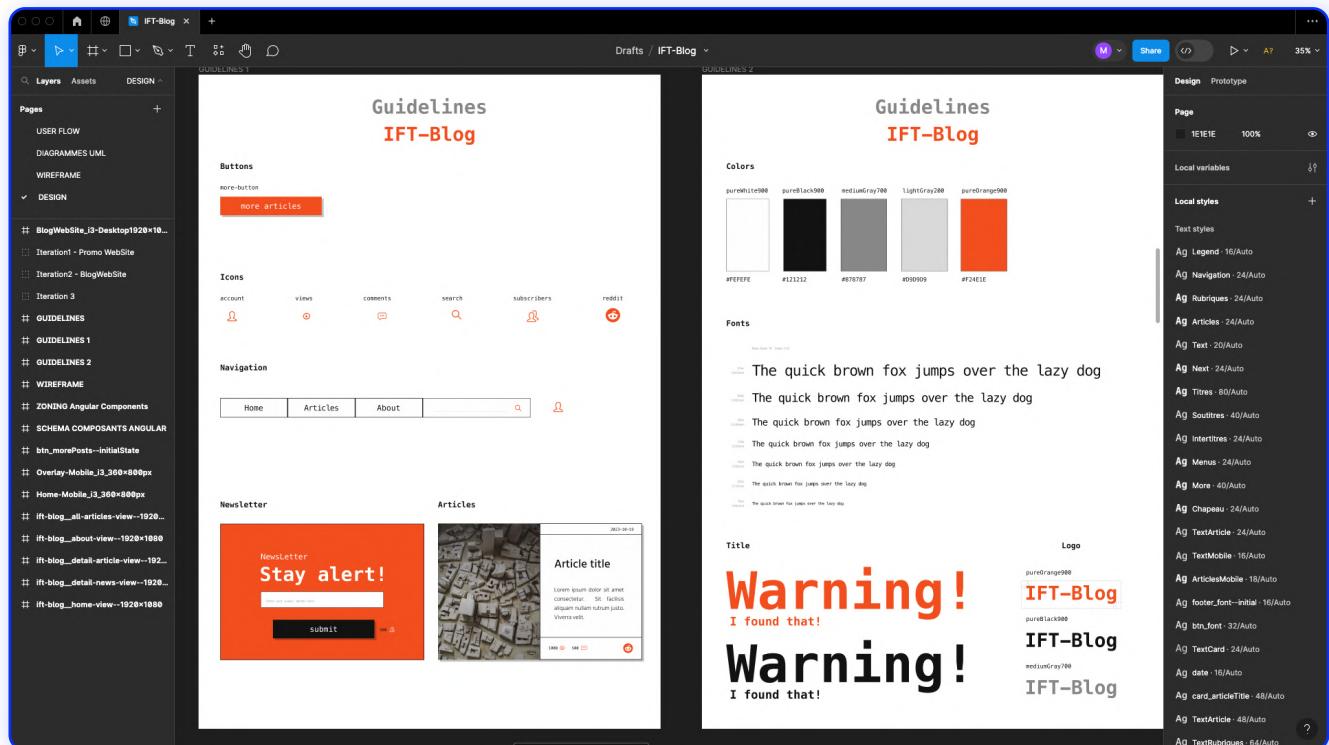


Figure I. Charte graphique

# IV. GESTION DE PROJET

## IV.I Définition

Vitalis Studio utilisait un large panel d'outils pour gérer ses projets. Cette rubrique présente l'ensemble des méthodes et des outils qui ont été utilisés pour le suivi de ce projet.

### IV.2 Entreprise et organisation du travail

#### Environnement de travail

Vitalis Studio est un jeune studio de jeux vidéo (création février 2023). Ses locaux sont sis au 3, rue Lamartine à Lyon 3ème.

Durant ma période de stage, l'entreprise comptait trois personnes. M.Éric SCHEIBER, CEO/Game designer/développeur Unity® et son associé, M.Adrien CONSTANS, ainsi que M.Simon CORTE, Game Programmer à mi-temps.

Le Studio faisait également appel à plusieurs Freelances selon les corps de métiers nécessaires au projet, notamment Sound designer, Animateurs et développeurs spécialisés (FX et SFX par exemple).

Les locaux du Studio étant inappropriés pour accueillir tous les intervenants, le Home Office était la règle pendant ce stage. Aussi, chacun travaillait avec son propre matériel informatique.

Pour ma part, j'ai réalisé toutes les productions durant ce stage sur un MacBookPro (2017) 13.3 pouces Retina équipé avec un processeur Intel i7 cadencé à 2.5 Ghz, de 16 Go de RAM et d'une carte graphique Intel Iris Plus Graphics 640.

Le stockage des fichiers de travail s'effectuait sur un disque dur externe indépendant de 2 To de type Lacie Rugged Secure. Un deuxième disque du même type était dédié au backup en continu via TimeMachine, le logiciel de réPLICATION de MacOS.

#### Organisation de l'équipe

M. Simon CORTE était à plein temps sur le développement Unity® en matière de Game Programming (programmation des niveaux et des interactions dans le jeu).

Pour ma part, mon temps était divisé en deux périodes distinctes : mes matinées de 9h à 12h, étaient dédiées au développement sous Unity® (C#), et plus spécifiquement sur la partie User interface (UI) du jeu, alors que mes après-midis étaient dévolus au développement web.

### IV.3 Suivi du projet

Le suivi global du projet était assuré par l'intermédiaire des Cérémonies SCRUM et des différents outils de suivi mis en place par Vitalis Studio.

## Méthodologie Agile SCRUM

L'équipe de Vitalis Studio utilisait la méthodologie Agile pour organiser sa production, notamment le cadre de travail SCRUM<sup>1</sup>.

Un premier Sprint Planning<sup>2</sup>, global en début de stage avait permis d'établir la production générale pour l'ensemble de l'équipe de Développement en tenant compte des spécialités de chacun.

La production était organisée en Sprint<sup>3</sup>. Tous les lundis matin à 10h, se tenait un Sprint Planning pour déterminer les tâches à réaliser sur la semaine. Ces tâches étaient fixées sur la base de deux Product Backlog distincts, l'un pour la production sous Unity®, tenu par Éric SCHEIBER, l'autre pour le développement web, tenu par moi-même, car seul développeur Web de l'équipe. Durant cette réunion, nous utilisions l'outil Trello® pour visualiser graphiquement la charge de travail globale. Ce support servait également de Sprint Backlog<sup>4</sup>, à l'équipe de Développement.

Tous les jours à 9h, l'équipe de développement tenait des Daily Scrum<sup>5</sup>, pour aborder les problématiques à venir dans le Sprint et les étapes déjà réalisées précédemment dans le même Sprint.

En milieu de semaine, avait lieu la Sprint Review<sup>6</sup>. Cette réunion avait pour objectif principal de faire le point sur l'avancement du Sprint, voire de réorienter la charge de travail pour permettre un aboutissement en fin de semaine.

Enfin, tous les vendredis à 16h se tenait une Sprint Retrospective<sup>7</sup>. Cette étape marquait généralement la fin du Sprint en cours et permettait à l'équipe d'envisager le prochain Sprint.

## Outils

Les outils employés au sein de Vitalis Studio étaient répartis en trois catégories : logiciels, frameworks et applications.

Dans le cadre du présent projet, les logiciels utilisés ont été les suivants :

- ◆ Figma® : un logiciel de prototypage vectoriel permettant la création de prototypes d'applications de toutes natures, directement utilisables. Le prototypage permet de faciliter les test utilisateurs et ainsi obtenir un feed-back de qualité avant la mise en production du produit.
- ◆ VS Code® : un IDE (Environnement de Développement Intégré) qui a permis la réalisation du code source de l'application.
- ◆ Photoshop® : logiciel de retouche photo de l'éditeur Adobe, ce software a servi notamment à l'optimisation des images bitmap au format webp.
- ◆ Illustrator® : logiciel de création vectorielle, Illustrator® a notamment servi à la création de certains Assets graphiques tels que les pictogrammes par exemple.
- ◆ GitHub Desktop® : cette application desktop permet de réaliser le versionning du projet vers le site web GitHub® via une interface graphique, sans passer par une invite de commandes.

<sup>1</sup>. Crée par Sutherland et Schwaber.

<sup>2</sup>. Réunion de planification des Sprints.

<sup>3</sup>. Période de production.

<sup>4</sup>. Document de suivi du Sprint.

<sup>5</sup>. Réunion quotidienne (15 min).

<sup>6</sup>. Revue du Sprint écoulé.

<sup>7</sup>. Réunion dédiée à identifier les points forts, les points faibles et les opportunités d'amélioration.

Les bibliothèques et frameworks utilisés ont été les suivants :

- ◆ Angular® : un framework basé sur le langage web TypeScript®. Angular® a été utilisé pour développer toute la partie front-end de l'application.
- ◆ Symfony® : un framework basé sur le langage php. Symfony® est utilisé pour produire la partie back-end et le requêtage vers les différentes bases de données du site.
- ◆ API Platform® : API Platform® est un ensemble d'outils php open source. Ils permettent la création d'API Web (RESTFull dans cadre du présent projet) et offrent une infrastructure complète pour développer des API avec Symfony®.
- ◆ PHPUnit® : framework de test PHP® utilisé pour l'écriture et l'exécution des tests unitaires et fonctionnels dans Symfony®. Cet outil facilite la gestion des tests selon le jeu d'essais (voir VIII p.56 "Présentation du jeu d'essais").

Concernant les applications, les solutions utilisées ont été les suivantes :

- ◆ GitHub® : cette plateforme nous a permis de stocker et de versionner chaque étape de la réalisation du produit par l'intermédiaire d'un "Repository", un dossier de dépôt propre au projet.
- ◆ Trello® : Trello® est une application de gestion de projet en ligne qui utilise un système de tableaux dits "Kanban" pour aider les équipes à organiser leurs tâches. Trello® est populaire pour sa simplicité et son interface intuitive.

## IV.4 Planification et objectifs

### Planification

Un calendrier général de production était tenu par Éric SCHEIBER en tant que Chef de Projet.

En outre, un diagramme de Gantt a été réalisé spécifiquement pour le suivi de ma production en tant que stagiaire. Régulièrement actualisé, ce support permettait à toute l'équipe de visualiser en une fois la durée de chaque période de production et les grandes étapes de la réalisation, ainsi que les tâches et sous-tâches nécessaires.

La figure ci-dessous présente un extrait du diagramme de Gantt utilisé pendant ce projet. Le diagramme complet est disponible en annexe n°3.

### Objectifs

Les objectifs de stage qui m'ont été confié étaient doubles. D'abord réaliser la conception, le prototypage et le développement de l'interface utilisateur (UI) du jeu vidéo, puis concevoir et développer l'application IFT, objet du présent dossier.

## IV.5 Plan de qualité

L'appréciation de la qualité des productions réalisées à l'issue de chaque Sprint s'est faite par l'intermédiaire d'un plan de qualité.

Le plan de qualité est un document qui indique de façon détaillée la stratégie mise en place par l'équipe de développement pour garantir

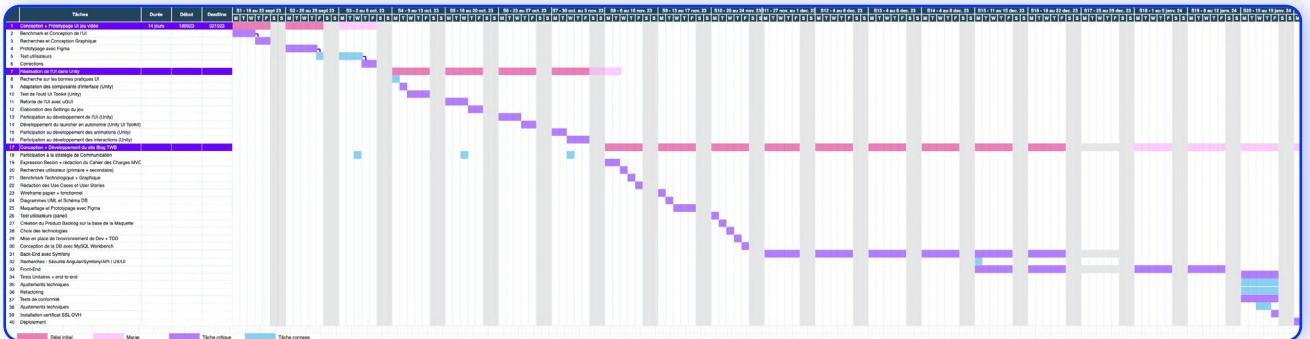


Figure 2. Diagramme de Gantt

que le produit final respecte les normes, les attentes et les exigences établies dans le cahier des charges. Outre les critères de qualité, le plan de qualité détermine aussi les processus, les méthodes et les normes à suivre tout au long du cycle de développement pour assurer la conformité du produit, notamment au regard des normes et des standards

de l'industrie du web. Ce document comprenait des points de vigilance tels que la sécurité, le respect de l'Architecture MVC.

La figure ci-dessous présente un extrait du plan de qualité utilisé pour ce projet. Le document complet est disponible en annexe n°2.

Aa Subject	≡ Details (cf. Cahier des Charges MVP)	⌚ Status	≡ Comment
<u>Mise en ligne</u>	Le site est en ligne	To do	
<u>Fiabilité</u>	Les tests unitaires et fonctionnels ont été réalisés pour les features principales (article)	Done	
<u>Password</u>	Les mots de passe sont hachés dans la DB	Done	
<u>Programmation</u>	Les exigences de programmation sont satisfaites	Done	
<u>MVC</u>	L'architecture MVC est respectée dans Symfony	Done	
<u>MVVM</u>	L'architecture MVVM est respectée dans Angular	Done	
<u>Requêtes</u>	Les requêtes SQL sont préparées avec prepare statement (API custom)	Done	
<u>Commentaires</u>	Le code source est commenté en back et front	Done	
<u>Accessibilité</u>	l'application est accessible à tous les utilisateurs, y compris ceux ayant des besoins spécifiques (attributs ARIA)	Ongoing	A améliorer

Figure 3. Extrait du plan de qualité

# V. SPÉCIFICATIONS FONCTIONNELLES

## V.I Définition

Les spécifications fonctionnelles décrivent les fonctionnalités, les actions et les processus de l'application. Dans cette section seront détaillées les exigences que l'application doit avoir du point de vue de l'utilisateur. Cette partie s'appuie notamment sur l'utilisation du langage de modélisation normalisé UML2.5. Les différents diagrammes sont consultables en annexe n°4.

## V.2 Diagrammes UML

Plusieurs diagrammes UML complémentaires ont été utilisés ensemble pour modéliser les différents aspects de l'application : un diagramme de cas d'utilisation, un diagramme de séquence et un diagramme de classes.

### Diagramme de cas d'utilisation

Séquences d'actions réalisées par le système, les cas d'utilisation produisent un résultat affichable à l'écran et conforme aux attentes d'un utilisateur donné.

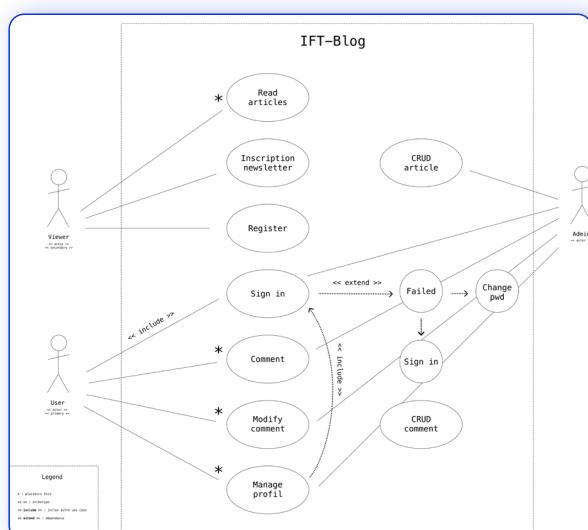


Figure 4. Diagramme UML de cas d'utilisation

Les diagrammes de cas d'utilisation spécifient le comportement attendu de manière visuelle. Ces schémas décrivent les tâches accomplies par l'application pour produire le résultat escompté, selon un scénario nominal écrit en parallèle (voir V.2 p.18 "Scénario nominal et alternatif"). La figure numéro 4 présente le diagramme de cas d'utilisation du projet.

### Diagramme de classes

Ce diagramme représente la structure statique de l'application. Il identifie les classes, leurs attributs, leurs méthodes (ou opérations), et les relations entre elles (comme l'héritage par exemple).

Les classes sont représentées sous forme de boîtes avec des compartiments pour les noms de classe, les attributs et les méthodes.

Les relations entre les classes sont indiquées par des lignes reliant les classes avec des indications de rôles par exemple.

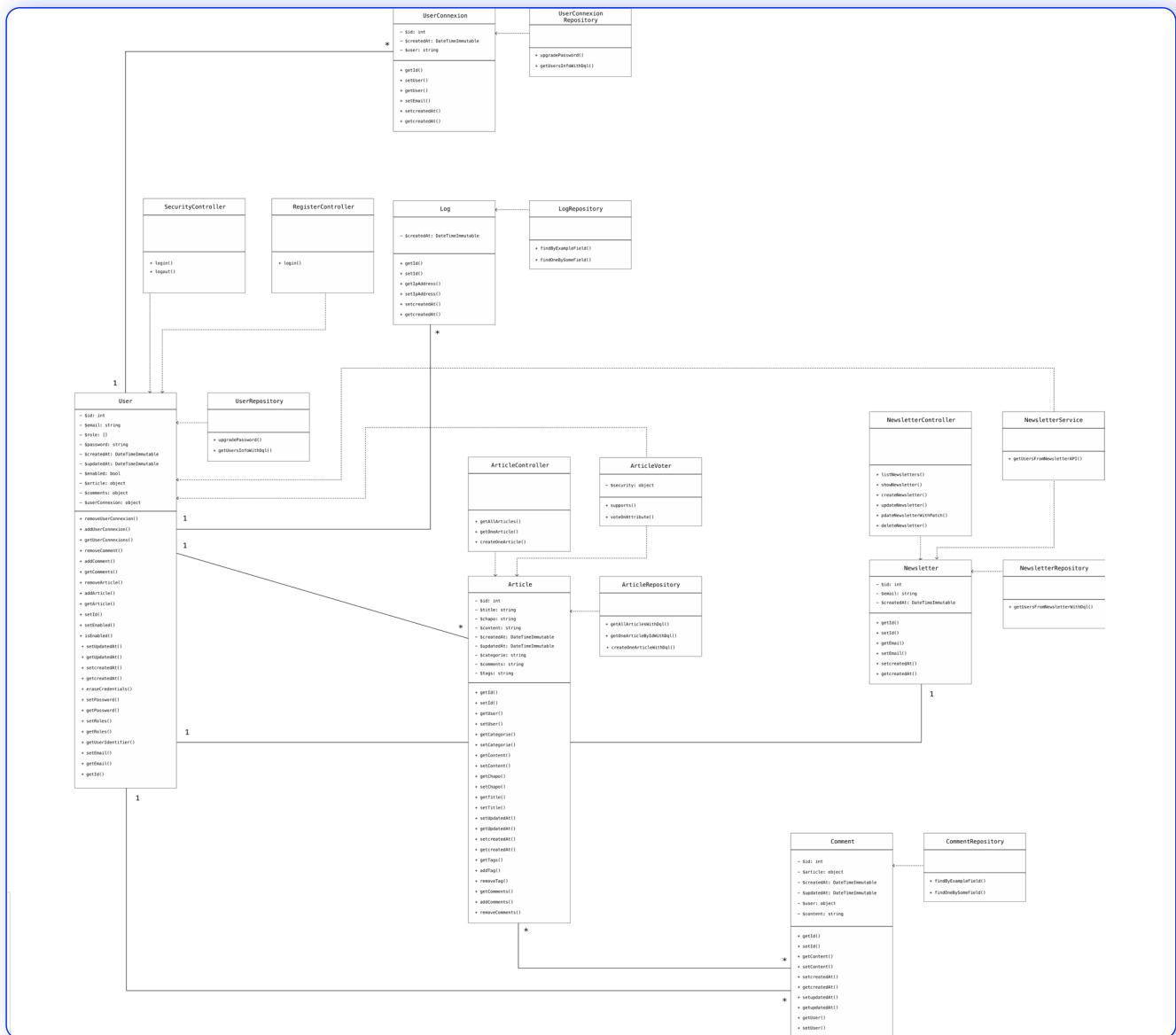


Figure 5. Diagramme UML de classes

### Diagramme de séquence

Ce diagramme montre la séquence des messages échangés entre différents objets ou entités de l'application, dans un ordre chronologique.

Il met en évidence comment les différents éléments du système interagissent les uns avec les autres pour concrétiser le scénario modélisé dans le diagramme de cas d'utilisation.

Les objets sont représentés par des rectangles contenant le nom de l'objet, et les informations sont représentées par des flèches entre ces objets, indiquant la séquence des interactions.

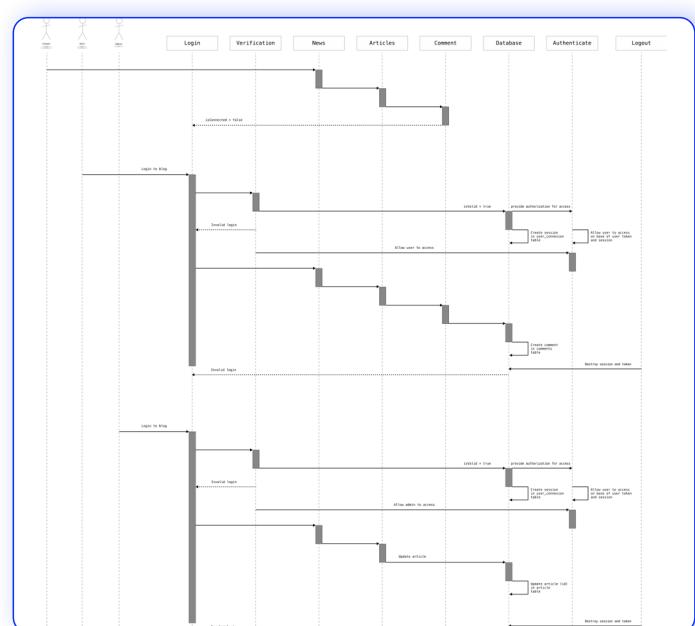


Figure 6. Diagramme UML de séquence

## Scénario nominal USER

- I - L'utilisateur membre s'identifie sur le site à l'aide du bouton "account".
- 2 - Le système renvoie un formulaire d'authentification.
- 3 - L'utilisateur entre son adresse email et son mot de passe.
- 4 - Le système vérifie la cohérence des informations sur la table "user" de la base de données et crée un token de session user.
- 5 - Le système renvoie une vue de confirmation à l'utilisateur membre.
- 6 - L'utilisateur membre est connecté au blog.
- 7 - L'utilisateur membre consulte l'article en news du blog.
- 8 - Le système renvoie le détail de la news.
- 9 - L'utilisateur membre consulte les autres articles du blog.
- I0 - Le système retourne la vue appropriée.
- II - L'utilisateur membre commente un article du site.
- I2 - Le système affiche l'interface de saisie.
- I3 - L'utilisateur membre valide son commentaire.
- I4 - Le système insère le commentaire dans la table "comments" de la base de données.
- I5 - Le système renvoie une vue de confirmation de prise en compte du commentaire.
- I6 - L'utilisateur membre se déconnecte.
- I7 - Le système détruit le token et la session.
- I8 - L'utilisateur membre est déconnecté.

## V.3 Scénario nominal et alternatif

Les scénarios décrivent de manière détaillée les cas d'utilisation précédemment identifiés. Dans ce projet, deux types de scénarios ont été utilisés : nominal et alternatif.

Le scénario nominal représente un enchaînement idéal de séquences alors que le scénario alternatif évoque une utilisation détournée, conduisant soit au résultat attendu par l'utilisateur, soit à une erreur.

Bien qu'il n'existe pas de formatage imposé par UML2.5 pour la rédaction des scénarios, la méthode employée pour ce projet est la représentation dite de Larman. Cette méthode sépare l'acteur de l'action. Les différents scénarios sont consultables en annexe n°5.

La figure numéro 7 présente le scénario nominal sous forme de tableau de Larman.

ACTOR	ACTION
1 - L'utilisateur membre s'identifie sur le site à l'aide du bouton "account".	2 - Le système renvoie un formulaire d'authentification.
3 - L'utilisateur entre son adresse email et son mot de passe.	4 - Le système vérifie la cohérence des informations sur la table "user" de la base de données et crée un token de session user.
6 - L'utilisateur membre est connecté au blog.	5 - Le système renvoie une vue de confirmation à l'utilisateur membre.
7 - L'utilisateur membre consulte l'article en news du blog.	8 - Le système renvoie le détail de la news.
9 - L'utilisateur membre consulte les autres articles du blog.	10 - Le système retourne la vue appropriée.
11 - L'utilisateur membre commente un article du site.	12 - Le système affiche l'interface de saisie.
13 - L'utilisateur membre valide son commentaire.	14 - Le système insère le commentaire dans la table "comments" de la base de données.
16 - L'utilisateur membre se déconnecte.	15 - Le système renvoie une vue de confirmation de prise en compte du commentaire.
18 - L'utilisateur membre est déconnecté.	17 - Le système détruit le token et la session.

Figure 7. Scénarios nominal méthode Larman

## V.4 Histoires utilisateur (user stories)

Dans le cadre de la réalisation de ce projet, j'ai été amené à rédiger les user stories.

Les user stories sont la description des cas d'utilisation (voir V.I p 16 "Diagrammes UML"), rédigé du point de vue de l'utilisateur final. Elles permettent de concrétiser les fonctionnalités de l'application d'une part et de mieux se représenter l'usage.

Les user stories ont été rédigées de façon classique, c'est à dire en trois points :

- ◆ AS (En tant que)
- ◆ I WANT (objectif)
- ◆ SO THAT (bénéfice).

Par exemple : "En tant qu'administrateur, je veux pouvoir créer, modifier ou supprimer un article".

En outre, elles ont été contrôlées avec la méthode INVEST<sup>1</sup>, puis affinées tout au long du projet, notamment en termes de definition of done<sup>2</sup>, avec le langage Gherkin.

Le langage Gherkin vient compléter la description des User Stories en apportant une situation de départ (Given), une action (When) et un résultat (Then). Si l'on reprend l'exemple de la user story précédente :

- ◆ GIVEN : un administrateur connecté.
- ◆ WHEN : l'administrateur modifie l'article.
- ◆ THEN : les modifications de l'article sont enregistrées dans la base de données.

Couplé au langage Gherkin, les user stories permettent d'avoir une vision précise des fonctionnalités, mais aussi de leur temporalité (début, fin).

Une fois rédigées, les user stories ont été pondérées avec la méthode du planning poker<sup>3</sup>. Ce score permet de mesurer la difficulté d'implémentation des fonctionnalités et ainsi évaluer au mieux la planification des différents Sprints.

Le planning poker est une technique utilisée dans la méthode Agile pour estimer l'effort nécessaire à la réalisation des user stories (et/ou du développement des tâches) lors de la planification des Sprints ou itérations.

### V.4.1 Product backlog

Une fois les user stories rédigées et validées par l'équipe de conception, elles peuvent être référencées dans le product backlog<sup>4</sup> du projet.

Les différentes entrées du product backlog sont alors organisées selon le score de pondération précédemment établi.

Les fonctionnalités qui offraient le plus de valeur ajoutée à l'application ont été développées en priorité. Par exemple l'interface utilisateur ou encore l'inscription sur le site.

La figure numéro 8 (voir page suivante) montre un exemple de product backlog utilisant les user stories. Le product backlog du projet est consultables en annexe n°6.

<sup>1</sup>. Acronyme de independante, negotiable, valuable, estimable, small, testable. Critères d'évaluation de la qualité d'une user story.

<sup>2</sup>. En méthodologie Agile SCRUM, le Definition of Done est un support qui indique les critères d'achèvement d'une implémentation.

<sup>3</sup>. Moyen d'estimer la complexité des tâches à réaliser par un système de pondération basé sur la suite de Fibonacci.

<sup>4</sup>. Document qui recense toutes les user stories et les fonctionnalités à implémenter.

User	# ID	Status	Priority	Theme	Item (User Story)	Aa Feature	# Story Point
ADMIN	230929	Done	High	CRUD	<p><u>Gherkin :</u>  Given un utilisateur connecté avec un article existant  When l'utilisateur modifie son article  Then les modifications sont enregistrées avec succès</p>	<u>Modification des articles</u>	21
ADMIN	23091210	Done	High	CRUD	<p><u>US :</u> En tant qu'ADMINISTRATEUR je veux pouvoir supprimer mes articles.</p> <p><u>Gherkin :</u>  Given un utilisateur connecté avec un article existant  When l'utilisateur supprime son article  Then l'article est supprimé de manière permanente</p>	<u>Suppression des articles</u>	21

Figure 8. Extrait du product backlog du projet

## V.5 Expérience utilisateur (UX)

L'UX du site a été conçu en amont de la réalisation. Cette étape clef de la conception de l'application s'est concentrée sur deux aspects : la recherche utilisateur d'une part et l'expérience utilisateur d'autre part.

La recherche utilisateur tout d'abord, permet une compréhension profonde de l'utilisateur final.

Cette étape constituait un enjeu majeur pour Vitalis Studio, car connexe au lancement de son produit et pour sa communication sur les canaux utilisés par la cible..

Ainsi, la recherche utilisateur s'est organisée elle-même en plusieurs étapes clés : la recherche primaire et la recherche secondaire.

La recherche dite "primaire" en UX Design correspond à la collecte de données directement auprès des utilisateurs cibles (entretiens utilisa-

teurs, tests, questionnaires). La recherche dite "secondaire" quant à elle, consiste à rechercher de la documentation existante sur la problématique concernant le produit final ou l'application.

Paradoxalement, la recherche secondaire a été réalisée avant la recherche primaire. Cette façon de procéder permet en effet une économie de temps et de moyens substantielle : en se basant sur des données existantes on peut affiner plus rapidement les attentes et les usages de la cible.

Une fois la recherche secondaire menée et les attentes des utilisateurs finaux identifiées, la phase de recherche primaire a été conduite par l'équipe de conception de Vitalis Studio.

Cette dernière étape de recherche s'est cependant concentrée sur les tests des différentes interfaces utilisateurs (produit et application).

Tous ces éléments préalables ont permis de dresser un portrait précis des utilisateurs. Pour les matérialiser de façon formelle et pour que toute l'équipe de conception puisse garder à l'esprit les attendus, une fiche de proto-persona<sup>1.</sup> a été produite. Cette fiche a servi tout autant aux designers qu'aux développeurs du Studio et a permis de corriger certaines User Stories pour affiner au mieux les fonctionnalités notamment.

Enfin, la partie technique de l'expérience utilisateur s'est concentrée sur des métriques précis comme les performances (temps de chargement), l'aisance de navigation (simplicité de l'interface, hiérarchisation des éléments d'UI) et les fonctionnalités attendues par l'utilisateur final (découvertes lors de la phase de recherche utilisateur).

Pour vérifier ces mesures tant quantitatives que qualitatives tout au long du processus de développement, l'équipe a ciblé les critères d'analyse selon trois thématiques : l'utilisabilité, le taux d'engagement et le taux de conversion.

- ♦ Utilisabilité : dans le cadre de ce blog, l'utilisabilité est mesurée par le temps passé à réaliser une tâche précise. Par exemple, le temps de création d'un compte pour un nouvel utilisateur. Si ce score se situe dans une fourchette attendue (soit 1 min maximum), cela signifie que l'interaction est optimale.

- ♦ Taux d'engagement : essentiellement mesuré par le temps total passé sur le site et le partage sur les réseaux sociaux. Ces marqueurs permettent de mesurer l'intérêt du lecteur pour les contenus proposés. Dans ce cas de figure, plus le temps passé sur le site est long, plus l'inté-

rêt est grand. Concernant les réseaux sociaux, si le nombre total de partages sur une période donnée (par exemple une semaine) est élevé, cela indique que l'utilisateur du site est devenu "prescripteur" : il recommande la lecture des contenus.

- ♦ Taux de conversion : dans le cadre de ce blog qui ne fait la promotion d'aucun produit, il est entendu par "conversion", l'intérêt de l'internaute pour les contenus du blog. Ainsi, le nombre d'interactions enregistrées permet de donner un indicateur fiable. Le nombre de commentaires laissés, l'utilisation du formulaire de contact et l'inscription à la newsletter sont des points de mesures concernant le taux de conversion. Ces indicateurs permettent de se faire une idée de la confiance, les éventuelles recommandations et la probabilité d'action de l'internaute avec le site et la communauté sous-jacente. Le proto-persona et les recherches secondaires du projet sont consultables en annexe n°7.

REMARQUE : les recherches primaires ne sont pas disponibles dans ce dossier de projet pour des questions de protection industrielle.

## V.6 Parcours utilisateur (user flow)

Le user flow est la représentation graphique de l'itinéraire qu'un utilisateur suit à travers l'application pour atteindre un objectif spécifique ou accomplir une tâche donnée. Les user flows sont essentiels pour comprendre l'expé-

<sup>1.</sup> C'est à partir du ressenti de l'équipe concernant les futurs utilisateurs que les proto-personas sont conçus. Cette pratique s'oppose aux personas qui eux, sont conçus à partir de données réelles. Dans les deux cas, ces documents représentent l'utilisateur final.

rience utilisateur et pour concevoir des interfaces intuitives et efficaces. Dans le cadre de ce projet, le user flow produit incluait les éléments suivants :

- ◆ écrans suivant les différentes interfaces (profil, article news, article standard, login, etc),
- ◆ les actions effectuées par l'utilisateur étaient indiquées sous forme de séquences logiques chiffrées,
- ◆ des écrans secondaires indiquaient les choix multiples,
- ◆ les différentes étapes de validation.

La figure numéro 9 représente le user flow de l'application IFT :

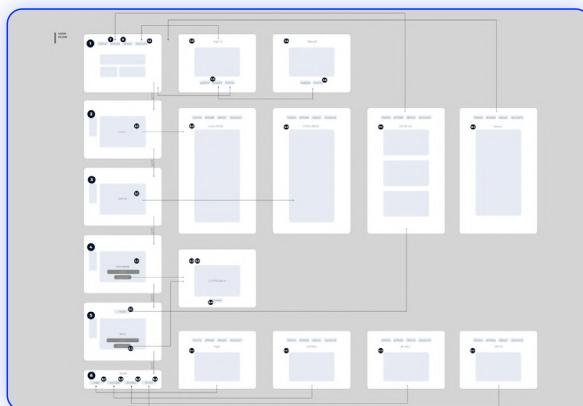


Figure 9. User flow

## V.7 interface utilisateur (UI)

L'interface utilisateur du blog et ses différents composants ont été réalisés en deux étapes distinctes : une phase de prototypage avec Figma®, puis une phase de développement.

### Prototypes

En tout premier lieu, un prototype basse fidélité (dit "low-fid") a été réalisé pour fixer le concept général du site et donner une première approche du design graphique. Puis un prototype

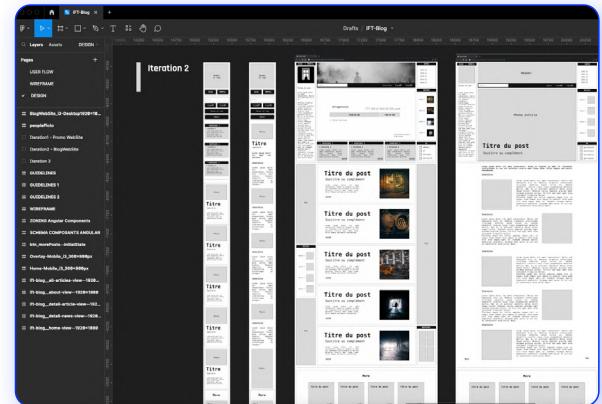


Figure 10. Itération « site années 90 »

haute fidélité ("high-fid") a permis de faire tester l'interface et la navigation à des utilisateurs.

Le prototype low-fid a permis notamment de réorienter le design général. En effet, l'équipe de conception étant partie au départ sur un concept "site des années 90". Ce premier prototype a permis de démontrer une incohérence entre la campagne de communication envisagée et la cible du blog. En outre, cette étape a également permis de trier les fonctionnalités.

Enfin, le prototype high-fid a permis d'obtenir des retours utilisateurs pertinents, notamment dans l'organisation des éléments de la page principale du blog. La figure numéro 10 montre le prototype réalisé avec Figma :

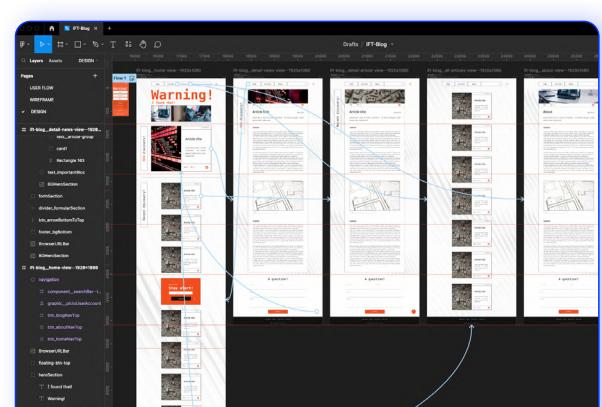


Figure 11. Prototype Figma®

## Développement

La phase de développement de l'interface utilisateur du site s'est également déroulée par étapes.

J'ai tout d'abord réalisé un zoning général dans une approche "desktop first", c'est-à-dire en débutant la conception par l'affichage sur écran d'ordinateur, cette approche correspondant plus aux usages réels de la cible (voir annexe n°7 "UX Design"). Toutefois, la problématique du responsive design (design s'adaptant aux écrans de mobiles ou autres) a été prise en compte dès ce moment de la conception. Cette double approche a permis de se focaliser sur les éléments fondamentaux de l'interface tout en offrant une expérience conforme aux attentes des utilisateurs. De plus, ce type de conception permet une meilleure gestion des ressources du site, ce qui s'inscrit dans un principe d'éco-conception comme vu dans le cahier des charges du projet (voir III.2.2 p.8 "Contraintes techniques").

Pour cela, j'ai utilisé le langage de balisage HTML pour bâtir la structure générale du site (sections, modules, boutons, etc), puis j'ai utilisé les grid CSS pour positionner les différentes zones.

Cette étape m'a permis de valider auprès de l'équipe de conception la bonne organisation générale du site tout en amorçant la réalisation du responsive design.

Ensuite, j'ai réalisé un wireframe<sup>1.</sup> en suivant cette même logique de conception. Pour accélérer la réalisation de cette étape, j'ai utilisé la bibliothèque Materialize<sup>®</sup>. L'utilisation du wireframe fonctionnel a permis de figer les différents éléments du blog (news, articles,

commentaires, etc) et de procéder en interne aux premiers tests d'utilisabilité de l'interface. Ces tests préliminaires ayant démontré que l'interface se comportait conformément aux retours utilisateurs obtenus par l'intermédiaire du prototype high-fid Figma<sup>®</sup>, j'ai pu procéder immédiatement à la réalisation concrète du site.

Remarque : cette étape du wireframe pourrait être jugée comme faisant "doublon" avec les tests utilisateurs. Or, celle-ci a permis de vérifier le comportement effectif de l'interface sur différents navigateurs, à savoir : Chrome (78%), Edge (10,3%), Firefox (4,8%) et Safari (3,9%) <sup>2.</sup>

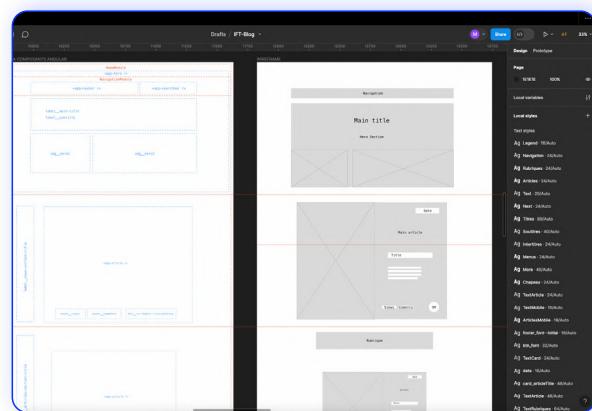


Figure I2. Wireframe

J'ai réalisé tout d'abord les différentes vues du site en HTML et SCSS, toujours en privilégiant une approche Desktop-First, mais en incluant le responsive design (*via* media queries). J'ai ensuite découpé l'ensemble des sections et modules du site pour les répartir dans des Modules et composants Angular<sup>®</sup>. La figure numéro I2 représente le wireframe fonctionnel de l'application. Les différents visuels de conception du projet sont consultables en annexes n°8.

1. Représentation basique d'une interface utilisateur.

2. Chiffres octobre 2023. Source : W3Schools.com.

# VI. SPÉCIFICATIONS TECHNIQUES

## V.I Définition

Les spécifications techniques définissent les exigences techniques relatives à la conception, au développement et à l'implémentation de l'application. Elles incluent également les détails sur les technologies, les bonnes pratiques en matière d'écriture de code, les informations relatives à la base de données et à l'architecture de l'application.

### VI.I Choix technologiques et argumentaire

Pour réaliser ce Blog, j'ai proposé à l'équipe de Vitalis Studio d'utiliser les technologies suivantes :

- ◆ Le framework Angular® 16.2.10 pour organiser la logique de la partie front-end.
- ◆ Le framework Symfony® 6.3 pour toute la partie back-end.
- ◆ API REST avec API Platform® pour gérer la communication entre le front et le back-end.
- ◆ MySQL® 5 pour le langage de requête à la base de données.
- ◆ PHPUnit® pour la partie tests unitaires (back).
- ◆ SCSS pour la gestion des styles.

#### Pourquoi utiliser des frameworks ?

L'usage de frameworks tant pour la réalisation du Front que du Back a été motivée par plusieurs aspects. Tout d'abord, en raison du délai de réalisation. En effet, deux mois et demi étaient alloués au planning pour la réalisation du Blog, de novembre à décembre 2023 (voir IV.4 p.14

"Planification et objectifs"). Aussi, ne connaissant pas ma vélocité en matière de développement en tant qu'apprenant et étant le seul développeur web au sein de Vitalis Studio, tous les outils capables de réduire les délais de Développement constituaient un atout pour le succès du projet.

