ATTENBOROUGH Elliott FISA 3 Informatique

CARLIER Amandine

TP - Technologie du Web

Réalisation du site Web :

Chatoon

Monsieur DEGROOTE Jérôme

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE

Université Polytechnique HAUTS-DE-FRANCE

Introduction

Dans le cadre du module de technologie web, nous avons eu l'occasion de réaliser un petit site web contenant plusieurs pages. Ainsi, au départ de notre projet, nous avons donc dû trouver l'idée sur laquelle allait reposer notre site. Étant tous deux amoureux des animaux et passionnés de chats, c'est donc naturellement que nous sommes partis sur un site web qui allait simuler un blog de chats. Nous avons appelé notre site : Chatoon.

Afin de contextualiser notre site Chatoon, il est essentiel de préciser que ce site internet qui possède trois acteurs donc trois objets au sein de son architecture, nous avons l'objet Chat, Personne (qui possède un chat) et Commentaire (le commentaire d'une personne sur un chat). Nous reviendrons plus tard sur l'architecture en détail de ce site.

Celui-ci réutilise alors les technologies que nous avons eu l'opportunité d'étudier dans ce même modules telles que Spring Boot, Jersey ou spring-data et hsqldb. En plus ce ces dernières, nous avons choisi de créer notre front via le la bibliothèque JQuery car nous avons eu l'opportunité de la manipuler durant nos séances de cours.

Bien évidemment, ce site web nous permet de faire des appels aux méthodes CRUD de nos ressources et il respecte les normes REST que nous avons pu voir en cours.

Ce rapport va se diviser en plusieurs parties, une première traitera des fonctionnalités implémentées au cœur du site web, une seconde de la gestion du projet. Puis, nous distinguerons la partie backend du projet de la partie frontend, pour terminer sur les difficultés rencontrées lors de la réalisation globale de Chatoon.

Fonctionnalités implémentées

Comme expliqué lors de l'introduction de ce projet, notre site Chatoon comporte trois objets, un objet Chat, Personne et Commentaire. Ainsi, nous avons fait en sorte de pouvoir effectuer des méthodes CRUD ("create", "read", "update", "delete") sur chacun d'entre eux.

Voyons alors un récapitulatif de ce qu'il est possible de faire avec notre site web (une explication plus en détail de l'utilisation de celui-ci est disponible plus tard dans ce rapport, ainsi que dans le readme de notre projet) :

- Pour l'objet Chat :
    - Ajouter un chat ;
    - Récupérer l'ensemble des chats ;
    - Récupérer un chat spécifique via son identifiant ;
    - Supprimer un chat via son identifiant ;
    - Ajouter un commentaire sur un chat spécifique ;
    - Modifier partiellement un chat via son identifiant ;
    - Modifier la totalité d'un objet chat via son identifiant également ;
- Pour l'objet Personne :
    - Ajouter une personne ;
    - Récupérer l'ensemble des personnes ;
    - Récupérer une personne spécifique via son identifiant ;
    - Supprimer une personne via son identifiant ;
    - Modifier partiellement une personne via son identifiant ;
    - Modifier la totalité d'un objet personne via son identifiant également ;
- Pour l'objet Commentaire :
    - Supprimer un commentaire via son identifiant ;
    - Modifier partiellement un commentaire via son identifiant ;
    - Modifier la totalité d'un objet commentaire via son identifiant ;

Voyons désormais comment nous avons choisi de gérer notre projet.

Gestion du projet

Pour la gestion de ce projet nous avons choisi de répartir le travail de la manière suivante :

- Elliott a travaillé sur de toute la partie front en JQuery, bibliothèque Javascript, qu'il a déjà eu l'occasion de prendre en main. De plus, il s'est également occupé de compléter la rédaction de toute la partie front contenu dans ce rapport et dans le readme.
- Amandine s'est occupée de toute la partie back de Chatoon et de sa conception (DTO, exceptions, contrôleurs, services et repositories) ainsi que de la rédaction du readme et du rapport.

Il nous semblait judicieux de procéder de cette façon puisque Elliott a un peu plus d'expérience dans le domaine du développement front et Amandine dans celui du back de part ses activités en entreprise.

De plus, nous devons préciser que la création de notre diagramme de classes, exposé plus tard dans ce rapport a été effectué en binôme afin que nous partions tous les deux dans la même direction.

Ainsi, nous avons pu avancer tous deux à notre rythme tout au long de ce projet. De surcroît, pour faciliter notre travail en collaboration, nous avons travaillé par le biais de GitHub, ce qui a été très utile. Cependant, nous n'avons pas exploité à 100% cet outil puisque, pour la simplicité d'utilisation de celui-ci, nous sommes restés sur la gestion d'une unique branche, ce qui en entreprise ne se serait pas effectué de la sorte.

Voici ci-dessous le lien GitHub de notre site web :

https://github.com/AvesSeven/chatoon.git

Vous constaterez que nous avons choisi de laisser notre base de données au sein de ce projet GitHub, ce qui encore une fois n'est pas à faire dans le cadre professionnel, afin d'une part de vous donner un jeu de données pour le test de notre site web, d'autre part de permettre à Elliott ainsi que moi-même de travailler avec les même données.

De plus, nous avons découvert que l'outil Postman pouvait également s'utiliser de manière coopérative, ce qui nous a permis de simplifier et de partager la réalisation de nos requêtes HTTP via cet outil.

