Rapport de My-Business

« Our Building Manager »

Emile Cyimena & Benoit Vankoningsloo

Ifosup 2021

Bachelier Informatique de Gestion – Ifosup – Juin 2021

Table des matières

1. Introduction 5
2. Consignes & Enoncé 6
    Consignes 6
    Enoncé choisi – Projet C : My Building 9
3. Contraintes communes demandées 10
4. Fonctionnalités 11
5. Base de données 13
6. suivi de projet / Design 14
7. Code 15
    Notifications 15
    Requêtes ajax/XHR 15
8. Routes 16
9. Points d'attention d'une fonctionnalité 20
10. Autocritique 22
10. Conclusion 23
11. Remerciements 24

1. Introduction

Pour ce projet, nous avons décidé de former un groupe de travail composé de Emile Cyimena et Benoit Vankoningsloo.

Notre choix de projet s'est porté sur la propostion « My Building » – projet qui vise à permettre la gestion de la communication entre locataires / propriétaires et syndic d'immeuble au travers d'un mini web site.

Le délai imparti pour mener à bien ce projet a été de 19 jours calendrier.

Notre collaboration pour ce projet s'est basée sur les outils suivants :

- Edi pour php : PhpStorm
- Suivi de projet : Jira
- Documentation : Confluences
- Version : GitHub
- Design : Xd adobe

Emile a drivé la cohérence de l'ensemble du code et Benoit l'ensemble du projet en complétant certaines parties du code sous les conseils d'Emile.

2. Consignes & Enoncé

Consignes

Examen 5IDV2 - Juin 2021

Chargé de cours : Benjamin Delbar
IFOSUP WAVRE

1. Projet à réaliser
2. Examen oral
3. Répartition des points
4. FAQ

Par groupe de 2 impérativement.
Vous avez le choix parmi **3 propositions de projets** (voir en annexe). Les 3 propositions comprennent des contraintes assez similaires. Il n'y a donc pas de choix stratégique avec un projet plus "facile" qu'un autre.
Je vous conseille donc de prendre le projet qui vous parle le plus.
Peu importe le projet choisi, il faut **respecter les contraintes communes** présentes dans le point ci-dessous.
Les fiches de projets comportent un texte reprenant les désirs du client (reprenant des informations pour vous aider à réaliser les fonctionnalités obligatoires et complémentaires), une liste des fonctionnalités obligatoires et une liste des fonctionnalités complémentaires.
Le projet comporte des **fonctionnalités obligatoires** et des **fonctionnalités complémentaires**, les fonctionnalités obligatoires sont, vous l'aurez deviné obligatoire, et les complémentaires vous permettront d'augmenter votre degré de maîtrise, il est vivement conseillé d'essayer d'intégrer l'une ou l'autre fonctionnalité complémentaire évidemment. Libre à vous d'ajouter d'autres fonctionnalités qui vous paraissent intéressantes et qui ne sont pas reprises, notez-les bien dans votre rapport.
Les 3 derniers cours sont majoritairement dédiés à vous laisser la possibilité de réaliser le projet "en classe". Je serai présent et disponible pour répondre à vos questions directement pendant le cours. *Comme d'habitude, vous pouvez toujours me poser vos questions en dehors des heures de cours via un message privé sur Discord ou par mail.*
Si nécessaire et à la demande de plusieurs élèves, il est possible de prendre une partie du cours pour revoir un point en particulier. Par exemple : "Revoir comment fonctionne le routeur ; expliquer le fonctionnement de git, … ".
Votre présence sur le google meet est requise pendant ces cours, **les présences seront prises au début du cours** et libre à vous de rester par la suite ou non.

Contraintes communes

Le projet doit fonctionner sur un serveur Apache, avec une base de données **mySQL** et être écrit en **PHP**.
Au niveau du code :
Vous devez utiliser une architecture comprenant des **modèles, des contrôleurs et des vues.**
Vous devez **sécuriser vos formulaires** et le traitement des données
Vous devez **protéger les données sensibles** (password) avec une sécurité "Hash + Salt" au minimum.
Vous devez utiliser des **requêtes Ajax/XHR** pour **au moins 1 fonctionnalité.**
Vous devez utiliser un **routeur** ainsi qu'un **virtual host**.
Il n'est pas obligatoire d'utiliser du javascript (à l'exception de la requête XHR) mais cela est fortement conseillé.
Un minimum d'efforts doit être apporté à la mise en page : Pas besoin de CSS, mais une structure HTML correcte, ajouter des labels à vos champs, ajouter un titre à vos formulaires.