Ensuite, les frameworks tels qu'Angular® et Symfony® offrent également des outils pour minimiser les tâches redondantes sans valeur ajoutée, me permettant ainsi de me concentrer sur la logique de l'application.

De plus, les frameworks imposent des normes et des bonnes pratiques de codage, ce qui permet une meilleure structuration du code et donc une meilleure lisibilité et une garantie de maintenance à long terme.

Enfin, les frameworks intègrent souvent des fonctionnalités de sécurité par défaut. Ils protègent contre des vulnérabilités courantes, comme les attaques XSS (Cross-Site Scripting) ou CSRF (Cross-Site Request Forgery), en fournissant des mécanismes de prévention.

## Pourquoi utiliser Angular® en version 16 et Symfony® en version 6 alors que des versions plus récentes existent ?

Même si Angular® 17 et Symfony® 7 venaient de sortir au moment où ce projet entrait dans sa phase de développement, l'usage des versions immédiatement inférieures (respectivement 16.2.10 et 6.3) a été motivé par le fait de pouvoir disposer d'une base de connaissances solides et de méthodes éprouvées. En effet, avec une nouvelle version aussi récente, peu de contenus existaient, risquant de réduire ma vitesse en cas de problème.

### VI.I.I front-end avec Angular®

Plusieurs raisons ont motivé mon choix d'utiliser Angular® pour ce projet :

- ◆ Angular® offre une structure solide par modules et composants ce qui permet une réutilisabilité du code importante. Il offre aussi des fonctionnalités avancées pour la gestion des données, la navigation, le routage et la mise en œuvre de l'interface utilisateur.
- ◆ Outre les arguments cités précédemment (voir page précédente "Pourquoi utiliser des frameworks ?"), Angular® est basé sur le langage TypeScript®. En tant que surcouche de JavaScript®, ce langage offre une meilleure structuration du code grâce au typage statique. Ceci sécurise le code lors de la compilation et évite du temps de débogage.
- ◆ En tant que framework conçu par les équipes de Google®, Angular® dispose d'une vaste communauté de développeur d'une part et d'un support à long terme d'autre part.

- ◆ Bien que jugé difficile à apprendre, Angular® dispose d'une grande quantité de ressources en ligne pour faciliter son apprentissage.

### VI.I.2 back-end avec Symfony®

Concernant Symfony®, j'ai décidé d'utiliser ce framework pour les raisons suivantes :

- ◆ Symfony® est l'un des frameworks php les plus anciens et bien établi. À ce titre il dispose d'une grande communauté en soutien. Par ailleurs, la documentation officielle de Symfony® est complète et bien faite, ce qui en facilite l'apprentissage.
- ◆ Ensuite, pour sa structure MVC. Symfony® offre en effet une architecture de type Model View Controller (MVC) ce qui permet de séparer clairement la logique, la présentation et les requêtes. Cela favorise également une organisation propre du code pour une meilleure maintenabilité.
- ◆ Symfony® intègre de plus des fonctionnalités de sécurité comme l'encodage des mots de passe, la protection contre les injections SQL ou encore la randomisation des id utilisateurs (Universal User ID ou UUID), etc.
- ◆ L'environnement de Symfony® fournit des outils de test intégrés tels que PHPUnit et Behat.
- ◆ Enfin, Symfony® dispose d'une bonne scalabilité. C'est-à-dire qu'il est conçu pour évoluer avec les besoins du projet, notamment en cas de forte croissance du trafic et des fonctionnalités.

### VI.I.3 API Platform®

L'utilisation de la solution API Platform® allait de pair avec le framework Symfony®. En effet, faisant partie de l'environnement de Symfony®, API Platform® facilite le développement d'API REST (ou GraphQL). API Platform® utilise les annotations de Symfony® et fournit une configuration par défaut pour créer rapidement des points de terminaison.

Par ailleurs, les points suivants sont venus appuyer mon choix :

- ◆ API Platform® inclut une validation automatique des requêtes entrantes. Ce qui simplifie la gestion des erreurs et respecte les standards du web tels que LD et JSON (LD+JSON) pour la sémantique des données et Swagger/OpenAPI pour la documentation.
- ◆ Cette solution offre également des fonctionnalités de gestion de la persistance des données par l'ORM Doctrine, également intégré à Symfony®.
- ◆ Les API créées avec API Platform® peuvent être facilement utilisées dans les applications Angular® en utilisant les Services HTTP (HttpClient) pour consommer les données et ainsi interagir avec le back-end.
- ◆ Enfin, API Platform® facilite la mise en place de stratégies de sécurité de l'API pour contrôler l'accès aux ressources *via* des mécanismes d'authentification et d'autorisation.

En adoptant cette approche, j'ai voulu séparer la logique métier (dans Symfony®) de la présentation (dans Angular®) et utiliser Symfony® comme fournisseur de services API. Cette méthode permet également de satisfaire à l'exigence d'évolutivité demandée dans le cahier des charges MVP.

### VI.2 Architecture d'application et environnement de développement

Ce projet répond aux standards actuels du Web. Aussi a-t-il été conçu de façon à répondre aux exigences du Pattern de conception Model-View-Controller (MVC).

Cependant, dans le cadre du framework Angular®, un autre Pattern de conception a été utilisé : le Model-View-ViewModel (MVVM).

#### VI.2.1 Modèle MVC et MVVM

Deux grands modèles d'architecture ont été utilisés durant ce projet, à savoir : le modèle MVC et le modèle MVVM.

##### Architecture MVC

Le Design Pattern MVC (Model-View-Controller) est un modèle architectural largement répandu dans le développement web (comme logiciel). Ce Pattern organise et sépare les différentes strates d'une application. Le but étant de faciliter la maintenance, mais aussi son évolutivité, tout en assurant une bonne lisibilité du code source.

Cette Architecture a été particulièrement représentée dans le framework Symfony®, où ce pattern se traduit comme suit :

- ◆ Model (Modèle) : dans Symfony®, la partie modèle gère la couche logique de l'application ainsi que la communication avec la base de données. Le modèle ne dépend pas de la vue ou du contrôleur. Il est indépendant.
- ◆ View (Vue) : la vue correspond à la couche d'interface utilisateur (UI). Son rôle est d'afficher

les données du modèle pour permettre aux utilisateurs d'interagir avec l'application.

Cette couche ne devrait jamais contenir de logique métier et se contenter d'afficher les informations.

- ◆ Controller (Contrôleur) : agit comme une sorte d'API entre la vue et le modèle. Le contrôleur gère les interactions de l'utilisateur et met à jour le modèle et/ou la vue en conséquence.

## Architecture MVVM

À l'instar du MVC, le modèle MVVM offre une structure de code qui sépare les données, la logique métier de l'affichage, c'est-à-dire l'interface utilisateur.

Bien qu'Angular® dispose de ses propres concepts architecturaux comme les services, les composants ou les directives, l'utilisation du pattern MVVM, permet de garantir une séparation nette des responsabilités entre le modèle, la "vue (template)" et le "vue-modèle (logique)".

Cette répartition facilite la maintenance et la responsabilisation du code comme stipulé par les principes de programmation SOLID (voir VI.4.2 p.30 "POO et principes SOLID").

En outre, cette organisation permet de réutiliser les modules et les composants et de les tester plus facilement, ce qui contribue à une meilleure qualité logicielle et répond au principe SOLID<sup>1</sup> dit de Liskov (voir VI.4.2 p.30 "POO et Principes SOLID").

Dans le cadre d'Angular®, l'architecture MVVM se détaille ainsi :

- ◆ Model (Modèle) : il représente généralement les données récupérées à partir des services, souvent stockées dans des variables ou des objets au sein des composants.
- ◆ View (Vue) : les fichiers de modèles (templates) sont utilisés pour créer la vue. Ces fichiers définissent la structure de l'interface utilisateur et affichent les données du modèle.
- ◆ ViewModel (Vue-Modèle) : dans le cadre d'Angular®, le rôle du vue-modèle est souvent joué par les composants. Les composants interagissent avec les modèles de données, exposent les données au template et gèrent les événements de l'interface utilisateur. Les méthodes et les fonctions définies dans les composants (viewmodel) sont utilisées pour manipuler les données et mettre à jour le Modèle.

### VI.2.2 Base de données

#### Dictionnaire des données

Un dictionnaire des données est une documentation détaillée qui décrit les entités (tables), les attributs (colonnes), les relations, les règles de validation et d'autres informations sur les données dans une base de données.

Le dictionnaire des données est utilisé pour comprendre la structure de la base de données, pour la documentation, la communication entre les équipes de développement et pour assurer

<sup>1</sup>. Les principes SOLID sont un ensemble de 5 principes de conception qui visent à faciliter la programmation logicielle pour rendre le code source plus évolutifs, flexibles et faciles à maintenir. Ces principes ont été introduits par Robert C. Martin dans son ouvrage « Clean code ».

la cohérence dans la manipulation des données.

La figure ci-dessous montre un extrait du dictionnaire de données du projet. Le tableau complet est consultable en annexe n°9.

Aa Data	Type	Lenght	DB code	Exemple
<u>Article content</u>	LONGTEXT	/	content	Article complet
<u>Tag_id</u>	INT	/	idtag	4
<u>Mot clé</u>	VARCHAR	70	keyword	strange
<u>Category_id</u>	INT	/	idcategorie	5
<u>Category_name</u>	VARCHAR	70	name	discovery
<u>Newsletter_id</u>	INT	/	idnewsletter	6
<u>Newsletter_creation</u>	DATETIMEIMMUTABLE	/	createdAt	20/02/2023   12:00
<u>Newsletter_email</u>	VARCHAR	70	email	simonin@int.dev

Figure I3. Dictionnaire de données

### Matrice de dépendances fonctionnelles

Une matrice des dépendances fonctionnelles (MDF) répertorie les dépendances entre les attributs dans une base de données relationnelles. Elle identifie les relations de dépendance entre les attributs, ou la valeur d'un attribut. Elle est utilisée pour normaliser une base de données en éliminant anomalies. Les clés primaires sont matérialisées par un symbole (\*) alors que les différents éléments associés sont représentés par un chiffre (I). Les clés étrangères quant à elles sont identifiées par un élément présent dans plusieurs colonnes.

La figure ci-dessous montre un extrait de la Matrice de dépendances fonctionnelles du projet. Le tableau complet est consultable en annexe n°9.

Aa DB code	User	User Connexion	Article	Tag	Category	Newsletter	Comment
<u>idcategorie</u>				*			
<u>name</u>				1			
<u>idnewsletter</u>				*			
<u>createdAt</u>				1			
<u>email</u>				1			
<u>idcomment</u>					*		
<u>createdAt</u>					1		
<u>updatedAt</u>					1		

Figure I4. Matrice de dépendances fonctionnelles

### Schéma de base de données

Ces différents outils ont permis de modéliser la base de données. Dans un premier temps, j'ai tout d'abord réfléchi sur papier aux relations et aux différents rôles que ce blog allait comporter, conformément au cahier des charges MVP. Les différentes tables de la base de données dans ce projet sont classées en deux catégories : les tables en relation et indépendantes.

À l'aide des documents précédemment générés, j'ai pu traduire ce schéma dans MySQL® WorkBench® en utilisant la fonctionnalité graphique du logiciel. L'éditeur permet de visualiser directement les tables et leurs relations, puis de générer les requêtes SQL nécessaires à leur création. Ce schéma est consultable en annexe n°10.

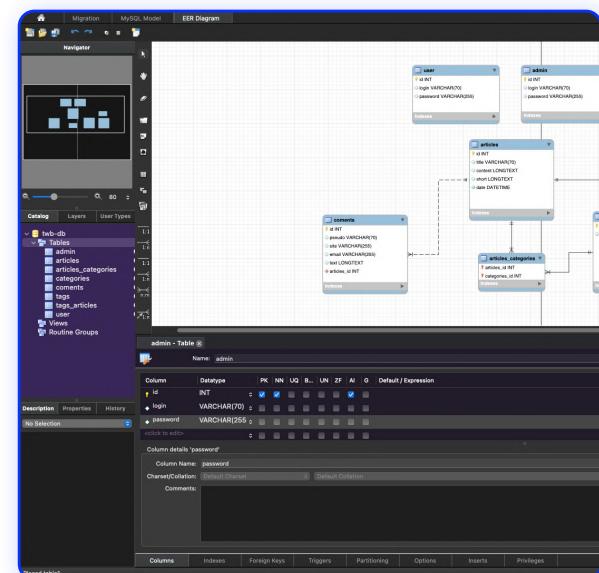


Figure I5. MySQL WorkBench®

### VI.2.3 Environnement de développement

L'environnement de développement regroupe l'ensemble des outils utilisés pour réaliser l'application. Dans le cadre de ce projet, les outils étaient classés en deux catégories : conception et développement.

## Conception

- ◆ Figma® : prototypage de l'application et réalisation des diagrammes UML.
- ◆ MySQL WorkBench® : pour le schéma de base de données.
- ◆ Notion® : pour les écrits du projet.

## Développement

- ◆ Visual Studio Code® : l'IDE m'ayant permis d'utiliser les frameworks Angular® et Symfony®.
- ◆ Angular® : framework TypeScript® front-end.
- ◆ Symfony® : framework php back-end.
- ◆ API Platform® : a servi à générer les points de terminaison REST nécessaires pour faire communiquer le front et le back-end.
- ◆ Karma/Jasmine et Jest : pour la réalisation des tests unitaires.
- ◆ Cypress® : une solution de test end-to-end.
- ◆ BurpSuite® : pour les tests de sécurité.

## VI.3 Domaine et hébergement

La partie hébergement de la solution a nécessité une étude spécifique. En effet, le site devant évoluer selon l'avancement du développement du produit, comme stipulé dans le cahier des charges MVP (voir III.I.3 p.7 "Évolutions envisagées"), l'hébergement devait répondre également à certaines exigences techniques notamment en matière de scaling <sup>1</sup>, mais aussi en termes de base de données et de compatibilité avec

les frameworks utilisés. Ainsi, les solutions suivantes ont été étudiées : hébergement mutualisé, Virtual Private Serveur (VPS), et Cloud Hosting.

Dans le cadre de ce projet, la solution Cloud avec Amazon Web Service (AWS) ou l'utilisation d'un VPS était préconisée (notamment au regard de la scalabilité). Cependant, dans un contexte de budget - très - contraint en raison de la jeunesse du Studio, la solution avec hébergement mutualisé chez OVH a été privilégiée.

Dans ce type d'hébergement, plusieurs sites web partagent les mêmes ressources d'un serveur, puissance de calcul, la bande passante et d'autres ressources du même serveur physique. Un site hébergé dans ce type d'hébergement est isolé dans un container. Les performances étant partagées, si l'un des clients prend plus de ressources, il pénalise les autres « colocataires ».

Toutes les ressources étant utilisées de manière prévisible, la facturation l'est donc tout autant ce qui permet un contrôle du budget important. En outre, ce type d'hébergement est souvent la meilleure alternative pour les projets modestes disposant d'un budget contraint.

## VI.4 Exigences de programmation

Pour satisfaire aux exigences du cahier des charges MVP du projet, un certain nombre de points étaient à respecter. Cette qualité a notamment été suivie par l'intermédiaire d'un plan de qualité (voir IV.4 p.XX "Plan de qualité"). Les points les plus importants sont détaillés ci-dessous.

<sup>1</sup>. Il s'agit de la mise à l'échelle de l'infrastructure. Il existe 2 types des scaling (dimensions) : horizontal ou vertical. Vertical : ajout de composants (ram, dd, CPU, etc) pour ajouter des ressources. Horizontal : ajoute des machines les unes à la suite des autres pour encore plus de puissance.

## VI.4.I Convention et documentation

### Convention de nommage

Une convention de nommage aide à rendre le code plus lisible, cohérent et facilement compréhensible pour les développeurs qui seront amenés à intervenir tout au long de la vie du produit. Dans le cadre de ce projet, une convention de nommage globale a été mise en place. Celle-ci était subdivisée en plusieurs chapitres. Chaque chapitre représentant une partie distincte du projet (Angular®, Symfony® et styles). La convention complète est disponible en annexe n°II du présent dossier.

#### Convention de Nommage Symfony

Cette Convention de Nommage suit la Convention en vigueur sur le Framework Symfony et son écosystème. Celle-ci utilise également la Convention du langage PHP, langage sur lequel est basé ce Framework.

##### Les fichiers et dossiers

###### Fichiers des Entities, Fixtures et Repositories :

Les noms des fichiers seront en `PascalCase`, reflétant le nom de la classe correspondante.

Exemple : `ApiUserController.php`, `ProductDetails.php`, `NewsletterRepository.php`

###### Noms de dossiers :

Conservez la structure des dossiers selon la fonctionnalité ou le domaine, en utilisant également `PascalCase` pour refléter le nom de la classe principale.

Exemple : `/userManagement`, `/ProductCatalog`, `/SharedComponents`

###### Fichiers des Tests Unitaires :

la collaboration entre les membres de l'équipe de développement par un support commun. Elle facilite également le transfert de connaissances lors de l'intégration de nouveaux membres dans l'équipe ou lors de la passation du projet à un autre développeur et réduit le risque d'erreurs lors de l'évolution ou de la maintenance du code existant.

En plus de ce support, la code base du projet a été commentée tout au long du processus de développement (voir VII p.36 "Réalisations du Candidat"), notamment pour expliciter la logique des méthodes employées.

## VI.4.2 POO et Principes SOLID

### POO

Outre les concepts de Programmation Orientée objet (POO) les plus élémentaires tels que l'utilisation des classes, des objets et des méthodes, les principes listés ci-dessous ont également été largement utilisés durant ce projet pour organiser, structurer et concevoir une code base robuste et pérenne :

- ◆ L'abstraction : les détails d'implémentation ont été cachés au maximum pour ne donner aux objets que les données qui leur étaient réellement utiles. Par exemple en utilisant le mécanisme d'injection de dépendances dans Angular® par l'intermédiaire d'un constructeur : `constructor(private http: HttpClient){}`, où les détails du service HttpClient sont masqués, mais néanmoins utilisables dans l'objet.
- ◆ L'Encapsulation : les données et les méthodes qui manipulent la classe sont regroupées et per-

Figure I6. Convention de nommage

### Documentation du projet

La documentation du présent projet décrit le fonctionnement, l'utilisation et la structure de l'application. Ce document est crucial pour donner le même niveau d'information à tous les développeurs intervenant sur le projet. Une documentation précise permet en outre de mieux comprendre la code base et favorise

mettent de contrôler l'accès à ces données. Par exemple, dans le cadre d'un controller Symfony® `private Custom ApiService $custom ApiService`. Ici, l'attribut `$custom ApiService` est déclaré comme "private", ce qui le rend accessible uniquement à l'intérieur de la classe NewsletterController.

- ◆ L'Héritage : permet à une classe "Fille" (Enfant) "d'hériter", c'est-à-dire de pouvoir disposer, utiliser les attributs et les méthodes de sa classe "Mère" (Parente). Dans cet exemple : `class NewsletterController extends AbstractController{}`, la classe NewsletterController hérite de la classe AbstractController de Symfony®. Ce mécanisme permet à NewsletterController de bénéficier des fonctionnalités fournies par AbstractController.
- ◆ Le Polymorphisme<sup>1</sup> : le polymorphisme se réfère à la capacité des objets de classes différentes à être manipulés par le biais d'une interface ("contrat" spécifiant les différentes méthodes qu'une classe doit implémenter). En d'autres termes, faire usage de méthodes avec le même nom, mais avec des implementations spécifiques à chaque classe.
- ◆ La Généricité : consiste à créer des méthodes ou des classes "génériques", c'est-à-dire ne se souciant pas du type de données qu'elles utilisent, dans le but d'être réutilisables. Par exemple avec les Observables dans Angular® : `getUserList(): Observable<User[]>` où le `<User[]>` dans `Observable<User[]>` indique le type "générique" de données que l'Observable émettra. Ici, il s'agit d'un tableau (array) de user, mais il pourrait s'agir de tout autre type de retour.

## Les principes SOLID

Tout au long du développement de cette application, tant au niveau front-end que back-end, je me suis attaché à respecter au maximum les principes SOLID. Au nombre de cinq, ils représentent des grands principes de conception à suivre dans toute base de code. Ils permettent de concevoir une application "dans les règles de l'art", c'est-à-dire réutilisable, maintenable, évolutive et faiblement couplée :

- ◆ Single-responsibility (responsabilité unique) : le principe de responsabilité unique stipule qu'un composant doit avoir un objectif ou une responsabilité claire. Il doit se concentrer sur une seule fonctionnalité ou un seul comportement.
- ◆ Open-closed (ouvert/fermé) : ce principe signifie que les composants doivent être ouverts pour ajouter de nouveaux comportements ou fonctionnalités, mais fermés pour que le code existant reste inchangé.
- ◆ Liskov Substitution (substitution de Liskov (LSP)) : ce principe constitue l'un des fondamentaux de la POO. Il met l'accent sur le fait que les composants enfants devraient pouvoir remplacer leurs composants parents sans affecter le comportement de l'application.
- ◆ Interface Segregation (ségrégation des interfaces (ISP)) : ce principe suggère qu'un code ne devrait pas dépendre de méthodes qu'il n'utilise pas. En outre, les interfaces doivent être ciblées et adaptées aux besoins spécifiques du client et ne pas mettre en œuvre des fonctionnalités inutiles.

---

<sup>1</sup>. Il existe deux types de polymorphismes en POO : statique (une classe a plusieurs méthodes avec le même nom, mais des signatures différentes) et dynamique (les objets de la classe dérivée peuvent être traités comme des objets de la classe de base). En Angular®, le polymorphisme est présent lorsque l'on travaille avec des services ou des composants qui utilisent des interfaces communes.

- ♦ Dependency Inversion (inversion de dépendance (DIP)) : ce principe indique qu'une entité devrait toujours dépendre d'abstractions. En d'autres termes, que les composants de haut niveau ne devraient pas dépendre des composants de bas niveau pour favoriser un couplage faible et augmenter la modularité et faciliter la maintenance de l'application.

pour coordonner l'accès à une ressource partagée (par exemple, pour assurer la connexion à la base de données - voir VII p.36 « Réalisations du candidat »).

Il s'agit d'une méthode statique qui retourne toujours la même instance de la classe. J'ai également utilisé un constructeur avec modificateur d'accès "private" pour empêcher l'instanciation directe.

#### VI.4.3 Design patterns

Les design patterns sont des structures de code éprouvées qui offrent des approches pour résoudre des problèmes connus. Ces outils permettent de concevoir des applications avec une code base robuste. Divisés en trois grandes familles (creational, structural et behavior) l'utilisation des designs patterns constitue une bonne pratique à suivre en développement web.

Comme dit précédemment, plusieurs patterns de conception ont été utilisés dans ce projet pour satisfaire aux exigences de qualité attendues.

Outre les patterns MVC et MVVM utilisés respectivement avec Symfony® et Angular® (voir VI.2.1 p.26 « Modèles MVC et MVVM »), les modèles de conception Singleton et Facade ont également été utilisés :

##### Singleton

Le Singleton est un design pattern creational. Son rôle est de contrôler l'instanciation d'une classe pour s'assurer qu'il n'y a qu'une seule instance de cette classe en mémoire pendant toute la durée de l'application. Dans le cadre de ce projet, j'ai utilisé ce pattern dans des situations où une seule instance d'une classe est nécessaire

##### Facade

Le Facade est un design pattern structural qui fournit une interface simplifiée pour interagir avec le système, qui lui est complexe. En d'autres termes, le design pattern Facade est une couche qui masque la complexité interne du programme tout en offrant une interface simple qui permet une meilleure interaction avec le système.

Ce pattern a essentiellement été utilisé avec Angular®, notamment pour fournir un point d'entrée unique à l'application. Il s'agit d'une classe TypeScript® qui contient les méthodes qui coordonnent les interactions entre les différents composants de l'application (voir VII p.36 « Réalisations du candidat »).

#### VI.5 Accessibilité

L'accessibilité était un autre point d'importance présent dans le cahier des charges de l'application. Dans le cadre de ce projet, l'accessibilité visait à développer l'application de manière qu'elle soit utilisable par toutes les personnes, y compris par des personnes en situation de handicap (incapacités physiques, sensorielles voire cognitives).

Durant la conception, je me suis attaché à respecter au maximum les principaux points évoqués dans le RGAA et la norme internationale WCAG (voir III.2.2 p.8 "Contraintes techniques") :

- ◆ Présenter les contenus de manière claire et perceptible en utilisant un contraste de couleur adéquat. Cela inclut l'usage de peu de couleurs (une dominante et deux teintes adjacentes), une police d'écriture facilement lisibles et des balises HTML de médias alternatifs telles que "alt" qui fournit une description des images afin que les utilisateurs non ou mal voyants puissent comprendre le contenu.
- ◆ Faire en sorte que tous les utilisateurs puissent naviguer et interagir avec le site facilement, notamment grâce à une interface utilisateur disposant de peu d'éléments et d'une grande surface d'interaction et en prenant en compte la navigation au clavier.
- ◆ Développer l'application de manière qu'elle soit compatible avec les technologies d'assistance, notamment en intégrant des balises ARIA. Les attributs ARIA (Accessible Rich Internet applications) ajoutent des informations sémantiques à des éléments HTML pour faciliter l'utilisation des technologies d'assistances telles que les Lecteurs d'écran (JAWS ou VoiceOver), les navigateurs vocaux ou encore les claviers alternatifs.

## VI.6 Sécurité

En développement web, la sécurité concerne la protection des systèmes, des données et des utilisateurs contre les menaces potentielles telles

que les cyberattaques, les fuites de données ou les failles de sécurité. Cela implique la mise en œuvre de mesures et de bonnes pratiques pour prévenir les vulnérabilités et assurer la confidentialité, l'intégrité et la disponibilité des données.

En matière de sécurité de l'application IFT, j'ai eu une vigilance toute particulière sur les points suivants :

- ◆ Informations sensibles : les messages d'erreurs ou d'avertissemens ne contiennent pas d'informations sensibles sur l'application ou les utilisateurs.
- ◆ Injection SQL : les requêtes SQL n'ont jamais été construites avec des informations utilisateurs. De plus, l'instruction "Prepared Statement" a été utilisée dans les différents framework chaque fois que nécessaire. L'utilisation de caractères spéciaux dans les données a été minimisée.
- ◆ CSRF ("seasurf") : des tokens anti-CSRF ont été utilisés dans les frameworks pour empêcher les attaques de type Cross-Site Request Forgery et vérifier l'origine des requêtes.
- ◆ URL avec ID utilisateur : les données d'identification des utilisateurs, comme l'ID par exemple, ne figure jamais dans les routes (URL). Les identifiants (ID) ont été constamment randomisés par l'utilisation de la fonctionnalité UUID <sup>1</sup>.
- ◆ Contrôle côté serveur inexistant : tout message d'erreur affiché en Front possède son pendant côté serveur.
- ◆ Input : "*Ne jamais faire confiance à ce que saisit l'utilisateur*". Suivant ce principe, tout champ

<sup>1</sup>. UUID signifie « Universally Unique Identifier » (Identifiant Universel Unique en français). C'est une chaîne de caractères utilisés pour identifier de manière unique une ressource, objet ou entité dans un système informatique.

de type input bénéficie d'un contrôle des données saisies, notamment par l'utilisation d'expressions régulières. De plus, lors de toute modification (profil, données, etc), une vérification du rôle de l'utilisateur (super-admin, admin, user) a été effectuée.

- ◆ Mass Assignment : en matière de développement orienté objet, seuls les attributs nécessaires au fonctionnement d'une classe figurent dans celle-ci.
- ◆ XSS : tout caractère spécial présent dans le code de l'application a été échappé <sup>1</sup> pour éviter le Cross Site Scripting (XSS).

De plus, les scripts comprennent les méthodes de transformation des caractères spéciaux suivantes : HTMLEncode(), HTMLDecode() et EncoderForHtmlattribute().

- ◆ Récupération/modification de mot de passe contraignante : la récupération du mot de passe est contraignante. C'est-à-dire que le nombre d'étapes pour parvenir à la récupération/modification effective a été maximisé.

Par exemple, en mettant en place tout d'abord une action à effectuer par l'utilisateur (bouton) pour faire la demande de récupération/modification du mdp, puis un email de vérification est envoyé sur l'adresse email de l'utilisateur avec un lien spécifique autorisant la modification, ensuite une confirmation de modification sera envoyée par email et enfin, une demande de reconnexion sera demandée à l'utilisateur pour achever le processus.

- ◆ Randomisation : la randomisation des données est conforme au "Secure Random" proposé par les frameworks.

- ◆ Component Vulnerability : les dépendances du projet ont été mises à jour dès que nécessaire (voir IX p.57 "Description d'un cas de veille sécuritaire : accès aux données via Doctrine").
- ◆ Contrôle du work flow : tous les cas d'usages (use cases) du parcours utilisateur (user flow) ont été pris en compte pour éviter que l'utilisateur ne puisse outrepasser une étape de contrôle.
- ◆ Cookie JWT : les cookies JSON Web Tokens ne contiennent pas d'informations sensibles, même chiffrées, telles que les mots de passe par exemple. Ils comportent en revanche une date d'échéance et ont été configurés avec l'attribut "HTTPONLY" pour éviter les attaques XSS. Par ailleurs, l'utilisation du protocole HTTPS a été mis en place au niveau de l'hébergement.
- ◆ Unprotected Primary Channel : l'accès "Admin" n'est pas disponible depuis n'importe quelle route du site. De plus, aucune autre URL Admin additionnelle n'est présente dans l'URL conduisant à l'Admin (ex : xxx/admin/admin/admin.html) afin de garantir la bonne protection de toutes les vues de l'application. Remarque : dans l'idéal, la page Admin aurait dû être placée derrière un VPS, mais le manque de budget alloué au projet n'a pas permis la mise en place de cette protection.
- ◆ SSRF : pour éviter les attaques de type serveur Side Request Forgery, chaque requête au serveur est vérifiée pour que la destination de réponse soit bien conforme et attendue par le serveur.
- ◆ Journalisation des évènements : un journal des logs de connexions et des différents évènements utilisateurs a été mis en place. Ce dernier est chiffré pour garantir la confidentialité des utilisateurs.

## VI.7 Maintenabilité et évolutivité

### Maintenabilité

La maintenabilité de l'application, c'est-à-dire sa capacité à être maintenue à jour selon les standards actuels du code, a été optimisée en organisant le code en modules logiques pour séparer les fonctionnalités.

De plus, la logique métier a été répartie dans des services pour éviter la duplication du code et ainsi permettre des modifications plus faciles.

Enfin, le code a été documenté par le biais de commentaires et de supports spécifiques pour que d'autres développeurs puissent rapidement s'approprier l'application dans un but de maintenance.

### Évolutivité

L'évolutivité renvoie à la notion d'augmentation de la charge ou du volume de données sans compromettre les performances de l'application. Ainsi, une architecture évolutive permettra au site de croître de manière efficace en réponse à la demande croissante des utilisateurs. Bien que cette notion concerne essentiellement l'hébergement (scalabilité horizontale ou verticale), elle concerne également la code base. Par exemple, des bibliothèques comme RxJS ont été utilisées pour la gestion d'états ou encore API Platform® pour les requêtes à la base de données, facilitant ainsi la gestion des données à mesure que l'application évoluera.

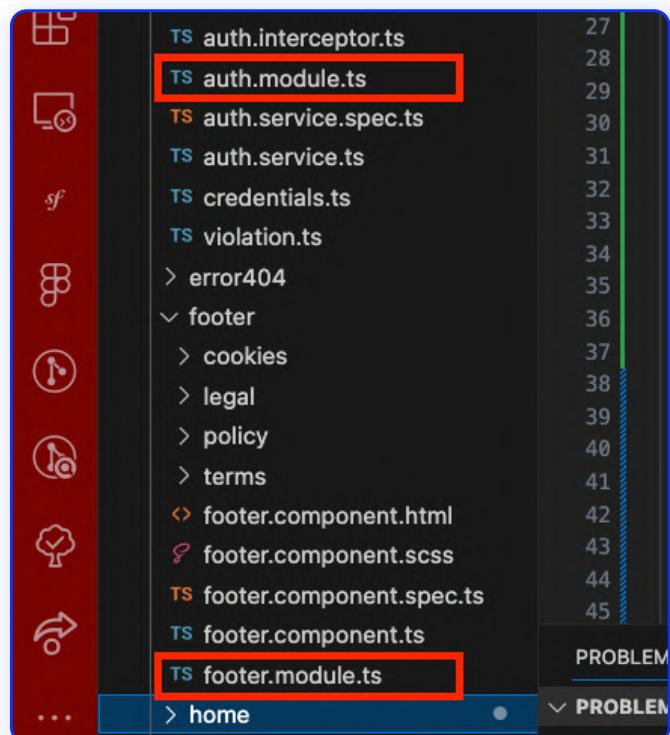


Figure I7. Organisation modulaire

```
detail-article
edit-article
form
hero
list-article
loader
more-button
news
newsletter
border-article.directive.spec...
border-article.directive.ts
card.service.spec.ts
card.service.ts
category-article.pipe.ts
home.component.html
home.component.scss
home.component.spec.ts
home.component.ts
home.module.ts M
mock-article-list.ts
model
navigation
navbar
searchbar
navigation.module.ts
```

```
35   { path: 'form/articles', title: 'Form Article' };
36   { path: 'edit/articles/:id', title: 'Edit Article' };
37   { path: 'articles/:id', title: 'Hero Article' };
38   { path: '**', title: '404 Error' };
39 ];
40
41
42 @NgModule({
43   declarations: [
44     HomeComponent,
45     HeroComponent,
46     NewsComponent,
47     ListArticleComponent,
48     DetailArticleComponent,
49     AddArticleComponent,
50     EditArticleComponent,
51     NewsletterComponent,
52     LoaderComponent,
53     BorderArticleDirective,
54     MoreButtonComponent,
55     ArticleFormComponent,
56     ArticleCategoryColorPipe,
57     CreateFormComponent,
58   ],
59   imports: [
60     CommonModule,
61     BrowserModule,
62     FormsModule,
63     ReactiveFormsModule,
64     NavigationModule,
```

Figure I8. Le module regroupe les composants réutilisables

I. L'échappement de caractères est un concept utilisé en informatique pour inclure des caractères spéciaux dans des chaînes de caractères avec symbole « \ » (antislash).

# VII. RÉALISATIONS DU CANDIDAT

Cette section présente les différentes réalisations du projet. Elle est organisée par thématiques de travail (conception, front-end, back-end, API, performances et déploiement) et par fonctionnalités.

## Conception : prototypage

Comme précisé dans le chapitre V (voir p.22 « Interface utilisateur »), un prototype haute fidélité a été réalisé avec Figma®. Ce support a servi

aux tests utilisateur, préalables au développement. Trois itérations ont été réalisées. Seule la version finale est présentée ici.

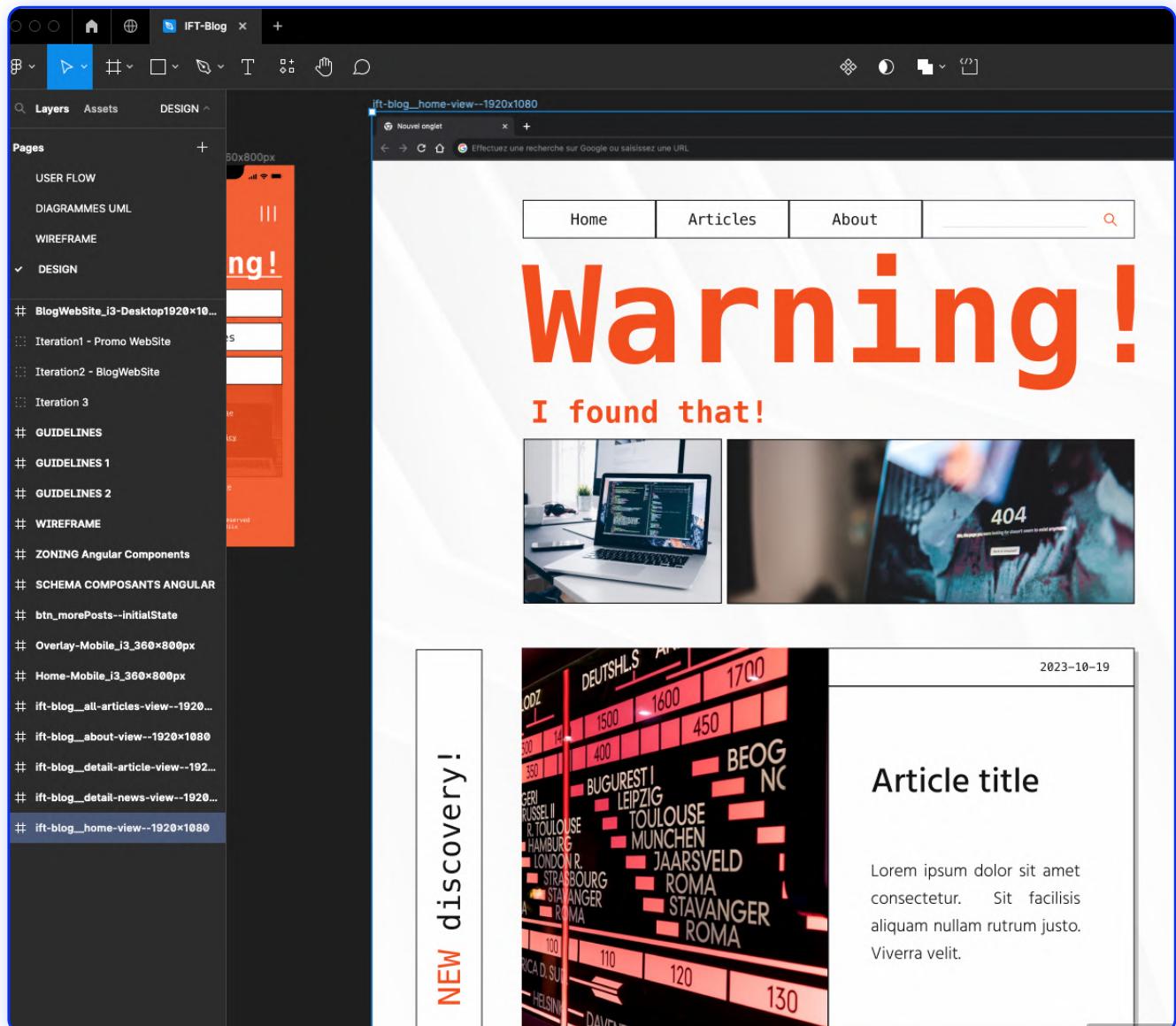


Figure I9. Conception de la page home avec Figma®

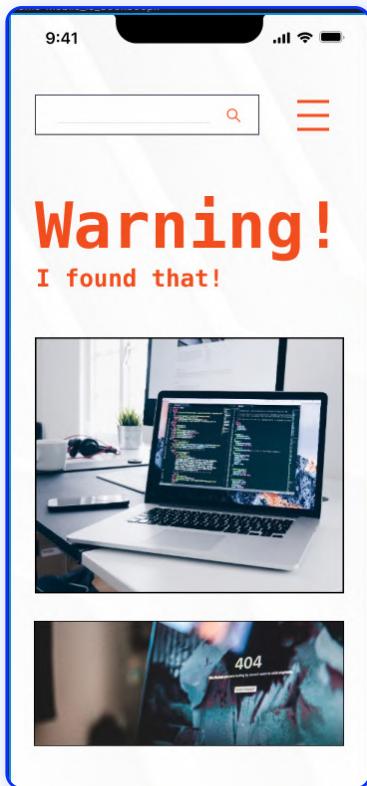


Figure 20. Page home



Figure 21. Exemple d'article



Figure 22. Menu overlay mobile

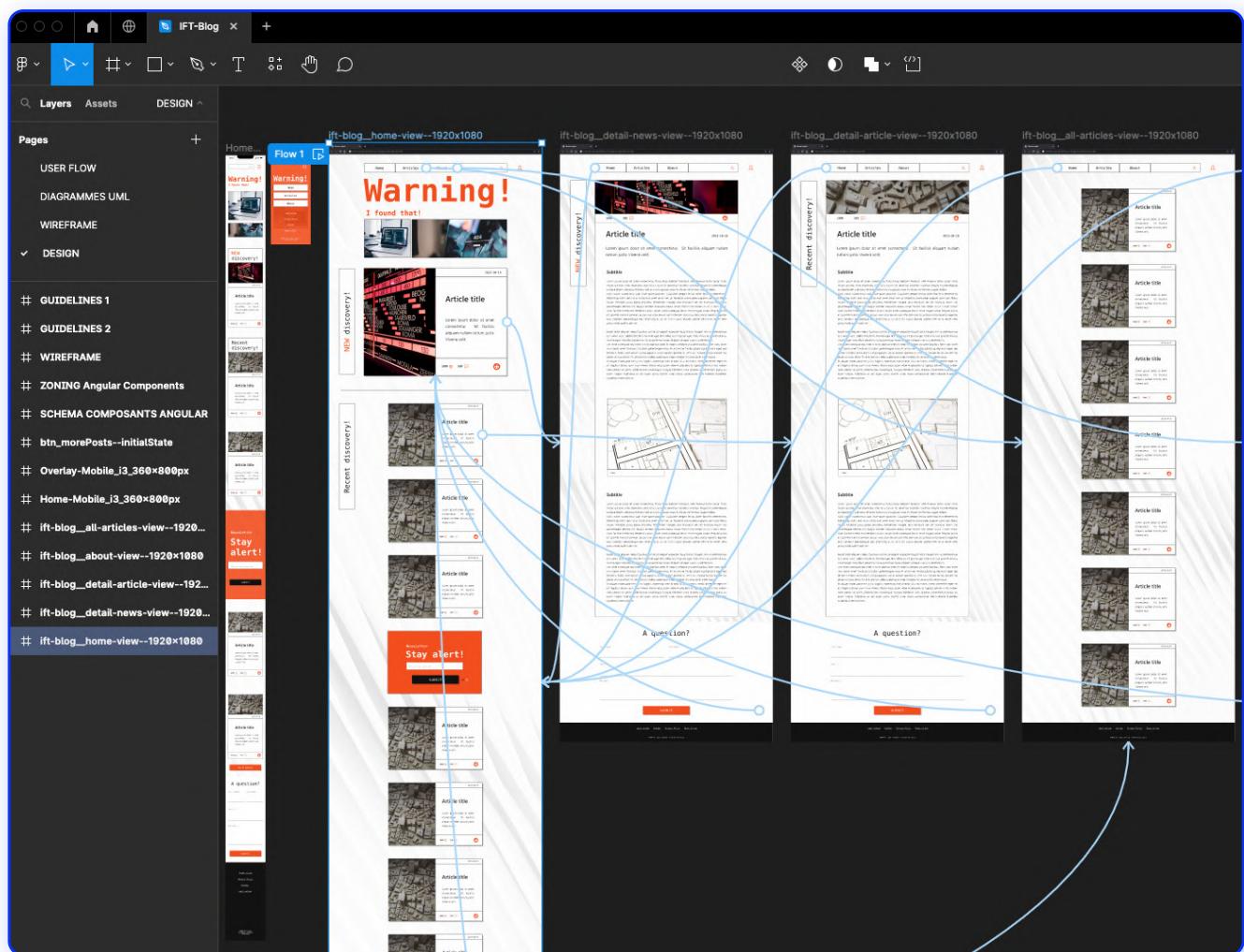


Figure 23. Mise en place des interactions

# Front-end Angular® : authentification

Mécanisme essentiel pour sécuriser les zones restreintes de l'application, l'authentification désigne le processus de vérification des autorisations de l'utilisateur. En d'autres termes, seuls les utilisateurs connus peuvent accéder à certaines fonctionnalités. Il comprend deux volets : login et register. Une fois qu'un utilisateur s'est identifié, le serveur génère un token d'authentification (JSON Web Token ou JWT). Ce token est ensuite utilisé pour authentifier les requêtes futures de l'utilisateur sans avoir besoin de demander une reconnexion.