Maintenant que nous avons fait le point sur notre gestion de ce projet, intéressons plus en détails dans la prochaine section sur la partie backend de celui-ci réalisé par Amandine.

Applicatif Backend

Pour cette section, nous allons premièrement expliquer les différents contrats d’échanges qui s’imposent sur chaque fonctionnalité.

Complétons alors ce que nous expliquions en introduction :

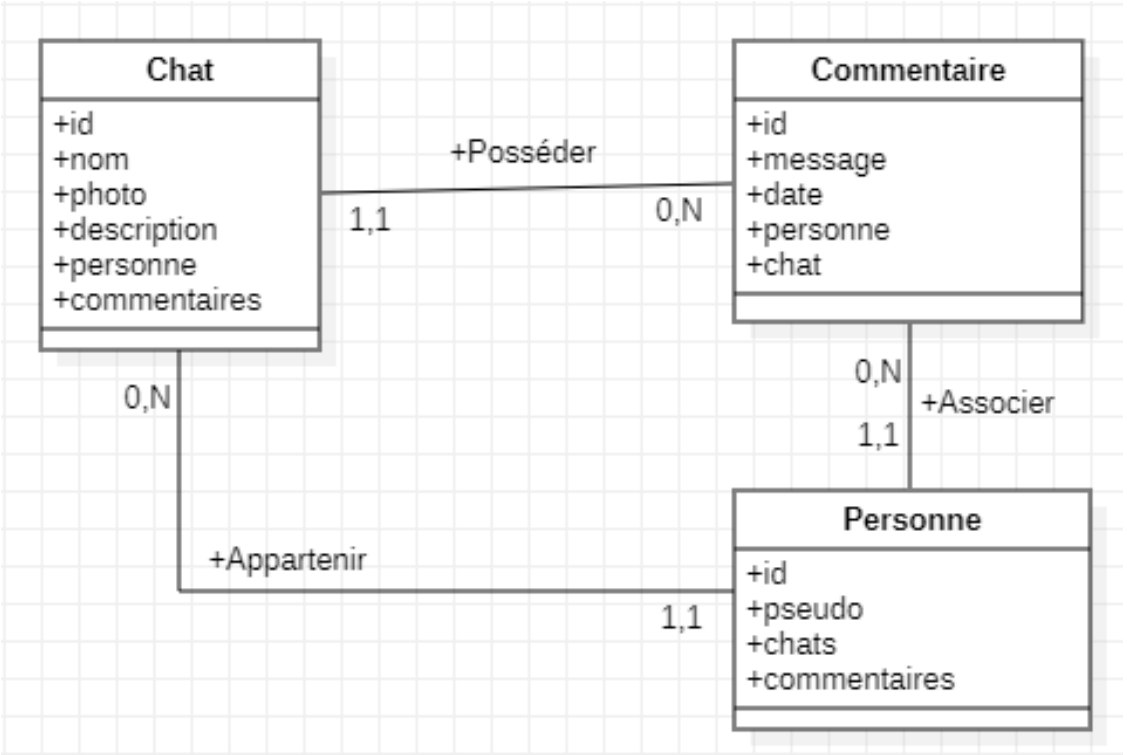
- Pour les chats :
    - Pour ajouter un chat :
        - La route est : localhost:8080/api/v1/chats avec la méthode POST ;
        - Il faut donner son nom, sa description et l'identifiant de son propriétaire sous le format JSON (un exemple est disponible dans le readme) ;
    - Récupérer l'ensemble des chats :
        - La route est : localhost:8080/api/v1/chats avec la méthode GET ;
    - Récupérer un chat spécifique via son identifiant :
        - La route est : localhost:8080/api/v1/chats/:id avec la méthode GET (:id représente une variable correspondant à l'identifiant d'un chat) ;
    - Supprimer un chat via son identifiant :
        - La route est : localhost:8080/api/v1/chats/:id avec la méthode DELETE (:id représente une variable correspondant à l'identifiant d'un chat) ;
    - Ajouter un commentaire sur un chat spécifique :
        - La route est : localhost:8080/api/v1/chats/:id/commentaires avec la méthode PUT (:id représente une variable correspondant à l'identifiant d'un chat) ;
        - Il faut donner le message (ou commentaire) et l'identifiant de de la personne qui a écrit le message sous le format JSON (un exemple est disponible dans le readme) ;
    - Modifier partiellement un chat via son identifiant :
        - La route est : localhost:8080/api/v1/chats/:id avec la méthode PATCH (:id représente une variable correspondant à l'identifiant d'un chat) ;
        - L'attribut modifié doit être donné dans le format JSON (un exemple est disponible dans le readme) ;

    - Modifier la totalité d'un objet chat via son identifiant également :
        - La route est : localhost:8080/api/v1/chats/:id avec la méthode PUT (:id représente une variable correspondant à l'identifiant d'un chat) ;
        - Il faut donner son nom, sa description et l'identifiant de son propriétaire sous le format JSON (un exemple est disponible dans le readme)