Vous devez **faire persister l'authentification** d'un utilisateur Faites attention aux erreurs SQL, transmettez un message compréhensible pour un non-développeur lorsqu'une erreur arrive.
Faites attention aux URL, est-ce que votre projet fonctionne toujours si le virtual host à un domaine différent ?

Délivrables

Code source
Via un lien **git** (github / gitlab / au choix)
Le code source **doit contenir un dossier dump** reprenant le dump de votre base de données.
Il n'est pas obligatoire que votre code soit commenté mais il vous est demandé de nommer vos fonctions, méthodes, classes et variables le plus clairement possible.
Base de données
Le dump de la base de données doit être pré-rempli de quelques entrées pour chaque table.
Rapport
Le rapport doit être rendu au format PDF, et être envoyé en même temps que le code source.
Le rapport doit contenir
- Une page de garde avec le nom de votre projet, ainsi que vos noms et prénoms.
- La répartition du travail, qui a fait quoi ?
- Le schéma relationnel de votre base de données
- La liste des routes accessibles de votre projet sous le format standard précisé en annexe
- Une description des éventuels problèmes rencontrés et des solutions trouvées
- La liste des fonctionnalités implémentées
- *Optionnel* : Une zone libre où vous pouvez mettre en avant une fonctionnalité / un bout de code / un diagramme / ...
- Une autocritique de quelques lignes par rapport à votre travail

Echéance

Le travail est à rendre (Code source + Dump Base de données + Rapport) pour **le samedi 19 juin 2021 à 18 :00 au plus tard**.
Envoi après le 19 juin : seconde session

Ce qui est important

Une fonctionnalité développée à 100% et testée vaut mieux que 2 fonctionnalités qui ne marchent qu'à moitié.
L'important est de montrer par votre code et votre rapport que vous avez compris ce qui a été vu en classe et que **vous comprenez ce que vous avez codé**.
Il est préférable de faire un travail "moche" mais dont vous comprenez chaque ligne, plutôt qu'un travail "beau" qui est un Frankenstein de morceaux de codes que vous ne comprenez pas.

Liste des projets

Vous trouverez ci-dessous une description des 3 projets possibles. Une liste plus détaillée avec les fonctionnalités est disponible dans les documents correspondants.
Projet A : Online Class
Suite à la crise du Coronavirus, vous êtes contactés par l'IFOSUP pour créer une plateforme de cours en ligne.
Projet B : E-Score
Un site d'information sur l'esport à décider de mettre en place une plateforme pour les équipes ou elles peuvent s'inscrire et ajouter leurs résultats.
Projet C : My Building
Un syndic vous a contacté pour créer une plateforme permettant de gérer les problèmes qui surviennent dans les immeubles sous leur responsabilité.

2. Examen oral

Examen oral

Les horaires de passages seront définis ensemble le lundi 14 juin.

Vous pouvez échanger votre heure de passage avec un autre groupe, dans ce cas merci de me prévenir.

Déroulement de l'examen oral

L'examen oral dure 30 minutes (grand maximum) et se déroule en étapes :
10 minutes maximum : Présentation de votre projet, de ce que vous avez implémenter, des choix techniques que vous avez fait. Vous pouvez ici mettre en avant une fonctionnalité ou un bout de code dont vous êtes particulièrement fier ou qui est intéressant par exemple.
15 minutes maximum : Session de questions & réponses.
Exemples de question :
- *Justifier pourquoi vous avez choisi d'implémenter tel ou tel fonctionnalité*
- *Dans ton UserDAO, ta méthode store () override la méthode store() de ton DAO abstrait, pourquoi?*

Les questions sont faites en fonction de votre projet. Néanmoins certaines questions plus génériques seront tirées aux hasards parmis une liste, tel que :
- *Si le client te demande de permettre à un produit d'avoir plusieurs catégories, qu'est-ce que cela implique comme changement dans ton code ?*
- *Comment t'y prendrais-tu pour ajouter la fonctionnalité : "Des commentaires peuvent être écrit sur une feuille de match"*

5 minutes : Feedback sur le cours & formalités administratives
L'examen oral se déroulera sur Discord, avec pour vous la possibilité de partager votre écran pour plus facilement répondre aux questions / montrer une présentation / ...
Il est fortement recommandé (mais pas obligatoire) de préparer un support pour votre présentation. Sous la forme d'un PowerPoint ou autres.
Cela vous aidera à bien présenter votre projet et vous aidera aussi à anticiper les questions les plus probables.
Si vous ne disposez pas du matériel nécessaire pour l'examen oral, vous pouvez faire une demande au secrétariat pour présenter votre examen depuis l'IFOSUP.