La réalisation de cette partie a nécessité la création de plusieurs composants Angular®. Les composants sont les briques essentielles de toute application Angular®. Ils sont composés d'un template (vue html) et d'une logique (extensions). Ils appartiennent à un module qui les regroupe de façon cohérente.

## Module « auth »

Un module est une mécanique qui permet de regrouper des composants, des directives, des pipes et des services liés entre eux. Ils permettent d'organiser le code par fonctionnalité. Outre le fait qu'ils rendent les dépendances lisibles (« imports »), ils permettent l'encapsulation des données et d'utiliser le lazy loading d'Angular® : le module n'est chargé que si l'utilisateur en a besoin, ce qui permet de meilleures performances et répond aux principes d'éco-conception évoqués précédemment.

La gestion de l'authentification côté front a nécessité la conception des éléments suivants :

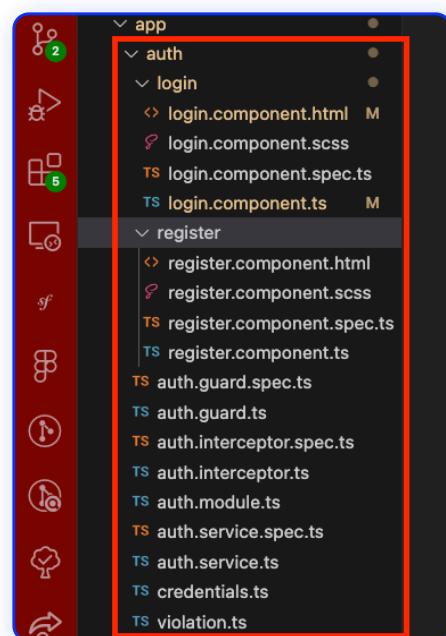


Figure 24. Éléments d'authentification

Les éléments les plus significatifs sont présentés et argumentés ci-après.



```
src
  app
    auth
      login
        login.component.html M
        login.component.scss S
        login.component.spec.ts TS
        login.component.ts M
      register
        register.component.html M
        register.component.scss S
        register.component.spec.ts TS
        register.component.ts M
      auth.guard.spec.ts TS
      auth.guard.ts M
      auth.interceptor.spec.ts TS
      auth.interceptor.ts M
      auth.module.ts TS
      auth.service.spec.ts TS
      auth.service.ts M
      credentials.ts M
      violation.ts M
    error404
    footer
  home
    add-article
    article-form
    card
    card-detail
    detail-article
    edit-article
```

11     const routes: Routes = [  
12       { path: 'login', component: LoginComponent },  
13       { path: 'register', component: RegisterComponent }  
14     ];  
15  
16     @NgModule({  
17       declarations: [ LoginComponent, RegisterComponent ],  
18       imports: [ CommonModule, SharedModule, ReactiveFormsModule, FormsModule, RouterModule.forChild(routes) ],  
19       exports: [ RouterModule, ReactiveFormsModule, FormsModule ]  
20     },  
21     providers: [  
22       { provide: HTTP\_INTERCEPTORS, useClass: AuthInterceptor, multi: true },  
23       { provide: Storage, useValue: window.localStorage }  
24     ])  
25  
26     export class AuthModule {}  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
999  
1000  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047<br

Figure 25. Module « auth »

## Service « auth »

En Angular®, les services sont des classes TypeScript qui centralisent des méthodes spécifiques ou des fonctionnalités réutilisables, comme l'accès aux données, la manipulation de données, les appels HTTP, etc. Ils permettent de faire appel aux méthodes qu'ils regroupent dans différentes classes de l'application. Ils sont par nature des "Singletons" (voir chapitre VI.4.3 p.32 « Design Patterns »).

L'utilisation d'un service permet de faire usage du principe d'abstraction en POO et de respecter le principe SOLID de Single Responsabiliby (voir chapitre VI.4.2 p.30 « POO et principes SOLID »).

Dans l'extrait de code ci-dessous, le service auth invoque le service HttpClient via le constructeur (injection de dépendance) :

```
private http: HttpClient
```

Celui-ci va permettre de requêter l'API :

```
this.http.post(USER_API, account)
```

La méthode `.pipe` est utilisée pour chaîner des opérateurs RxJS (bibliothèque qui gère les requête asynchrones) à l'observable `<AuthResponse>` renvoyée par la requête HTTP. L'opérateur `map` transforme le résultat de la requête (de type AuthResponse), stocke le token dans le `localStorage` et met à jour `authState` pour indiquer que l'utilisateur est connecté. `catchError` intercepte les erreurs et diffuse un message.

```
// User registration
register(account: { email: string; password: string; fullName: string }) {
  return this.http.post(USER_API, account);
}

// User authentication
authenticate(credentials: Credentials): Observable<AuthResponse> {
  return this.http.post<AuthResponse>(AUTH_API, credentials).pipe(
    map((result: AuthResponse) => {
      // Store the token in localStorage on successful authentication
      localStorage.setItem('token', result.token);
      // Update authState to indicate user is logged in
      this.authState.next(true);
      return result;
    }),
    catchError((error: any) => {
      throw new Error(error); // Throw an error for the calling code to
    })
  );
}

// User logout
logout() {
  localStorage.removeItem('token'); // Remove token from localStorage
  this.authState.next(false); // Update authState to indicate user is not
}
```

Figure 26. auth.service.ts

## Formulaire « login »

L'authentification sur le site était une partie fondamentale de ce projet. Dans le cadre de IFT-Blog, seul l'administrateur devait pouvoir poster, modifier ou supprimer des articles. Ce rôle spécifique est attribué par le biais d'une authentification par formulaire de login.

Le processus d'authentification côté back-end a été réalisé en premier lieu. Celui-ci est détaillé en page 46.

Concernant le front-end, j'ai tout d'abord entrepris de créer un composant Angular dédié, puis de réaliser le formulaire dans le template (vue) de ce composant. La protection des champs de texte a été assurée dans un premier temps par

l'intermédiaire d'expressions régulières (figure 28). Leur rôle est de vérifier les caractères entrés par l'utilisateur et ainsi prévenir les injections SQL (voir chapitre VI.7 p.33 « Sécurité »).

J'ai ensuite construit la logique sur cette base en utilisant le service FormBuilder. Cette classe utilitaire d'Angular simplifie la création des groupes et contrôles des formulaires (figure 29). De plus, FormBuilder offre l'accès à des méthodes avancées de protection des formulaires telles que les Validators. Ces fonctions permettent de personnaliser le contrôle des données entrées par l'utilisateur en précisant une expression régulière ou en utilisant un contrôle dédié, par exemple : **[Validators.email]** (figure 27).

```
form: FormGroup; // Angular Reactive Form for login inputs

constructor(
  private formBuilder: FormBuilder, // Service for building Angular form
  private auth: AuthService, // Authentication service
  private router: Router, // Angular router for navigation
  private ui: UIService // Service for managing UI state
) {
  // Initialize the form with validation rules
  this.form = this.formBuilder.group({
    email: ['', [Validators.required, Validators.email]],
    password: ['', Validators.required],
  });
}

// Function triggered on form submission
onSubmitFormLogin() {
  if (this.form.invalid) {
    return; // Stop execution if form is invalid
  }

  // Enable loading screen
  this.ui.activateLoading();

  // Call the authentication service to log in
  this.auth.authenticate(this.form.value).subscribe({
    next: (result) => {
      console.log('RESULT', result);
      // Disable loading screen
      this.ui.deactivateLoading();
      this.errorMessage = 'Error!';
      // Navigate to the home page on successful login
      this.router.navigateByUrl('/home');
    },
    error: (error) => {
      // Disable loading screen
      this.ui.deactivateLoading();

      if (error.status === 401) {
        // Handle unauthorized access error
        this.errorMessage =
      }
    }
  });
}
```

Figure 28. Utilisation d'une regex

```
<div class="form-group">
  <input
    type="email"
    formControlName="email"
    class="form-control"
    required
    [class.is-invalid]="form.get('email')?.invalid && for
    placeholder="Email address"
    aria-describedby="login-email-description"
  />
  <!-- Using ARIA description for help assistance technol
  <div id="Login-email-description" aria-hidden="true">
    Please, enter a valid email address into the text fie
  </div>
  <div class="invalid-feedback" *ngIf="form.get('email')?.
```

Figure 29. Utilisation de formControle

Figure 27. login.component.ts

## Authentification par tokens JWT

Le choix de la technologie d'authentification JWT (JSON Web Token) était basé sur la nature du projet. Ce blog ne traitant pas de données hautement sensibles (données bancaires par exemple), un format standardisé tel que JWT s'imposait.

En outre, les JWT permettent de transporter des informations d'authentification, mais également d'autorisation. Ce qui simplifie la gestion des droits d'accès nécessaires à l'utilisation de certaines fonctionnalités de l'application.

La gestion des tokens JWT côté front-end avec Angular se déroule comme suit :

- ♦ l'utilisateur est authentifié sur le site grâce au token obtenu côté serveur (figure 32).
- ♦ l'Interceptor Angular (voir page suivante) ajoute les entêtes (headers).
- ♦ le token est alors envoyé au serveur avec une requête HTTP. Cette requête est exécutée par le service Auth et HttpClient (voir p.39).
- ♦ le token est stocké de manière sécurisée dans le navigateur par l'intermédiaire du service LocalStorage (figure 32).
- ♦ le token expire après une certaine durée ou lorsque l'utilisateur se déconnecte. L'expiration peut-être gérée soit côté front-end, soit côté back-end avec Lexik dans le cadre de cette application.

Pour indiquer à l'utilisateur qu'il est connecté ou non, une icône spécifique dans la navigation change selon le statut de connexion (figures 31).

The screenshot shows a 'Sign in' form with two input fields: 'Email address' and 'Password'. Below the 'Password' field, there is an error message: 'Valid email address required!'. At the bottom of the form are two buttons: 'SIGN IN' and 'REGISTER'.

Figure 30. Formulaire login (form.component.html)

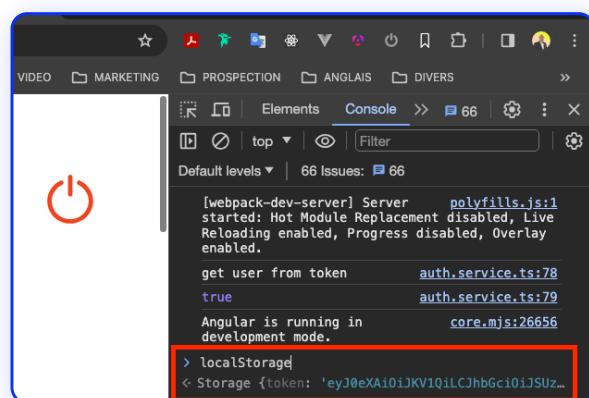


Figure 32. Présence du token de sécurité (connexion)

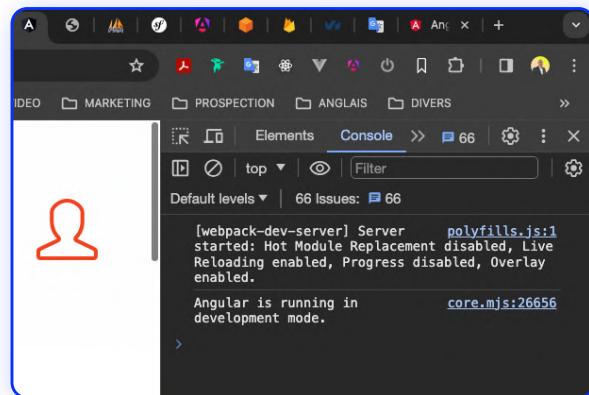


Figure 31. Utilisateur déconnecté

## Interceptor

Un Interceptor permet de manipuler les requêtes HTTP en ajoutant des entêtes (headers).

Les headers précisent l'objet de la requête (ex : Authorization ou Bearer pour l'authentification par token).

Dans ce projet, l'utilisation des Interceptors a permis d'assurer l'interception des requêtes HTTP sortantes. Cette étape permet d'ajouter un header d'autorisation contenant le token JWT lorsqu'un utilisateur est authentifié et ce, de façon automatique. Cela évite de répéter le même code d'ajout d'en-têtes dans chaque service ou chaque appel et assure une meilleure maintenabilité du code *in fine*.

Les Interceptors dépendent de l'interface **HttpInterceptor**. Cette interface expose une méthode **intercept()** qui prend une requête HTTP en paramètre **intercept(req: HttpRequest<any>, next: HttpHandler)** et renvoie une observable de la requête modifiée **Observable<HttpEvent<any>>** (figure 33).

```
import { Injectable } from '@angular/core';
import { HttpRequest, HttpHandler, HttpEvent, HttpInterceptor } from '@angular/common/http';
import { Observable } from 'rxjs';
import { AuthService } from './auth.service';

@Injectable()
export class AuthInterceptor implements HttpInterceptor {

  constructor(
    private auth: AuthService // AuthService Dependency Injection that gives a token as needed
  ) {}

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    // Check if the user is not authenticated
    if (!this.auth.isAuthenticated()) {
      // If not authenticated, proceed with the original request
      return next.handle(req);
    }

    // If authenticated, retrieve the token from the AuthService
    const token = this.auth.getToken();

    // Clone the request and add the Authorization and Bearer headers with the token
    const clonedReq = req.clone({
      headers: req.headers.append('Authorization', 'Bearer ' + token),
    });

    // Proceed with the modified request
    return next.handle(clonedReq);
  }
}
```

Figure 33. auth.interceptor.ts

## Guard

Dans Angular, un Guard désigne un moyen de protéger les routes suivant des conditions définies. Ces conditions peuvent être basées sur l'authentification, les autorisations, ou d'autres règles métier. En d'autres termes, un Guard contrôle le comportement du routeur.

Dans le cadre de IFT-Blog, les Guards ont été utilisés pour contrôler l'authentification ou les rôles. Ce cloisement des responsabilités permet une meilleure maintenabilité. Ainsi, AuthGuard assure le contrôle de l'authentification par l'intermédiaire de la méthode isAuthenticated() du service auth (figure 35), alors qu'AdminGuard contrôle le rôle administration avec la méthode isAdmin() (figure 34).

```
import { inject } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from './auth.service';

export const AdminGuard = () => {
  const authService = inject(AuthService);
  const router = inject(Router);

  // Check if the user is authenticated as an admin
  if(authService.isAdmin()){
    // If authenticated as an admin, allow access
    return true;
  }

  // If not authenticated as an admin, navigate to the home page
  router.navigate(['/home']);
  return false;
}
```

Figure 34. admin.guard.ts

○ ○ ○

```
import { inject } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from './auth.service';

// Refactored with Guard Function and inject(),
// because CanActivate class is now deprecated
export const AuthGuard = () => {
  const authService = inject(AuthService); // Make DI with inject() method
  const router = inject(Router);

  // Check if the user is authenticated using the AuthService
  if(authService.isAuthenticated()){
    console.log('AUTH GUARD', authService);
    // If authenticated, allow access to the route
    return true;
  }

  // If not authenticated, navigate to the login page and deny access
  router.navigate(['/login']);
  return false;
}
```

Figure 35. auth.guard.ts

# Front-end Angular® : article de blog

## Liste d'article et CRUD article

La fonction première du site IFT-Blog, conformément au cahier des charges MVP, était d'informer les internautes issus de la communauté cible. L'affichage d'une liste d'articles constituait donc une fonctionnalité essentielle de l'application.

Celle-ci a été développée en plusieurs étapes. J'ai tout d'abord réalisé la partie back-end avec Symfony®, puis j'ai procédé à la mise en place d'API Platform®, avant de développer la partie front-end avec Angular®.

La première étape consistait en la création d'une « entity » Article. Cette classe PHP® a été élaborée sur la base du schéma de base de données (voir annexe n°10). Elle construit l'objet Article à partir des attributs **\$id**, **\$title**, **\$chap**, **\$content**, **\$createdAt**, **\$updatedAt**, **\$category** et **\$user**.

À partir de cette entité, j'ai pu construire les points de terminaison REST avec API Platform®. Pour ce faire, l'attribut PHP® 8 **#**[[ApiRe-source](#)] a été configuré avec différentes opérations (figure 36), puis l'attribut **#**[[Groups](#)] a été apposé sur chacun des attributs de classe. Les Groups d'API Platform® permettent de gérer la sérialisation de chaque attribut de classe (figure 37).

Enfin, la partie front-end a elle-même été subdivisée en trois étapes. Dans un premier temps, j'ai réalisé la structure d'un article en utilisant le data binding avec les interpolations d'Angular `{{ article.title }}` pour les champs dynamiques ainsi que la directrice `*ngFor` pour créer une boucle nécessaire à l'affichage d'une liste en Angular (figure 38).

Puis j'ai réalisé la logique du composant « list-article » avant de terminer par les styles.

La dernière étape consistait en la création d'un service Angular. Nommé `article.service.ts`, ce fichier centralise toutes les méthodes utiles à une entité donnée et son contenu peut être appelé partout dans l'application par le mécanisme d'injection de dépendance.

Les méthodes CRUD (acronyme représentant les opérations de base que l'on peut effectuer sur des données : Create, Read, Update et Delete), ont été centralisées dans le service article (figures 39 et 40) pour respecter le principe SOLID de responsabilité unique (single responsibility) d'une part et d'autre part, pour permettre un appel plus aisé de ces méthodes ailleurs dans l'application.

Figure 36. End points REST avec API Platform®

```
58 #ORM\Column
59 #[Groups ([
60     'read:Article:item',
61     'read:Article:collection',
62     'read:Article:itemmp:public',
63     'read:Article:collection:public'
64 )]
65 private ?int $id = null;
66
67 #[ORM\Column(length: 30)]
68 #[Groups ([
69     'read:Article:item',
70     'read:Article:collection',
71     'read:Article:itemmp:public',
72     'read:Article:collection:public'
73 )]
74 #[Assert\Length(min: 2, max: 30, minMessage: "Title must be 30
75 private ?string $title = null;
```

Figure 37. Groups API Platform®

```

<!-- Displaying a container for each article using Angular's ngFor directive -->
<div
  class="container__articleSection"
  *ngFor="let article of articleList$ | async; let i = index; trackBy: trackByFn;"
  routerLink="/public_articles/{{ article.id }}"
  articleBorderCard
>
  <!-- Left column containing the main image of the article -->
  <div class="bloc__image-column">
    <img
      loading="lazy"
      class="img__article-main"
      [src]="imgArticleMain"
      alt="Image of a high-tech green and black 3D plane"
    />
  </div>

  <!-- Center column containing article details -->
  <div class="bloc__text-column">
    <div class="label__date">{{ article.createdAt | date : "dd/MM/yy" }}</div>
    <div class="label__update">{{ article.updatedAt | date : "dd/MM/yy" }}</div>
    <div class="label__article-title">{{ article.title }}</div>
    <div class="label__article-chapo">{{ article.chapo }}</div>
  </div>

```

Figure 38. Utilisation de la directive structurelle \*ngFor et des interpolations pour les champs dynamiques

```

// CRUD: Get collection
getArticleListFromDb(): Observable<Article[]> { // This method creates an Observable to listen for
the response from the database,
  return this.http
    .get<any>(`${this.apiUrl}/public_articles`)
    .pipe(map((response: any) => response['hydra:member'] as Article[])); // and decode the
database JSON Object into an array with map(),
}

// CRUD: Get
getArticleByIdFromDb(articleId: number): Observable<Article | undefined> {
  return this.http
    .get<Article>(`${this.apiUrl}/public_articles/${articleId}`)
    .pipe(
      map((response: any) => response as Article),
      tap((response) => this.log(response)),
      catchError((error) => this.handleError(error, undefined))
    );
}

```

Figure 39. Méthodes CRUD « Get » pour appeler un article et « Get Collection » pour appeler une liste d'articles

```

// CRUD: Post
addArticle(article: Article): Observable<Article> {
  const httpOptions = {
    headers: new HttpHeaders({ 'Content-Type': 'application/json' })
  };

  return this.http
    .post<Article>('api/articles/add', article, httpOptions)
    .pipe(
      // This Method return a Article Type Object with cast "<Article>" 
      // tap((response) => this.log(response)),
      catchError((error) => this.handleError(error, null))
    );
}

```

Figure 40. Méthodes CRUD « Post » pour créer un article dans la base de données

## Back-end Symfony® : login

### Login côté serveur avec Lexik

Lexik est un bundle Symfony® qui facilite l'utilisation de JSON Web Tokens (JWT) pour l'authentification côté serveur. Ce bundle réalise la création et la validation des tokens JWT, permettant ainsi la mise en place d'une authentification basée sur les tokens.

Ce type d'authentification est dite « state less », c'est-à-dire sans historique. L'ensemble des informations de connexion sont donc intégrées au token (identifiant, autorisations, mot de passe, etc). Dans le cadre de ce projet, l'authentification state less était préconisée en raison de l'utilisation d'une API REST avec API Platform.

Ces informations sont chiffrées et sont signées numériquement pour assurer leur intégrité. Elles sont réparties en trois parties : l'en-tête (header), la charge utile (payload) et la signature, puis inscrites sous la forme d'une chaînes de caractères encodées au format JSON.

Lexik utilise ce mécanisme à partir de l'entité « User » dans Symfony®, puis stocke les clés de signature (nécessaire au chiffrement et déchiffrement des informations) de façon sécurisée (figure 41).

La configuration de Lexik permet de préciser les clés publiques et privées ainsi que la durée de validité des tokens côté serveur (figure 42). Par exemple, pour faciliter les essais lors de la phase de développement, une durée de 86400 secondes avait été paramétrée pour la validité des tokens.

The screenshot shows the VS Code interface with the Explorer sidebar on the left and the Editor on the right. The Explorer sidebar shows a project structure with a 'jwt' folder expanded, containing 'private.pem' and 'public.pem' files, both of which are highlighted with a red box. The Editor shows the contents of the 'public.pem' file, which is a PEM certificate starting with '-----BEGIN PUBLIC KEY-----' and ending with '-----END PUBLIC KEY-----'. The 'private.pem' file is shown in a separate tab.

```
-----BEGIN PUBLIC KEY-----  
MIIBIjANBgkqhkiG9w0BAQEFA  
jdRAKchIdaNDy5rjiCCAQfskj  
IKnIdcWT3ULMmPORoJbUkv5  
s3mjAUqlnPsy6jaKVHce1B60F  
kyGzv2r4u08v7BALyV8ATbvY  
qgBv1l66A0Bt3/Dlyvg6LhwVT  
AQIDAQAB  
-----END PUBLIC KEY-----
```

Figure 41. Clés de chiffrement

The screenshot shows the VS Code interface with the Explorer sidebar on the left and the Editor on the right. The Explorer sidebar shows a project structure with a 'config' folder expanded, containing a 'jwt' folder which in turn contains 'private.pem' and 'public.pem' files. The Editor shows the contents of the 'lexik\_jwt\_authentication.yaml' file, which is a YAML configuration file for Lexik JWT authentication. A red box highlights the 'lexik\_jwt\_authentication:' section, which includes parameters like 'secret\_key', 'public\_key', 'pass\_phrase', 'token\_ttl', and 'user\_identity\_field'. The file also lists other configuration files like 'api\_platform.yaml', 'cache.yaml', etc.

```
lexik_jwt_authentication:  
  secret_key: '%env(resolve:  
    public_key: '%env(resolve:  
      pass_phrase: %env(JWT_PAS  
      token_ttl: 86400 # 24 ho  
      user_identity_field: email
```

Figure 42. Configuration de Lexik

## Controller « Newsletter »

Un contrôleur est une classe PHP qui gère la logique métier selon le design pattern Modèle Vue Contrôleur. En Symfony®, le contrôleur est le point d'entrée vers le back-end. Son rôle est de traiter les requêtes HTTP entrantes, d'interagir avec le modèle de données (DTO) et de renvoyer une réponse au client (requête sortante).

Dans le cadre de ce projet où une solution telle qu'API Platform a été utilisée, les contrôleurs sont inutiles. Cependant, à des fins pédagogiques, une API « custom » a été réalisée à partir de l'entité « Newsletter » (classe « Mère » qui détermine l'objet en POO). Le code ci-dessous (figure 43) présente le contrôleur dédié à cette API.

Les points de terminaison REST (ends-points) sont générés par l'attribut PHP8 `##[Route()]` pour chaque méthode du CRUD (Get, Post, Delete et Put).

```
public function showNewsletter(Newsletter $newsletter): Response
{
    return $this->json($newsletter);
}

// CRUD: Post - Create a new newsletter
#[Route('/api/custom/public_newsletters_post', name: 'new')]
public function createNewsletter(Request $request, EntityManagerInterface $entityManager)
{
    $newsletter = new Newsletter;
    $data = json_decode($request->getContent(), true);

    // Set properties from the request data
    $newsletter->setId($data['id']);
    $newsletter->setEmail($data['email']);

    // Persist and flush changes to the database
    $entityManager->persist($newsletter);
    $entityManager->flush();

    return $this->json($newsletter);
}

// CRUD: Put - Update a newsletter by replacing its data
#[Route('/api/custom/protected_newsletters_put/{id}', name: 'update')]
public function updateNewsletter(Request $request, EntityManagerInterface $entityManager, Newsletter $newsletter): Response
{
    $data = json_decode($request->getContent(), true);

    // Set properties from the request data
    $newsletter->setEmail($data['email']);

    // Persist and flush changes to the database
    $entityManager->persist($newsletter);
    $entityManager->flush();

    return $this->json($newsletter);
}
```

Figure 43. NewsletterController.php

## Repository « Newsletter »

Dans le contexte de Symfony et de l'ORM Doctrine, un repository est une classe PHP chargée de gérer l'accès aux données. Chaque entité dispose d'un repository.

Dans le cadre de l'API « custom » réalisée pour l'entité Newsletter de ce projet, le repository contient la requête SQL personnalisée (ou DQL pour Doctrine Query Langage) responsable de l'injection des informations dans la base de données.

Dans l'extrait ci-dessous, le service `EntityManager` appelle la connexion à la base de données, puis la requête DQL est écrite et stockée dans la variable `$dql`. La méthode `prepare()` (Prepare Statement) sécurise la requête selon les recommandations de la documentation Symfony® et des bonnes pratiques de sécurité en vigueur. Les paramètres `:id` et `:email` sont ensuite liés à la requête préparée, puis la requête est exécutée `$preparedQuery->execute();`

```
// Fetch information from newsletters using DQL (Doctrine Query Language)
public function fetchInfoFromNewsletterWithDql(): array
{
    $entityManager = $this->getDoctrine()->getManager(); // Create a query and execute it
    $dql = $entityManager->createQuery( // DQL request which return Array
        'SELECT newsletter.id, newsletter.email
         FROM App\Entity\Newsletter AS newsletter'
    );
    $result = $dql->getResult();
    return $result;
}

// Insert information in newsletters DB using DQL
public function insertEmailFromNewsletter(int $id, string $email): void
{
    // Use EntityManager to execute query with parameters
    $entityManager = $this->getDoctrine()->getManager();
    $connection = $entityManager->getConnection();

    $dql = "INSERT INTO newsletter (id, email, createdAt)
           VALUES (:id, :email, NOW())";
    $preparedQuery = $connection->prepare($dql);
    $preparedQuery->bindParam(':id', $id);
    $preparedQuery->bindParam(':email', $email);
    $preparedQuery->execute();
}
```

Figure 44. NewsletterRepository.php

## Journalisation des logs avec Monolog

La journalisation des logs consiste à enregistrer les événements importants survenus dans l'application. Les logs permettent un suivi détaillé des activités, notamment en termes de sécurité ou de débogage.

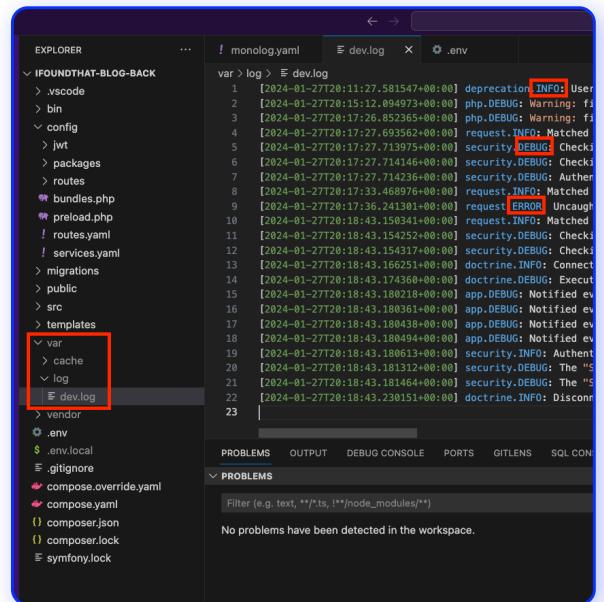
Les messages peuvent provenir de différentes parties de l'application, comme de l'authentification ou des services, etc et traiter d'événements front comme d'événements back.

Dans le cadre de ce projet, j'ai utilisé la bibliothèque Monolog avec Symfony®. Elle offre divers gestionnaires de logs (handlers) qui renseignent différents niveaux de logs : error, info et debug (figure 45).

Monolog se configure selon les différents environnements de développement (développement ou production), depuis le fichier monolog.yaml disponible dans config/packages/monolog.yaml.

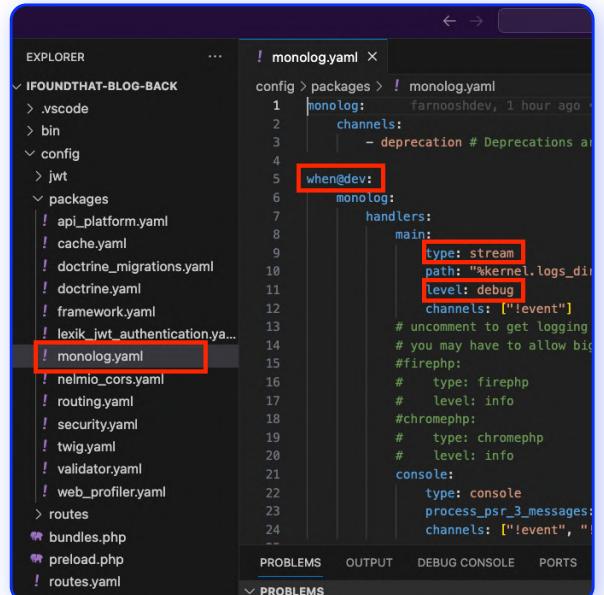
Dans ce projet, j'ai configuré le gestionnaire de logs sur « stream » pour un enregistrement en continu des logs dans un fichier donné (dev.log). Le niveau de log est configuré sur « debug » pour enregister les messages de tous les niveaux de log (figure 46).

Ainsi configuré, Monolog peut monitorer n'importe quelle partie de l'application.



```
var > log > dev.log
1 [2024-01-27T20:11:27.581547+00:00] deprecation: INFO User
2 [2024-01-27T20:15:12.059473+00:00] php.DEBUG: Warning: fi
3 [2024-01-27T20:17:26.852365+00:00] php.DEBUG: Warning: fi
4 [2024-01-27T20:17:27.693562+00:00] request: INFO: Matched
5 [2024-01-27T20:17:27.713975+00:00] security: DEBUG
6 [2024-01-27T20:17:27.714146+00:00] security: DEBUG: Checks
7 [2024-01-27T20:17:27.714236+00:00] security: DEBUG: Author
8 [2024-01-27T20:17:33.468976+00:00] request: INFO: Matched
9 [2024-01-27T20:17:36.241301+00:00] request: ERROR: Uncough
10 [2024-01-27T20:18:43.150341+00:00] request: INFO: Matched
11 [2024-01-27T20:18:43.154252+00:00] security: DEBUG: Checks
12 [2024-01-27T20:18:43.154317+00:00] security: DEBUG: Checki
13 [2024-01-27T20:18:43.166251+00:00] doctrine: INFO: Connect
14 [2024-01-27T20:18:43.174350+00:00] doctrine: DEBUG: Execut
15 [2024-01-27T20:18:43.180218+00:00] app: DEBUG: Notified ev
16 [2024-01-27T20:18:43.180351+00:00] app: DEBUG: Notified ev
17 [2024-01-27T20:18:43.180438+00:00] app: DEBUG: Notified ev
18 [2024-01-27T20:18:43.180494+00:00] app: DEBUG: Notified ev
19 [2024-01-27T20:18:43.180613+00:00] security: INFO: Authent
20 [2024-01-27T20:18:43.181312+00:00] security: DEBUG: The "s
21 [2024-01-27T20:18:43.181464+00:00] security: DEBUG: The "s
22 [2024-01-27T20:18:43.230151+00:00] doctrine: INFO: Discou
```

Figure 45. Enregistrement des logs dans dev.log



```
EXPLORER ... ! monolog.yaml
! IFOUNDTHAT-BLOG-BACK
> .vscode
> bin
> config
> jwt
> packages
! api_platform.yaml
! cache.yaml
! doctrine_migrations.yaml
! doctrine.yaml
! framework.yaml
! lexik_jwt_authentication.ya...
! monolog.yaml
! nelmio_cors.yaml
! routing.yaml
! security.yaml
! twig.yaml
! validator.yaml
! web_profiler.yaml
> routes
! bundles.php
! preload.php
! routes.yaml
PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS SQL CON
! PROBLEMS
Filter (e.g. text, **/*.ts, **/*node_modules/*)
No problems have been detected in the workspace.
```

```
config > packages > ! monolog.yaml
1 monolog: farnooshdev, 1 hour ago
2
3
4
5 when:dev:
6 monolog:
7   handlers:
8     main:
9       type: stream
10      path: "%kernel.logs_dir"
11      level: debug
12      channels: ["!event"]
13      # uncomment to get logging
14      # you may have to allow big
15      #firephp:
16      #  type: firephp
17      #  level: info
18      #chromephp:
19      #  type: chromephp
20      #  level: info
21      console:
22        type: console
23        process_psr_3_messages:
24        channels: ["!event", "
```

Figure 46. Configuration en développement

## Voters

Dans le contexte de Symfony, un Voter est un composant utilisé dans le système de sécurité pour déterminer si un utilisateur a le droit d'effectuer une action spécifique (ex : CRUD).

Lorsqu'une décision d'accès doit être prise, le système invoque le Voter enregistré pour cette décision. Il est responsable de la logique de validation. La méthode `isGranted()` précise le rôle attendu côté serveur pour une action donnée : `case 'ARTICLE_CREATE'`.

Dans IFT-Blog, des Voters ont été implémentés, entre autres, pour contrôler l'accès au CRUD des articles du site (figure 47).

Cette méthode offre une protection supplémentaire contre certaines cyber-attaques.

```
protected function voteOnAttribute(string $attribute, mixed $subject, TokenInterface $token)
{
    $user = $token->getUser();
    if (!$user instanceof UserInterface) {
        return false;
    }

    switch ($attribute) {
        case 'ARTICLE_READ':
            if($this->security->isGranted('ROLE_ADMIN') || $subject->
                isGranted('ROLE_ADMIN')) {
                return true;
            }
            break;
        case 'ARTICLE_CREATE':
            if($this->security->isGranted('ROLE_ADMIN') || $subject->
                isGranted('ROLE_ADMIN')) {
                return true;
            }
            break;
        case 'ARTICLE_EDIT':
            if($this->security->isGranted('ROLE_ADMIN') || $subject->
                isGranted('ROLE_ADMIN')) {
                return true;
            }
            break;
        case 'ARTICLE_DELETE':
            if($this->security->isGranted('ROLE_ADMIN') || $subject->
                isGranted('ROLE_ADMIN')) {
                return true;
            }
            break;
    }
}
```

Figure 47. Voter rattaché à l'entity Article

## Constraints (Assert)

En Symfony, une contrainte est une règle qui est appliquée à une propriété d'une entité (ex : email).

Les contraintes sont utilisées en tant que validateur. Elles vérifient la validité des données au niveau du DTO (Data Transfer Object) côté serveur avant que celles-ci ne soient persistées en base de données.

Ces contraintes sont appliquées à des propriétés d'une entité pour spécifier la règle de validation à respecter. Pour ce faire, l'attribut `##[Assert\...]` PHP 8 est utilisé avant la déclaration de la propriété (figure 48).

À l'instar des Voters, l'utilisation des Contraintes renforce la sécurité de l'application.

```
new Patch(
    security: "is_granted('FORM_EDIT', object)",
),
],
)

private ?int $id = null;

#[ORM\Column(length: 70)]
#[Assert\Length(min: 2, max: 70, minMessage: 'Your name is too short.')]
private ?string $name = null;

#[ORM\Column(length: 70)]
#[Assert\Email(message: 'This email {{ value }} is not a valid email address.')]
private ?string $email = null;

#[ORM\Column(type: Types::TEXT)]
#[Assert\Length(min: 2, max: 3500, minMessage: 'Your message is too short.')]
private ?string $message = null;

public function getId(): ?int
{
    return $this->id;
}

public function setId(int $id): static
{
    $this->id = $id;

    return $this;
}
```

Figure 48. Exemple d'utilisation de l'attribut Assert

# API Platform®

## Article

API Platform® est une solution open source qui propose un ensemble d'outils qui accélèrent le développement d'API REST.

API Platform utilise des attributs PHP 8 pour créer rapidement des points de terminaison REST, c'est-à-dire, des routes (URLs) pour les méthodes de création, d'ajout, de modification ou de suppression de contenu dans une base de données (Post, Get, Put, Delete).

Elle respecte les standards du web tels que LD+JSON pour la sémantique des données (figure 50) et génère automatiquement une documentation interactive (Swagger/OpenAPI) (figure 49).

La gestion des erreurs est simplifiée par un mécanisme de validation sécurisé des requêtes REST.

API Platform s'est imposée pour sa compatibilité avec les frameworks utilisés dans le projet (API Platform est basé sur Symfony®) et pour la rapidité de création des end points REST. La structure proposée par API Platform ® a permis d'accélérer le développement grâce cette documentation. Un atout de poids dans un planning fortement contraint !

L'implémentation dans le code se réalise côté back-end, directement dans l'entité Symfony® (« Article » par exemple). Un attribut PHP® 8 **##[ApiResource]** est placé au-dessus de l'attribut de classe qui doit être piloté par API Platform®. Cet attribut est paramétrable pour chaque requête et des routes personnalisées peuvent être créées selon les besoins spécifiques du projet (figure 51).

The screenshot shows the API Platform documentation for the Article endpoint. It includes a list of HTTP methods and their corresponding URLs:

- GET /api/articles: Retrieves the collection of Article resources.
- POST /api/articles: Creates a Article resource.
- GET /api/articles/{id}: Retrieves a Article resource.
- PUT /api/articles/{id}: Replaces the Article resource.
- DELETE /api/articles/{id}: Removes the Article resource.
- PATCH /api/articles/{id}: Updates the Article resource.
- GET /api/public\_articles: Retrieves the collection of Article resources.
- GET /api/public\_articles/{id}: Retrieves a Article resource.

Below the Article section, there is a link to the Categorie endpoint: GET /api/categories.

Figure 49. Documentation des end points REST

The screenshot shows the API Platform documentation for the Login Check endpoint. It includes a POST method for creating a user token:

- POST /api/login: Creates a user token.

Details for the POST method:

- Creates a user token.
- Parameters: No parameters.
- Request body (required): The login data. Example Value: Schema: { "email": "string", "password": "string" }
- Responses:
  - Code 200: User token created. Media type: application/json. Content Accept header: application/json. Example Value: Schema: { "token": "string" }

Figure 50. Structure et type des données

```
Entity > Article.php > PHP Intelephense > Article
#[ApiResource()
operations: [
    new GetCollection(),
    new Post(),
    new Get(),
    new Put(),
    new Delete(),
    new Patch(),
    new Get()
],
```

The code snippet shows the implementation of the Article entity with various operations:

- new GetCollection(): You, 3 weeks ago · [api]-update-article-entity-w · normalizationContext: ['groups' => ['read:Article:collection']]
- new Post(): security: "is\_granted('ROLE\_ADMIN')", # Post method is only available to ROLE\_ADMIN users.
- new Get(): normalizationContext: ['groups' => ['read:Article:item']]
- new Put(): security: "is\_granted('ARTICLE\_EDIT', object)", # Voter's specific context
- new Delete(): security: "is\_granted('ARTICLE\_DELETE', object)"
- new Patch(): security: "is\_granted('ARTICLE\_EDIT', object)"
- new Get(): normalizationContext: ['groups' => ['read:Article:item:public']], # name:'public\_article', uriTemplate:'public\_articles/{id}'
- new GetCollection(): normalizationContext: ['groups' => ['read:Article:collection:public']] name:'public\_articles', uriTemplate:'public\_articles'

Figure 51. Implémentation d'API Platform®

## Accessibilité

## Balises ARIA

Pour rendre un site web accessible à un large éventail de personnes, y compris celles porteuses de handicap, l'utilisation des attributs HTML ARIA (Accessible Rich Internet Applications) constituent une solution efficiente. C'est dans cet objectif que cet outil a été implémenté dans le projet IFT-Blog.