- Pour les personnes :
    - Ajouter une personne :
        - La route est : localhost:8080/api/v1/personnes avec la méthode POST ;
        - Il faut donner son pseudo sous le format JSON (un exemple est disponible dans le readme) ;
    - Récupérer l'ensemble des personnes :
        - La route est : localhost:8080/api/v1/personnes avec la méthode GET ;
    - Récupérer une personne spécifique via son identifiant :
        - La route est : localhost:8080/api/v1/personnes/:id avec la méthode GET (:id représente une variable correspondant à l'identifiant d'une personne) ;
    - Supprimer une personne via son identifiant :
        - La route est : localhost:8080/api/v1/personnes/:id avec la méthode DELETE (:id représente une variable correspondant à l'identifiant d'une personne) ;
    - Modifier partiellement une personne via son identifiant :
        - La route est : localhost:8080/api/v1/personnes/:id avec la méthode PATCH (:id représente une variable correspondant à l'identifiant d'une personne) ;
        - L'attribut modifié doit être donné dans le format JSON (un exemple est disponible dans le readme) ;
    - Modifier la totalité d'un objet personne via son identifiant également :
        - La route est : localhost:8080/api/v1/personnes/:id avec la méthode PUT (:id représente une variable correspondant à l'identifiant d'une personne) ;
        - Il faut donner son pseudo sous le format JSON (un exemple est disponible dans le readme) ;

- Pour les commentaires :
    - Supprimer un commentaire via son identifiant :
        - La route est : localhost:8080/api/v1/commentaires/:id avec la méthode DELETE (:id représente une variable correspondant à l'identifiant d'un commentaire) ;
    - Modifier partiellement un commentaire via son identifiant :
        - La route est : localhost:8080/api/v1/commentaires/:id avec la méthode PATCH (:id représente une variable correspondant à l'identifiant d'un commentaire) ;
        - L'attribut modifié doit être donné dans le format JSON (un exemple est disponible dans le readme) ;
    - Modifier la totalité d'un objet commentaire via son identifiant :
        - La route est : localhost:8080/api/v1/commentaires/:id avec la méthode PUT (:id représente une variable correspondant à l'identifiant d'un commentaire) ;
        - Il faut donner son pseudo sous le format JSON (un exemple est disponible dans le readme).

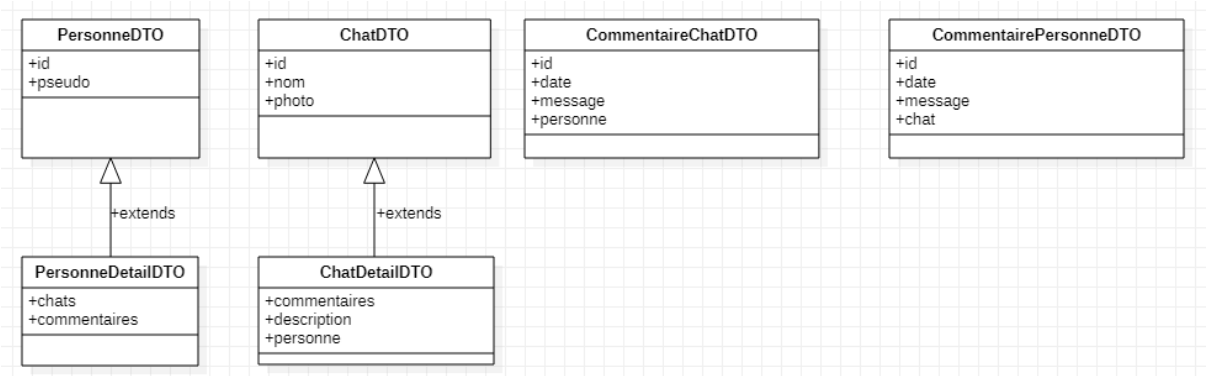
À présent, penchons nous sur le diagramme de classes qui a été effectué, ainsi que sur les choix fait au sujet de la conception :



Sur le diagramme de classe ci-dessus, nous pouvons constater l'ensemble des liens entre les trois acteurs ainsi que leurs attributs respectifs. Ainsi, nous remarquons qu'un chat possède un ou plusieurs commentaires et un commentaire fait référence à un seul chat et une seule personne. Puis une personne peut être associée à plusieurs commentaires ainsi qu'à plusieurs chats, alors qu'un chat n'appartient qu'à une seule personne.

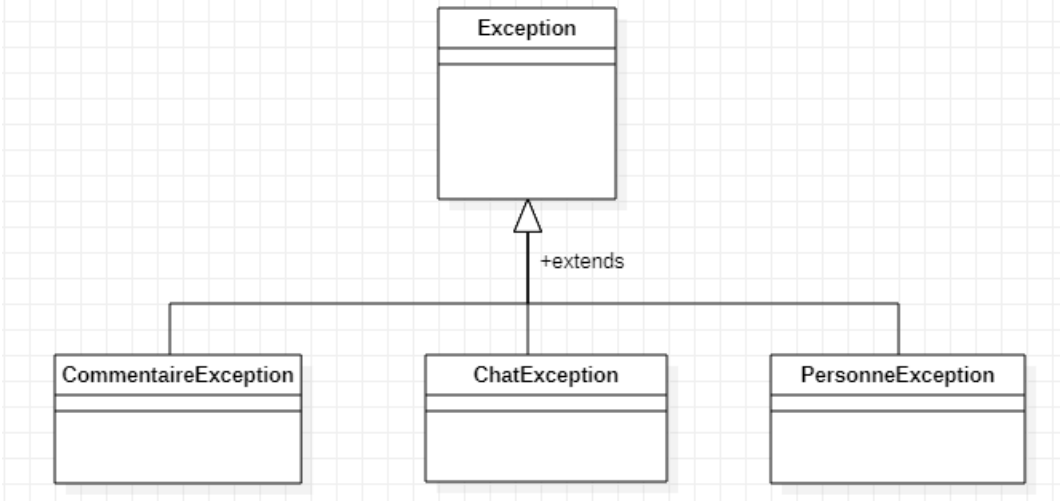
Ainsi, ce diagramme nous permet de mieux visualiser l'architecture globale de notre site web et plus particulièrement la partie back. Cependant cette conception n'a pas été suffisante, il nous à alors fallu concevoir des DTO ("Data Object Transfert", pour le transfert des données du back vers le front et respectivement), des exceptions afin de gérer les différents cas d'erreurs et des contrôleurs, des services et des repositories pour limiter les problèmes de persistances liés à la base de données (puisqu'un objet peut avoir trois états "detached", "persistent" et "transient").