Présence à l'examen oral

Il vous est demandé d'être présent sur discord au minimum **5 minutes avant** votre heure de passage. Profitez-en pour tester votre micro et déjà ouvrir les fichiers que vous voudriez montrer.
De mon côté
Si un retard au niveau des passages venait à s'accumuler, un message sera publié dans le canal du cours sur discord pour vous prévenir et permettre aux suivants d'arriver plus tard.
De votre côté
Il est important de me prévenir au plus tôt **via un message privé sur Discord** (et non par mail car je serai fort probablement occupé en session avec un autre élève) en cas de retard.
Dans la mesure du possible nous définirons une nouvelle heure de passage le même jour.
Un retard non prévenu, ou une absence de votre part lors de votre passage entraîne automatiquement une seconde session.

3. Répartition des points

Examen : 70%
Répartition :
- Travail : 60%
- Oral : 40%

Travaux remis pendant le cours : 30%
Répartition :
- Travail "Pokédex" MVC + Ajax : 30%
- Travail Chenil : 70%

Je vous souhaite un excellent travail, et vous remercie pour votre écoute

Description

Un syndic vous a contacté pour créer une plateforme permettant de gérer les problèmes qui surviennent dans les immeubles sous leur responsabilité.

Fonctionnalités obligatoires

- 2 rôles : syndic et résident
- Utilisateurs unique sur l'adresse e-mail
- Formulaire d'inscription de résident
- Formulaire de connection
- Création - modification - suppression - lecture d'immeubles & lecture de la liste des habitations (syndic)
- Lecture d'immeubles et de la liste des résidents par habitations
- Gestion des propriétaires d'appartements
- Création - modification - suppression - lecture de communications (syndic)
- Lecture de communications (résident)
- Permettre aux résidents de créer des "tickets", comme par exemple "Fuite d'eau dans le hall"
- Permettre au syndic de marquer les tickets comme réglé, ou en attente

+ Contraintes communes

Fonctionnalités complémentaires

- Soumettre l'acceptation de rejoindre un immeuble à la validation du syndic
- Ajouter la notion de propriétaire-résident
- Gestion des résidents d'un immeuble par le syndic
- Tout autres fonctionnalités que vous avez envie d'implémenter

Demande du client

Ceci reprend des informations brutes transmises par la personne ayant eu contact avec le client, si vous ne faites que les fonctionnalités obligatoires par exemple, il est possible que certaines informations ne s'appliquent donc pas.

C'est pour eux un portail, ils insistent donc pour qu'un utilisateur qui n'est pas connecté n'ait accès qu'au formulaire d'enregistrement et de connexion.

Le syndic aura un compte créé et modifier en base de données pour être "Syndic". Il n'a donc pas besoin d'un formulaire d'inscription.

Lors de l'inscription d'un résident, il doit choisir parmi les résidences et précise son appartement grâce à la liste des habitations disponibles dans la résidence. Plusieurs résidents peuvent habiter dans le même appartement.

Le syndic dispose donc de la liste de ses résidences, et peut les gérer. Par résidences il peut créer des "communications" visibles par les résidents. Du style "Ascenseur en panne : réparation prévue jeudi" et pouvoir y préciser un titre, une description et potentiellement une date.

Ils aimeraient aussi que les résidents puissent créer des tickets, pour prévenir directement le syndic si un problème survient. Du genre "Fuite d'eau dans le garage !", les tickets peuvent avoir un titre, une description, sont liés au résident qui le crée et ont plusieurs états.

Non traité - en attente et traité. Une fois un ticket créé, uniquement le syndic peut en changer l'état.

Ils aimeraient aussi que ce portail puisse distinguer les propriétaires des locataires et donc spécifier un propriétaire pour les appartements.

Ils aimeraient aussi, qu'à terme, il puisse spécifier que le propriétaire réside dans un appartement, pour ne pas devoir encoder le propriétaire comme un "locataire"

Des questions, des précisions, des informations complémentaires à demander au client?