Les attributs ARIA les plus utilisés sont localisés autour des champs de formulaires ou des points d'interaction (ex : boutons) (figures 52 et 53).

Il existe plusieurs attributs ARIA qui peuvent être utilisés selon les cas de figure. Les plus utilisés sont : **aria-label**, **aria-describedby** ou encore **aria-haspopup**.

Cependant, il est à noter que l'utilisation des balises sémantiques HTML seules peuvent déjà permettre aux technologies d'assistance de se repérer dans le code et fournir une information pertinente à l'utilisateur porteur de handicap. Les attributs ARIA permettent une description plus précise des interactions.

## Conception graphique

La conception graphique joue également un rôle important dans l'accessibilité. C'est pourquoi le design de cette application priviliege les contrastes forts et des éléments d'interface larges (figure 54), tant en version desktop qu'en version mobile.

La police de caractères et les couleurs participent à la visibilité et à l'utilisabilité du site.

La maquette est présentée en détail en annexe n°8.

```
<div class="form-group">
<input
  type="password"
  formControlName="password"
  class="form-control"
  [class.is-invalid]="form.get('password')?.invalid && form.get('password').dirty"
  placeholder="Password"
  required
  pattern="^a-zA-Z0-9éàç;,!?:!ÉÈÀÇÈÌÒÙÛéèàçéìòùû\s]+$"
  aria-describedby="login-password-description"
/>
<!-- Using ARIA description for help assistance technology -->
<div id="login-password-description" aria-hidden="true">
  Please, enter your password into the text field selected.
</div>
<div class="invalid-feedback" *ngIf="form.get('password')?.invalid">
  Password required!
</div>
</div>
<div class="alert alert-danger" *ngIf="errorMessage">
  {{ errorMessage }}
</div>
<button type="submit" class="btn btn-danger deep-orange darken-8">
  Sign In
</button>
<button routerLink="/register" class="btn btn-link deep-orange darken-8">
  Register
</button>
```

Figure 52. Liaison de l'id avec aria-describedBy

```
v class="container_wrapper">
div class="label_newsletter-subtitle">Newsletter</div>
div class="label_newsletter-title">Stay alert!</div>
form [formGroup]="newslettersForm" (ngSubmit)="onSubmitNewsletter"
<div class="input_email-field">
  <input
    type="email"
    id="email"
    formControlName="email"
    class="form-control"
    required
    placeholder="Your email"
    aria-describedby="newsletter-form-email-description"
  />
  <!-- Using ARIA description for help assistance technology -->
  <div id="newsletter-form-email-description" aria-hidden="true">
    Please, enter your email into the text field selected.
  </div>
</div>
<div class="section center">
  <button
    type="submit"      You, 27 seconds ago * Uncommitted changes
    class="btn-large black white-text center"
    aria-label="submit newsletter form"
  >
    Submit
  </button>
</div>
```

Figure 53. Précision de l'action avec aria-label

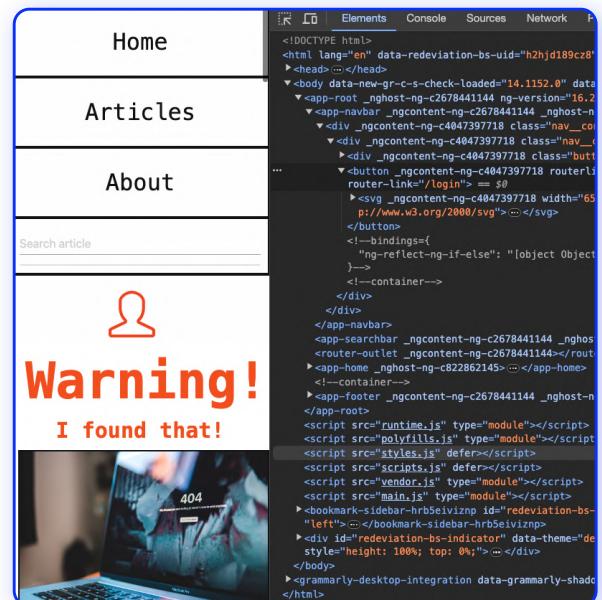


Figure 54. Conception des éléments d'UI

## Tests unitaires

### Tests unitaires

Dans le cadre de ce projet, les tests unitaires ont essentiellement été réalisés sur les éléments du back-end.

L'objectif était de tester individuellement les méthodes nécessaires au fonctionnement de l'application, telles que les méthodes du CRUD « Article » ou la méthode Post pour l'inscription à la newsletter par exemple.

Parfois, l'erreur est provoquée volontairement dans le but de vérifier tous les cas de figure (succès ou échec), selon les données prévues en entrée et en sortie de test.

Le protocole des différents tests effectués est détaillé en annexe n°12.

### Tests fonctionnels

Contrairement aux tests unitaires qui se concentrent sur la vérification d'une seule unité de code, les tests fonctionnels cherchent à vérifier la bonne marche d'une fonctionnalité entière.

Dans le cadre du projet IFT-Blog, l'objectif était de tester les fonctions les plus essentielles pour l'utilisateur final comme la visualisation des articles.

Ces tests étaient basés sur des cas d'utilisation présentés dans le jeu d'essais (voir VIII p.56 « Présentation du jeu d'essais »).

Le framework de test PHPUnit® a été utilisé pour réaliser ces différents tests.

```
public function getEntity(): Newsletter
{
    // Create a new instance of the Newsletter entity and set a pre-defined email
    return (new Newsletter())
        ->setEmail('aaa@aaa.aaa');
}

// Test for invalid email validation in the Newsletter entity
public function testInvalidEmail(): void
{
    // Bootstrap the Symfony kernel to set up the testing environment
    self::bootKernel();
    // Get the container from the kernel for accessing services
    $container = static::getContainer();
    // Create a Newsletter entity with a valid email
    $newsletter = $this->getEntity();
    // Set an invalid email for testing purposes
    $newsletter->setEmail('aaa');
    // Use the validator service to validate the Newsletter entity
    $errors = $container->get('validator')->validate($newsletter);
    // Assert that there is exactly one validation error
    $this->assertCount(1, $errors);
}
```

Figure 55. Test où l'erreur est provoquée

```
public function testGetCollection(): void
{
    // Create 100 articles using our factory
    ArticleFactory::createMany(10);

    // The client implements Symfony HttpClient's 'HttpClientInterface'
    $response = static::createClient()->request('GET', '/api/articles');

    // Asserts that the response is successful
    $this->assertResponseIsSuccessful();

    // Asserts that the returned content type is JSON-LD (the default)
    $this->assertResponseHeaderSame('content-type', 'application/ld+json');

    // Asserts that the returned JSON is a superset of this one
    $this->assertJsonContains([
        '@context' => '/api-contexts/Article',
        '@id' => '/api/articles',
        '@type' => "hydra:Collection",
        "hydra:totalItems" => 10,
        'hydra:member' => []
    ]);

    // Because test fixtures are automatically loaded between each test
    $this->assertCount(10, $response->toArray()['hydra:member']);
}
```

Figure 56. Vérification du CRUD Article

## Performances

### Accessibilité

L'accessibilité a été testée avec les outils Wave® et Google Lighthouse®. Cependant, des résultats contradictoires apparaissent. Une vérification poussée de l'accessibilité du site serait à envisager.

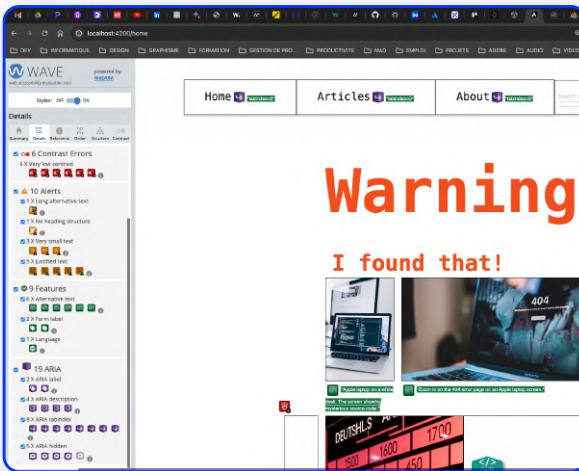


Figure 57. Résultats d'accessibilité avec Wave®

### Performances

Les performances du site ont été testées avec Google Lighthouse® pour un score de 80%. Des améliorations seront nécessaires pour viser une éco-conception efficiente.

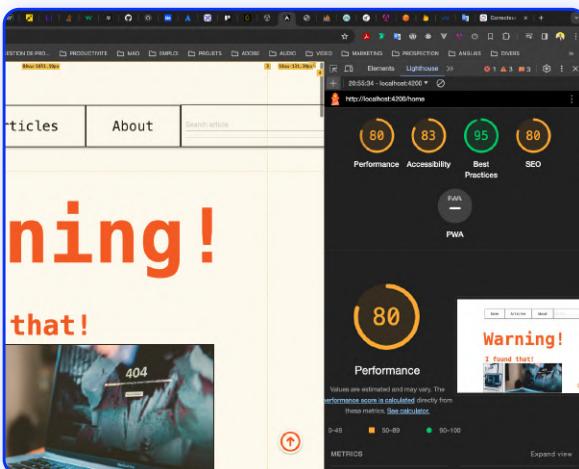


Figure 58. Résultats avec Google Lighthouse®

## Sécurité

### Test de pénétration

Burp Suite® permet d'identifier les vulnérabilités de sécurité dans les applications web par un système de monitorage du trafic.

Dans le cadre de ce projet, l'application a été testée en utilisant un serveur local. En conséquence, la fiabilité des résultats serait à vérifier lorsque l'application sera en ligne. Une étude poussée du trafic (requêtes) devrait être envisagée pour déterminer avec exactitude les probables failles de sécurité.

A screenshot of the Burp Suite traffic monitor. The left pane shows a list of captured items added to the site map, including various URLs and their corresponding status codes and MIME types. The right pane contains sections for 'Task configuration', 'Task progress', and 'Task log', which detail the process of capturing traffic and analyzing responses.

Figure 59. Monitorage du trafic local

A screenshot of the Burp Suite response editor. It shows a detailed view of a captured request and its corresponding response. The response tab displays the raw HTTP response, which includes headers, status code (HTTP/2 200), and the response body. The response body contains HTML code, including a script tag that triggers a modal dialog with the text 'I found that!'.

Figure 60. Vérification des failles de sécurité

# Déploiement

## Préparation

La préparation du déploiement de l'application s'est déroulée en plusieurs étapes : d'abord en réalisant un diagramme de déploiement puis en configurant les différents frameworks

pour la production. La configuration du certificat SSL auprès de l'hébergeur (OVH) a également été réalisée durant cette phase préparatoire.

## Diagramme de déploiement

Ce diagramme UML représente la disposition physique des composants matériels nécessaires au fonctionnement de l'application. Il permet de visualiser la manière dont les éléments sont

connectés et déployés. La visualisation des composants et de leurs interactions respectives avec le matériel, permet de déployer et maintenir l'application de manière optimale.

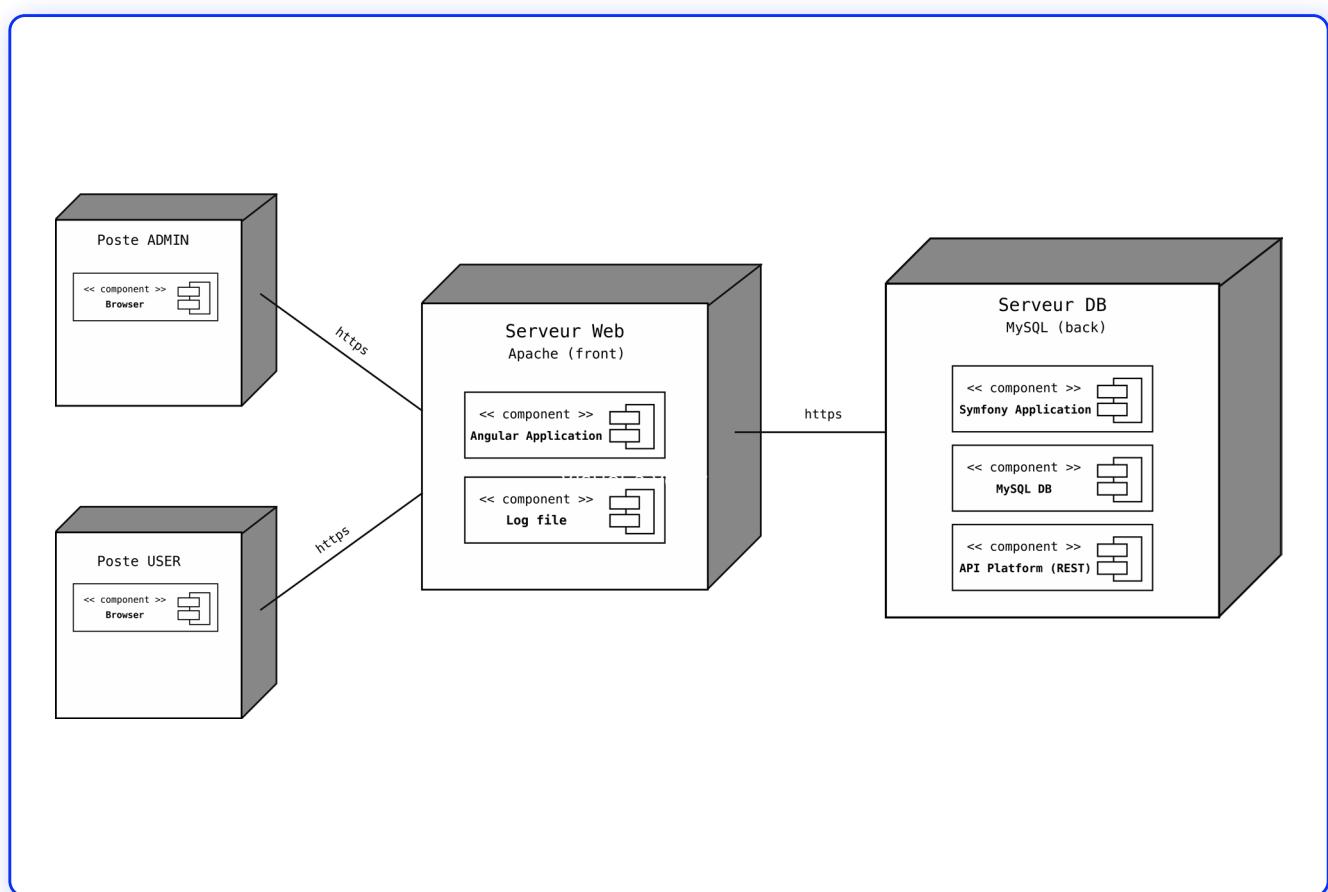


Figure 61. Diagramme UML de déploiement

## Observation

Le temps aura manqué pour procéder au déploiement de l'application durant la période de stage en entreprise. Ainsi, seules les étapes préliminaires qui ont effectivement été réalisées sont présentées dans ce chapitre.

## Installation du certificat SSL/TLS OVH

L'installation du certificat SSL (Secure Sockets Layer) OVH a permis d'obtenir le protocole HTTPS (Hyper Text Transfer Protocol Secure).

### I. Attribution du certificat SSL :

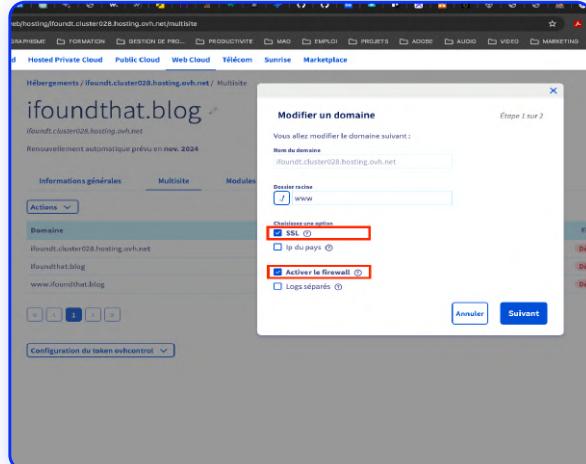


Figure 62. Attribution du certificat (entrée multisite)

### 2. Génération du certificat SSL :

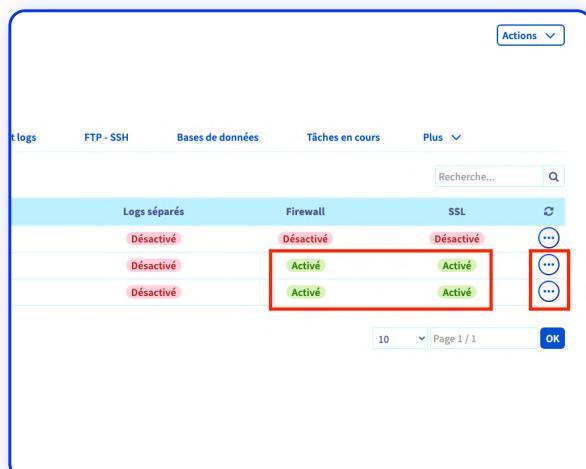


Figure 63. Génération du SSL et activation du Firewall

### 3. Génération du certificat SSL :

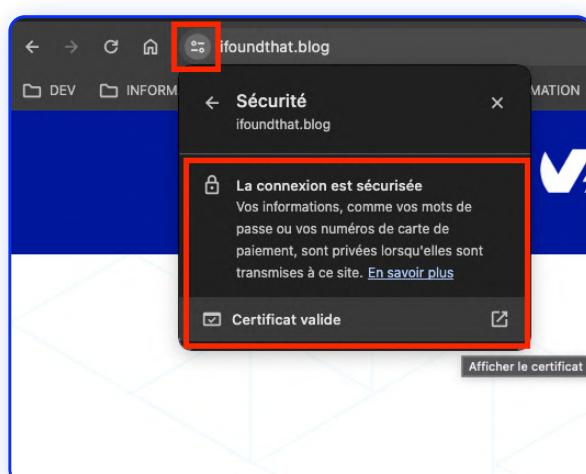


Figure 64. Vérification de la prise en compte du SSL

## Configuration des frameworks

La configuration du certification SSL varie selon les framework. Pour des raisons de sécurité, les chemins ont été masqués.

### I. Modification du build dans Angular® :

```

15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
      "prefix": "app",
      "schematics": {
        "build": 4
      },
      "builder": "@angular-devkit/build-angular:browser",
      "options": {
        "outputPath": "dist/ifoundthat-blog-frontend",
        "index": "src/index.html",
        "main": "src/main.ts",
        "polyfills": [
          "zone.js"
        ],
        "tsConfig": "tsconfig.app.json",
        "inlineStyleLanguage": "scss",
        "assets": [
          "src/favicon.ico",
          "src/assets"
        ],
        "styles": [
          "node_modules/materialize-css/dist/css/materialize.min.css",
          "src/styles.scss"
        ],
        "scripts": [
          "node_modules/materialize-css/dist/js/materialize.min.js"
        ]
      },
      "configurations": {
        "production": {
          "ssl": [
            {
              "key": "[PATH]",
              "cert": "[PATH]",
              "ca": "[PATH]"
            }
          ],
          "budgets": [
            {
              "type": "initial"
            }
          ]
        }
      }
    }
  }
}

```

Figure 65. Modification du fichier angular.json

### 2. Ajout des informations SSL dans Symfony® :

```

> .vscode
> bin
> config
> migrations
> public
> src
> templates
> var
> vendor
> .env
$ .env.local
$ .gitignore
# composer.override.yaml
# composer.yaml
() composer.json
() composer.lock
# symfony.lock

23 # Format described at https://www.doctrine-project.org
24 # IMPORTANT! You MUST configure your server version
25 #
26 # DATABASE_URL="sqlite:///{kernel.project_dir}/var
27 # DATABASE_URL="mysql://app!:ChangeMe@127.0.0.1:3306"
28 # DATABASE_URL="mysql://app!:ChangeMe@127.0.0.1:3306"
29 # DATABASE_URL="postgresql://app!:ChangeMe@127.0.0.1:5432"
30 ##< doctrine/doctrine-bundle ###
31 #
32 ##> nelmio/cors-bundle #####
33 CORS_ALLOW_ORIGIN='https://localhost:127.0.0.1:4200'
34 ##< nelmio/cors-bundle #####
35 #
36 ##> lexik/jwt-authentication-bundle #####
37 JWT_SECRET_KEY='kernel.project_dir/config/jwt/pri
38 JWT_PUBLIC_KEY='kernel.project_dir/config/jwt/pub
39 JWT_PASSPHRASE='819f9c8394e1e4c5d12a66c7ceec3d5191
40 ##< doctrine/doctrine-bundle #####
41 #
42 # SSL configuration
43 APP_ENV=prod
44 APP_DEBUG=0
45 APP_KEY=base64:[KEY]
46 #
47 # SSL certificate
48 SSL_KEY=[PATH]
49 SSL_CERT=[PATH]
50 SSL_CA=[PATH]

PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS
PROBLEMS TERMINAL
Filter (e.g. text, */*, ts, tsx)
mathieu@MacBook-Pro-de-Mathieu:~/ifoundthat-blog$ 

```

Figure 66. Modification des variables d'environnement

### 3. Modification d'API Platform® :

```

> .vscode
> bin
> config
> jwt
> packages
> api_platform.yaml
! cache.yaml
! doctrine_migrations.yaml
! doctrine.yaml
! framework.yaml
! lexik_jwt_authentication.yaml
! nelmio_cors.yaml
! routing.yaml
! security.yaml
! twig.yaml
! validator.yaml
! web_profiler.yaml
> routes
# bundles.php
# preload.php
! routes.yaml
! services.yaml
> migrations
> public
> src
> templates
> var
> vendor
> .env

6   json: { application/json }
docs_formats:
  json_id: [application/ld+json]
  json_compatible_json: [application/vnd.openapi+json]
  html: [text/html]
defaults:
  stateless: true
  cache_headers:
    vary: ['Content-Type', 'Authorization']
  extra_properties:
    standard_put: true
    rfc_7207_compliant_errors: true
  event_listeners_backward_compatibility_layer:
    keep_legacy_inflector: false
  security:
    api_key:
      enabled: true
      type: http
      name: Authorization
      header: Bearer
    http_client:
      default_options:
        method: https
        verify: false
        verify_peer: false
        trusted_certificates:
          - [PATH]

```

Figure 67. Modification du fichier api\_platform.yaml

# VIII. PRÉSENTATION DU JEU D'ESSAIS

## V.III Définition

Le jeu d'essais (ou « Jeu de données de test ») de ce projet fait référence à un ensemble spécifique de données, préparées en amont du développement, destinées à tester l'application selon divers cas de tests et ainsi vérifier que l'application se comporte comme prévu dans les différents scénarios (voir V.2 p.16 “Diagrammes UML”).

Les cas de test présentés ici sont issus à la fois des scénarios nominaux et alternatifs, ainsi que des user stories, notamment la partie rédigée en langage Gherkin (Given, When, Then).

Ils mesurent de nombreux aspects du code source. En outre, il est à noter que certaines étapes sont volontairement destinées à induire un résultat d'erreur.

Bien que les jeux d'essais en général puissent être utilisés avec des tests d'intégration ou encore des tests fonctionnels, dans le cadre de ce projet spécifique, le présent jeu d'essais est effectivement cantonné aux tests unitaires.

La figure ci-dessous montre un extrait du jeu d'essai utilisé pour ce projet. Le jeu d'essai complet est disponible en annexe n°12.

Aa Case Type	≡ Description	≡ Test Step	≡ Expected Result	>Status	≡ Comment
<u>Affichage des articles</u>	Vérifier l'affichage d'une liste d'article. Données en entrée : 'GET', '/api/articles'	Un utilisateur non connecté accède à la home page.	Une liste des articles de la base de données est visible. Données en sortie : confirmation de la méthode assertResponsesSuccessful() de la classe ApiTestCase (KernelTestCase)	Pass	CRUD article
<u>Enregistrement email</u>	Vérifier l'enregistrement d'une adresse email valide depuis le formulaire newsletter. Données en entrée : 'aaa@aaa.aaa'	Un utilisateur non connecté accède à la home page. Il saisit son adresse email dans le formulaire d'inscription à la newsletter.	L'email provenant du formulaire newsletter est enregistrée. Données en sortie : confirmation de la méthode assertResponsesSuccessful() de la classe ApiTestCase (KernelTestCase)	Pass	Persistance email dans table newsletter
<u>Enregistrement email</u>	Vérifier l'enregistrement d'une adresse email invalide depuis le formulaire newsletter. Données en entrée : 'aaa'	Un utilisateur non connecté accède à la home page. Il saisit son adresse email dans le formulaire d'inscription à la newsletter.	L'email provenant du formulaire newsletter n'est pas enregistrée. Données en sortie : erreur de la méthode assertCount(0, \$errors); de la classe ApiTestCase (KernelTestCase)	Pass	Erreur provoquée

Figure 68. Extrait du jeu d'essais

# IX. VEILLE SÉCURITAIRE

## IX. Description d'une veille sécuritaire : accès aux données via Doctrine

Cette veille s'est effectuée dans le cadre de l'activité type n°1 du titre Concepteur Développeur d'Applications : "Développer des composants d'accès aux données", comme stipulé dans le référentiel de compétences (RC) page 7.

### Description du besoin d'information

La partie back-end du projet a été réalisée sous Symfony®. Ce framework php utilise "Doctrine" comme composant d'accès aux données. Doctrine est une bibliothèque ORM (Object Relational Mapping) pour Symfony®. Elle permet aux développeurs de manipuler les données d'une base de données sous forme d'objet php. Cependant, comme toute bibliothèque, Doctrine peut comprendre des failles de sécurité qu'il convient de connaître pour anticiper de potentielles attaques. Partant de ce constat, il m'a fallu d'abord identifier ces menaces potentielles puis comprendre ces failles avant de trouver une solution pour y remédier.

### Méthodologie de la veille

Pour effectuer cette veille sécuritaire sur la bibliothèque Doctrine, j'ai suivi la méthodologie suivante :

I. Consultation des sources officielles telles que le site officiel de Doctrine (rubrique ORM), ainsi que leur blog pour m'informer des mises à jour et des annonces importantes.

2. Consultation de la documentation officielle et du changelogs (journal des modifications) de Doctrine pour connaître les modifications récentes apportées à l'ORM. Cela m'a également permis de découvrir que Doctrine est souvent vulnérable aux problèmes de sécurité de type "Mass Assignment" (affectations de masse). Doctrine n'est pas vulnérable à ce problème au départ. Cependant, les entités (classe php spécifique à Doctrine qui représente une fonction métier de l'application) peuvent comprendre des vulnérabilités à l'affectation de masse dès lors que des méthodes telles que updateFromArray() ou updateFromJson() sont incorporées au programme.

3. Consultation du dépôt GitHub® officiel de Doctrine, notamment les sections Security Advisories et pour vérifier la présence de correctifs de sécurité récemment publiés.

4. Consultation des alertes CVE (Common Vulnerabilities and Exposures) relatives à Doctrine. Cette partie m'a permis de découvrir deux failles respectivement référencées CVE-2015-5723 et CVE-2021-43822.

5. Consultation du détail des failles CVE sur les sites du NVD (National Vulnerability Database)

et du CWE (Common Weakness Enumeration). Pour chaque faille identifiée sur le site du CVE, un renvoi est fait sur le site du NVD qui détaille la vulnérabilité et propose une solution technique pour la combler. Un complément d'information est également délivré sur le site du CWE par un lien présent sur le site du NVD (ex : CWE-89).

6. Échange avec un développeur back-end Symfony®/Doctrine confirmé Farnoosh Ashrafi-Ghalati pour confronter nos veilles respectives.

## Critères de sélection

Le principal critère de sélection des informations recueillies pendant cette veille sécuritaire a été la fiabilité. En effet, la provenance des informations par le biais des sources officielles de Doctrine, dépôt GitHub® et Blog notamment, garantissent leur fiabilité et leur exactitude.

En outre, des sources éprouvées telles que le CVE et la NVD sont également un gage de sérieux des données récoltées.

## Mots-clés utilisés

Pour mener à bien cette veille sécuritaire, j'ai été amené à réaliser plusieurs recherches distinctes. J'ai utilisé l'anglais pour pouvoir disposer des sources les plus officielles et à jour possibles.

D'abord une première recherche généraliste sur Google® avec les mots-clés suivants : "Doctrine", "ORM", "Security", "Watch".

Puis une recherche plus spécifique avec les mots-clés suivants : "Doctrine ORM", "Security breach", "CVE", "NVD".

## Solutions trouvées

Même si les sources officielles n'évoquaient pas de menaces récentes en date de la recherche (26/12/23) et dans la version de Doctrine employée sur le projet (version 2.17.1), les alertes CVE (CVE-2021-43822 et CVE-2015-5723), dont le détail était visible sur le site du NVD, m'ont permis d'identifier un problème récurrent d'injection SQL via Doctrine. La solution proposée sur le site du NVD consistait en la montée de version de Doctrine.

De plus, la documentation officielle de Doctrine proposait d'utiliser une « liste blanche » pour éviter les attaques de type affection de masse (Mass Assignment, référencée CWE-915). Cette approche autorise explicitement les actions, routes ou accès spécifiques définis et rejette le reste.

## Site(s) consulté(s)

Les recherches réalisées sur Google® concernant cette veille sur la sécurité des composants d'accès aux données avec Doctrine m'ont conduit dans un premier temps à des sites de références tels que la documentation officielle et le site web de l'ORM. Puis, dans un second temps, j'ai ciblé plus avant ma recherche en consultant les sites se rattachant à Doctrine, notamment via GitHub®. Enfin, j'ai consulté les sites du CVE, du NVD et de l'OWASP pour comprendre la nature de ces failles d'une part et vérifier les correctifs proposés. La liste des sites consultés est disponible en annexe n°14.

## Implémentation

Dans un premier, j'ai entrepris d'incorporer une liste blanche. J'ai choisi d'appliquer celle-ci sur le champ de recherche du site. La barre de recherche étant gérée par Angular® en front, j'ai utilisé les fonctionnalités de filtrage d'API Platform® puis en ai déterminé le contenu avec une classe Symfony® spécifique nommée « SearchFilter ».

## I. Création de la classe SearchFilter selon les recommandations de la doc d'API Platform®.

```

namespace App\Filter;

use ApiPlatform\Doctrine\Orm\Filter\AbstractFilter;
use ApiPlatform\Doctrine\Orm\Util\QueryNameGeneratorInterface;
use ApiPlatform\Metadata\Operation;
use Doctrine\ORM\QueryBuilder;
use Symfony\Component\ PropertyInfo\Type;

final class SearchFilter extends AbstractFilter // Custom search filter for API Platform
{
    protected function filterProperty(string $property, $value, QueryBuilder $queryBuilder, Operation $operation, array $context = []): void
    {
        // Condition: check if the property is ready for filter
        if (
            !$this->isPropertyEnabled($property, $resourceClass) ||
            !$this->isPropertyMapped($property, $resourceClass)
        ) {
            return;
        }

        $parameterName = $queryNameGenerator->generateParameterName($property); // Generate
        $queryBuilder // Add a WHERE clause using REGEX for search (nota: REGEXP is a specific
        ->andWhere(sprintf('REGEXP(%s, :%s) = 1', $property, $parameterName)) // sprintf
        ->setParameter($parameterName, $value); // Bind value with variable $parameterName
    }
}

```

BLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS SQL CONSOLE V TERMINAL

Figure 69. SearchFilter.php

## 2. Déclaration du filtre dans api\_platform.yaml.

L'entité concernée ici est « Article ».

```

status_code: 200
cache_headers:
    vary: ['Content-Type', 'Authorization', 'Origin']
extra_properties:
    standard_put: true
    rfc_7207_compliant_errors: true
event_listeners_backward_compatibility_layer: false
keep_legacy_inflector: false

# SearchFilter
filter:
    metadata:
        'App\Entity\Article': # The entity to secure
            filters:
                - 'search_filter' # Filter alias

```

BLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS SQL CONSOLE

Figure 70. Déclaration du filtre

## 3. Application de l'attribut **#[ApiFilter]** sur les champs « content », « title » et « chapo ».

```

private ?string $title = null;
#[ORM\Column(type: Types::TEXT)]
#[Groups ({
    'read:Article:item',
    'read:Article:collection',
    'read:Article:item:public',
    'read:Article:collection:public'
})]
private ?string $chapo = null;
#[ApiFilter(SearchFilter::class)]
#[ORM\Column(type: Types::TEXT)]
#[Groups ({
    'read:Article:item',
    'read:Article:collection',
    'read:Article:item:public',
    'read:Article:collection:public'
})]
private ?string $content = null;

```

Figure 71. Application de l'attribut « ApiFilter »

Dans un second temps, j'ai procédé à la montée de version de Doctrine.

## I. Vérification de la dernière version stable sur le site officiel de Doctrine (doctrine-project.org).



Figure 72. Vérification de la version de Doctrine

## 2. Contrôle de la version actuelle dans VSCode.

```

{
    "require": {
        "php": ">=8.1",
        "ext-ctype": "*",
        "ext-iconv": "*",
        "api-platform/core": "^3.2" (v3.2.7),
        "doctrine/doctrine-bundle": "^2.11" (2.11.1),
        "doctrine/doctrine-migrations-bundle": "^3.3" (3.3.0),
        "doctrine/orm": "2.17" (2.17.1),
        "nelmio/cors-bundle": "^2.4" (2.4.0),
        "phpdocumentor/reflection-docblock": "^5.3" (5.3.0),
        "phpstan/phpdoc-parser": "^1.24" (1.24.4),
        "symfony/asset": "6.3.*" (v6.3.8),
        "symfony/console": "6.3.*" (v6.3.8),

```

Figure 73. Vérification dans composer.json

## 3. Après un commit Git, dans le Terminal, mise à jour de Doctrine à partir de la commande :

`composer require doctrine/orm:2.17.2`

```

mathieu@MacBook-Pro-de-Mathieu:~/foundthat-blog-back % composer require doctrine/orm:2.17.2
./composer.json has been updated
Running composer update doctrine/orm
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 0 removals
- Upgrading doctrine/orm (2.17.1 => 2.17.2)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 0 installs, 1 update, 0 removals
- Upgrading doctrine/orm (2.17.1 => 2.17.2): Extracting archive
Generating autoload files
65 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
Run composer recipes at any time to see the status of your Symfony recipes.
Executing script cache:clear [OK]
Executing script assets:install public [OK]
No security vulnerability advisories found.

```

Figure 74. Mise à jour effective de Doctrine

## 4. Vérification dans composer.json, puis test de l'application.

```

{
    "license": "proprietary",
    "minimum-stability": "stable",
    "prefer-stable": true,
    "require": [
        "php": ">=8.1",
        "ext-ctype": "*",
        "ext-iconv": "*",
        "api-platform/core": "3.2" (v3.2.7),
        "doctrine/doctrine-bundle": "^2.11" (2.11.1),
        "doctrine/doctrine-migrations-bundle": "3.3" (3.3.0),
        "doctrine/orm": "2.17.2" (2.17.2),
        "nelmio/cors-bundle": "2.4" (2.4.0),
        "phpdocumentor/reflection-docblock": "5.3" (5.3.0),

```

Figure 75. Doctrine 2.17.2 est installé

# X. SITUATION DE TRAVAIL

## X. Situation de travail : sécurisation d'une application Angular®

Pour être en totale conformité avec le cahier des charges MVP du projet, notamment au regard de la sécurité (voir III.2.2 p.8 "Contraintes techniques"), j'ai été amené à effectuer une recherche dédiée à la sécurisation des applications réalisées sous Angular®.

### Description du besoin d'information

L'application étant de type blog, cette recherche était motivée par le fait de minimiser au maximum les failles de sécurité les plus connues du framework Angular® d'une part, tout en prenant en compte les spécificités de l'application d'autre part (formulaire de contact, appel à une API notamment).

### Méthodologie de la recherche

Pour réaliser cette recherche, j'ai employé une méthodologie en sept points. Celle-ci s'est organisée comme suit :

1. Reformulation de la problématique : il m'est en effet apparu comme essentiel de clarifier le problème pour organiser la recherche de façon suffisamment large.

2. Choix de mots-clés pertinents : dans le but d'obtenir des résultats les plus précis possibles, j'ai sélectionné et listé les mots-clés adéquats.

3. Formulation d'un prompt et affinage avec une question centrale : toujours pour guider la recherche, j'ai rédigé tout d'abord un prompt

complet puis j'ai affiné ce prompt par une unique question claire et spécifique afin de guider la recherche au maximum.

4. Utilisation de l'IA (ChatGPT et Bard) : sur la base du prompt réalisé précédemment, j'ai utilisé l'IA pour me fournir une première orientation sur le sujet.

5. Vérification des informations : après avoir recoupé puis additionné les informations des IA, j'ai entrepris une recherche plus traditionnelle sur la base d'une unique question centrale (étape 3). Cette recherche m'a permis de vérifier les informations recueillies précédemment en consultant des sources officielles en ligne afin de valider les propos des IA tout en approfondissant mes connaissances sur ces points précis. J'ai également entrepris à cette étape d'élargir le champ de cette recherche par l'intermédiaire du site de l'ANSSI.

6. Revue de code : puis, j'ai réalisé un examen du code source d'un site réalisé avec Angular® pour vérifier *in situ* les meilleures pratiques et solutions à mettre en œuvre.

7. Implémentation des solutions : enfin, après confirmation auprès de développeurs séniors,

j'ai procédé à la mise en œuvre des solutions identifiées dans le projet.

En combinant des méthodes de recherche en ligne avec l'utilisation d'outils comme l'IA, la validation des informations et l'examen pratique du code, j'ai pu obtenir des informations variées et fiables afin de prendre une décision éclairée avant toute implémentation.

### Reformulation de la problématique

Suivant cette méthodologie, j'ai tout d'abord entrepris de reformuler l'objet de ma recherche (« *La sécurité web avec le framework Angular®* »), en problématique. J'ai donc abouti au paragraphe suivant :

« *Angular® ne sécurise pas complètement une application. C'est pourquoi il convient d'identifier les failles de sécurité web les plus connues du framework tout en prenant en compte les spécificités de mon application (formulaire, commentaires, profil utilisateur, barre de recherche) qui est de type "blog". Pour cela, j'ai besoin de connaître les bonnes pratiques actuelles de développement relatives à ce framework en matière de sécurité, mais également de connaître plus largement les failles de sécurité provenant de l'écosystème d'Angular®* ».

### Mots-clés, prompt et question centrale

La problématique étant désormais élargie, j'ai pu isoler les mots-clés les plus pertinents pour ma recherche, à savoir : « *Angular®* », « *Failles de sécurité* », « *Sécurité web* », « *Blog* », « *Bonnes pratiques* », « *Ecosystème Angular®* », « *Actuelles* » (2024).

J'ai alors pu incorporer ces différents mots-clés dans le prompt ci-dessous. Ce prompt était destiné aux IA comme précisé à l'étape 4 de la méthodologie suivie :

« *Je souhaite sécuriser au maximum une Web application réalisée avec le framework Angular® en version 16. J'ai donc besoin de connaître d'une part les principales failles de sécurité de l'écosystème Angular® et d'autre part, les bonnes pratiques actuelles de développement dans ce cadre précis. Je précise que l'application à sécuriser utilise les fonctionnalités suivantes : des formulaires (contact, newsletter et inscription de l'utilisateur), la possibilité de laisser un commentaire et d'utiliser une barre de recherche. L'application utilise en outre la solution API Platform® pour réaliser ses requêtes à la base de données. La base de données est en MySQL® 5. Pourrais-tu, en tant qu'expert en sécurité web et Angular®, m'indiquer, sous forme de liste à points, les étapes à suivre pour concrètement sécuriser une application de type blog* ? ».

Les réponses complètes des IA consultées sont disponibles en annexe n°13.

J'ai ensuite entrepris de synthétiser les notions évoquées dans le prompt en une unique question centrale :

« *Quelles sont les meilleures pratiques de sécurité en 2024 pour prévenir les vulnérabilités et les attaques sur une web application Angular® ?* ».

Puis, j'ai traduit cette question centrale en anglais afin d'organiser la recherche sur des sites anglophones, réputés plus à jour que les sites francophones. Voici le texte traduit destiné à une recherche traditionnelle sur Google® :

« What are the best security practices in 2024 to prevent vulnerabilities and attacks on an Angular® web application? ».

Ces différents éléments textuels m'ont permis d'une part, d'obtenir un premier champ de recherche et de compréhension grâce aux réponses des IA et d'autre part d'ajouter leurs réponses respectives pour obtenir une liste exhaustive des failles de sécurité et recommandations à suivre.

### Site(s) consulté(s)

Comme indiqué dans les étapes 4 et 5 de la méthodologie de recherche, la question centrale m'a permis de consulter les sources en ligne officielles.

Le site de la documentation d'Angular® proposait un guide relatif à la sécurité du framework mentionnant les éléments évoqués par l'IA tels que les failles XSS ou CSRF.

En outre, le site officiel d'Angular® renvoyait au site de l'OWASP (Open Worldwide application Security Project), source que je connaissais d'une conférence donnée par Gaëtan ELOUET lors de la Devoxx 2023. Cette conférence en ligne m'a ensuite mené au site CheatSheetSeries de l'OWASP qui détaille toutes les failles de sécurité et les solutions à appliquer pour chacune.

Le détail des sites consultés se trouve en annexe n°15.

### Critères de sélection

La sélection s'est faite de façon assez naturelle étant donnée la réputation des sources consultées telle que la documentation officielle d'Angular® ou l'OWASP dont la réputation et le sérieux m'avait été confirmé lors de la Conférence Devoxx 2023 précédemment évoquée.

### Solutions trouvées

Les solutions proposées au travers de ces différentes sources sont énumérées ci-dessous. Certaines sont présentées en détail dans le chapitre VII p.36 « Réalisations du candidat » :

1. Activer HTTPS auprès de l'hébergeur.
2. Utiliser un service d'authentification, tel que Lexik ou AuthO.
3. Utiliser les tokens JWT (JSON Web Tokens) pour gérer l'authentification et les autorisations.
4. Utiliser les directives Angular® formControl et formGroup pour valider les données entrées par l'utilisateur (voir ci-contre).
5. Échapper les caractères spéciaux (front et back).
6. Sécuriser API Platform® en vérifiant la validité des données transmises à l'API.
7. Protéger les formulaires en utilisant Validator d'Angular®.
8. Protéger la barre de recherche des injections SQL en validant les résultats avant de les valider, par l'utilisation d'expressions régulières.
9. Faire une mise à jour régulièrement des dépendances d'Angular® ainsi que toutes les dépendances tierces.
10. Effectuer un test de pénétration pour identifier et corriger les vulnérabilités restantes.