Nous avons alors les DTO ci-dessous :



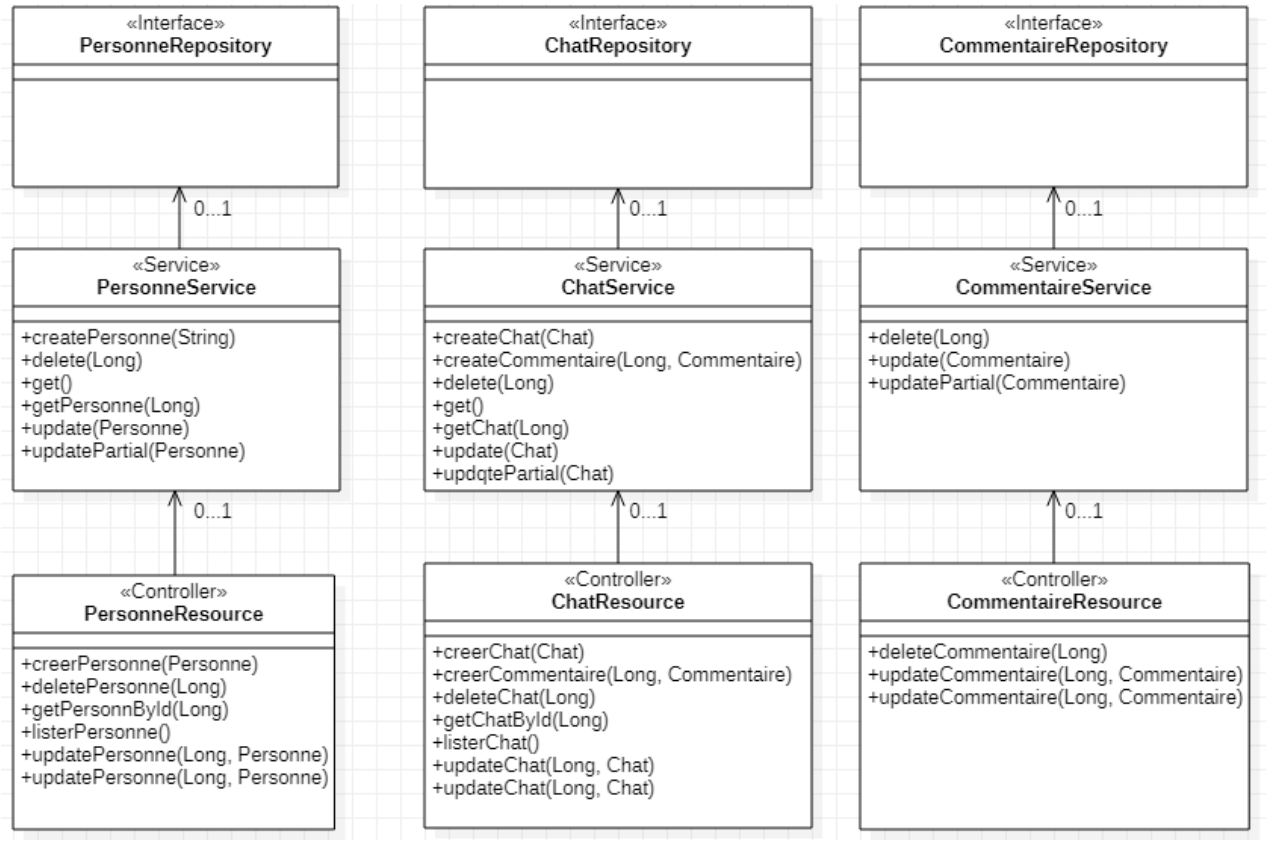
Nous pouvons remarquer que ces DTO sont en quelque sorte des simplifications des objets en base de données. De plus, nous constatons que nous avons deux DTO pour les personnes et les chats car en fonction de la méthode CRUD appelée, nous ne souhaitons pas forcément avoir toutes les informations de ces objets.

Puis, nous avons les classes exceptions suivantes :



Celles-ci nous permettent de gérer les cas d'erreurs concernant les objet Commentaire, respectivement Chat et Personne, en fonction de l'erreur obtenue au cours de l'exécution de certaines méthodes.

Enfin, nous avons les différentes couches avec les contrôleurs, les services et les repositories pour chaque objet :



Par conséquent, pour chaque type d'objet, nous allons avoir un contrôleurs qui va permettre de valider les données passées en entrée (que ça soit dans la route ou dans les informations au format JSON), le service va lui être capable de manipuler les différentes données en fonction de la méthode appelée et enfin le repository va communiquer avec la base de données directement.