Une proposition de fonctionnalités qui pourrait rendre la plateforme plus agréable ?

Contactez le client en précisant le projet My Building soit par e-mail à b.delbar@xsi.io soit via discord.

3. Contraintes communes demandées

Le projet tourne sous wamp avec setup d'un vhost

Apache : erver web.

mysql : base de données relationnelle.

php : langage de programmation du projet / version 7.4

4. Fonctionnalités

Rôles utilisateurs

On retrouve quatre rôles dans la table « rôle »

- Syndic
- Locataire
- Propriétaire non-résident
- Propriétaire résident

Chacun de ces utilisateurs est rendu unique par son adresse email

Formulaire d'inscription

Le formulaire d'inscription est unique et est accessible depuis la home page

Il permet entre autres le choix des rôles utilisateurs sauf celui de syndic.
Le rôle de syndic se fait via encodage directement dans la base de données.

Un point d'attention est apporté aux choix possibles pour le propriétaire résident : en effet, il peut choisir d'une part sa résidence et d'autre part sa ou ses propriété(s) car ce statut donne la possibilité d'avoir plusieurs propriétés tout en étant résident.

Formulaire de connexion

Le formulaire de connexion est accessible via la home page.
À tout moment, il permet de voir le statut de connexion

Validation d'un nouvel utilisateur

Nous avons placé la validation d'un utilisateur à rejoindre un appartement au niveau de la validation de l'inscription par le syndic et ceci dans un menu spécifique

Le syndic pourra valider par cette vue l'accès à l'application.

Le syndic pourra valider l'appartenance à un appartement en tant que locataire / résident et également en tant que propriétaire.

Communication par le syndic

Le syndic peut créer une communication spécifique pour chaque immeuble.

Chaque résident pourra voir la communication de son immeuble.

Une information de date de création et mise à jour est également disponible.

Système de tickets

Chaque utilisateur (locataire et résident) peut créer un ticket à destination du syndic.

Les trois statuts sont : en attente, non traité et traité ; ils sont utilisés et gérés par le syndic.

Une information de date de création et mise à jour est également disponible.

Gestion des immeubles

Les immeubles sont gérés par le syndic. Il peut les créer, les mettre à jour et les supprimer.

Une vue de la liste des immeubles appelés « résidence » est disponible avec le nombre d'appartements par résidence et le nom de la ville dans laquelle la résidence se trouve.

Gestion des appartements

Les appartements sont gérés par le syndic. Il peut les créer, les mettre à jour et les supprimer. Un immeuble doit exister pour pouvoir créer un appartement.

Une vue avec la liste des appartements par immeuble est disponible incluant le nombre de locataires s'y trouvant ainsi que le nom du propriétaire.

Liste des utilisateurs

Une vue des utilisateurs enregistrés sur l'application est disponible pour le syndic.

Le syndic peut voir la fiche et supprimer un utilisateur.

Votre compte

Chaque utilisateur qui a un accès validé par le syndic peut avoir accès à son compte.

L'utilisateur pourra créer et voir ses tickets.

L'utilisateurs pourra voir les communications liées à son immeuble

L'utilisateur peut modifier ses données personnelles hormis sa résidence et sa propriété.

Le statut de propriétaire n'a pas de vue sur les tickets ni sur les communications.