### Implémentation

Avant de procéder à l'implémentation et conformément avec les deux dernières étapes de la méthodologie, j'ai recueilli l'avis de développeurs web plus expérimentés. J'ai ainsi sollicité Hakim Saoudi développeur front-end Angular®/Java pour évoquer avec lui ces solutions.

Enfin, je me suis appuyé sur la documentation officielle d'Angular®, notamment en consultant le guide de sécurité, pour sécuriser le principal point de vulnérabilité de l'application : les formulaires. Les étapes d'implémentation du service Validator et des directives FormControl et formGroup sont décrites ci-après :

### I. Importation des classes FormGroup, FormControl et Validator dans la logique du composant (form.component.ts).

```

import { Component } from '@angular/core';
import { FormModel } from 'src/app/model/form';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Observable, catchError } from 'rxjs';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-form',
  templateUrl: './form.component.html',
  styleUrls: ['./form.component.scss']
})

export class FormComponent {
  formModel: FormModel = { name: '', email: '', message: '' }
  submitted = false;

  private apiUrl = 'https://127.0.0.1:8000/api'; // URL from API Platform
}

```

Figure 76. Importation

### 2. Instanciation des classes FormGroup et FormControl (toujours dans form.component.ts).

```

export class FormComponent {
  formModel: FormModel = { name: '', email: '', message: '' }
  submitted = false;

  private apiUrl = 'https://127.0.0.1:8000/api';

  form = new FormGroup({
    name: new FormControl('', [Validators.required]),
    email: new FormControl('', [Validators.required, Validators.email]),
    message: new FormControl('', [Validators.required])
  }, {});

  constructor()
}

```

Figure 77. Instanciation

3. Répartition des Validators pour chaque champ du formulaire. Notez que le champ email dispose d'une méthode de vérification spécifique (fournie par le Validator). Les méthodes « required » vérifient que le champ est rempli.

```

export class FormComponent {
  formModel: FormModel = { name: '', email: '', message: '' }
  submitted = false;

  private apiUrl = 'https://127.0.0.1:8000/api'; // URL from API Platform

  form = new FormGroup({
    name: new FormControl('', [Validators.required]),
    email: new FormControl('', [Validators.required, Validators.email]),
    message: new FormControl('', [Validators.required])
  }, {});
}

constructor()

```

Figure 78. Répartition des Validators et required

4. Dans le template (vue) du formulaire, intégration de la directive d'attribut [FormGroup] dans la balise <form> du formulaire (form.component.html).

Puis répartition des formControlName dans chaque champ du formulaire pour que la logique puisse suivre le contenu de ce champ.

```

<div class="title_form center">A question?</div>
<form [formGroup]="form" (ngSubmit)="onSubmitForm(formModel)">
  <div>
    <label for="name">Name</label>
    <input id="name" type="text" formControlName="name"/>
    <div class="error-container">
      <span>*ngIf="form.controls.name.touched && form.controls.name.errors?.['required']>
        Please, complet this field.
      </span>
    </div>
  </div>
  <div>
    <label for="email">Email</label>
    <input id="email" type="email" formControlName="email"/>
    <div class="error-container">
      <span>*ngIf="form.controls.email.touched && form.controls.email.errors?.['email']>
        Please, complet this field.
      </span>
      <span>*ngIf="form.controls.email.touched && form.controls.email.errors?.['email']>
        Invalid email.
      </span>
    </div>
  </div>
  <div>
    <label for="message">Message</label>
    <input id="message" type="text" formControlName="message"/>
    <div class="error-container">
      <span>
    </span>
    </div>
  </div>
</form>

```

Figure 79. Répartition des FormGroup et FormControl

## **ANNEXES**

# LISTE DES ANNEXES

**Annexe n°1 :** Charte graphique

**Annexe n°2 :** Plan de qualité

**Annexe n°3 :** Diagramme de Gantt

**Annexe n°4 :** Diagrammes UML 2.5

**Annexe n°5 :** Scénarios nominal et alternatifs

**Annexe n°6 :** Product backlog

**Annexe n°7 :** UX Design

**Annexe n°8 :** Visuels de conception

**Annexe n°9 :** Dictionnaire de données et matrice

**Annexe n°10 :** Schéma de base de données

**Annexe n°11 :** Convention de nommage

**Annexe n°12 :** Présentation du jeu d'essais et tests

**Annexe n°13 :** Résultats de recherche par IA

**Annexe n°14 :** Veille sécuritaire : sites consultés

**Annexe n°15 :** Situation de travail : sites consultés

**Annexe n°16 :** Webographie

**Annexe n°17 :** Références bibliographiques

**Annexe n°18 :** Lexique

# ANNEXE N°1

## Charte graphique

Cette annexe présente la charte graphique créée spécifiquement pour IFT-Blog.

### Guidelines IFT-Blog

#### Buttons

more-button



#### Icons

account



views



comments



search



subscribers



reddit



#### Navigation



#### Newsletter



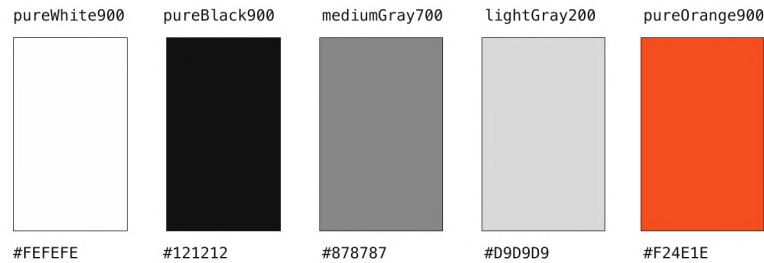
#### Articles



# Guidelines

## IFT-Blog

### Colors



### Fonts

Base Value: 16 Scale: 1.25

61px 3.812rem The quick brown fox jumps over the lazy dog

49px 3.062rem The quick brown fox jumps over the lazy dog

39px 2.438rem The quick brown fox jumps over the lazy dog

31px 2.038rem The quick brown fox jumps over the lazy dog

25px 1.622rem The quick brown fox jumps over the lazy dog

20px 1.250rem The quick brown fox jumps over the lazy dog

16px 1.000rem The quick brown fox jumps over the lazy dog

### Title

# Warning!

I found that!

# Warning!

I found that!

### Logo

pureOrange900  
**IFT-Blog**

pureBlack900

**IFT-Blog**

mediumGray700

**IFT-Blog**

# ANNEXE N°2

## Plan de qualité

Cette annexe présente le plan de qualité qui a été suivi pour ce projet.

Aa Subject	≡ Details (cf. Cahier des Charges MVP)	⊖ Status	≡ Comment
<u>Mise en ligne</u>	Le site est en ligne	To do	
<u>Fiabilité</u>	Les tests unitaires et fonctionnels ont été réalisés pour les features principales (article)	Done	
<u>Password</u>	Les mots de passe sont hachés dans la DB	Done	
<u>Programmation</u>	Les exigences de programmation sont satisfaites	Done	
<u>MVC</u>	L'architecture MVC est respectée dans Symfony	Done	
<u>MVVM</u>	L'architecture MVVM est respectée dans Angular	Done	
<u>Requêtes</u>	Les requêtes SQL sont préparées avec prepare statement (API custom)	Done	
<u>Commentaires</u>	Le code source est commenté en back et front	Done	
<u>Accessibilité</u>	l'application est accessible à tous les utilisateurs, y compris ceux ayant des besoins spécifiques (attributs ARIA)	Ongoing	A améliorer
<u>Performances</u>	l'application répond rapidement aux interactions de l'utilisateur et minimise les temps de chargement.	Ongoing	A améliorer
<u>Documentation</u>	Une documentation complète a été fournie pour le code source, l'architecture de l'application et les processus de déploiement	Ongoing	
<u>Maintenabilité</u>	Le code de manière modulaire et documentez-le efficacement pour faciliter la maintenance continue.	Done	

Aa Subject	≡ Details (cf. Cahier des Charges MVP)	▽ Status	≡ Comment
<u>Compatibilité</u>	L'application fonctionne correctement sur différents navigateurs web et appareils	Done	
<u>Gestion des erreurs</u>	Des mécanismes de gestion des erreurs ont été mis en place	Done	
<u>Performances SEO</u>	La structure de l'application favorise son classement dans les moteurs de recherche	Done	
<u>Respect de la Loi LCEN</u>	La Loi LCEN (Loi pour la Confiance dans l'Économie Numérique) relative à la responsabilité des acteurs du web a été consultée	Done	
<u>Respect du CIL</u>	Les mentions légales obligatoires définies par le CIL (Code de l'Information Légale) sont respectées	Ongoing	Non implémenté
<u>Respect du RGAA</u>	Les principaux points du RGAA (référentiel français d'accessibilité) ont été pris en compte au titre de guideline	Done	
<u>Respect du RGS</u>	Les principales recommandations RGS (Référentiel Général de Sécurité) ont été prise en compte	Done	
<u>Respect du WCAG</u>	Les principales recommandations du WCAG (Web Content Accessibility Guidelines) ont été prise en compte	Done	
<u>Respect reco ANSSI</u>	Les principales recommandations de l'ANSSI ont été suivies	Done	
<u>Respect reco CNIL</u>	Les principales recommandations de la CNIL ont été suivies	Done	
<u>Respect reco norme ISO/IEC 27001</u>	Des mécanismes de contrôle d'accès appropriés aux informations ont été mise en place	Done	
<u>Responsive Design</u>	Le site est Responsive (desktop, mobile, tablette)	Done	
<u>Rôles</u>	Les rôles utilisateurs sont défini et fonctionnel	Done	
<u>Sauvegarde et récupération</u>	Mise en place d'un plan de reprise d'activité (PRA)	Ongoing	Confié à la DevOps

# ANNEXE N°3

## Diagramme de Gantt

Cette annexe présente le diagramme de Gantt utilisé pour planifier le projet. Ce calendrier comprend deux périodes distinctes correspondant aux deux tâches réalisées en stage : développement sous Unity® (du 18/09/23 au 06/II/23) et développement web (du 06/II/23 au 24/01/24).

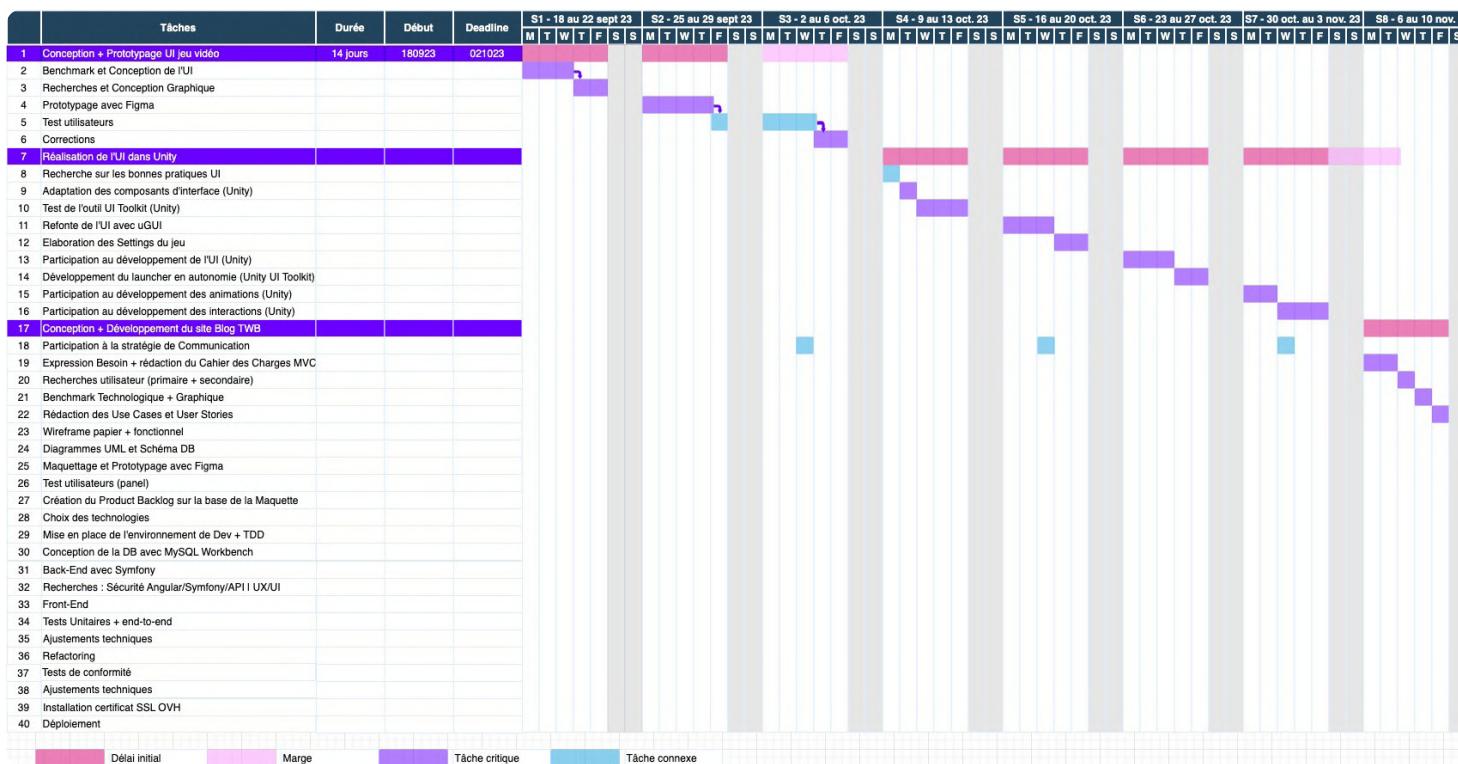
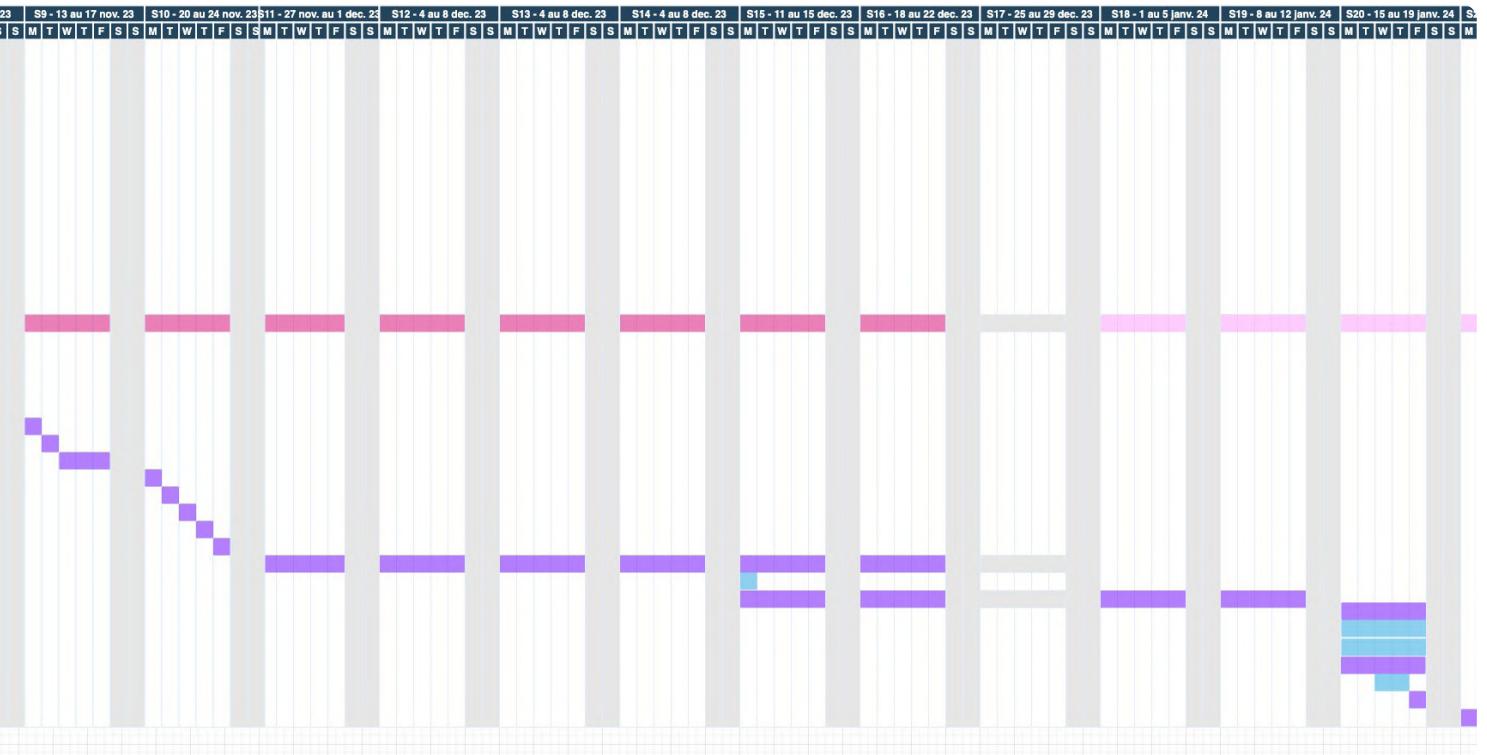


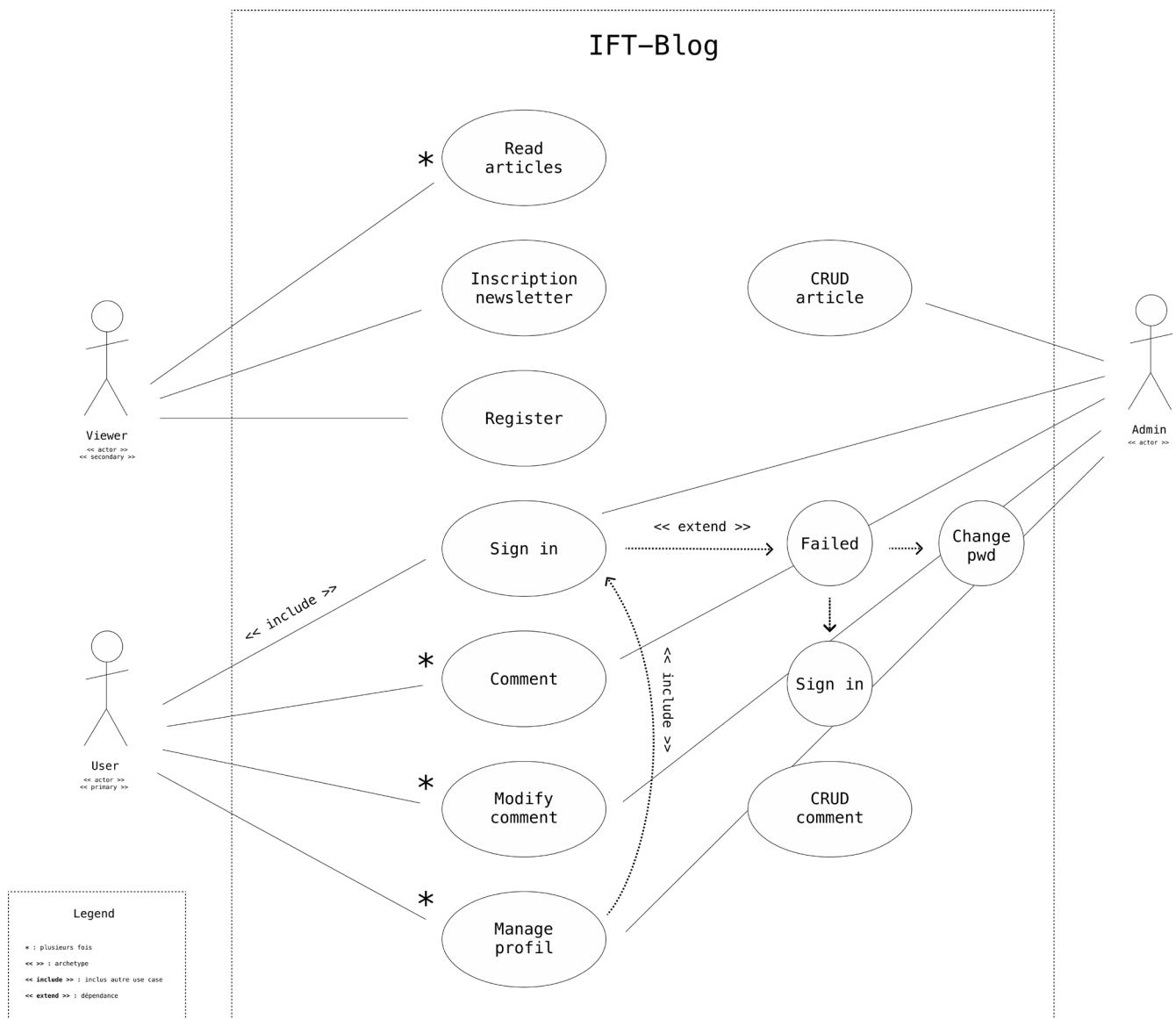
Diagramme de Gantt du projet IFT-Blog



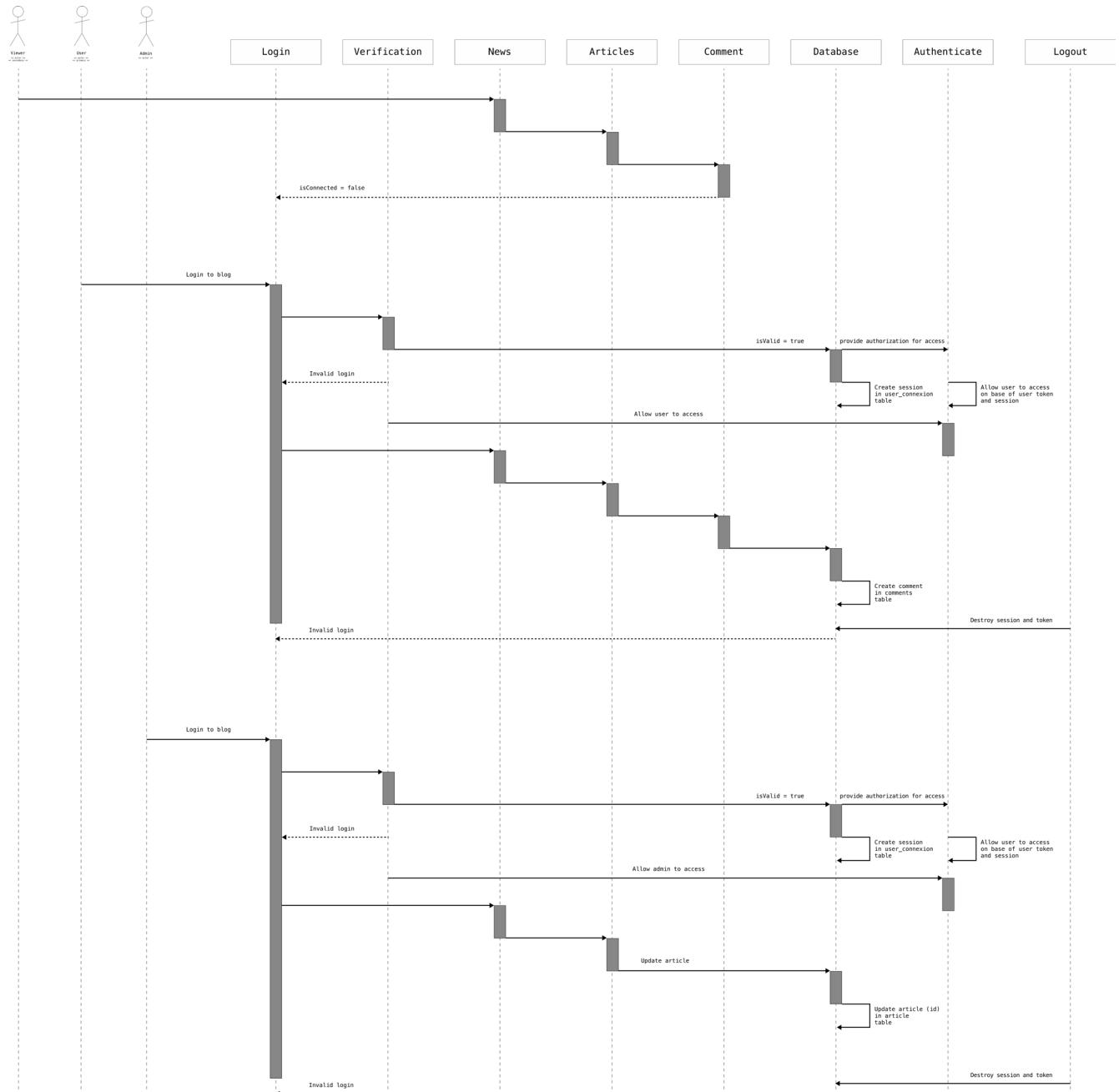
## ANNEXE N°4

# Diagrammes UML 2.5

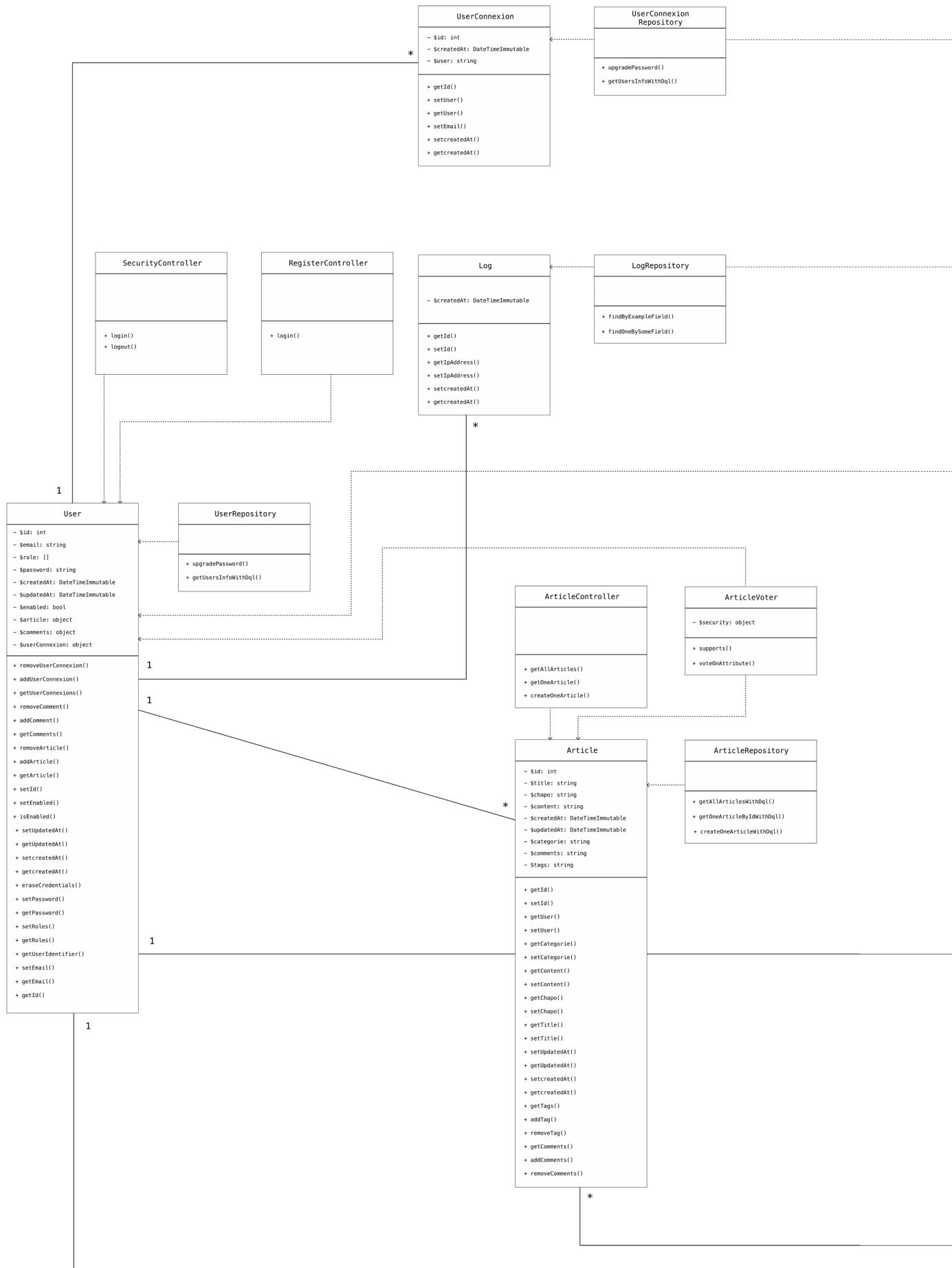
## Diagramme de cas d'utilisation

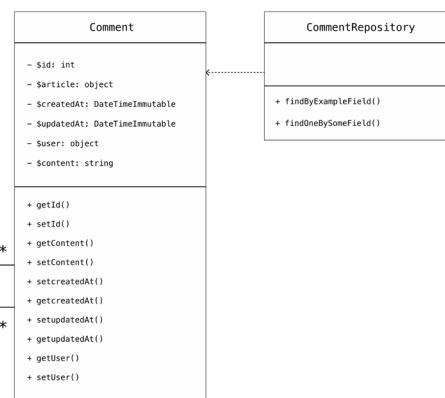
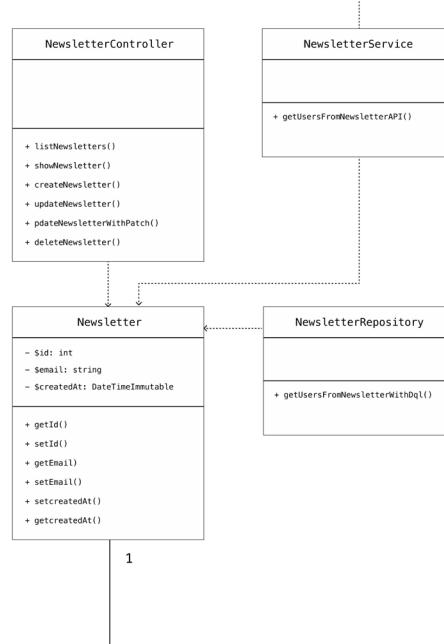
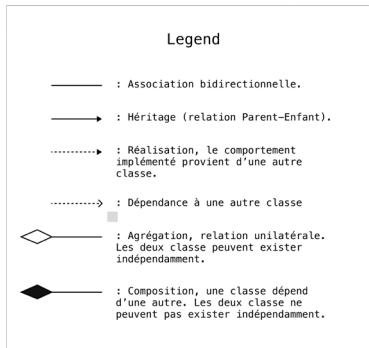
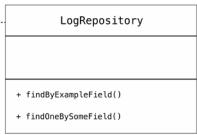
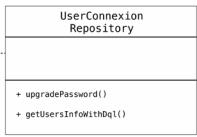


## Diagramme de séquence



## Diagramme de classe





# ANNEXE N°5

## Scénarios nominal et alternatif (Larman)

Cette annexe présente les différents scénarios d'utilisation de l'application selon la méthode de Larman (cf annexe n°15 « Références bibliographiques »).

≡ ACTOR	≡ ACTION
1 - L'utilisateur membre s'identifie sur le site à l'aide du bouton "account".	2 - Le système renvoie un formulaire d'authentification.
3 - L'utilisateur entre son adresse email et son mot de passe.	4 - Le système vérifie la cohérence des informations sur la table "user" de la base de données et crée un token de session user. 5 - Le système renvoie une vue de confirmation à l'utilisateur membre.
6 - L'utilisateur membre est connecté au blog.	
7 - L'utilisateur membre consulte l'article en news du blog.	8 - Le système renvoie le détail de la news.
9 - L'utilisateur membre consulte les autres articles du blog.	10 - Le système retourne la vue appropriée.
11 - L'utilisateur membre commente un article du site.	12 - Le système affiche l'interface de saisie.
13 - L'utilisateur membre valide son commentaire.	14 - Le système insère le commentaire dans la table "comments" de la base de données. 15 - Le système renvoie une vue de confirmation de prise en compte du commentaire.
16 - L'utilisateur membre se déconnecte.	17 - Le système détruit le token et la session.
18 - L'utilisateur membre est déconnecté.	

Tableau du scénario nominal « utilisateur membre » (user)

≡ ACTOR	≡ ACTION
1 - L'Admin s'identifie sur le site à l'aide du bouton "account".	2 - Le système renvoie un formulaire d'authentification.
3 - L'Admin accède à un formulaire d'authentification et entre son adresse email et son mot de passe.	4 - Le système vérifie la cohérence des informations sur la table "user" de la base de données et crée un token de session admin.  5- Le système renvoie une vue de confirmation à l'utilisateur Membre.
6 - L'Admin est connecté au blog.	
7 - L'Admin modifie un article du site.	8 - Le système retourne le formulaire de modification d'un article.
9 - L'Admin valide sa modification.	10 - Le système intègre la correction dans la table "article" de la base de données, sur l'article concerné.  11 - Le système renvoie une vue de confirmation de prise en compte du commentaire.
12 - L'Admin se déconnecte.	13 - Le système détruit le token et la session.
14 - L'Admin est déconnecté.	

Tableau du scénario nominal « Admin »

7 - L'utilisateur viewer consulte l'article en news du blog.	8 - Le système renvoie le détail de la news.
9 - L'utilisateur viewer consulte les autres articles du blog.	10 - Le système retourne la vue appropriée.
11 - L'utilisateur viewer commente un article du site.	12 - Le système vérifie que l'utilisateur viewer est bien connecté.  - 13 - Le système renvoie la vue "login".

Tableau du scénario alternatif « utilisateur non-connecté » (viewer)

# ANNEXE N°6

## Product backlog

Ce tableau représente l'ensemble des user stories du projet et les fonctionnalités associées.

▽ User	# ID	Status	▽ Priority	☰ Theme	☰ Item (User Story)	Aa Feature	# Story Point
USER	2309116	Done	High	UI	<p><u>US</u> : En tant qu'UTILISATEUR, je veux pouvoir naviguer sur le site grâce à une interface simple.</p> <p><u>Gherkin</u> :</p> <pre>Given un utilisateur When l'utilisateur utilise le site Desktop Then les composants d'interface respectent les standards UI</pre>	<u>Interface utilisateur</u>	13
USER	2309113	Done	High	RESPONSIVE	<p><u>US</u> : En tant qu'UTILISATEUR, je veux pouvoir consulter le site sur Mobile et Desktop.</p> <p><u>Gherkin</u> :</p> <pre>Given un utilisateur When l'utilisateur utilise le site sur Mobile Then le Design s'adapte à son device</pre>	<u>Design Responsive</u>	13
USER	230911	In progress	High	SIGN UP	<p><u>US</u> : "En tant qu'UTILISATEUR, je veux pouvoir m'inscrire sur la plateforme en fournissant mon adresse e-mail et un mot de passe".</p> <p><u>Gherkin</u> :</p> <pre>Given : un utilisateur non enregistré When : l'utilisateur fournit son adresse e- mail et un mot de passe Then : le compte utilisateur est créé avec succès</pre> <p>formulaire d'inscription avec les champs email (text field), et</p>	<u>Inscription d'un utilisateur sur la plateforme</u>	13

⌚ User	# ID	Status	⌚ Priority	☰ Theme	☰ Item (User Story)	Aa Feature	# Story Point
					<p>confirmation email, mot de passe (text field), politique de confidentialité (checkbox + message "RGPD friendly 🌱 - Désinscription à tout moment et effacement de vos données personnelles automatique après 5 ans"), une checkbox "please contact me by email (newsletter)", un lien "Already Member? Log in now!" qui renvoie à la page Sign In.</p>		
USER	230912	In progress	High	SIGN IN	<p><u>US</u> : En tant qu'UTILISATEUR, je veux pouvoir me connecter à mon compte avec mon adresse e-mail et mon mot de passe.</p> <p><u>Gherkin</u> :</p> <pre>Given un utilisateur enregistré When l'utilisateur se connecte avec son adresse e-mail et son mot de passe Then l'utilisateur est redirigé vers son tableau de bord</pre>	<a href="#"><u>Connexion d'un utilisateur à la plateforme</u></a>	13
USER	230917	In progress	Medium	COMENT	<p><u>US</u> : En tant qu'UTILISATEUR, je veux pouvoir ajouter des commentaires sur les articles publiés par d'autres utilisateurs.</p> <p><u>Gherkin</u> :</p> <pre>Given un utilisateur connecté et un article publié When l'utilisateur ajoute un commentaire à l'article Then le commentaire est affiché sous l'article</pre>	<a href="#"><u>Commenter un article</u></a>	34

⌚ User	# ID	Status	⌚ Priority	☰ Theme	☰ Item (User Story)	Aa Feature	# Story Point
USER	230918	In progress	Low	COMENT	<p><u>US :</u> En tant qu'UTILISATEUR, je veux pouvoir modifier ou supprimer mes propres commentaires.</p> <p><u>Gherkin :</u> Given un utilisateur connecté avec un commentaire existant When l'utilisateur modifie ou supprime son commentaire Then les modifications ou la suppression du commentaire sont enregistrées</p>	<u>Suppression/modification de commentaires par l'utilisateur</u>	13
USER	230919	Done	High	USE	<p><u>US :</u> En tant qu'UTILISATEUR, je veux pouvoir parcourir les articles publiés.</p> <p><u>Gherkin :</u> Given plusieurs articles publiés When l'utilisateur parcourt les articles Then l'utilisateur peut voir une liste d'articles</p>	<u>Visualisation d'une liste d'articles de blog sur une page</u>	3
USER	2309110	In progress	Medium	SEARCH	<p><u>US :</u> En tant qu'UTILISATEUR, je veux pouvoir rechercher des articles en fonction de mots-clés ou de sujets.</p> <p><u>Gherkin :</u> Given plusieurs articles publiés, une zone de recherche When l'utilisateur entre un mot clé dans la zone de recherche Then l'utilisateur peut voir tous les articles et commentaires comprenant ce mot clé</p>	<u>Recherche sur le site</u>	34
USER	2309111	Done	Low	SHARE	<u>US :</u> En tant qu'UTILISATEUR, je veux pouvoir partager le contenu d'un article sur Reddit.	<u>Partage sur Reddit</u>	3

▽ User	# ID	Status	▽ Priority	≡ Theme	≡ Item (User Story)	Aa Feature	# Story Point
					<p><u>Gherkin :</u>  Given un utilisateur connecté  When l'utilisateur clique sur l'icône Reddit OU sur l'icône Partager  Then l'utilisateur peut voir une pop-up s'afficher avec le lien à copier/coller</p>		
USER	2309112	In progress	Low	PRIVACY	<p><u>US :</u> En tant qu'UTILISATEUR, je veux être informé de la politique de confidentialité du site lors de mon inscription.</p> <p><u>Gherkin :</u>  Given un utilisateur non enregistré  When l'utilisateur doit cocher une case relative à la politique de confidentialité  Then un message d'erreur apparaît si la case n'est pas cochée</p>	<u>Information de la politique de confidentialité</u>	3
USER	2309114	In progress	Low	ACCESSIBILITY	<p><u>US :</u> En tant qu'UTILISATEUR, je veux pouvoir naviguer sur le site par le clavier.</p> <p><u>Gherkin :</u>  Given un utilisateur  When l'utilisateur utilise les touches du clavier  Then la navigation s'effectue</p>	<u>Navigation par clavier</u>	13
USER	2309115	Not started	Low	ACCESSIBILITY	<p><u>US :</u> En tant qu'UTILISATEUR, je veux pouvoir accéder à un menu accessibilité.</p> <p><u>Gherkin :</u>  Given un utilisateur  When l'utilisateur active un item "accessibilité"  Then les touches du clavier indiquées permettent de naviguer</p>	<u>Menu dédié à l'accessibilité proposant différentes options</u>	34

⌚ User	# ID	Status	⌚ Priority	≡ Theme	≡ Item (User Story)	Aa Feature	# Story Point
ADMIN	230921	In progress	High	CRUD	<p><u>US</u> : En tant qu'ADMINISTRATEUR, je veux pouvoir gérer les comptes d'utilisateurs (désactiver, supprimer).</p> <p><u>Gherkin</u> :</p> <pre>Given un administrateur connecté When l'administrateur gère les comptes utilisateurs Then les modifications sont effectuées avec succès</pre>	<u>Gestion des comptes utilisateurs</u>	21
ADMIN	230922	In progress	High	COMENT	<p><u>US</u> : En tant qu'ADMINISTRATEUR, je veux pouvoir modérer les commentaires (approuver, rejeter, supprimer).</p> <p><u>Gherkin</u> :</p> <pre>Given un administrateur connecté avec des commentaires en attente When l'administrateur modère les commentaires Then les commentaires sont approuvés ou rejettés selon la décision de l'administrateur</pre>	<u>Modération des commentaires</u>	34
ADMIN	230923	Not started	Low	MODERATION	<p><u>US</u> : En tant qu'ADMINISTRATEUR, je veux pouvoir désactiver temporairement un utilisateur en cas de non-respect des règles de la plateforme.</p> <p><u>Gherkin</u> :</p> <pre>Given un administrateur connecté et un utilisateur en infraction When l'administrateur</pre>	<u>Désactivation d'un utilisateur</u>	21

⌚ User	# ID	Status	⌚ Priority	☰ Theme	☰ Item (User Story)	Aa Feature	# Story Point
					désactive temporairement la publication d'articles par cet utilisateur Then l'utilisateur ne peut pas publier de nouveaux articles pendant la période de désactivation		
ADMIN	230924	Not started	Low	STATISTIC	<p><u>US</u> : En tant qu'ADMINISTRATEUR, je veux pouvoir visualiser des statistiques sur l'utilisation de la plateforme (nombre d'articles, commentaires, utilisateurs, etc.).</p> <p><u>Gherkin</u> :</p> <pre>Given un administrateur connecté When l'administrateur demande des statistiques Then les statistiques sont affichées avec succès</pre>	<a href="#">Dashboard des Statistiques + Google Analytics</a>	13
ADMIN	230925	Not started	High	SECURITY	<p><u>US</u> : En tant qu'ADMINISTRATEUR, je veux pouvoir recevoir des notifications ou des rapports sur les activités suspectes ou non conformes.</p> <p><u>Gherkin</u> :</p> <pre>Given un administrateur connecté When des activités suspectes ou non conformes sont détectées Then l'administrateur reçoit des notifications ou des rapports</pre>	<a href="#">Notifications de sécurité par email</a>	55
ADMIN	230926	Not started	Low	NEWSLETTER	<p><u>US</u> : En tant qu'ADMINISTRATEUR, je veux pouvoir envoyer le contenu d'un article ou</p>	<a href="#">Newsletter</a>	34

⌚ User	# ID	Status	⌚ Priority	☰ Theme	☰ Item (User Story)	Aa Feature	# Story Point
					<p>plusieurs directement par email aux abonnés.</p> <p><u>Gherkin :</u>  Given un administrateur connecté  When l'administrateur peut sélectionner les articles par le biais de checkbox, puis cliquer sur un bouton  Then l'administrateur reçoit une demande de confirmation avant envoi avec une sélection de la base de données Abonnée</p>		
ADMIN	230927	In progress	Low	RGPD	<p><u>US :</u> En tant qu'ADMINISTRATEUR, je veux que les données des utilisateurs soient automatiquement supprimées après 5 ans pour respecter le RGPD.</p> <p><u>Gherkin :</u>  Given les données personnelles d'un utilisateur enregistrées dans la DB  When automatique à la date anniversaire de l'inscription à N+5  Then un email d'information est envoyé à J-1 semaine puis à J-1 et à la suppression pour confirmation</p>	<u>Suppression automatique des données utilisateurs</u>	8
ADMIN	230928	Done	High	CRUD	<p><u>US :</u> En tant qu'ADMINISTRATEUR, je veux pouvoir rédiger un nouvel article de blog.</p> <p><u>Gherkin :</u>  Given un utilisateur connecté  When l'utilisateur rédige un nouvel article</p>	<u>Edition des articles</u>	34

⌚ User	# ID	Status	⌚ Priority	☰ Theme	☰ Item (User Story)	Aa Feature	# Story Point
					de blog Then l'article est sauvegardé en brouillon		
ADMIN	230929	Done	High	CRUD	<u>US</u> : En tant qu'ADMINISTRATEUR, je veux pouvoir éditer mes articles existants.  <u>Gherkin</u> : Given un utilisateur connecté avec un article existant When l'utilisateur modifie son article Then les modifications sont enregistrées avec succès	<u>Modification des articles</u>	3
ADMIN	23091210	Done	High	CRUD	<u>US</u> : En tant qu'ADMINISTRATEUR je veux pouvoir supprimer mes articles.  <u>Gherkin</u> : Given un utilisateur connecté avec un article existant When l'utilisateur supprime son article Then l'article est supprimé de manière permanente	<u>Suppression des articles</u>	3
ADMIN	2309211	Done	High	CRUD	<u>US</u> : En tant qu'ADMINISTRATEUR, je veux pouvoir publier mes articles pour qu'ils soient visibles par d'autres utilisateurs.  <u>Gherkin</u> : Given un utilisateur connecté avec un article en brouillon When l'utilisateur publie son article Then l'article devient visible pour tous les utilisateurs	<u>Autoincrémentation du composant app-articles dans body-section</u>	3
ADMIN	2309212	Not started	Low	SECURITY	<u>US</u> : En tant qu'ADMINISTRATEUR, je veux que les logs soient conservés et consultables dans un	<u>Journalisation des Logs</u>	34

⌚ User	# ID	Status	⌚ Priority	≡ Theme	≡ Item (User Story)	Aa Feature	# Story Point
					<p>journal.</p> <p><u>Gherkin</u></p> <p>:</p> <p>Given l'utilisateur se connecte puis se déconnecte</p> <p>When l'administrateur se connecte à son espace</p> <p>Then un item permet de visualiser le journal des logs (pseudo + horodatage de la connexion)</p>		
ADMIN	230913	Done	Low	ECOCONCEPTION	<p><u>US</u> : En tant qu'ADMINISTRATEUR, je veux que le site soit conçu de manière responsable.</p> <p><u>Gherkin</u></p> <p>:</p> <p>Given un utilisateur ou administrateur</p> <p>When le site est scrollé</p> <p>Then les images ne sont chargées que si c'est nécessaire</p>	<a href="#"><u>Lazy Loading et suivi des bonnes pratiques</u></a>	13
ADMIN	230914	Not started	Low	CMS	<p><u>US</u> : En tant qu'ADMINISTRATEUR, je veux pouvoir administrer mon site seul.</p> <p><u>Gherkin</u></p> <p>:</p> <p>Given un administrateur connecté</p> <p>When l'admin utilise le dashboard du CMS Sulu intégré au site</p> <p>Then les articles peuvent être créés, modifiés ou supprimés depuis le dashboard Sulu</p>	<a href="#"><u>Intégration du site au Skeleton Sulu</u></a>	55



# ANNEXE N°7

## UX Design : recherche secondaire

Cette annexe présente les captures d'écran des différentes sources consultées dans le cadre de la recherche utilisateur nécessaire à la conception du site IFT-Blog.