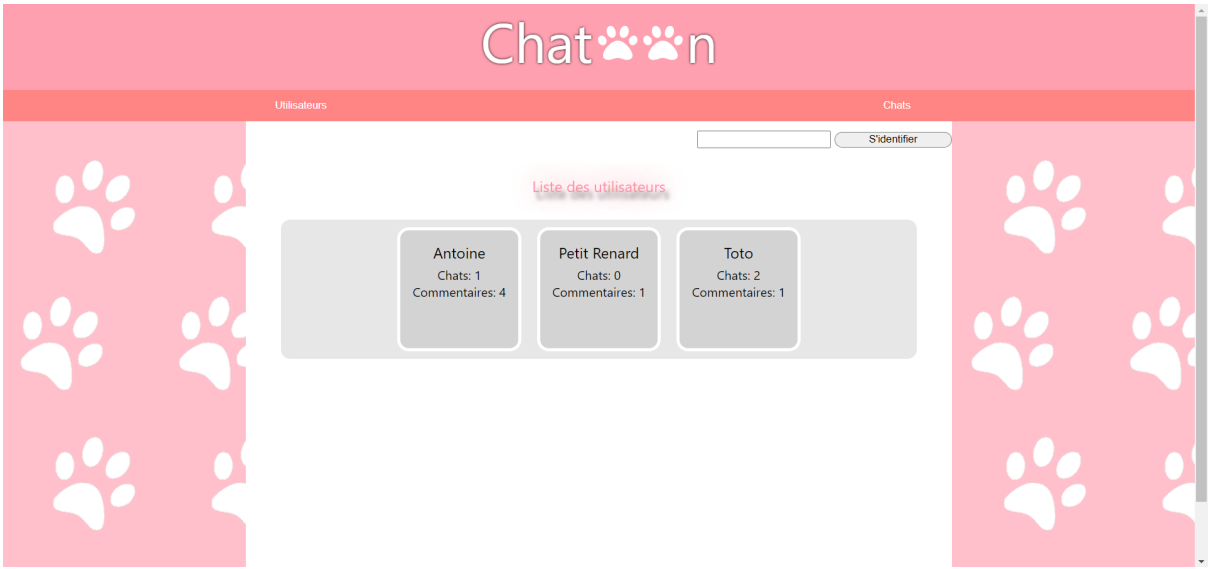
Comme expliqué en introduction de cette section, l'intérêt de diviser la partie backend en contrôleurs, services et repositories est grand puisqu'il permet une bonne gestion des différents états de nos objets circulant au sein de notre architecture et donc de ne pas avoir de problème avec nos objets. De surcroît, cette architecture que nous pouvons nommée d'architecture n-tiers (ou 3-tiers) est très intéressante car elle nous permet une certaine souplesse quant à l'ORM ("Object-Relational Mapping") que nous souhaitons utiliser. En effet, si nous souhaitons changer notre ORM, il nous suffit de changer le mode de fonctionnement de nos repositories seulement, nous n'avons pas besoin de changer l'intégralité du back.

Pour ce qui est de la base de données, nous avons utilisé le système de base de données (SGBD) imposé qui était HSQLDB. Celui-ci été très intéressant à utiliser car nous pouvons notre base de données était visualisable par le biais d'un .jar founi avec le SGBD. De plus, l'ajout, la suppression ou la modification des objets n'a pas été compromis par ce SGBD, nous ne retenons que du positif de ce dernier.

À présent, passons aux explications de la partie front vu par Elliott.

Applicatif Frontend

Pour cette section, nous allons vous présenter les fonctionnalités vues dans la partie backend à travers le site web. Commençons donc par la page d'accueil.



Sur la page d'accueil l'utilisateur a plusieurs choix :

- Se connecter :

Afin de se connecter l'utilisateur n'a qu'à entrer son nom dans le champ texte à côté du bouton "S'identifier" en haut à droite de la page. Son nom sera ensuite affiché en haut de la page ainsi qu'un bouton de déconnexion et un bouton d'édition de nom d'utilisateur.

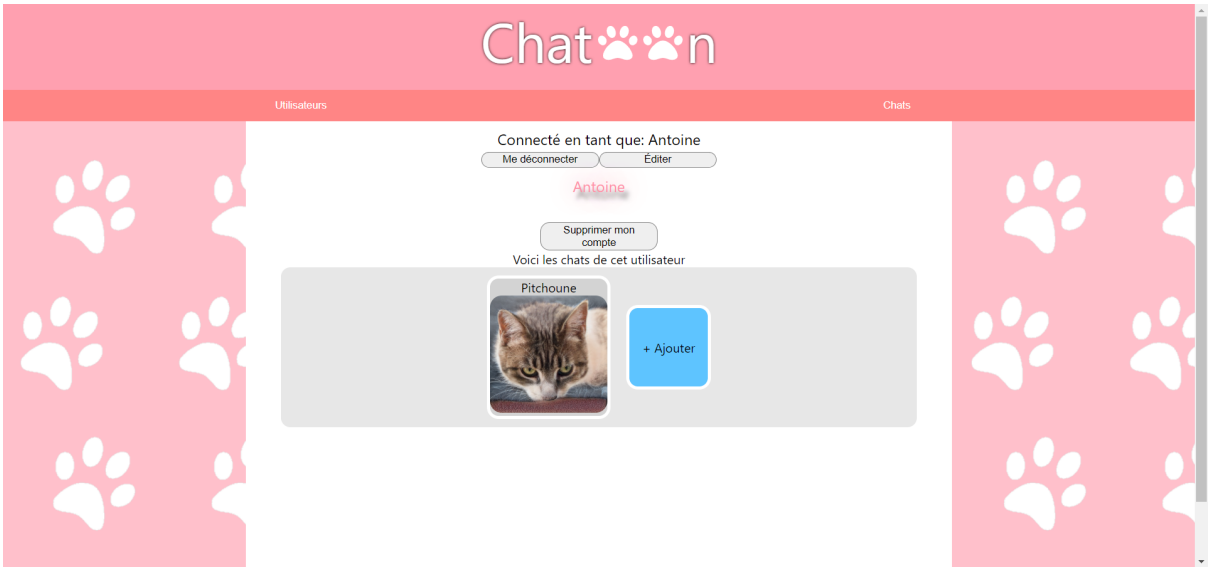
- Se créer un compte :