### Les source consultées

Les chiffres principaux ont été trouvés sur le site de l'Agence Française pour le Jeu Vidéo qui faisait référence à l'étude annuel du média We Are Social dans son Digital Report 2023 :

**“Digital Report 2023”**, Agence Française pour le Jeu Video (AFJV) [en ligne], 2023. Disponible à l'adresse : [https://www.afjv.com/news/lI08l\\_digital-report-2023-datas-tendances-gaming.htm](https://www.afjv.com/news/lI08l_digital-report-2023-datas-tendances-gaming.htm) [consultée le 08/lI/23].

Les avis utilisateurs ont été consultés sur la platform Steam®. Le jeu Madison® a servi de référence pour l'analyse des attendus des utilisateurs :

**“Madison®”**, Steam [en ligne], 2023. Disponible à l'adresse : <https://store.steampowered.com/app/l670870/MADiSON/> [consultée le 08/lI/23].

Les différents réseaux sociaux du jeu Madison® ont également été consultés pour élargir la source des avis.

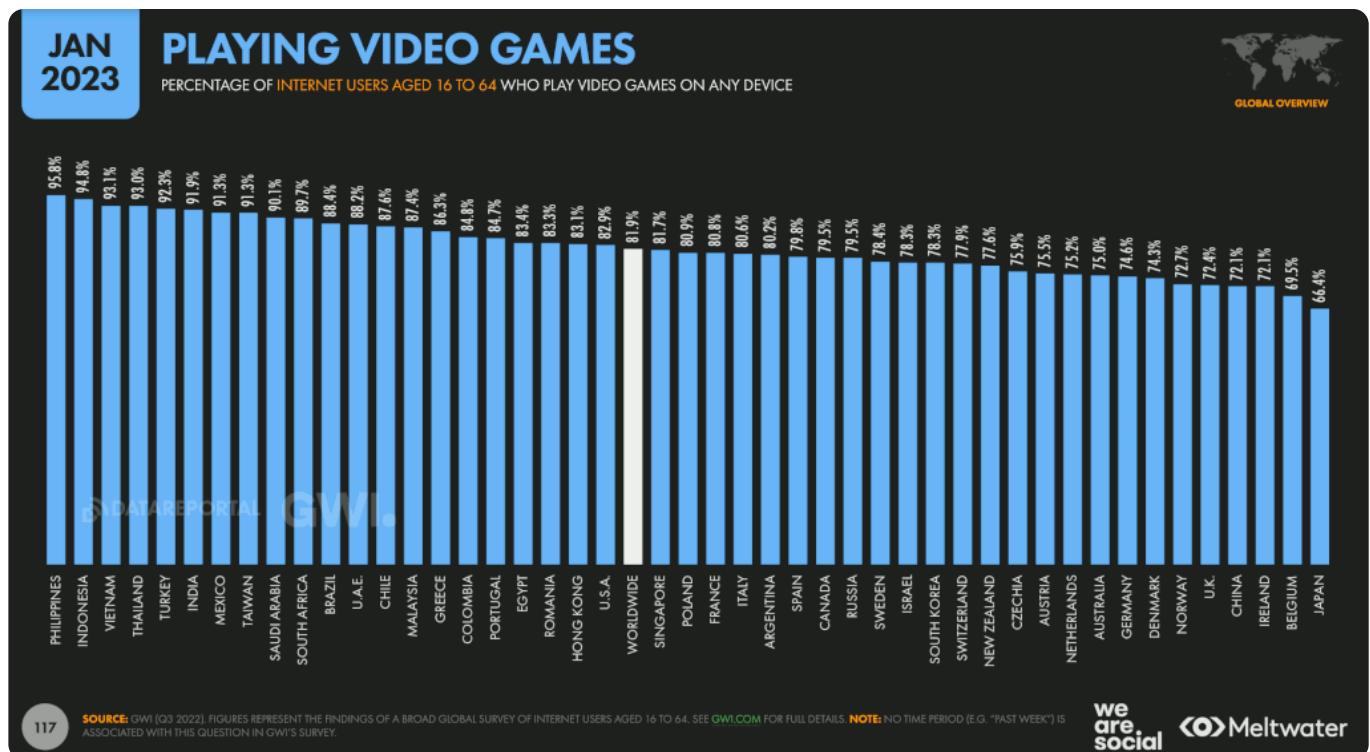
Puis, afin de bien cerner le vocabulaire utilisé par les utilisateurs cibles, la plateforme YouTube® a été consultée également :

**“Je joue à Madison et c'est fou !”**, Carole Quintaine [en ligne], 2023. Disponible à l'adresse : [https://www.youtube.com/watch?v=jQ8ty\\_GtHzU](https://www.youtube.com/watch?v=jQ8ty_GtHzU) [consultée le 08/lI/23].

## Principaux chiffres



66,2% des internautes de 16-64 ans jouent à des jeux vidéo sur smartphone / 37,9% sur ordinateur / 25,3% sur console de salon / 16,5% sur tablette / 8,7% avec casque de VR.

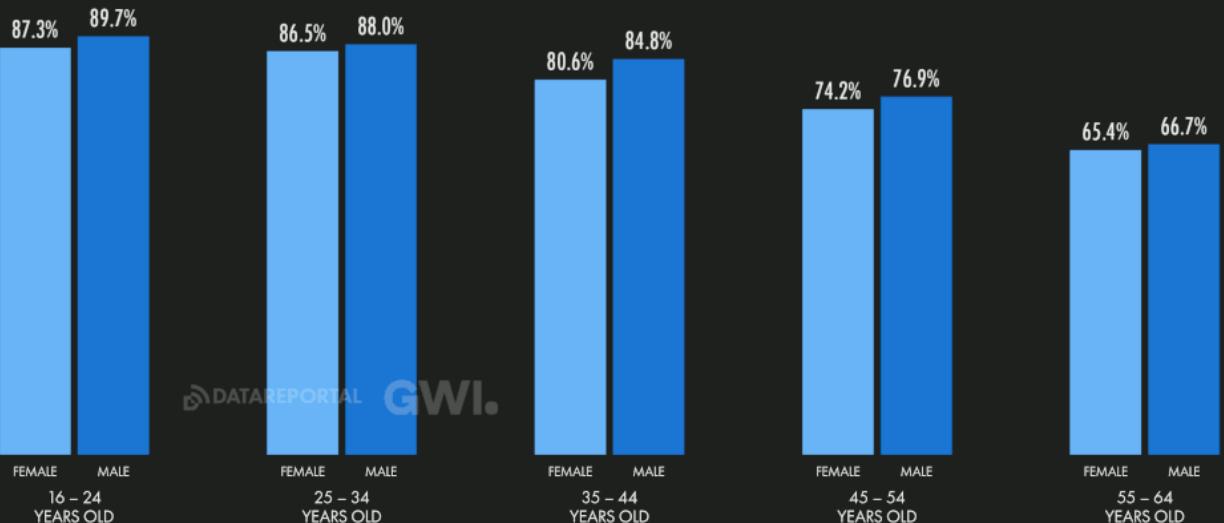


81,9% des internautes de 16-64 ans jouent à des jeux vidéos dans le monde (80,8% pour la France).

JAN  
2023

## PLAYING VIDEO GAMES

PERCENTAGE OF INTERNET USERS WHO PLAY VIDEO GAMES ON ANY DEVICE



118

SOURCE: GWI (Q3 2022). FIGURES REPRESENT THE FINDINGS OF A BROAD GLOBAL SURVEY OF INTERNET USERS AGED 16 TO 64. SEE [GWI.COM](#) FOR FULL DETAILS. NOTE: NO TIME PERIOD (E.G. "PAST WEEK") IS ASSOCIATED WITH THIS QUESTION IN GWI'S SURVEY.

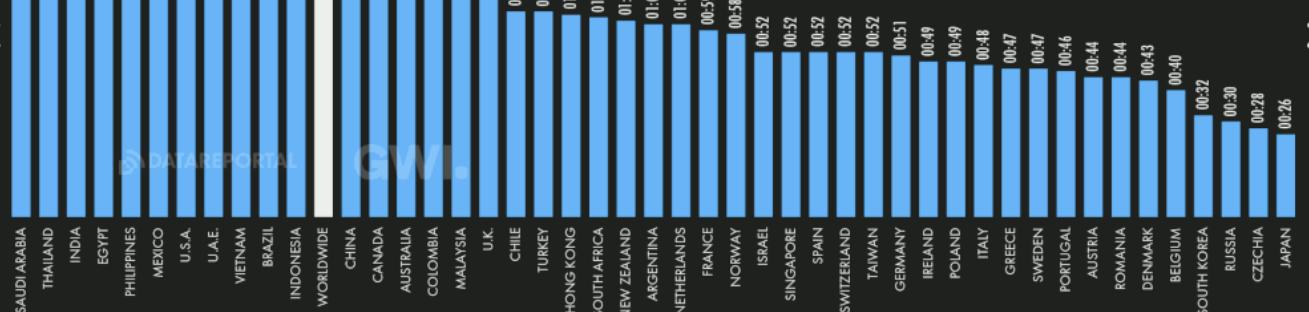
**we  
are,  
social** Meltwater

Parité dans les jeux vidéo, par tranches d'âge.

JAN  
2023

## DAILY TIME SPENT USING A GAMES CONSOLE

AVERAGE AMOUNT OF TIME (IN HOURS AND MINUTES) THAT INTERNET USERS AGED 16 TO 64 SPEND USING A GAMES CONSOLE EACH DAY



120

SOURCE: GWI (Q3 2022). FIGURES REPRESENT THE FINDINGS OF A BROAD GLOBAL SURVEY OF INTERNET USERS AGED 16 TO 64. SEE [GWI.COM](#) FOR FULL DETAILS.

**we  
are,  
social** Meltwater

Chaque jour, les internautes de 16-64 ans passent 1h14 à jouer sur une console de jeux vidéo dans le monde (59 minutes pour la France).

JAN  
2023

## MOST POPULAR VIDEO GAME FORMATS

PERCENTAGE OF INTERNET USERS WHO PLAY VIDEO GAMES EACH MONTH WHO ALSO PLAY EACH GENRE OF VIDEO GAME ON ANY DEVICE



16 TO 24 YEARS OLD		25 TO 34 YEARS OLD		35 TO 44 YEARS OLD		45 TO 54 YEARS OLD		55 TO 64 YEARS OLD	
SHOOTER	66%	SHOOTER	64%	SHOOTER	56%	ACTION ADVENTURE	44%	ACTION ADVENTURE	33%
ACTION ADVENTURE	62%	ACTION ADVENTURE	60%	ACTION ADVENTURE	54%	SHOOTER	43%	SHOOTER	32%
SIMULATION	43%	RACING	45%	SPORTS	42%	PUZZLE PLATFORM	34%	PUZZLE PLATFORM	31%
M.O.B.A.	42%	SPORTS	44%	RACING	41%	RACING	32%	SPORTS	24%
SPORTS	41%	M.O.B.A.	43%	PUZZLE PLATFORM	40%	SPORTS	31%	RACING	23%
RACING	40%	SIMULATION	43%	STRATEGY	39%	STRATEGY	30%	SIMULATION	22%
BATTLE ROYALE	40%	STRATEGY	43%	SIMULATION	38%	SIMULATION	29%	STRATEGY	22%
STRATEGY	40%	PUZZLE PLATFORM	41%	ACTION PLATFORM	35%	ACTION PLATFORM	26%	ACTION PLATFORM	20%
PUZZLE PLATFORM	36%	FIGHTING	38%	M.O.B.A.	35%	FIGHTING	25%	ONLINE BOARD GAMES	19%
FIGHTING	34%	ACTION PLATFORM	37%	FIGHTING	34%	M.O.B.A.	24%	M.O.B.A.	18%

122

SOURCE: GWI (Q3 2022). FIGURES REPRESENT THE FINDINGS OF A BROAD GLOBAL SURVEY OF INTERNET USERS AGED 16 TO 64. SEE [GWI.COM](#) FOR FULL DETAILS.

NOTE: PERCENTAGES REPRESENT SHARE OF INTERNET USERS IN EACH AGE GROUP WHO PLAY VIDEO GAMES EACH MONTH, AND WHO ALSO SAY THEY PLAY EACH GENRE OF VIDEO GAME



Les types de jeux vidéo les plus populaires, par tranche d'âge.

## Avis utilisateurs

Votre magasin    Nouveaux et à découvrir    Catégories    Boutique des points    Actualités    Labs    Rechercher

Tous les jeux > Aventure > franchise BLOODY GAMES > MADISON

**MADISON**    Hub de la communauté





**MADISON** est un jeu d'horreur psychologique à la première personne qui offre une expérience immersive et terrifiante. A l'aide d'un appareil photo, connectez le monde des vivants à celui de l'au-delà. Résolvez des énigmes, explorez votre

87 % des 2,025 évaluations des utilisateurs et utilisatrices pour ce jeu sont positives.

EVALUATIONS RECENTES : très positives (7/10)

TOUTES LES ÉVAL. : très positives (2,025)

DATE DE PARUTION : 8 juil. 2022

DÉVELOPPEMENT : BLOODY GAMES

EDITION : BLOODY GAMES

Tags populaires des utilisateurs et utilisatrices pour ce jeu :











Le jeu Madison® a servi de référence pour le recueil des avis utilisateurs.

## ÉVALUATIONS LES PLUS PERTINENTES GLOBALES

 **Proutman**  
179 produits enregistrés  
26 évaluations

 **Recommandé**  
19.1 h en tout (4.8 heure(s) lors de l'évaluation)

ÉVALUATION PUBLIÉE LE 9 JUILLET 2022

Pour le moment je n'ai toujours pas fini le jeu mais je donne déjà mon avis pour ceux qui hésiteraient!  
Sachez que je joue énormément aux jeux d'horreurs indé et j'ai eu un coup de cœur pour ce jeu.  
J'y ai joué 4h et pour le moment je suis totalement conquis par l'ambiance, l'histoire, les personnages, le sound design et les décors. Perso, je n'avais pas du tout aimé Visage (me demandez pas pourquoi). Mais ce jeu, j'aime énormément, les énigmes demandent parfois un peu de fil à retordre mais tout reste assez logique et bien scripté. Franchement pour le moment, je vous le conseille !  
Voici le résumé de mes 4h de jeux en 40 minutes de vidéo :  
<https://youtu.be/ExxE8q9U2X0>

[LIRE LA SUITE](#)

Cette évaluation vous a-t-elle été utile ?

 Oui  Non  Amusante  Récompenser

27 personnes ont trouvé cette évaluation utile  
1 personne a trouvé cette évaluation amusante



 **Mademoiselle Castle...**  
6 évaluations

 **Recommandé**  
10.2 h en tout (7.2 heure(s) lors de l'évaluation)

ÉVALUATION PUBLIÉE LE 7 AOUT 2022

Je ne sais pas par quoi commencer, ce jeu est absolument un "hors norme" du style horreur.  
Je ne fais que du jeu d'horreur, j'ai souvent peur mais je tiens le coup quand même, cependant Madison m'a mise dans un état second.

## PUBLIÉES RÉCEMMENT

 **LACARROTEFROIDE** 11.5 heures  
ÉVALUATION PUBLIÉE LE 6 JANVIER  
It's just the best horror game ever made.

Cette évaluation vous a-t-elle été utile ?

 Oui  Non  Amusante  Récompenser

 **Miss FoxKat** 16.0 heures  
ÉVALUATION PUBLIÉE LE 28 DÉCEMBRE 2023  
One of the scariest games I've ever played. The puzzles change each time so that you can replay without getting bored of the gameplay. The story is downright horrifying and the voice acting is well done. I absolutely LOVE this game and recommend it to anyone who likes first person shooters.

Cette évaluation vous a-t-elle été utile ?

 Oui  Non  Amusante  Récompenser

 **Xionix** 9.9 heures  
ÉVALUATION PUBLIÉE LE 26 DÉCEMBRE 2023  
FLIPPANT

Cette évaluation vous a-t-elle été utile ?

 Oui  Non  Amusante  Récompenser

 **Laure.Noobieland YT** 9.8 heures  
ÉVALUATION PUBLIÉE LE 7 JUILLET 2023

## Avis utilisateurs sur Steam®

 **[ES.GR] masterriders**  
220 évaluations

 **Recommandé**  
3.6 h en tout

ÉVALUATION PUBLIÉE LE 8 JANVIER 2023

Produit reçu gratuitement

Tout d'abord merci à Doya pour ce cadeau. ^^  
Madison est un jeu horrifique produit par le studio argentin Bloodous Games, sortis en 2022 il a rapidement gravi les échelons de l'horreur pour se hisser au grade de jeux d'horreur la plus effrayante de l'année, mais est-ce la vérité? Ce jeu terrifiant en 2022 l'est-il toujours autant aujourd'hui?  
Graphiquement reste-t-il dans la norme alors que les avancées techniques vont de plus en plus loin?

### Ce qui nous terrifie le plus? Est-ce nous même?

Dans cette sombre aventure vous jouerez le rôle de Luca, un homme qui a été possédé par un être étrange et à moitié aveugle qui a forcé votre personnage à terminer un rituel

[LIRE LA SUITE](#)

Cette évaluation vous a-t-elle été utile ?

 Oui  Non  Amusante  Récompenser

6 personnes ont trouvé cette évaluation utile

8



 **atraout19**  
224 produits enregistrés  
3 évaluations

 **Recommandé**  
7.8 h en tout

ÉVALUATION PUBLIÉE LE 15 JUILLET 2022

Habituer au Survival horror old school (silent hill, project zero, etc...) ce jeu était moyen bon  
je m'explique il n'est pas mauvais d'où ma recommandation, j'ai eu peur mais uniquement car ça seul est unique mécanique de peur est le jump scare ... j'aurai aimé plus d'ambiance, plus d'inconnu, plus de " non non non je n'irai pas par là y a pas moyen!"  
Certes le jeu est court 7h en premier run pour moi.  
Les énigmes sont correctes, on galère juste ce qu'il faut mais après quelque temps on comprend ce qu'il faut faire donc parfait (un manque de cohérence entre énigme et atmosphère nous sort de l'immersion).  
L'histoire ne m'a pas transporté... le développement de l'intrigue

[LIRE LA SUITE](#)

Cette évaluation vous a-t-elle été utile ?

 Oui  Non  Amusante  Récompenser

 **Oui**  **Non**  **Amusante**  **Récompenser**  
1 personne a trouvé cette évaluation utile

 **BenJ36** 6.0 heures  
ÉVALUATION PUBLIÉE LE 16 AVRIL 2023

Jeu d'angoisse qui nous met rapidement dans l'ambiance. Petit frère éloigné de "Visage", je le trouve malgré tout un peu court dans sa durée de vie.  
Qualité graphique vraiment top, ambiance et quelques sursauts au rendez-vous. Je conseille vivement ce jeu!

Cette évaluation vous a-t-elle été utile ?

 Oui  Non  Amusante  Récompenser

1 personne a trouvé cette évaluation utile

 **RawoulDeTurquie** 7.5 heures  
ÉVALUATION PUBLIÉE LE 13 AVRIL 2023

"MADISON est un jeu d'horreur psychologique à la première personne qui offre une expérience immersive et terrifiante."  
Jamais aussi bien résumé...  
Je recommande très fortement

Cette évaluation vous a-t-elle été utile ?

 Oui  Non  Amusante  Récompenser

1 personne a trouvé cette évaluation utile

## Avis utilisateurs sur Steam®

 **SPARTAN**  
130 produits enregistrés  
15 évaluations

 **Recommandé**  
8.3 h en tout

ÉVALUATION PUBLIÉE LE 27 OCTOBRE 2023

---{Graphique}---

- Vous oubliez ce qu'est la réalité
- Magnifique
- Bien
- Décent
- Mauvais
- Ne le regardez pas trop longtemps
- MS-DOS

---{Gameplay}---

- Très bien
- Bien
- C'est juste du gameplay
- Meh
- Regardez plutôt la peinture sécher
- Ne le fais pas

---{L'audio}---

- Eargasm (littéralement)
- Très bien
- Bien
- Pas trop mal
- Mauvais
- Je suis maintenant sourd

---{Public}---

- Tous
- Adultes
- Ados
- Enfants

---{Configuration PC requise}---

- Demandez à la NASA si elle a un ordinateur d...
- Riche mec
- Rapide
- Décent
- Pomme de terre

---{Difficulté}---

- Âmes sombres
- Difficile
- Comme une gifle
- Utilisation importante du cerveau
- Facile
- Appuyez simplement sur « W »

---{Histoire}---

- Cela remplacera ta vie
- Charmant
- Bien
- Moyenne

Avis utilisateurs sur Steam®

 **BLOODIOUS GAMES**   
379 posts

 **BLOODIOUS GAMES**  @BLOODIOUSGAMES · 29 avr. 2022 ...  
Preorders are becoming available, and with awesome discounts!  
Stay tuned everyone   
Thank you! @MeridiemGames @videooca 

 **Tiendas VIDEO OCA** @videooca · 29 avr. 2022  
Ya podéis reservar la propuesta de terror psicológico con tintes de Project Zero y PT. de @BLOODIOUSGAMES

#Madison Possessed Edition llega a PS4/PS5/SWITCH el 24 de junio con cont. adicional: DLC, libro de arte, targetas,etc...  
[Voir plus](#)




4
7
40
...

 **BLOODIOUS GAMES**  @BLOODIOUSGAMES · 16 nov. 2021 ...  
Go comment and let them know you wanna hear some more of @MADISON\_Game  

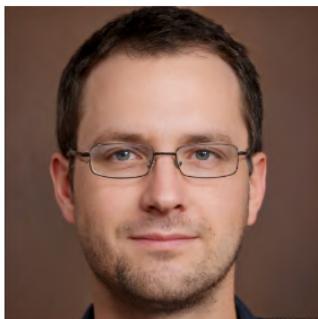
 **IGN**  @IGN · 16 nov. 2021  
Check out the first 12 minutes of creepy horror game, MADISON, exclusively with IGN. Solving puzzles and taking photos as you venture deeper into the nightmare. [bit.ly/3nknPVf](http://bit.ly/3nknPVf)

Page Twitter (X) du jeu Madison

## Proto-persona

Les différentes recherches utilisateurs effectuées on permis la réalisation d'un proto-persona. Ce document permet de garder à l'esprit qui est notre utilisateur final et pour qui le produit est conçu (démarche d'expérience utilisateur centrée sur l'utilisateur).

### "Johnny" | Ingénieur réseau



Curieux Méthodique Patient Introverti

#### Objectif

- Toujours en quête d'informations nouvelles
- Veut être le premier à disposer de ces informations
- Partage des informations et connaissances
- Perfectionniste

*"La connaissance est le pouvoir. L'information est libératrice."*

Age : **39 ans**

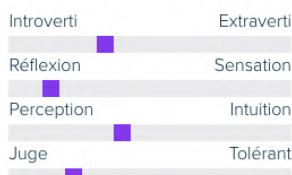
Emploi : **Ingénieur réseau**

Situation : **Marié**

Localisation: **Sarlat, France**

Caractère : **Curieux**

#### Personnalité



#### Frustration

- Manquer une information (importante ou non)
- Ne pas pouvoir diffuser cette information
- Peur du manque et peur de l'échec

#### Bio

"Johnny" (un pseudo pour les réseaux sociaux dont il est accro) est Ingénieur Système et Réseaux dans une grande entreprise informatique européenne. Ce métier, souvent cantonné devant un écran d'ordinateur, colle avec son tempérament introverti et calme. Sa passion pour l'informatique et les nouvelles technologies le pousse à rechercher toujours de l'information "fraîche". Il faut qu'il soit le premier au courant pour mieux rediffuser cette information. Volontaire et persévérant il est constamment à l'affût d'opportunité d'évolution et ce, dans tous les domaines de sa vie.

#### Motivation

Incitation



Crainte



## Conclusion

La recherche secondaire conduite en premier lieu lors de ce projet a permis de déterminer les sensibilités de la cible, notamment en matière de frustrations et d'intérêt, mais également en termes de communication et de marketing. Ainsi, une approche « indirecte » est préconisée. C'est-à-dire de manière narrative et immersive pour le futur utilisateur, plutôt que par du marketing traditionnel.

L'objectif serait de créer un engouement autour du produit, sans en révéler pour autant sa nature de jeu vidéo. En effet, le lecteur devrait être maintenu dans l'ignorance jusqu'au jour de la mise effective sur le marché.

Le concept de cette communication indirecte serait d'attiser la curiosité des acheteurs potentiels par l'activation du biais cognitif dit d'"heuristique catégorielle" : apporter des informations clés sur le produit de façon à conforter l'acheteur dans son choix, et de ce fait, créer les conditions propices à l'acte d'achat.

Cet engouement créé autour d'un « pseudo-mystère » devrait permettre de fédérer une communauté. C'est cette communauté "de niche" qui formera le gros des acheteurs du produit.

Ainsi, pour que cette stratégie fonctionne, le faisceau de "preuves" - qu'il s'agit bien d'un jeu vidéo - devrait converger vers le lecteur de façon qu'il parvienne lui-même à cette conclusion. Ce processus "d'auto-récompense" ("J'en étais sûr !") donne l'illusion au lecteur d'avoir fait partie du projet depuis ses prémisses. Ce sentiment facilite d'une part l'appropriation du produit et crée d'autre part un sentiment d'appartenance à la communauté. Ces sentiments peuvent alors être exploités dans une campagne de financement participatif sur KickStarter notamment. Les fonds ainsi récoltés (en échange de compensations cohérentes pour les backers) permettraient de financer la suite du développement du produit.

La recherche primaire ensuite, a permis de détermier les attendus des futurs acheteurs du produit et d'étudier les critères techniques les plus importants pour la cible autant concernant le produit final que concernant les différents supports de communication, notamment numériques. Ainsi, les utilisateurs cibles se sont avérés être très attentifs à l'accessibilité en général, aux possibilités de régler les interfaces ou options selon leurs besoins spécifiques. En outre, les résultats ont démontré un usage « desktop first » (écran d'ordinateur) en premier lieu, puis un relai sur mobile. Ce point devrait être pris en compte lors de la conception de tout produit numérique.

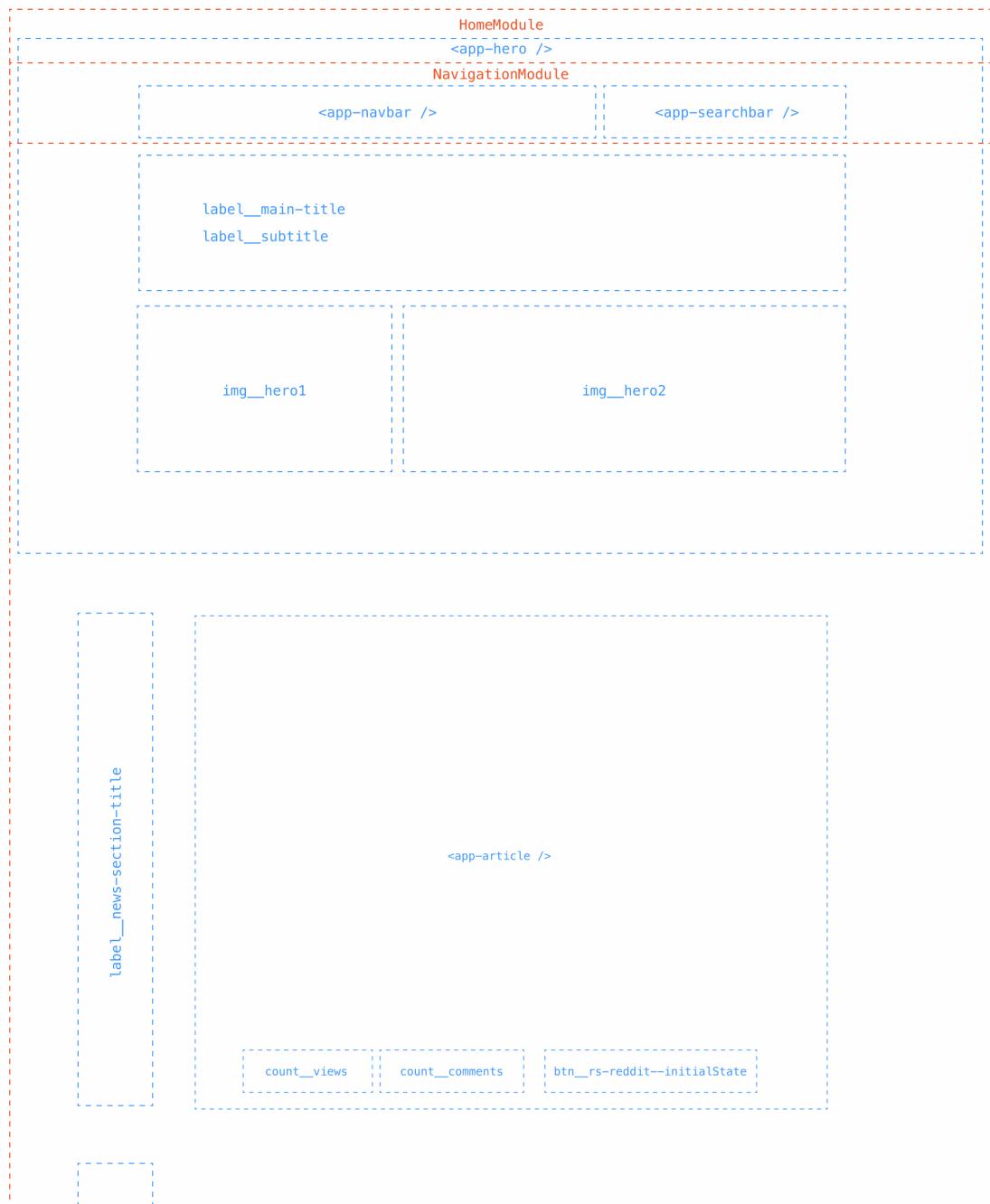
# ANNEXE N°8

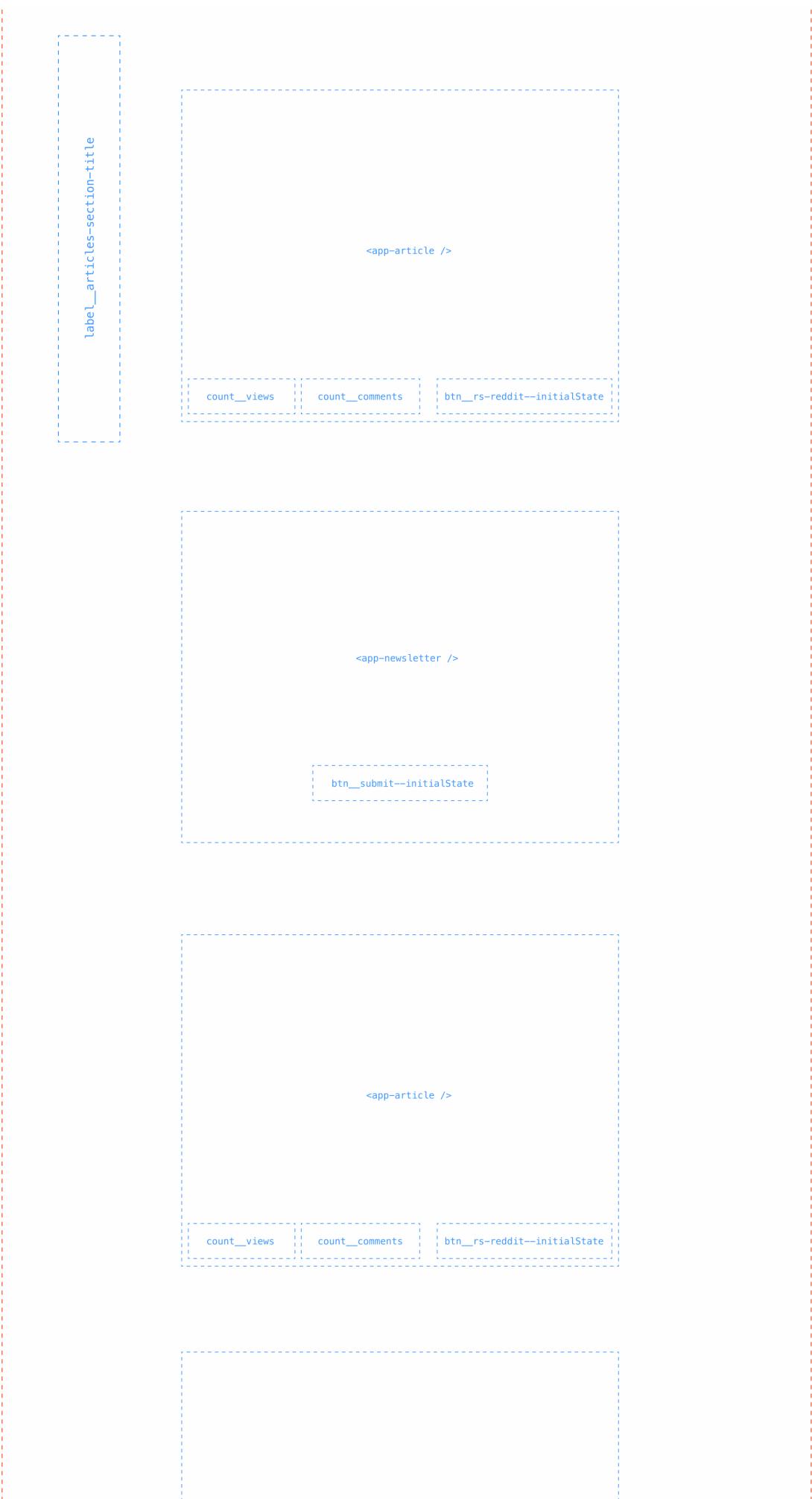
## Visuels de conception

Cette annexe regroupe les différents supports qui ont servi à la conception du site.

### Schéma de composants

Dans le framework Angular, représente l'agencement des différents composants de l'application.





```
<app-article />
```

count_views	count_comments	btn_rs--reddit--initialState
-------------	----------------	------------------------------

```
<app-article />
```

count_views	count_comments	btn_rs--reddit--initialState
-------------	----------------	------------------------------

```
<app-more-button />
```

```
<app-form />
```

```
btn_submit--initialState
```

FooterModule

```
<app-legal />
```

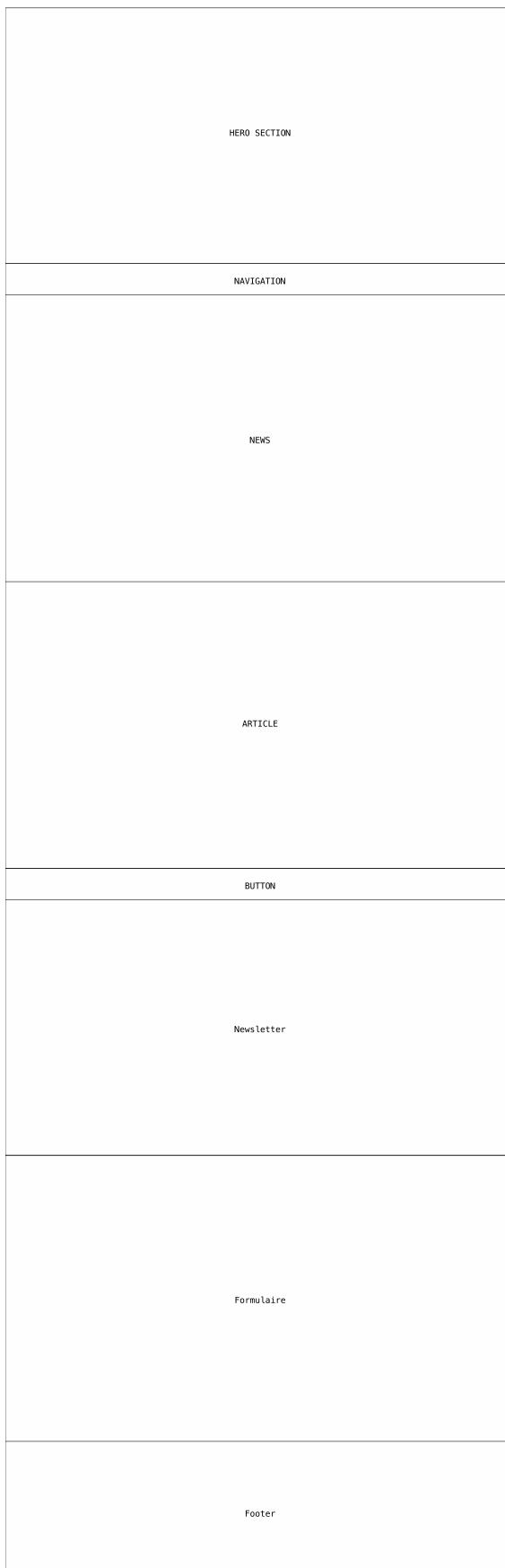
```
<app-cookies />
```

```
<app-pivacy />
```

```
<app-terms />
```

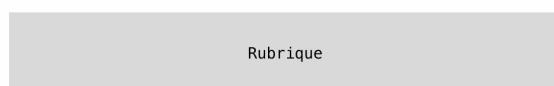
## Zoning

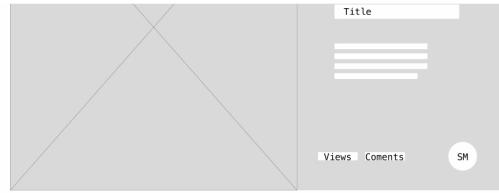
Représente les principales zones d'interactions de l'application.



## Wireframe

Représente les zones d'interactions dans une granularité plus fine.







button

A template for a form. It has a grey header with the word "Form" in white. Below the header is a white input field. The main body of the form is divided into several horizontal sections, each containing a white rectangular area. At the bottom of the form is a large grey section containing a black rectangular button with the word "button" in white.

Footer

Nouvel onglet Effectuez une recherche sur Google ou saisissez une URL

Home Articles About   

# Warning!

## I found that!



**NEW discovery!**



2023-10-19

### Article title

Lorem ipsum dolor sit amet consectetur. Sit facilisis aliquam nullam rutrum justo. Viverra velit.

1000  500  

**covery!**



2023-10-19

### Article title

Lorem ipsum dolor sit amet consectetur. Sit facilisis aliquam nullam rutrum justo. Viverra velit.

## Recent discovery!



2023-10-19

### Article title

Lorem ipsum dolor sit amet consectetur. Sit facilisis aliquam nullam rutrum justo. Viverra velit.

1000  500  



2023-10-19

### Article title

Lorem ipsum dolor sit amet consectetur. Sit facilisis aliquam nullam rutrum justo. Viverra velit.

1000  500  



2023-10-19

### Article title

Lorem ipsum dolor sit amet consectetur. Sit facilisis aliquam nullam rutrum justo. Viverra velit.

1000  500  

### NewsLetter

## Stay alert!

Enter your e-mail address here

submit 



2023-10-19



[more posts](#)

## A question?

First Name

Email \*

Message...

[submit](#)



**NEW** discovery!

**Article title** 2023-10-19

**Subtitle**

**Text**

**Image**

## A question?

First Name \_\_\_\_\_ Last Name \_\_\_\_\_

Email: \_\_\_\_\_

Message... \_\_\_\_\_

**submit**



## Prototype



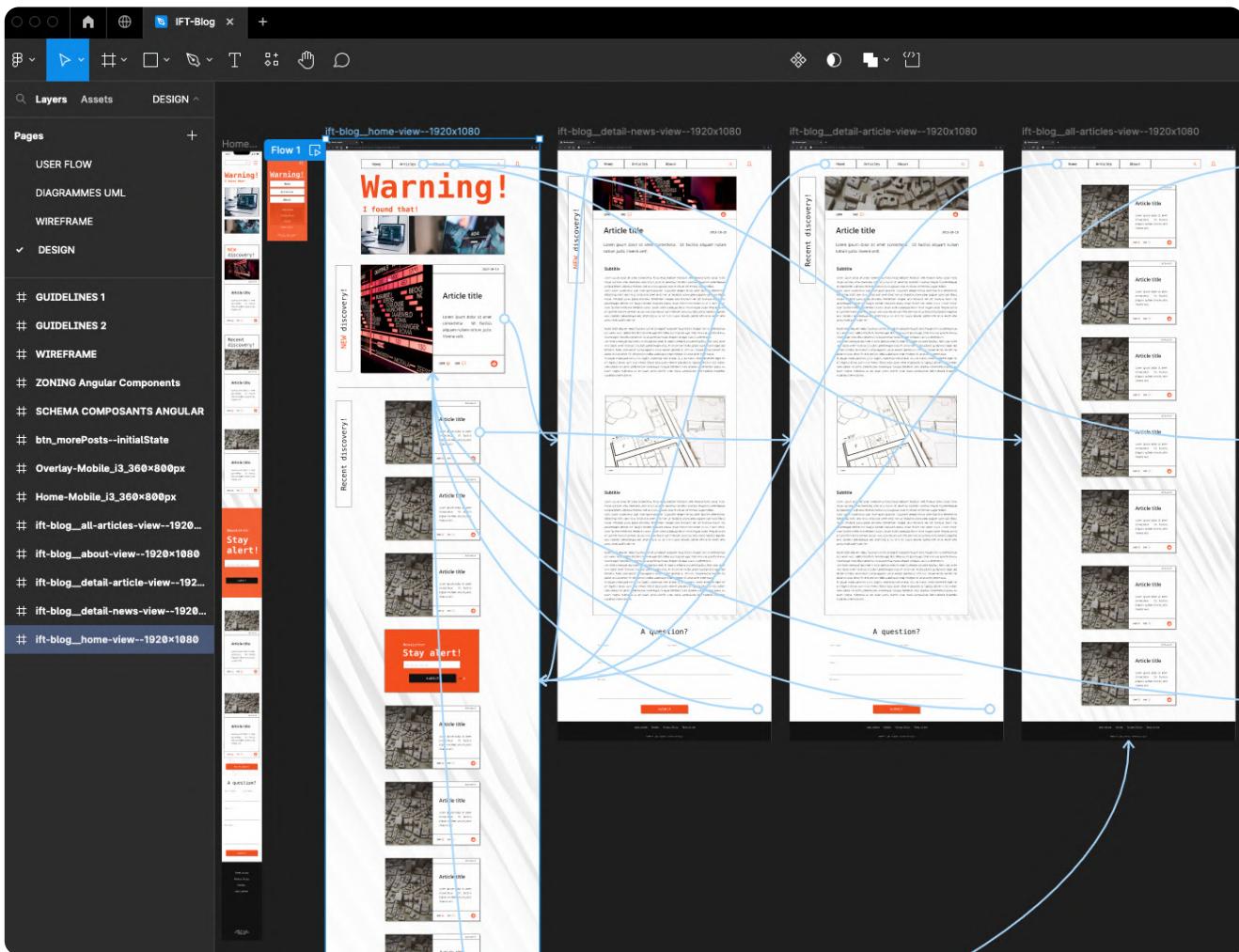
Page home



Exemple d'article

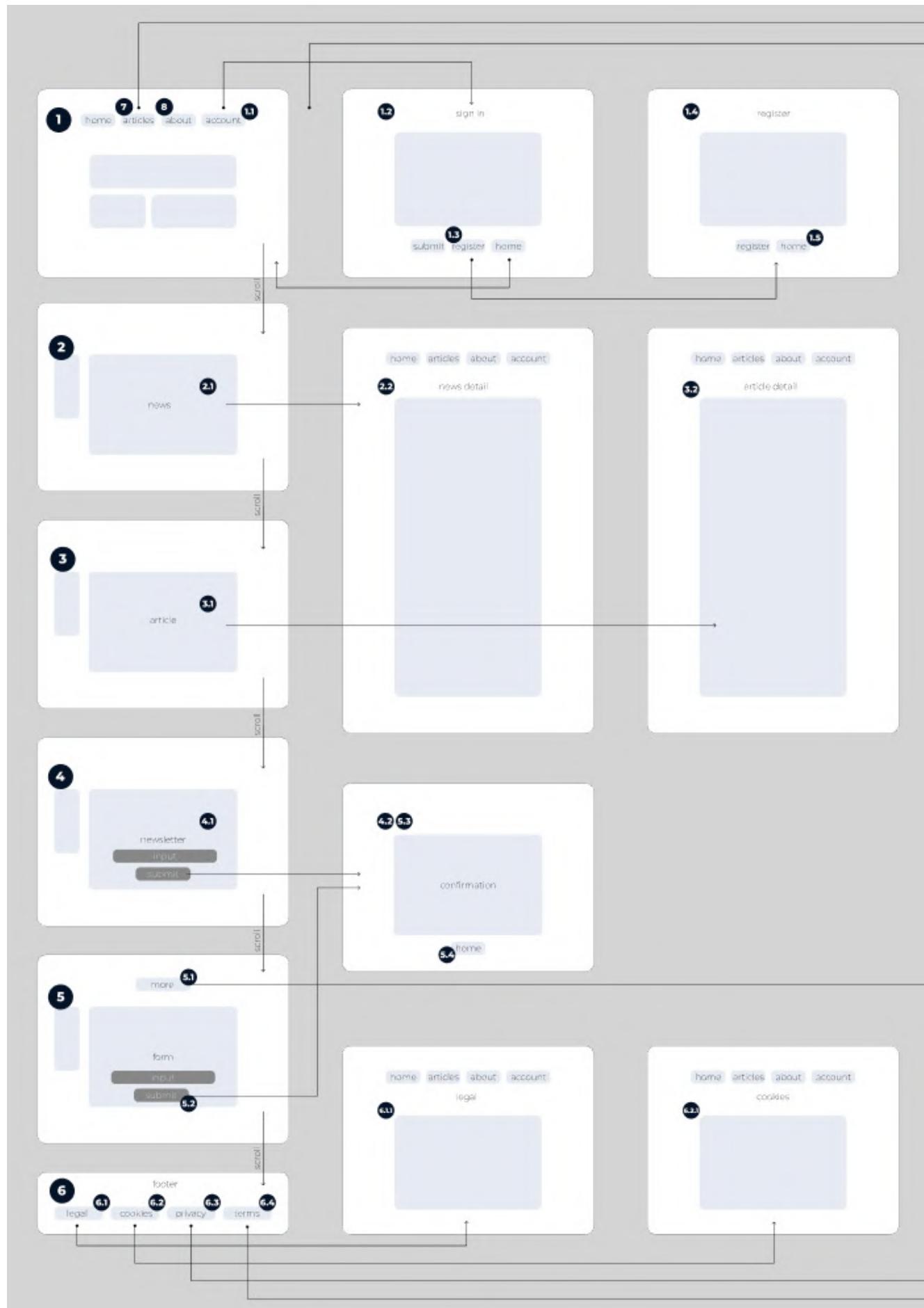


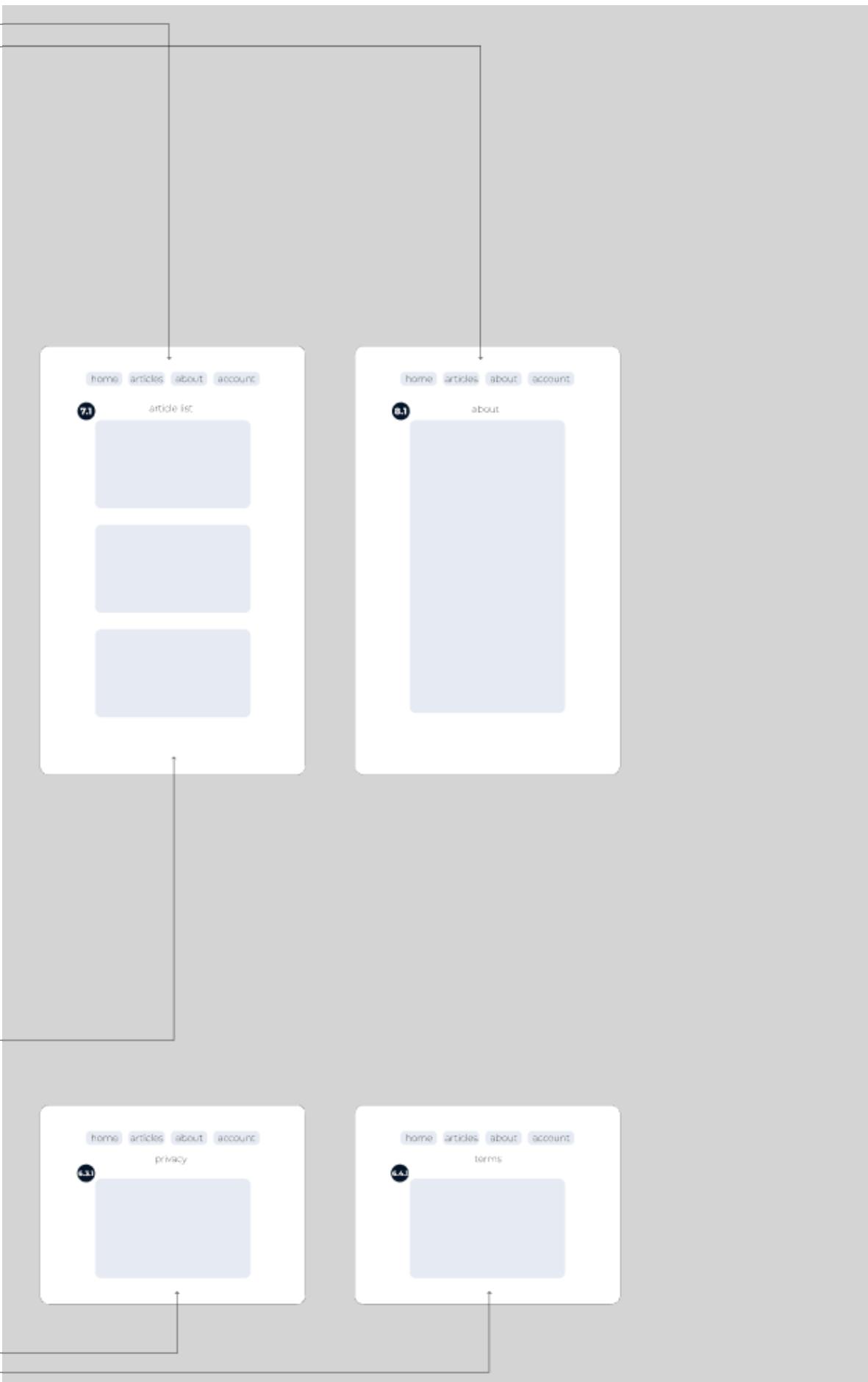
Menu overlay mobile



Mise en place des interactions

## User Flow





# ANNEXE N°9

## Dictionnaire des données et matrice des dépendances

### Dictionnaire des données

Aa Data	≡ Type	≡ Length	≡ DB code	≡ Exemple
<u>User id</u>	INT	/	iduser	1
<u>User email</u>	VARCHAR	70	email	mathieu@int.dev
<u>User password</u>	VARCHAR	70	psw	*****
<u>User role</u>	JSON	/	role	User   Admin   SuperAdmin
<u>User creation</u>	DATETIMEIMMUTABLE	/	createdAt	20/02/2023   12:00
<u>User update</u>	DATETIMEIMMUTABLE	/	updatedAt	20/02/2023   12:01
<u>IsAdminSuperAdmin</u>	BOOLEAN	/	isSuperAdmin	Yes   No
<u>Log id</u>	INT	/	idlog	2
<u>Log creation</u>	DATETIMEIMMUTABLE	/	createdAt	20/02/2023   12:00
ip	VARCHAR	255	ip	192.168.0.2
<u>Article id</u>	INT	/	idarticle	3
<u>Article creation</u>	DATETIMEIMMUTABLE	/	createdAt	20/02/2023   12:00
<u>Article update</u>	DATETIMEIMMUTABLE	/	updatedAt	20/02/2023   12:01
<u>Article title</u>	VARCHAR	255	title	Pourquoi les développeurs ont-ils des canards en plastique ?
<u>Article Chapo</u>	MEDIUMTEXT	/	chapo	Cil mo maio. Nequatu rendant moluptam re escia voluptati ommolent, ex eaquam dolorpori a sum faccum, etus dolo berchil molest facest valoribus.
<u>Newsletter id</u>	INT	/	idnewsletter	6
<u>Newsletter creation</u>	DATETIMEIMMUTABLE	/	createdAt	20/02/2023   12:00
<u>Newsletter email</u>	VARCHAR	70	email	simonin@int.dev

## Matrice de dépendances fonctionnelles

Aa DB code	UserService	UserConnexionService	ArticleService	TagService	CategoryService	NewsletterService	CommentService
<u>iduser</u>	*						
<u>email</u>	1						
<u>psw</u>	1						
<u>role</u>	1						
<u>createdAt</u>	1						
<u>updatedAt</u>	1						
<u>isSuperAdmin</u>	1						
<u>idlog</u>		*					
<u>createdAt</u>		1					
<u>ip</u>		1					
<u>idarticle</u>			*				
<u>createdAt</u>			1				
<u>updatedAt</u>			1				
<u>title</u>			1				
<u>chapo</u>			1				
<u>content</u>			1				
<u>idtag</u>				*			
<u>keyword</u>				1			
<u>idcategorie</u>					*		
<u>name</u>					1		
<u>idnewsletter</u>						*	
<u>createdAt</u>						1	
<u>email</u>						1	
<u>idcomment</u>							*

# ANNEXE N°10

## Tables et schéma de base de données

### Tables en relations

**user** : un utilisateur pouvant disposer de plusieurs articles, commentaires et connexions, la Table “user” est en relation One to Many (1:m) avec les Tables “user\_connexion”, “article” et “comment”

**user\_connexion** : est en relation Many to One (m:1) avec la Table “user”, car, comme vu précédemment, un utilisateur peut se connecter à plusieurs reprises sur le site.

**article** : est en relation One to One (1:1) avec la Table “category”, car dans ce blog, un article ne peut appartenir qu'à une seule catégorie.

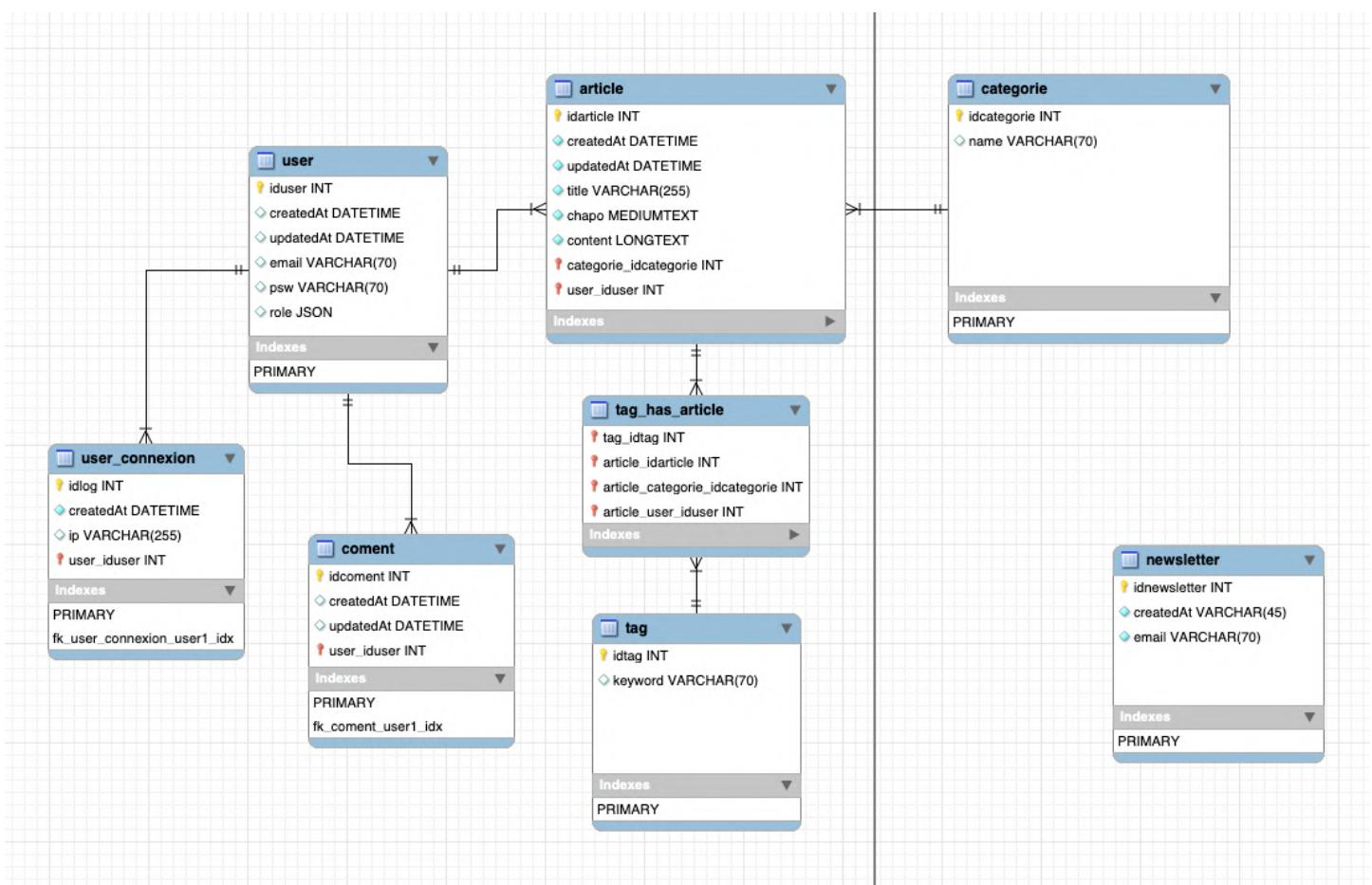
**category** : est en relation One to Many (1:m), car une même catégorie peut avoir plusieurs articles.

**comment** : est en relation Many to One à la fois avec les Tables “user” et “article”. En effet, plusieurs commentaires peuvent être rattachés à un seul article et un utilisateur peut proposer plusieurs commentaires.

**tag** : est en relation One To Many (1:m) avec la Table “article”, car un Tag peut s'appliquer à plusieurs articles.

### Table indépendante

**newsletter** : cette Table ne dispose d'aucune relation. Elle stocke simplement une adresse email.



# ANNEXE N°11

## Convention de nommage

De façon générale pour le projet, l'utilisation des conventions BEM, CamelCase et PascalCase sont largement représentées. Le détail pour chaque framework est présenté ci-dessous, dans l'ordre de la conception :

## Convention de Nommage Symfony

Cette Convention de Nommage suit la Convention en vigueur sur le Framework Symfony et à son écosystème. Celle-ci utilise également la Convention du langage PHP, langage sur lequel est basé ce Framework.

### Les fichiers et dossiers

#### Fichiers des Entities, Fixtures et Repositories :

Les noms des fichiers seront en `PascalCase`, reflétant le nom de la classe correspondante.

Exemple : `ApiController.php`, `ProductDetails.php`, `NewsletterRepository.php`

#### Noms de dossiers :

Conservez la structure des dossiers selon la fonctionnalité ou le domaine, en utilisant également `PascalCase` pour refléter le nom de la classe principale.

Exemple : `/UserManagement`, `/ProductCatalog`, `/SharedComponents`

## Fichiers des Tests Unitaires :

Les fichiers de tests unitaires pour un fichier `FooComponent.php` devraient être nommés `FooComponentTest.php` pour indiquer qu'il s'agit des tests unitaires associés à ce fichier.

Exemple : `UserListComponentTest.php`, `ProductDetailsComponentTest.php`

## Les Classes

### Classes de Controllers et Services :

Les noms de Classes doivent être écrits en `PascalCase`, démarrant par une majuscule, sans préfixe.

Exemple : `UserService`, `LoggingController`

## Les noms de Variables et Méthodes

### Noms des Méthodes, Variables :

Les noms de méthodes et variables seront en `camelCase`, reflétant leur objectif.

Exemple : `userName`, `productList`, `isLoggedIn`

### Noms des Constantes :

Les constantes devront être rédigées en `UPPER_CASE` avec des mots séparés par des underscores pour une meilleure lisibilité, comme dans la convention Angular.

Exemple : `API_URL`, `MAX_RETRIES`, `DEFAULT_TIMEOUT`

---

# Convention de Nommage ANGULAR

La présente Convention de Nommage s'applique au Framework Angular et à son écosystème. Elle suit également les Conventions propres aux langages JavaScript et TypeScript, Angular étant un Framework basé sur ces langages de programmation.

## Les fichiers et dossiers

### Fichiers de Composants, Modules et Services :

Les noms des fichiers doivent être en `kebab-case` pour la lisibilité et la cohérence, en utilisant des tirets pour séparer les mots.

Exemple : `user-list.component.ts`, `product-details.component.ts`, `shared.module.ts`

### Noms de dossiers :

Les dossiers devraient refléter la structure des fonctionnalités ou des domaines de l'application, et ils doivent également suivre la convention `kebab-case`.

Exemple : `/user-management`, `/product-catalog`, `/shared-components`

### Fichiers des Tests Unitaires :

Les fichiers de tests unitaires pour un fichier `foo.component.ts` devraient être nommés `foo.component.spec.ts` pour indiquer qu'il s'agit des tests unitaires associés à ce fichier.

Exemple : `user-list.component.spec.ts`, `product-details.component.spec.ts`

## Les Classes

### Classes de Composants :

Les noms des Classes des Composants doivent être écrits en `PascalCase`, démarrant par une majuscule, sans utiliser de préfixe. Les noms doivent être des noms significatifs et descriptifs.

Exemple : `AppComponent`, `UserListComponent`, `ProductDetailsComponent`

## **Classes de Directives :**

Les Directives personnalisées doivent également être en `PascalCase` et suivre les mêmes règles de nommage que les composants.

Exemple : `CustomDirective` , `HighlightDirective` , `ResizeDirective`

## **Classes de Services :**

Les Services doivent être nommés de manière descriptive et utiliser le suffixe `Service` en `PascalCase`. Les noms doivent refléter leur rôle ou leur fonctionnalité.

Exemple : `UserService` , `LoggerService` , `DataService`

## **Classes de Modules :**

Les noms de Modules doivent être en `PascalCase` et utiliser un suffixe représentant le domaine de l'application ou du feature set.

Exemple : `AdminModule` , `UserModule` , `SharedModule`

# **Les noms de Variables et Méthodes**

## **Noms des Méthodes :**

Les noms de Méthodes doivent être rédigés en `camelCase` et représenter clairement leur objectif.

Exemple : `userName` , `productList` , `isLoggedIn`

## **Noms des Variables :**

Les noms de Variables doivent être écrites en `camelCase` également et représenter clairement leur objectif.

Exemple : `userName` , `productList` , `isLoggedIn`

## **Noms des Constantes :**

Les Constantes devront être rédigées en `UPPER_CASE` avec des mots séparés par des underscores pour une meilleure lisibilité.

Exemple : `API_URL` , `MAX_RETRIES` , `DEFAULT_TIMEOUT`

---

# Convention de Nommage Styles

La présente Convention de Nommage utilise le standard BEM (Bloc, Element, Modifier). Elle est destinée à être appliquée avec la partie Style du projet (SCSS/CSS) :

## Les Classes

### Blocs (Blocks)

- Les noms des blocs représentent les composants principaux ou les éléments de haut niveau.
- Utilisez un préfixe ou un nom significatif pour identifier le bloc.
- Écrivez les noms des blocs en **kebab-case** pour une meilleure lisibilité.

Exemple :

```
.button {}  
.button-submit {}
```

### Éléments (Elements)

- Les éléments sont des parties spécifiques à l'intérieur des blocs.
- Utilisez un double underscore pour séparer les éléments du bloc.

- Les éléments doivent être écrits en **kebab-case**.

Exemple :

```
.nav__button {}  
.nav__button-item1 {}
```

## Modificateurs (Modifiers)

- Les modificateurs permettent de définir des variations ou des états des blocs ou des éléments.
- Utilisez un double tiret pour séparer le nom du modificateur.
- Les modificateurs devraient être écrits en **kebab-case**.
- Les modificateurs peuvent être appliqués aux blocs ou aux éléments.

Exemple :

```
.img--hover {}  
.img__carousel--onclick {}}
```

## Fonctions en SCSS

### Fonctions générales

- Utilisez des noms significatifs et descriptifs pour les fonctions.
- Évitez les noms trop génériques ; privilégiez des noms descriptifs pour indiquer leur rôle ou leur utilisation.

Exemple :

```
@function calculate-grid-width($columns, $gutter) {  
  
  @return $columns * $column-width + ($columns - 1) * $gutter;  
}  
  
@function format-currency($amount) {  
  @return formatted-amount;  
}
```

## Fonctions de mixins

- Les mixins sont des fonctions qui génèrent du code CSS.
- Utilisez des noms descriptifs pour les mixins, décrivant leur comportement ou leur utilisation.

Exemple :

```
@ mixin flex-container($direction, $justify, $align) {  
  display: flex;  
  flex-direction: $direction;  
  justify-content: $justify;  
  align-items: $align;  
}  
  
@ mixin hover-effect($property, $value) {  
  &:hover {  
    #{$property}: $value;  
  }  
}
```



# ANNEXE N°12

## Présentation du jeu d'essais et des tests

### Présentation du jeu d'essais

La réalisation de tests fonctionnels s'est faite sur la base d'un tableau de jeu d'essais.

Ce support regroupe les données en entrées, les étapes suivies et les données attendues.

Ces données sont issues de différents cas

d'utilisation. De ces cas, des données de code ont été proposées en entrée et en sortie pour réaliser ces tests.

Le cas d'utilisation présenté ici fait référence au CRUD de l'entity Article avec API Platform®.

Aa Case Type	☰ Description	☰ Test Step	☰ Expected Result	>Status	☰ Comment
<u>Affichage des articles</u>	Vérifier l'affichage d'une liste d'article. Données en entrée : 'GET', '/api/articles'	Un utilisateur non connecté accède à la home page.	Une liste des articles de la base de données est visible. Données en sortie : confirmation de la méthode assertResponseIsSuccessful() de la classe ApiTestCase (KernelTestCase)	Pass	CRUD article
<u>Enregistrement email</u>	Vérifier l'enregistrement d'une adresse email valide depuis le formulaire newsletter. Données en entrée : 'aaa@aaa.aaa'	Un utilisateur non connecté accède à la home page. Il saisit son adresse email dans le formulaire d'inscription à la newsletter.	L'email provenant du formulaire newsletter est enregistrée. Données en sortie : confirmation de la méthode assertResponseIsSuccessful() de la classe ApiTestCase (KernelTestCase)	Pass	Persistante email dans table newsletter
<u>Enregistrement email</u>	Vérifier l'enregistrement d'une adresse email invalide depuis le formulaire newsletter. Données en entrée : 'aaa'	Un utilisateur non connecté accède à la home page. Il saisit son adresse email dans le formulaire d'inscription à la newsletter.	L'email provenant du formulaire newsletter n'est pas enregistrée. Données en sortie : erreur de la méthode assertCount(0, \$errors); de la classe ApiTestCase (KernelTestCase)	Pass	Erreur provoquée
<u>Création d'article</u>	Vérifier la création d'un nouvel article. Données en entrée : 'POST',	L'admin accède à la section de gestion des articles et crée un nouvel article avec le	L'article est créé avec succès et est visible sur la page home et dans la liste des articles. Données en sortie : confirmation de la méthode assertResponseIsSuccessful()	Pass	CRUD article

Aa Case Type	≡ Description	≡ Test Step	≡ Expected Result	Status	≡ Comment
			méthode assertResponseIsSuccessful() de la classe ApiTestCase (KernelTestCase)		
<u>Suppression d'article</u>	Vérifier la suppression d'un article existant. Données en entrée : 'DELETE', '/api/articles/1'	L'admin accède à la section de gestion des articles et supprime un article existant.	L'article est supprimé avec succès et n'est plus visible sur la page home ni dans la liste des articles. Données en sortie : confirmation de la méthode assertResponseIsSuccessful() de la classe ApiTestCase (KernelTestCase)	Pass	CRUD article
<u>Recherche d'article</u>	Vérifier la fonction de recherche d'articles. Données en entrée : 'Greeting!'.	L'utilisateur saisie un mot clé dans la fonction de recherche sur la page d'accueil.	Les articles contenant le mot clé correspondant sont affichés dans la home page. Données en sortie : confirmation de la méthode assertResponseIsSuccessful() de la classe ApiTestCase (KernelTestCase)	Fail	Bug à déterminer
<u>Routing_front</u>	Vérifier que tous les liens sont fonctionnels. Données en entrées : ng run ift-blog:routes	L'utilisateur clic à plusieurs endroits de l'interface.	Le lien doit renvoyer l'utilisateur sur la page demandée avec une URL conforme. Données en sortie : confirmation de la méthode assertResponseIsSuccessful() de la classe ApiTestCase (KernelTestCase)	Pass	+ test manuel via Postman
<u>Routing_back</u>	Vérifier que tous les liens sont fonctionnels. Données en entrées : utiliser php bin/console debug:router	L'utilisateur clic à plusieurs endroits de l'interface.	Le lien doit renvoyer l'utilisateur sur la page demandée avec une URL conforme. Données en sortie : confirmation de la méthode assertResponseIsSuccessful() de la classe ApiTestCase (KernelTestCase)	Pass	+ test manuel via Postman
<u>Sécurité</u>	Vérifie que le token est bien présent dans localStorage.	L'admin se connecte au site (login).	La console indique la présence du token de session.	Pass	
<u>Securité</u>	Vérifier que les règles de mot de passe fonctionnent.	L'utilisateur (ou l'admin) crée un nouveau mot de passe.	Si le mot de passe correspond aux règles établies, il est accepté.	Pass	
<u>Sécurité</u>	Vérifier la sécurité de l'application.	Tenter d'accéder à des ressources	L'accès non autorisé est bloqué, et les erreurs de sécurité sont correctement	To do!	Effectuer un test d'intrusion via BurpSuite.

Aa Case Type	≡ Description	≡ Test Step	≡ Expected Result	▽ Status	≡ Comment
		sans être connecté et tenter d'accéder à des fonctionnalités d'administration sans les droits appropriés.	gérées (par exemple, une page de connexion s'affiche pour un utilisateur non connecté).		
<u>Vérification input min</u>	Vérifier que les input contiennent un nombre de caractères minimum (2).	L'utilisateur saisi moins de 2 caractères.	Un message d'erreur s'affiche indiquant que le nombre de caractères est insuffisant.	Pass	Assert côté back et regex côté front
<u>Vérification input max</u>	Vérifier que les input contiennent un nombre limité de caractères (70).	L'utilisateur saisi 71 caractères.	Un message d'erreur s'affiche indiquant que le nombre de caractères est dépassé.	Pass	Assert côté back et regex côté front
<u>UI</u>	Vérifier la navigation et l'expérience utilisateur.	L'utilisateur navigue sur l'application.	S'assurer que la navigation et que l'interface utilisateur est agréable et intuitive.	To do!	Via test utilisateur et End to End
<u>Ajout de Commentaire</u>	Vérifier l'ajout de commentaires sur un article.	L'utilisateur membre accède à un article spécifique et ajoute un commentaire.	Le commentaire est ajouté avec succès et apparaît immédiatement sous l'article.	To do!	NON IMPLEMENTE
<u>Gestion des Utilisateurs</u>	Vérifier la gestion des utilisateurs (administrateurs, éditeurs, et lecteurs).	L'admin accède à la section de gestion des utilisateurs et ajoute, modifie ou supprime un utilisateur.	Les opérations sur les utilisateurs sont effectuées avec succès, et les rôles sont correctement appliqués (par exemple, un administrateur a des droits étendus par rapport à un éditeur ou un lecteur).	Pass	Via Voters et Constraints back-end
<u>Gestion des Catégories</u>	Vérifier la création, modification et suppression de catégories.	L'admin accède à la section de gestion des catégories et ajoute, modifie, ou supprime une catégorie.	Les opérations sur les catégories sont effectuées avec succès, et les articles associés sont correctement classés dans les catégories.	To do!	NON IMPLEMENTE
<u>Partage sur les Réseaux Sociaux</u>	Vérifier la fonctionnalité de partage	L'utilisateur membre accède à un	L'article est partagé correctement sur les réseaux	Pass	Simple lien vers Reddit

Aa Case Type	≡ Description	≡ Test Step	≡ Expected Result	Status	≡ Comment
	d'articles sur les réseaux sociaux.	article spécifique et utilise les boutons de partage pour partager l'article sur reddit.	sociaux avec le lien et le contenu appropriés.		
<u>Gestion des erreurs</u>	Vérifier la gestion des erreurs lors de la soumission de formulaires.	L'admin soumet un formulaire de création d'article avec des données incorrectes.	Des messages d'erreur s'affiche.	Pass	
<u>Gestion des erreurs</u>	Vérifier la gestion des erreurs lors de la soumission de formulaires.	L'admin soumet un formulaire de création d'article avec toutes les données attendues.	Un message de validation s'affiche.	Pass	
<u>Compatibilité navigateur</u>	Vérifier la compatibilité avec différents navigateurs.	L'utilisateur ouvre l'application sur Chrome, Firefox, et Safari.	L'application fonctionne correctement sur tous les navigateurs cibles sans erreurs d'affichage ou de fonctionnement.	Pass	Vérifier la mise en page et les fonctionnalités.
<u>Gestion des sessions</u>	Vérifier la gestion des sessions utilisateur.	L'utilisateur membre se connecte à l'application, puis ferme le navigateur avant de le rouvrir. Enfin, il accède à nouveau à l'application.	L'utilisateur est automatiquement connecté après la réouverture du navigateur, montrant une gestion appropriée des sessions.	Pass	Via token JWT
<u>Performance des Requêtes</u>	Vérifier les performances lors de la récupération d'articles.	L'utilisateur accède à la page home, puis passe à la page articles.	La page se charge rapidement, même avec un grand nombre d'articles, assurant une bonne expérience utilisateur.	To do!	Mesurer le temps nécessaire pour charger une page d'articles.
<u>Performance des Requêtes</u>	Vérifier les performances lors de la	L'utilisateur accède à la page home,	La page se charge rapidement, même avec un grand nombre d'articles,	To do!	Mesurer le temps nécessaire

## Tests unitaires

Le protocole suivi pour la réalisation des tests unitaires était le suivant :

1 - Installation de l'environnement de test :

- Crédation du fichier **.env.test.local** Ce fichier contient les variables d'environnement locales propres aux tests et écrase le contenu du fichier **.env.test**, (uniquement nécessaire en production).

- Installation du chemin de la base de données initiale à copier-coller dans **.env.test.local**.

2 - Crédation d'un base de données de test pour PHPUnit avec la commande **symfony console d:d:c --env=test**

3 - Remplissage de la base de données de test avec la commande **symfony console d:m:m --env=test** puis **symfony console d:f:l --env=test**.

4 - Générer un test avec la commande **symfony console make:test**. Choisir le type de test désiré. Ici, **KernelTestCase**, puis choisir l'entity sur laquelle le test sera basé (ex : Newsletter.php). Nommez le fichier de test selon la convention suivante : « **NewsletterTest** ».

5 - Ecrire le test selon les besoins. Le test ci-dessous vise à évaluer la validité d'une adresse email entrée depuis l'input du composant Newsletter côté client.

6 - Lancement du test avec la commande **php bin/phpunit** puis analyse des résultats.

The screenshot shows the phpMyAdmin interface with the database 'ifl-db.test' selected. The left sidebar lists the schema with tables like 'article', 'categorie', 'comment', etc. The right panel shows a grid of tables with actions like 'Parcourir', 'Structure', and 'Insérer' for each.

Création de la base de données de test

```
// Test for invalid_email validation in the Newsletter entity
public function testInvalidEmail(): void
{
    // Bootstrap the Symfony kernel to set up the testing environment
    self::bootKernel();
    // Get the container from the kernel for accessing services
    $container = static::getContainer();
    // Create a Newsletter entity with a valid email
    $newsletter = $this->getEntity();
    // Set an invalid email for testing purposes
    $newsletter->setEmail('aaa');
    // Use the validator service to validate the Newsletter entity
    $errors = $container->get('validator')->validate($newsletter);
    // Assert that there is exactly one validation error
    $this->assertCount(1, $errors);
}

// Test for valid_email validation in the Newsletter entity
public function testValidEmail(): void
{
    self::bootKernel();
    $container = static::getContainer();
    $newsletter = $this->getEntity();
    $newsletter->setEmail('valid@example.com');
    $errors = $container->get('validator')->validate($newsletter);
    $this->assertCount(0, $errors);
}
```

Assertions testées

```
mathieu@MacBook-Pro-de-Mathieu ifoundthat-blog-back % php bin/PHPUnit 9.6.16 by Sebastian Bergmann and contributors.
Testing
..
Time: 00:00.218, Memory: 26.00 MB
OK (2 tests, 2 assertions)
mathieu@MacBook-Pro-de-Mathieu ifoundthat-blog-back %
```

Résultats des tests unitaires

○ ○ ○

```
<?php

namespace App\Tests\Unit\Entity;

use App\Entity\Newsletter;
use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;

class NewsletterTest extends KernelTestCase
{
    // Create and return a Newsletter entity with a predefined email
    public function getEntity(): Newsletter
    {
        // Create a new instance of the Newsletter entity and set a predefined
        return (new Newsletter())
            ->setEmail('aaa@aaa.aaa');
    }

    // Test for invalid email validation in the Newsletter entity
    public function testInvalidEmail(): void
    {
        // Bootstrap the Symfony kernel to set up the testing environment
        self::bootKernel();
        // Get the container from the kernel for accessing services
        $container = static::getContainer();
        // Create a Newsletter entity with a valid email
        $newsletter = $this->getEntity();
        // Set an invalid email for testing purposes
        $newsletter->setEmail('aaa');
        // Use the validator service to validate the Newsletter entity
        $errors = $container->get('validator')->validate($newsletter);
        // Assert that there is exactly one validation error
        $this->assertCount(1, $errors);
    }

    // Test for a valid email validation in the Newsletter entity
    public function testValidEmail(): void
    {
        self::bootKernel();
        $container = static::getContainer();
        $newsletter = $this->getEntity();
        $newsletter->setEmail('site@site.com');
        $errors = $container->get('validator')->validate($newsletter);
        $this->assertCount(1, $errors);
    }
}
```

Test unitaire réception ou échec email

## Tests fonctionnels

Le protocole suivi pour la réalisation des tests fonctionnel était le suivant :

- I - Installation de l'environnement de test :
  - Installation de la bibliothèque Dama® (manipulation de la base de données de test) avec la commande **composer require --dev dama/doctrine-test-bundle**.
  - Installation de la bibliothèque JsonSchema® (validation des schémas de données JSON) avec la commande **composer require --dev justinrainbow/json-schema**.

- Installation de la bibliothèque Foundry® (remplissage de base de données avec faker®) avec la commande **composer require zenstruck/foundry --dev**.

2 - Modification du fichier doctrine.yaml :

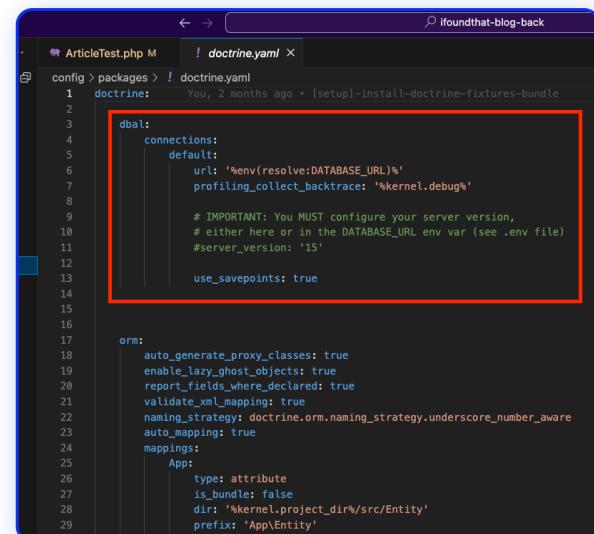
- Configuration DBAL (Database Abstraction Layer permet d'écrire dans tous types de langage de base de données) de Doctrine.