De la même manière que pour se connecter si l'utilisateur rentre un nom n'appartenant pas à la base de donnée, on fait une requête HTTP POST afin de créer un nouvel utilisateur. L'utilisateur est donc connecté à son nouveau compte comme vu précédemment.

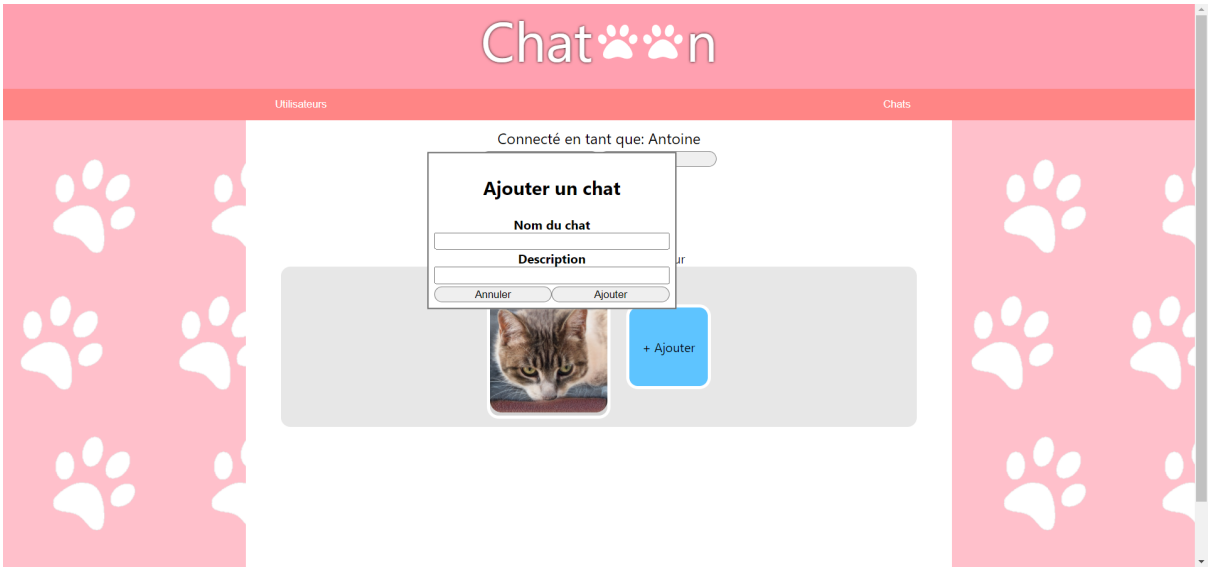
- Naviguer sur le site en tant qu'invité.

En tant qu'invité, l'utilisateur ne pourra que regarder les profils des autres utilisateurs ainsi que des chats. Il devra se connecter ou se créer un compte afin de pouvoir commenter ou créer un chat.

A partir d'ici l'utilisateur peut naviguer soit vers le profil d'un utilisateur soit vers la liste de tous les chats.