3 - Remplissage de la base de données de test avec la commande **symfony console make:factory --test**. Puis choisir l'entité visée.

4 - Générer un test avec la commande **symfony console make:test**. Choisir le type de test désiré. Ici, **KernelTestCase**, puis choisir l'entity sur laquelle le test sera basé (ex : Article.php). Nommez le fichier de test selon la convention suivante : « ArticleTest ».

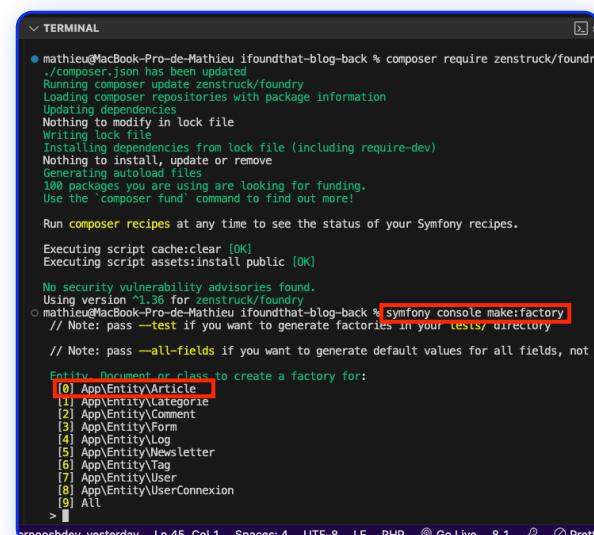
5 - Ecrire le test selon les besoins. Le test ci-contre vise à évaluer la validité des opérations CRUD d'un article du blog.

6 - Lancement du test avec la commande **php bin/phpunit** puis analyse des résultats.



```
config/packages/doctrine.yaml
1 doctrine: You, 2 months ago + [setup]-install-doctrine-fixtures-bundle
2
3     dbal:
4         connections:
5             default:
6                 url: '%env(resolve:DATABASE_URL)%'
7                 profiling_collect_backtrace: '%kernel.debug%'
8
9             # IMPORTANT: You MUST configure your server version,
10             # either here or in the DATABASE_URL env var (see .env file)
11             #server_version: '15'
12
13             use_savepoints: true
14
15
16 orm:
17     auto_generate_proxy_classes: true
18     enable_lazy_ghost_objects: true
19     report_fields_where_declared: true
20     validate_xml_mapping: true
21     naming_strategy: doctrine.orm.naming_strategy.underscore_number_aware
22     auto_mapping: true
23     mappings:
24         App:
25             type: attribute
26             is_bundle: false
27             dir: 'kernel.project_dir%/src/Entity'
28             prefix: 'App\Entity'
29             alias: App
```

Configuration de DBAL



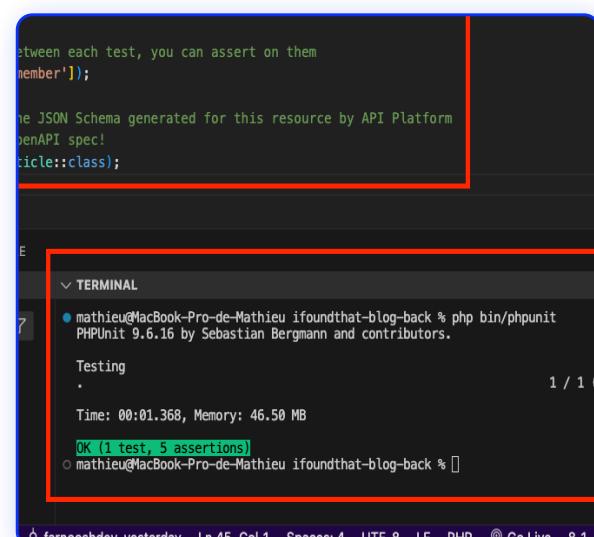
```
mathieu@MacBook-Pro-de-Mathieu ifoundthat-blog-back % composer require zenstruck/foundry
./composer.json has been updated
Running composer update zenstruck/foundry
Loading composer repositories with package information
Updating dependencies
Nothing to install, update or remove
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
100 packages are using are looking for funding.
Use the 'composer fund' command to find out more!

Run composer recipes at any time to see the status of your Symfony recipes.

Executing script cacher:clear [OK]
Executing script assets:install public [OK]

No security vulnerability advisories found.
Using version 1.36 for zenstruck/foundry
mathieu@MacBook-Pro-de-Mathieu ifoundthat-blog-back % symfony console make:factory
// Note: pass --test if you want to generate factories in your tests directory
// Note: pass --all-fields if you want to generate default values for all fields, not
Public __construct...to create a factory for:
[0] App\Entity\Article
[1] App\Entity\Categorie
[2] App\Entity\Comment
[3] App\Entity\Form
[4] App\Entity\Log
[5] App\Entity\Newsletter
[6] App\Entity\Tag
[7] App\Entity\User
[8] App\Entity\UserConnexion
[9] All
> 0
```

Remplissage de la base de données



```
mathieu@MacBook-Pro-de-Mathieu ifoundthat-blog-back % php bin/phpunit
PHPUnit 9.6.16 by Sebastian Bergmann and contributors.

Testing
.
Time: 00:01.368, Memory: 46.50 MB

OK (1 test, 5 assertions)
mathieu@MacBook-Pro-de-Mathieu ifoundthat-blog-back % []
```

Résultats

○ ○ ○

```
<?php

namespace App\Tests\Functional;

use ApiPlatform\Symfony\Bundle\Test\ApiTestCase;
use App\Entity\Article;
use App\Tests\Fixtures\Factories\ArticleFactory;
use App\Tests\Fixtures\Factories\UserFactory;
use Zenstruck\Foundry\Test\Factories;
use Zenstruck\Foundry\Test\ResetDatabase;
use function Zenstruck\Foundry\lazy;

class ArticleTest extends ApiTestCase
{
    // This trait provided by Foundry will take care of refreshing the database content to
    // state before each test
    use ResetDatabase, Factories;

    public function testGetCollection(): void
    {
        // Create 100 articles using our factory
        ArticleFactory::createMany(10);

        // The client implements Symfony HttpClient's `HttpClientInterface`, and the response
        // `ResponseInterface`
        $response = static::createClient()->request('GET', '/api/articles');

        $this->assertResponseIsSuccessful();

        // Asserts that the returned content type is JSON-LD (the default)
        $this->assertResponseHeaderSame('content-type', 'application/ld+json; charset=utf-8');

        // Asserts that the returned JSON is a superset of this one
        $this->assertJsonContains([
            '@context' => '/api-contexts/Article',
            '@id' => '/api/articles',
            '@type' => "hydra:Collection",
            "hydra:totalItems" => 10,
            'hydra:member' => []
        ]);

        // Because test fixtures are automatically loaded between each test, you can assert
        // the count of items
        $this->assertCount(10, $response->toArray()['hydra:member']);

        // Asserts that the returned JSON is validated by the JSON Schema generated for the
        // API Platform
        // This generated JSON Schema is also used in the OpenAPI spec!
        $this->assertMatchesResourceCollectionJsonSchema(Article::class);
    }
}
```

# ANNEXE N°13

## Résultats de recherche

Cette annexe résume les résultats retournés par les IAs conversationnelles ChatGPT et Bard.

### ChatGPT



You

Je souhaite sécuriser au maximum une web application réalisée avec le Framework Angular. Je souhaiterais donc connaître les bonnes pratiques de développement à suivre dans ce cadre précis. Je précise que l'application que je souhaite sécuriser utilise des formulaires de contacts ainsi que des commentaires et permet aux utilisateurs de s'inscrire sur le site par l'intermédiaire d'une base de donnée MySQL. Elle utilise en outre la solution API Platform pour réaliser ses requêtes à la base de données. Pourrais-tu, en tant qu'expert en sécurité web et Angular, m'indiquer, sous forme de liste à point, les étapes à suivre pour concrètement sécuriser une application de type "blog" ? D'avance merci à toi.



ChatGPT

Bien sûr, voici une liste de bonnes pratiques pour sécuriser une application Angular de type blog :

#### 1. Gestion des mots de passe :

- Utiliser des algorithmes de hachage sécurisés pour stocker les mots de passe dans la base de données.
- Encourager les utilisateurs à créer des mots de passe forts et à les protéger.

#### 2. Authentification et autorisation :

- Mettre en place un système d'authentification robuste comme JWT (JSON Web Token).
- Utiliser des Guards pour protéger les routes sensibles, n'autorisant l'accès qu'aux utilisateurs authentifiés.

#### 3. Protection contre les attaques XSS (Cross-Site Scripting) :

- Sanitiser les données entrantes provenant des utilisateurs pour éviter l'exécution de scripts malveillants.
- Utiliser des fonctionnalités de sécurité telles que l'interpolation sécurisée d'Angular (`{{ data }}`).



#### 4. Protection contre les attaques CSRF (Cross-Site Request Forgery) :

Message ChatGPT...



ChatGPT can make mistakes. Consider checking important information.

**4. Protection contre les attaques CSRF (Cross-Site Request Forgery) :**

- Utiliser des jetons anti-CSRF pour protéger les requêtes sensibles.
- S'assurer que les requêtes HTTP ne sont pas sensibles à ce type d'attaque.

**5. Sécurité des API :**

- Appliquer des stratégies CORS appropriées pour contrôler les requêtes d'origines différentes.
- Limiter l'accès aux API en utilisant des clés d'API, des jetons ou d'autres méthodes d'authentification.

**6. Validation des données :**

- Valider et filtrer toutes les données côté serveur pour éviter les injections SQL ou d'autres attaques par entrées malveillantes.

**7. Mises à jour régulières :**

- Garder le framework Angular, les bibliothèques tierces et les dépendances à jour pour bénéficier des correctifs de sécurité.

**8. Gestion des erreurs :**

- Gérer les erreurs côté client et côté serveur pour ne pas révéler d'informations sensibles.

**9. Sécurité des formulaires :**

- Utiliser les fonctionnalités de sécurité fournies par Angular pour la validation des formulaires et éviter les attaques d'injection.

**10. Tests de sécurité et audits réguliers :**

- Effectuer des tests de sécurité, des audits de code et des analyses de vulnérabilités régulières pour identifier et corriger les failles de sécurité potentielles.

Ces bonnes pratiques peuvent grandement améliorer la sécurité de votre application Angular. Toutefois, la sécurité est un processus continu et il est important de rester informé des nouvelles menaces et vulnérabilités pour maintenir un niveau de sécurité élevé.



Message ChatGPT...



## Google Bard

 Je souhaite sécuriser au maximum une web application réalisée avec le Framework Angular en version 16. J'ai donc besoin de connaître d'une part les principales failles de sécurité de l'écosystème Angular et d'autre part, les bonnes pratiques actuelles de développement dans ce cadre précis. Je précise que l'Application à sécuriser utilise les fonctionnalités suivantes : des formulaires (contact, newsletter et inscription de l'utilisateur), la possibilité de laisser un commentaire et d'utiliser une barre de recherche. L'Application utilise en outre la solution API Platform pour réaliser ses requêtes à la base de données. La base de données est en MySQL 5. Pourrais-tu, en tant qu'expert en sécurité web et Angular, m'indiquer, sous forme de liste à point, les étapes à suivre pour concrètement sécuriser une application de type "Blog" ?.

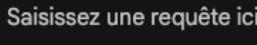


 **Principales failles de sécurité de l'écosystème Angular**

- **Cross-site scripting (XSS)** : une faille XSS permet à un attaquant d'injecter du code malveillant dans une application web, ce qui peut entraîner l'exécution de commandes ou le vol de données.
- **Injection SQL (SQLi)** : une faille SQLi permet à un attaquant d'injecter du code malveillant dans une requête SQL, ce qui peut entraîner l'accès non autorisé à des données ou la modification de données.
- **Attaques par force brute (brute-force)** : une attaque par force brute consiste à essayer de deviner un mot de passe ou une autre valeur secrète en essayant différentes combinaisons possibles.
- **Attaques de type CSRF (cross-site request forgery)** : une attaque CSRF permet à un attaquant d'inciter un utilisateur à effectuer une action qu'il ne voulait pas effectuer, comme effectuer un paiement ou divulguer des informations sensibles.
- **Attaques par injection de code (code injection)** : une attaque par injection de code permet à un attaquant d'injecter du code malveillant dans une application web, ce qui peut entraîner l'exécution de commandes ou le vol de données.

**Bonnes pratiques de développement pour sécuriser une application Angular**

 Saisissez une requête ici

Bard peut afficher des informations inexactes, y compris sur des personnes. Vérifiez donc ses réponses. [Votre confidentialité et Bard](#)

## Bonnes pratiques de développement pour sécuriser une application Angular

- **Utiliser le protocole HTTPS** : HTTPS permet de chiffrer les communications entre le navigateur et l'application, ce qui rend plus difficile pour les attaquants d'intercepter les données sensibles.
- **Implémenter une authentification et une autorisation solides** : l'authentification permet de vérifier l'identité d'un utilisateur, tandis que l'autorisation permet de contrôler les actions qu'un utilisateur peut effectuer.
- **Valider les données saisies par les utilisateurs** : il est important de valider les données saisies par les utilisateurs pour s'assurer qu'elles sont correctes et ne contiennent pas de code malveillant.
- **Sanitiser le contenu avant de l'afficher** : le sanitizer permet de supprimer les caractères dangereux du contenu avant de l'afficher dans l'application.
- **Mettre à jour régulièrement l'application** : les développeurs d'Angular publient régulièrement des mises à jour de sécurité qui corrigent les failles connues. Il est important de mettre à jour régulièrement l'application pour bénéficier de ces correctifs.

## Étapes à suivre pour sécuriser une application de type "Blog"

Voici les étapes à suivre pour sécuriser une application de type "Blog" réalisée avec Angular en version 16 :

1. **Activer HTTPS** : configurez votre serveur web pour qu'il serve l'application sur le protocole HTTPS.
2. **Implémenter une authentification et une autorisation solides** : utilisez un service d'authentification, tel que Keycloak ou Auth0, pour authentifier les utilisateurs et contrôler leurs actions.
3. **Valider les données saisies par les utilisateurs** : utilisez les directives `FormControl` et `FormGroup` pour valider les données saisies dans les formulaires.
4. **Sanitiser le contenu avant de l'afficher** : utilisez le service `DomSanitizer` pour supprimer les caractères dangereux du contenu avant de l'afficher dans l'application.
5. **Mettez à jour régulièrement l'application** : assurez-vous de mettre à jour l'application régulièrement pour bénéficier des correctifs de sécurité.



Saisissez une requête ici



Bard peut afficher des informations inexactes, y compris sur des personnes. Vérifiez donc ses réponses. [Votre confidentialité et Bard](#)

# ANNEXE N°14

## Veille sécuritaire : sites consultés

**“Doctrine Project ORM”**, Doctrine Object-Relational-Mapper for PHP [en ligne], 2024. Disponible à l'adresse : <https://www.doctrine-project.org/projects/orm.html> [consultée le 02/01/24].

**“Doctrine Blog”**, Doctrine Object-Relational-Mapper for PHP [en ligne], 2024. Disponible à l'adresse : <https://www.doctrine-project.org/blog/index.html> [consultée le 02/01/24].

**“Doctrine ORM Security”**, Doctrine Object-Relational-Mapper for PHP [en ligne], 2024. Disponible à l'adresse : <https://www.doctrine-project.org/projects/doctrine-orm/en/2.17/reference/security.html> [consultée le 02/01/24].

**“Doctrine ORM Release Notes 2.17.2”**, Official Doctrine ORM Project GitHub Repository [en ligne], 2024. Disponible à l'adresse : <https://github.com/doctrine/orm/releases> [consultée le 02/01/24].

**“Doctrine Security Advisories”**, Official Doctrine ORM Project GitHub Repository [en ligne], 2024. Disponible à l'adresse : <https://github.com/doctrine/orm/security/advisories> [consultée le 02/01/24].

**“Doctrine ORM Issues”**, Official Doctrine ORM Project GitHub Repository [en ligne], 2024. Disponible à l'adresse : <https://github.com/doctrine/orm/issues> [consultée le 02/01/24].

**“Doctrine SECURITY”**, Official Doctrine ORM Project GitHub Repository [en ligne], 2024. Disponible à l'adresse : <https://github.com/doctrine/orm/blob/old-prototype-3.x/SECURITY.md> [consultée le 02/01/24].

**“Doctrine UPGRADE»**, Official Doctrine ORM Project GitHub Repository [en ligne], 2024. Disponible à l'adresse : <https://github.com/doctrine/orm/blob/old-prototype-3.x/UPGRADE.md> [consultée le 02/01/24].

**“CVE-2021-43822”**, CVE catalog publicly disclosed cybersecurity vulnerabilities [en ligne], 2024. Disponible à l'adresse : <https://cve.mitre.org/cgi-bin/cvename.cgi?name=2021-43822> [consultée le 02/01/24].

**“CVE-2021-43822 Detail”**, CVE catalog publicly disclosed cybersecurity vulnerabilities [en ligne], 2024. Disponible à l'adresse : <https://nvd.nist.gov/vuln/detail/CVE-2021-43822> [consultée le 02/01/24].

**“CWE-89”**, CWE Common Weakness Enumeration [en ligne], 2024. Disponible à l'adresse : <https://cwe.mitre.org/data/definitions/89.html> [consultée le 02/01/24].

**“OWASP Mass Assignment Cheat Sheet”**, OWASP [en ligne], 2023. Disponible à l'adresse : [https://cheatsheets.owasp.org/cheatsheets/Mass\\_Assignment\\_Cheat\\_Sheet.html](https://cheatsheets.owasp.org/cheatsheets/Mass_Assignment_Cheat_Sheet.html) [consulté le 02/01/24].

Dans le cadre de l'implémentation, les sites suivants ont également été visités :

**“Doctrine configuration reference”**, Symfony php framework [en ligne], 2023. Disponible à l'adresse : <https://symfony.com/doc/6.4/reference/configuration/doctrine.html#doctrine-orm-configuration> [consulté le 02/01/24].

**“Doctrine Bundle”**, Symfony php framework [en ligne], 2023. Disponible à l'adresse : <https://symfony.com/bundles/DoctrineBundle/current/installation.html> [consulté le 02/01/24].

**“Doctrine Bundle”**, Symfony php framework [en ligne], 2023. Disponible à l'adresse : <https://symfony.com/bundles/DoctrineMigrationsBundle/current/index.html> [consulté le 02/01/24].

# ANNEXE N°15

## Situation de travail : sites consultés

**“Sécuriser un site web”**, ANSSI Recommandations pour la mise en œuvre d'un site web : maîtriser les standards de sécurité côté navigateur [en ligne], 2021. Disponible à l'adresse : <https://cyber.gouv.fr/publications/securiser-un-site-web> [consulté le 03/01/24].

**“Security”**, Official Angular Documentation [en ligne], 2023. Disponible à l'adresse : <https://angular.io/guide/security> [consulté le 02/01/24].

**“OWASP Top Ten”**, The Open Worldwide application Security Project (OWASP) [en ligne], 2023. Disponible à l'adresse : <https://owasp.org/www-project-top-ten/> [consulté le 02/01/24].

**“Je malmène ta prod en direct avec 15 failles de sécu (G.ELOUET)”**, DEVOXX FRANCE 2023, Conférence sur le développement informatique [en ligne], 2023. Disponible à l'adresse : <https://www.youtube.com/watch?v=Z0rgGrJLt-0> [consultée le 27/10/23 puis revisionnée le 02/01/24].

**“OWASP Cheat Sheet Series Project”**, OWASP [en ligne], 2023. Disponible à l'adresse : <https://cheatsheetseries.owasp.org/index.html> [consulté le 02/01/24].

# ANNEXE N°16

## Webographie

Tableau récapitulatif de tous les sites internets visités dans le cadre du projet.

Type	ThemeOK	Author/Source	Consult the...	Keywords	Selection criteria	URL
DEV UI	<a href="#">Screen statistics</a>		@2023/09/18			<a href="https://www.w3schools.com">https://www.w3schools.com</a>
DEV	<a href="#">Browser statistics</a>		@2023/09/18			<a href="https://www.w3schools.com">https://www.w3schools.com</a>
DESIGN	<a href="#">Inspiration</a>		@2023/09/18			<a href="https://www.awwwar.com">https://www.awwwar.com</a>
DESIGN UX	<a href="#">UX</a>		@2023/09/18			<a href="https://lawsofux.com">https://lawsofux.com</a>
DESIGN	<a href="#">hd creation</a>		@2023/09/18			<a href="https://handbcreations.com">https://handbcreations.com</a>
DEV	<a href="#">FW web</a>	Aman GOEL	@2023/09/18			<a href="https://hackr.io/blog/full-stack-web-development">https://hackr.io/blog/full-stack-web-development</a>
DEV	<a href="#">Accessibility</a>	Kodeco.com	@2023/09/26			<a href="https://www.kodeco.com">https://www.kodeco.com</a>
DEV	<a href="#">CSS in SVG</a>	Mozilla Dev	@2023/09/26			<a href="https://developer.mozilla.org">https://developer.mozilla.org</a>
DESIGN UX	<a href="#">Accessibility</a>	gov.uk	@2023/10/02			<a href="https://accessibility.blog.gov.uk">https://accessibility.blog.gov.uk</a>
DESIGN UI	<a href="#">Figma Proto</a>	gamedev-ressources	@2023/09/18			<a href="https://gamedev-ressources.com">https://gamedev-ressources.com</a>
DESIGN	<a href="#">Screen statistics</a>	W3schools	@2023/10/02			<a href="https://www.w3schools.com">https://www.w3schools.com</a>
DESIGN	<a href="#">Screen resolution</a>	StatCounter	@2023/10/02			<a href="https://gs.statcounter.com">https://gs.statcounter.com</a>
DEV	<a href="#">Flex box CSS</a>		@2023/10/04			<a href="https://css-tricks.com">https://css-tricks.com</a>
LEGAL	<a href="#">DSA</a>	vie-publique.fr	@2023/10/04			<a href="https://www.vie-publique.fr">https://www.vie-publique.fr</a>
DESIGN	<a href="#">SVG export</a>	Illustrator official doc	@2023/10/06	illustrator export SVG options	official doc	<a href="https://helpx.adobe.com/illustrator/exporting-svg.html">https://helpx.adobe.com/illustrator/exporting-svg.html</a>
DEV	<a href="#">Coupler CMS à Symfony</a>	Sulu.io	@2023/10/10	how to use a CMS with Symfony?	Sulu official web site	<a href="https://sulu.io/blog/a-guide-to-coupling-sulu-with-symfony">https://sulu.io/blog/a-guide-to-coupling-sulu-with-symfony</a>
DEV	<a href="#">Hébergement VueJS</a>	Back4App	@2023/10/10	hébergeurs compatibles VueJS		<a href="https://blog.back4app.com/hosting-vuejs">https://blog.back4app.com/hosting-vuejs</a>
DEV	<a href="#">Hébergement Symfony</a>	help.ovhcloud.com	@2023/10/10	hébergeurs compatibles Symfony	OVH official web site	<a href="https://help.ovhcloud.com/hosting-symfony">https://help.ovhcloud.com/hosting-symfony</a>
DEV	<a href="#">Hébergement Symfony</a>	platform.sh	@2023/10/10	idem	Recommandé par Symfony + Conseillé par F. ASHRAFI_GHALATI MAIS déploiement complexe	<a href="https://platform.sh/">https://platform.sh/</a>
DEV	<a href="#">Déploiement Symfony</a>	RSync	@2023/10/10	Deployment Symfony	Conseillé par F. ASHRAFI_GHALATI	<a href="https://fr.wikipedia.org/wiki/D%C3%A9ploiement_symfony">https://fr.wikipedia.org/wiki/D%C3%A9ploiement_symfony</a>
	<a href="#">Untitled</a>					<a href="https://testing-angular.com">https://testing-angular.com</a>
DEV WEB	<a href="#">FlowBite install</a>	GitHub FlowBite	@2023/10/16		Official	<a href="https://github.com/flowbite/flowbite">https://github.com/flowbite/flowbite</a>
DESIGN	<a href="#">Biases cognitifs</a>	ECN	@2023/10/15	bias cognitif achat		<a href="https://www.ecommerec.com/biases-cognitifs-aachat">https://www.ecommerec.com/biases-cognitifs-aachat</a>

:≡ Type	Aa ThemeOK	≡ Author/Source	Consult the...	≡ Keywords	≡ Selection criteria	🔗 URL
ACCESSIBILITY	<a href="#">Accessibiliiy</a>	Intopia	@2023/10/19		Interesting tools on map shape	<a href="https://intopia.digital/">https://intopia.digital/</a>
DEV	<a href="#">DB choice</a>	AWS	@2023/10/21		AWS reputation	<a href="https://aws.amazon.us/mysql#:~:text=MySQL%20structure%C3%A9s">https://aws.amazon.us/mysql#:~:text=MySQL%20structure%C3%A9s</a>
DEV WEB	<a href="#">DB choice and host</a>	OVH	@2023/10/21		OVH reputation	<a href="https://www.ovhcloud.com">https://www.ovhcloud.com</a>
DEV WEB	<a href="#">npm run watch script problem</a>	npmjs.com	@2023/10/21	npm run watch		<a href="https://www.npmjs.com">https://www.npmjs.com</a>
DEV WEB	<a href="#">npm run build script problem</a>		@2023/10/22	npm run build problem with Symfony		<a href="https://stackoverflow.com">https://stackoverflow.com</a>
DEV WEB	<a href="#">setting up symfony and vueJS</a>		@2023/10/22	how to install symfony and vuejs?		<a href="https://makina-corpus.com">https://makina-corpus.com</a>
DEV WEB	<a href="#">install typescript in a Symfony project</a>		@2023/10/22		MS official website	<a href="https://www.typescriptlang.org">https://www.typescriptlang.org</a>
DESIGN UI	<a href="#">settings menu options</a>	Interface In Game.com	@2023/10/24			<a href="https://interfaceingame.com">https://interfaceingame.com</a>
DESIGN UI	<a href="#">Figma Components</a>	Figma Learn	@2023/10/24			<a href="https://help.figma.com">https://help.figma.com</a>
DESIGN	<a href="#">Gherkin notation for US</a>	test quality	@2023/10/24			<a href="https://www.testquality.com/scenarios/~text=Gherkin">https://www.testquality.com/scenarios/~text=Gherkin</a>
DEV	<a href="#">Change Git Repo</a>	xenovation	@2023/10/26	how to change git repo with git cli		<a href="https://xenovation.cc">https://xenovation.cc</a>
SECURITY	<a href="#">Security failed</a>	OWASP cheat sheet	@2023/10/27	OWASP cheat sheet	OWASP reputation and G.Elouet meetup Devoxx 23	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">CSS security</a>	OWASP cheat sheet	@2023/10/27	OWASP cheat sheet	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">SQL security</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">PHP security</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">Web Service Security</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">Dependencies vulnerabilities</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">REST API security</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">JS security</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">Microservices security</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">JAVA security</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">JAVA web token</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">HTML security</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">.NET security</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">Docker security</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>
SECURITY	<a href="#">Database</a>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com">https://cheatsheetse.com</a>

:≡ Type	Aa ThemeOK	≡ Author/Source	Consult the...	≡ Keywords	≡ Selection criteria	🔗 URL
	<u>security</u>					
SECURITY	<u>UI security</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>Authentification security</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>Content security</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>CSRF prevention</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>XSS prevention</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>Denial service</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>Journalisation</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>DB request security</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>Threats modeling</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>User privacy protection</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>XML security</u>	idem	@2023/10/27	idem	idem	<a href="https://cheatsheetse.com/">https://cheatsheetse.com/</a>
SECURITY	<u>ANSSI</u>	cyber.gouv	@2023/12/27	recommandations cyber sécurité	ANSSI	<a href="https://cyber.gouv.fr/">https://cyber.gouv.fr/</a>
DEV WEB	<u>More features in Symfony webpack.json</u>		@2023/12/27		Official Symfony doc	<a href="https://symfony.com/doc">https://symfony.com/doc</a>
DEV WEB	<u>Hash user id</u>		@2023/12/27			<a href="https://www.php.net/">https://www.php.net/</a>
DEV	<u>BDD</u>		@2023/12/28			<a href="https://www.wefiit.co">https://www.wefiit.co</a>
DEV	<u>TDD</u>		@2023/12/28			<a href="https://openclassroo">https://openclassroo</a>
DEV WEB	<u>Angular strict mode</u>		@2023/11/01	ng new my-project --strict false		<a href="https://angular.io/guide">https://angular.io/guide</a>
DEV WEB	<u>SCSS</u>		@2023/11/01	css vs scss ?		<a href="https://stacklima.com">https://stacklima.com</a>
DEV WEB	<u>SCSS syntaxe</u>		@2023/11/01			<a href="https://sass-lang.com">https://sass-lang.com</a>
DEV	<u>Database schema</u>	IBM	@2023/11/03		IBM	<a href="https://www.ibm.com">https://www.ibm.com</a>
DEV WEB	<u>MySQL Workbench</u>	Graphic Art	@2023/11/03			<a href="https://grafikart.fr/tutoriels">https://grafikart.fr/tutoriels</a>
DEV WEB	<u>Database relationship</u>	Stackoverflow	@2023/11/03	database relations n:m, 1:n?	Stackoverflow	<a href="https://stackoverflow.com">https://stackoverflow.com</a>
SECURITY	<u>Secure JS code</u>	Maxmilian Schwartzmuller	@2023/11/05	js secure	The effectiveness of the teacher	<a href="https://academind.cc">https://academind.cc</a>
DEV WEB	<u>RESTful API</u>	Maximilian Schwartzmuller	@2023/11/05		idem	<a href="https://academind.cc">https://academind.cc</a>
DEV	<u>Git LFS</u>	git-lfs.com	@2023/11/06			<a href="https://git-lfs.com/">https://git-lfs.com/</a>
DEV	<u>Git ignore file</u>	techblobst	@2023/11/06	how to configure a gitignore file?		<a href="https://techblobst.com">https://techblobst.com</a>
DEV WEB	<u>Angular 17 release</u>		@2023/11/08	Angular 17	Maximilian Schwartzmuller	<a href="https://www.youtube.com">https://www.youtube.com</a>

Type	Theme	Author/Source	Consult the...	Keywords	Selection criteria	URL
DEV WEB	<a href="#">Standalone Component Angular</a>		@2023/11/09	Standalone components Angular	Official Angular Doc	<a href="https://angular.io/guide/standalone-components-angular">https://angular.io/guide/standalone-components-angular</a>
TEST	<a href="#">TDD Angular</a>	Angular Official Doc	@2023/11/10	TDD with Angular	Official Doc	<a href="https://testing-angular.com/tdd-angular">https://testing-angular.com/tdd-angular</a>
TEST	<a href="#">Cypress for Angular</a>	AOD	@2023/11/13			<a href="https://testing-angular.com/cypress-for-angular">https://testing-angular.com/cypress-for-angular</a>
DEV WEB	<a href="#">Architecture Angular</a>	Angular.fr	@2023/11/13			<a href="https://angular.fr/get-started/architecture">https://angular.fr/get-started/architecture</a>
TEST	<a href="#">Spectator for Angular</a>	Testing Angular.com	@2023/11/13			<a href="https://testing-angular.com/spectator-for-angular">https://testing-angular.com/spectator-for-angular</a>
DESIGN	<a href="#">Planning Pocker</a>		@2023/11/13			<a href="https://planningpoker.com/">https://planningpoker.com/</a>
TEST	<a href="#">Cypress tutorial</a>		@2023/11/14			<a href="https://www.browserstack.com/guides/cypress-tutorial">https://www.browserstack.com/guides/cypress-tutorial</a>
TEST	<a href="#">Karma Jasmine</a>		@2023/11/14			<a href="https://medium.com/@karma_jasmine">https://medium.com/@karma_jasmine</a>
DEV MOBILE	<a href="#">Ecoconception Mobile</a>		@2023/11/14			<a href="https://thetribe.io/ecoconception/mobile">https://thetribe.io/ecoconception/mobile</a>
DEV WEB	<a href="#">Ecoconception Web</a>		@2023/11/14			<a href="https://thetribe.io/ecoconception/web">https://thetribe.io/ecoconception/web</a>
DEV WEB	<a href="#">CSS Grid compatibility</a>		@2023/11/14			<a href="https://caniuse.com/css-grid-compatibility">https://caniuse.com/css-grid-compatibility</a>
DEV WEB	<a href="#">API for search bar</a>		@2023/11/14			<a href="https://www.algolia.com/doc/api/search-bar">https://www.algolia.com/doc/api/search-bar</a>
TEST	<a href="#">E2E with Angular</a>	Official Angular 17 doc	@2023/11/16		Official Doc	<a href="https://angular.dev/test-with-angular">https://angular.dev/test-with-angular</a>
DEV WEB	<a href="#">CSS Grid</a>	MDN	@2023/11/17		Mozilla Dev	<a href="https://developer.mozilla.org/en-US/docs/Web/CSS">https://developer.mozilla.org/en-US/docs/Web/CSS</a>
DEV WEB	<a href="#">CSS Grid with Angular</a>		@2023/11/17			<a href="https://www.ifourtech.com/courses/html-css-angular">https://www.ifourtech.com/courses/html-css-angular</a>
SECURITY	<a href="#">Api Platform</a>	Symfony Cast	@2023/11/21		Symfony Cast	<a href="https://api-platform.com">https://api-platform.com</a>
DEV WEB	<a href="#">Mobile screen size</a>		@2023/11/22			<a href="https://stackoverflow.com/questions/1895234/mobile-screen-size">https://stackoverflow.com/questions/1895234/mobile-screen-size</a>
DEV WEB	<a href="#">Media Queries</a>		@2023/11/22		W3C	<a href="https://www.w3.org/Learn/HTML/MediaQueries">https://www.w3.org/Learn/HTML/MediaQueries</a>
DESIGN	<a href="#">Figma to HTML/CSS</a>		@2023/11/22			<a href="https://medium.com/@figmadesign/figma-to-html-css-1f3a2a2a2a">https://medium.com/@figmadesign/figma-to-html-css-1f3a2a2a2a</a>
DEV WEB	<a href="#">Flowbite library</a>		@2023/11/23			<a href="https://flowbite.com/library">https://flowbite.com/library</a>
DEV WEB	<a href="#">Custom Flowbite Tailwind config</a>		@2023/11/23			<a href="https://flowbite.com/custom-flowbite-tailwind-config">https://flowbite.com/custom-flowbite-tailwind-config</a>
ACCESSIBILITY	<a href="#">WCAG 2.2 Techniques</a>		@2023/12/02		W3C	<a href="https://www.w3.org/WAI/standards-guidelines/wcag">https://www.w3.org/WAI/standards-guidelines/wcag</a>
ACCESSIBILITY	<a href="#">Attributs ARIA pour HTML et technologie d'assistance</a>		@2023/12/02			<a href="https://hubvisory.com/aria-attributes-for-html-and-assistive-technology">https://hubvisory.com/aria-attributes-for-html-and-assistive-technology</a>
ACCESSIBILITY	<a href="#">Accessibility in Angular</a>		@2023/12/02		AOD	<a href="https://angular.io/guide/accessibility">https://angular.io/guide/accessibility</a>
ECOCONCEPTION	<a href="#">RGESN</a>		@2023/12/03		Site du Gouvernement Français	<a href="https://ecoresponsabilite.gouv.fr/">https://ecoresponsabilite.gouv.fr/</a>

Type	Theme	Author/Source	Consult the...	Keywords	Selection criteria	URL
ECOCONCEPTION	<a href="#">GR491</a>		@2023/12/03			<a href="https://gr491.isit-eur.com">https://gr491.isit-eur.com</a>
TEST	<a href="#">E2E avec Angular</a>		@2023/12/11			<a href="https://angular.dev/test">https://angular.dev/test</a>
TEST	<a href="#">Jeu d'Essais et Plan de Tests</a>		@2023/12/23			<a href="https://fr.parasoft.com">https://fr.parasoft.com</a>
SECURITY	<a href="#">Angular Security</a>		@2024/01/02	What are the best security practices in 2024 to prevent vulnerabilities and attacks on an Angular web application?	Angular official Doc	<a href="https://angular.io/guide/security">https://angular.io/guide/security</a>
SECURITY	<a href="#">Untitled</a>		@2024/01/02		Official OWASP web site	<a href="https://owasp.org">https://owasp.org</a>
SECURITY	<a href="#">Sécurité des Composants d'accès aux données via Doctrine</a>		@2024/01/02	doctrine ORM security watch	Official Doctrine documentation web site	<a href="https://www.doctrine-project.org">https://www.doctrine-project.org</a>
SECURITY	<a href="#">Sécurité API Platform</a>		@2023/12/28	api platform doc security	Official API Platform doc	<a href="https://api-platform.com">https://api-platform.com</a>
SECURITY	<a href="#">ANSSI sécurité des sites web</a>		@2024/01/03	ANSSI sécurité site web	Site officiel de l'ANSSI	<a href="https://cyber.gouv.fr/">https://cyber.gouv.fr/</a>
DEV WEB	<a href="#">Angular prob [formGroup]</a>		@2024/01/04	site: stackoverflow trouble [formGroup] Angular and Module imports	Stackoverflow reputation	<a href="https://stackoverflow.com/questions/73035566/angular-problem-with-formgroup-in-a-formarray">https://stackoverflow.com/questions/73035566/angular-problem-with-formgroup-in-a-formarray</a>
SECURITY	<a href="#">Validation Symfony</a>		@2024/01/10	Validation Symfony	Official Symfony	<a href="https://symfony.com/doc/current/validation.html">https://symfony.com/doc/current/validation.html</a>
SECURITY	<a href="#">Voter Symfony</a>		@2024/01/10	Security Symfony doc	Official Doc	<a href="https://symfony.com/doc/current/voting.html">https://symfony.com/doc/current/voting.html</a>

# ANNEXE N°17

## Références bibliographiques

Cette annexe présente les ouvrages consultés lors de la réalisation du projet, selon leur thématique.

### Diagrammes UML

ROQUES Pascal, [UML2.5 par la pratique : études de cas et exercices corrigés](#) 8<sup>ème</sup> édition, Paris, Eyrolles, 2018.

LARMAN Craig, [Applying UML and Patterns](#), Upper Saddle River, Boston, Prentice Hall, 2004.

### Code

ROBERT C. Martin, [Clean code](#), Upper Saddle River, Boston, Prentice Hall, 2009.

NIXON Robert, [Learning PHP, MySQL & JavaScript](#), 6<sup>ème</sup> édition, Sebastopol, O'Reilly, 2021.

### Angular

DIENY Simon, [Apprendre Angular](#) 1<sup>er</sup> édition, Paris, Eyrolles, 2020.

# ANNEXE N°18

## Lexique

Cette annexe présent le lexique des termes techniques employés dans ce dossier.

### Headers

Un « header » fait référence à l'en-tête de la requête HTTP. Les requêtes SQL avec Postman sont généralement effectuées en utilisant une requête HTTP POST ou GET vers une API ou un service web qui traite la requête SQL. L'en-tête (header) dans ce contexte contient des métadonnées supplémentaires sur la requête, telles que le type de contenu accepté ou le type de contenu envoyé.

opérations futures, tandis que `flush()` applique réellement ces changements en base de données. Il est généralement recommandé d'appeler `flush()` après avoir effectué toutes les opérations `persist()` nécessaires pour assurer la cohérence des données entre votre application et la base de données.

### Serializer

Un serializer est un programme qui permet de prendre des objets complexes et de les transformer en chaîne de caractère, bien plus facile à transporter. Les plus courants sont des serializers qui vont convertir des éléments vers du JSON ou du XML, mais il en existe de nombreux autres. Il existe également des déserialiseurs qui permettent l'opération inverse.

### Slug

Un « slug » fait référence à une version conviviale pour les moteurs de recherche et les utilisateurs humains d'une chaîne de caractères. Les slugs sont souvent utilisés pour représenter des URL conviviales et compréhensibles (ex /apprendre-a-coder).

### DBAL

Est une couche d'abstraction qui offre une API pour communiquer avec les bases de données relationnelles. La bibliothèque DBAL permet d'écrire des requêtes indépendamment des modèles ORM, par exemple pour la construction de rapports ou directement pour manipulations de données.

### Persist/Flush

`persist()` indique à Doctrine (l'ORM de Symfony) de suivre une entité pour des

## **Changelog (journal des modifications)**

Fait référence à un document ou à une liste qui répertorie les modifications, les mises à jour, les correctifs et les ajouts effectués sur une application, un framework, une bibliothèque ou tout autre logiciel pertinent.

## **CVE**

Signifie « Common Vulnerabilities and Exposures » (Vulnérabilités et Expositions Courantes). Il s'agit d'un dictionnaire de référence international pour les vulnérabilités de sécurité connues. Les CVE sont des identifiants uniques assignés à chaque vulnérabilité de sécurité répertoriée dans la base de données CVE.

## **Assertion**

Dans le cadre des tests unitaires, une assertion est une déclaration qui permet de vérifier qu'une condition spécifique dans le code testé est vraie. Les assertions sont utilisées pour comparer le résultat attendu avec le résultat réel obtenu lors de l'exécution du test.

## **SPA**

Une Single Page Application (SPA) est un type d'application web qui fonctionne dans un navigateur et ne nécessite pas de rechargement de page pendant son utilisation. Contrairement aux sites web traditionnels, une SPA charge dynamiquement le contenu et actualise l'interface utilisateur sans recharger entièrement la page.

## **Observables**

Les Observables représentent une source de données potentielles qui peuvent émettre plusieurs valeurs de manière asynchrone dans le temps. Les Observables fournissent une façon pratique de gérer les flux de données, notamment les événements, les réponses de requêtes HTTP, et autres opérations asynchrones.

## **Endpoint**

Point d'entrée spécifique ou une URL permettant à une application, un service ou un utilisateur d'interagir avec une base de données ou un service de base de données.

## **Repository**

Dans le cadre de Symfony, c'est une classe PHP qui agit comme une passerelle entre votre application et la base de données.

## **Entity**

Est une classe PHP® spécifique qui représente une entité métier dans une application. Ces entités sont généralement utilisées avec Doctrine (ORM).

## **ORM (Object-Relational Mapping)**

Un ORM (ex : Doctrine), au lieu d'écrire des requêtes SQL directement, les développeurs travaillent avec des objets PHP® qui représentent les Tables et les données.