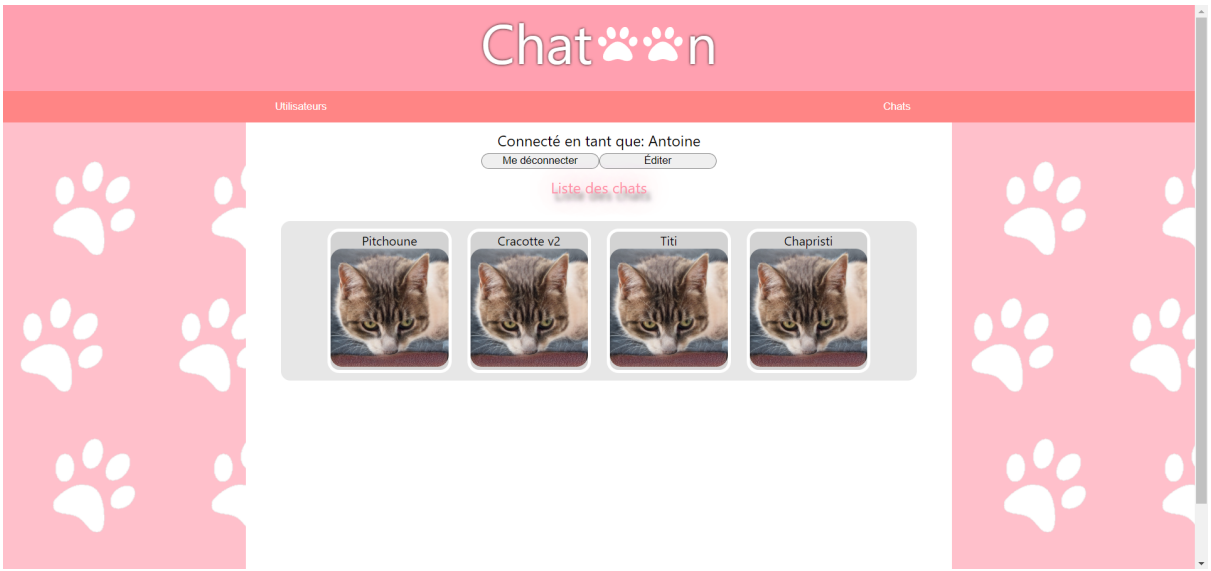


En cliquant sur son nom d'utilisateur ou sur la carte d'un compte existant, l'utilisateur arrive sur le profil correspondant. Sur cette page on affiche les chats appartenant au profil consulté ainsi que le nom du profil (en rose). S'il s'agit du compte de l'utilisateur alors il verra un bouton lui permettant de supprimer son compte et un autre lui permettant d'ajouter un chat.

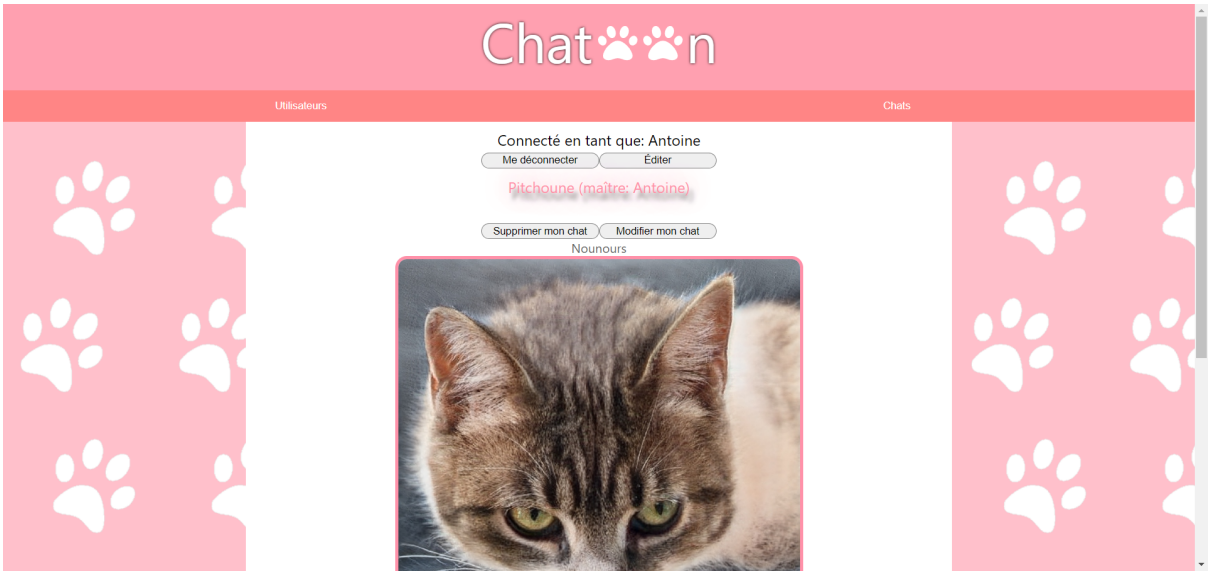


Lors de l'ajout d'un chat l'utilisateur n'a qu'à saisir le nom et la description d'un chat, puis lors de son clic sur le bouton "Ajouter" on ajoute le nouveau chat à la base de données et la page est rechargé pour afficher la nouvelle liste de chats de l'utilisateur.

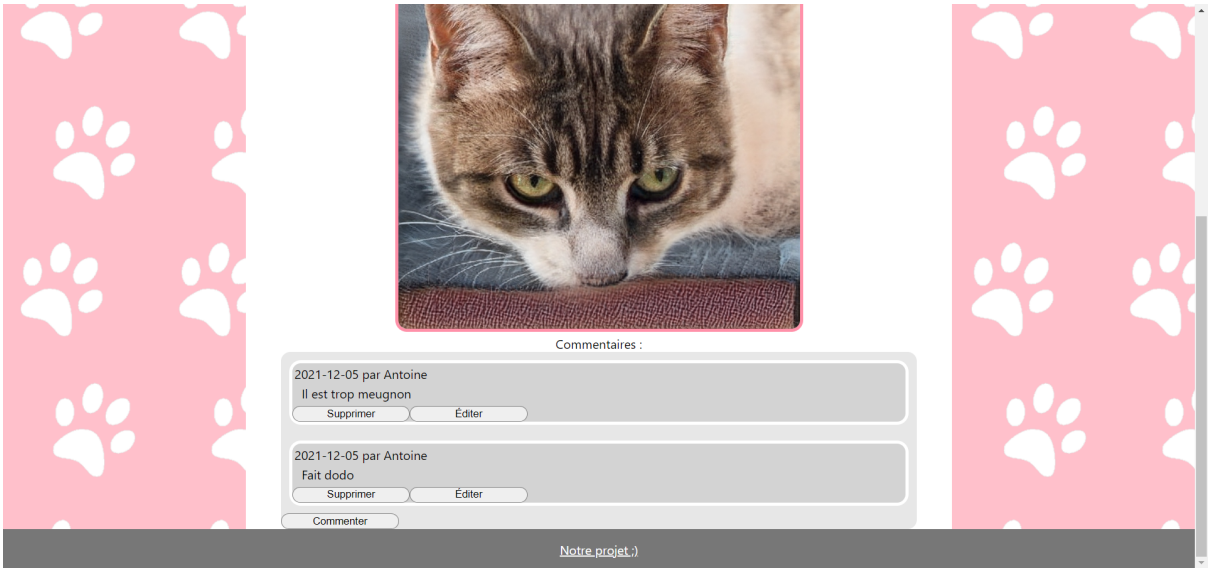
En cliquant sur le bandeau, l'utilisateur sera dirigé vers une page listant les utilisateurs ou les chats créés sur le site.



Comme depuis le profil d'un autre utilisateur, l'utilisateur actuel peut accéder au profil d'un chat en cliquant sur la carte correspondante.



Sur le profil d'un chat (et s'il s'agit de celui de l'utilisateur) on peut soit le supprimer, soit l'éditer afin de changer son nom et/ou sa description via un formulaire semblable à celui utilisé lors de sa création.



Plus bas, on peut voir les commentaires laissés par les autres utilisateurs sur le chats consulté. S'il s'agit de commentaires de l'utilisateur, il pourra alors les supprimer ou les éditer. Il peut également poster un nouveau commentaire grâce au bouton "Commenter".

Maintenant que vous savez comment le site fonctionne, libre à vous de naviguer sur ce dernier, créer des chats, poster des commentaires… N'hésitez pas à rafraîchir la page afin de voir de nouvelles photos de chats générées grâce à une AI sur le site https://thiscatdoesnotexist.com.

Difficultés rencontrées

Communes au binôme

Effectuer un travail en groupe n’est jamais chose aisée. En effet, nous ne travaillons pas forcément au même rythme et nous n'avons pas le même emploi du temps d’autant plus dans un mode de formation en alternance.

Néanmoins, nous pensons avoir réussi à trouver séparer le travail afin que chacun s’occupe d’une partie qu’il affectionne. Ainsi, malgré une mauvaise organisation temporelle, nous pensons avoir réussi à travailler, en binôme, sur ce même projet.

Côté Backend

Pour ce qui est de cette partie, Amandine a eu des problèmes d’organisation puisque n’étant pas habitué à de tel projet, il a fallu trouver par quel bout débuter celui-ci. Ce souci d’organisation a pu lui faire perdre du temps au départ, cependant, bien entouré, elle a su se recentrer sur ses objectifs et fournir un back clair fonctionnel.

Elle a également eu des problèmes de dépendances circulaires avec la base de données. En effet, au départ, le back n’avait pas découpage en service, alors elle n’arrivait pas à récupérer comme il se doit les objets Chat d’un objet Personne. Cela était dû au fait que le contrôleur possédait un objet Personne dont l’état était détaché provoquant ainsi une duplication des chats dans la liste des chats avec les commentaires (le “fetch Edge” ne répondant pas entièrement à notre problématique elle a eu des problèmes lors de la jointure des deux tables).

Côté Frontend

A la base le site devait être réalisé à partir de VueJS, mais plusieurs problèmes sont survenus lors de l'ajout des requêtes HTTP. De plus, la manière dont fonctionnent les variables et différentes méthodes de Vue sont assez difficile à assimiler pour un premier projet de cette envergure. Nous nous sommes donc tournés vers JQuery qui se rapproche beaucoup plus du JavaScript classique mais qui a l'avantage d'être plus rapide pour coder et aussi d'offrir la fonction ajax permettant de faire les requêtes HTTP nécessaire afin de communiquer avec le serveur.

Conclusion

Pour conclure sur ce projet, nous pouvons dire qu'il a été très enrichissant car il nous a permis de réaliser une application web et plus particulièrement un site web du début à la fin, du front au back. Il nous a également permis de mettre en pratique l'ensemble notions des qui nous ont été donnés en cours. Ainsi, malgré la dose de travail entre la partie back et front, nous sommes satisfait d'avoir pu rendre un site fonctionnel, ainsi que de ce que nous a apporté ce module tout au long du semestre. De plus, ce projet représente aussi pour nous l'opportunité de pouvoir recréer un projet entier de manière plus instinctive.

Comme tout projet, nous pouvons bien évidemment envisager de multiples améliorations. Celles-ci peuvent à la fois être côté back, en étant un petit peu plus puriste et en utilisant des DTO, des modèles métiers et des DAO ("Data Access Object") de manière plus directe et en faisant les conversions qui leurs sont associées. Mais elles peuvent également venir du côté front, en ajustant certaines choses grâce à certains frameworks, en prenant en compte les règles plus poussées concernant l'ergonomie des sites web ou en se basant sur l'organisme de standardisation W3C ("World Wide Web Consortium") par exemple. De plus, avec plus de temps nous aurions pu approfondir nos connaissances sur JUnit pour la réalisation des tests ou sur Swagger pour effectuer des tests sur nos requêtes HTTP et générer une documentation de notre code entre autres.