5. Base de données

Diagramme relationnel des entités

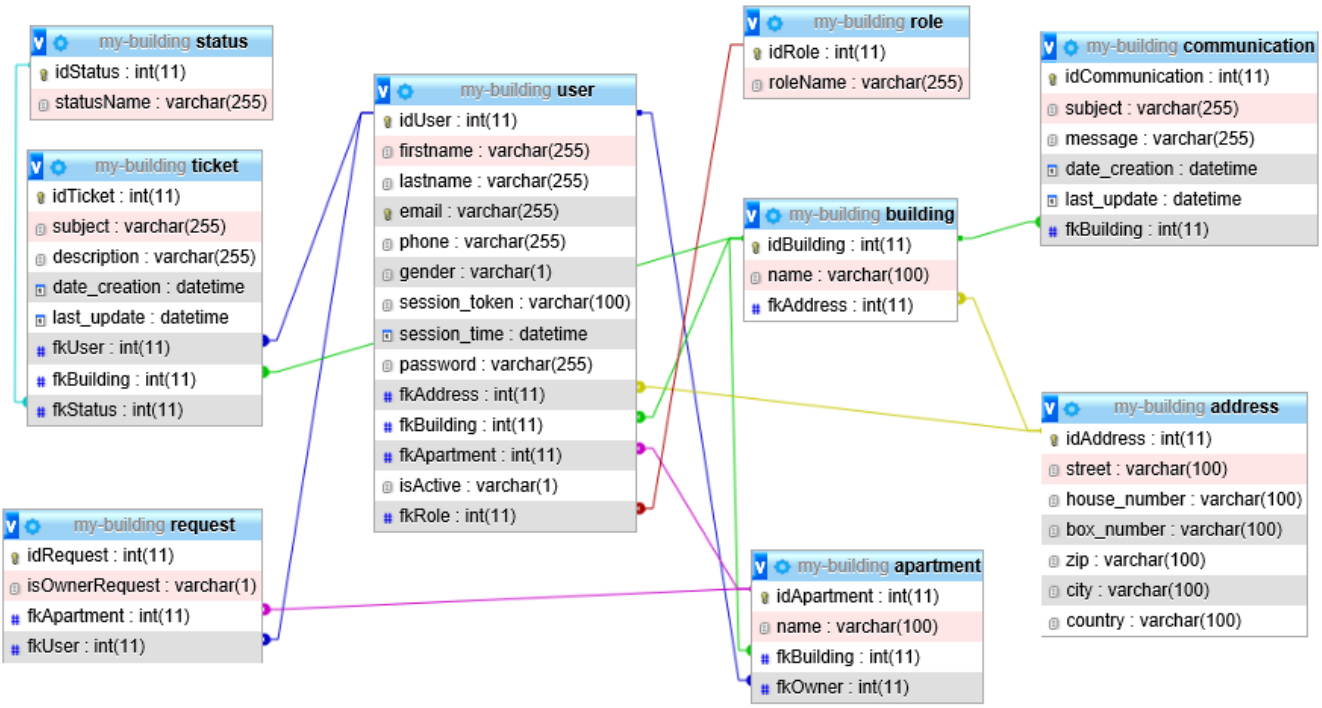
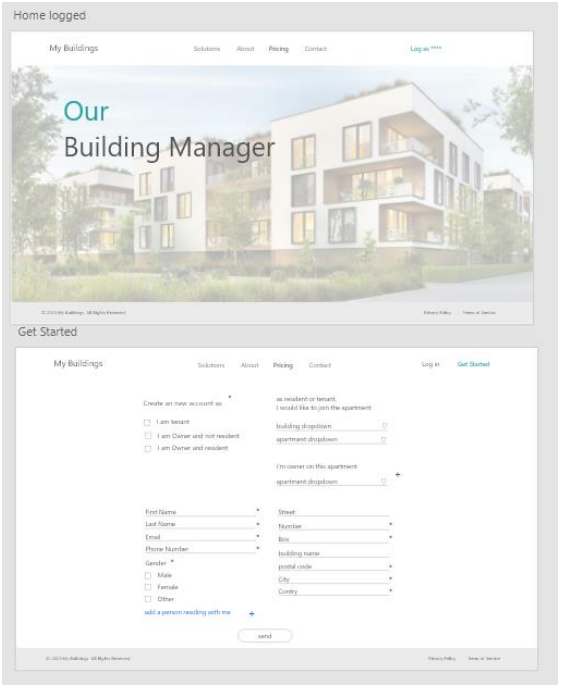
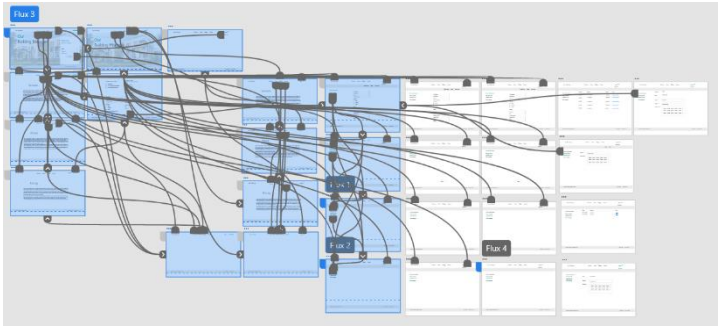


Figure 1 diagramme des relations d'entités

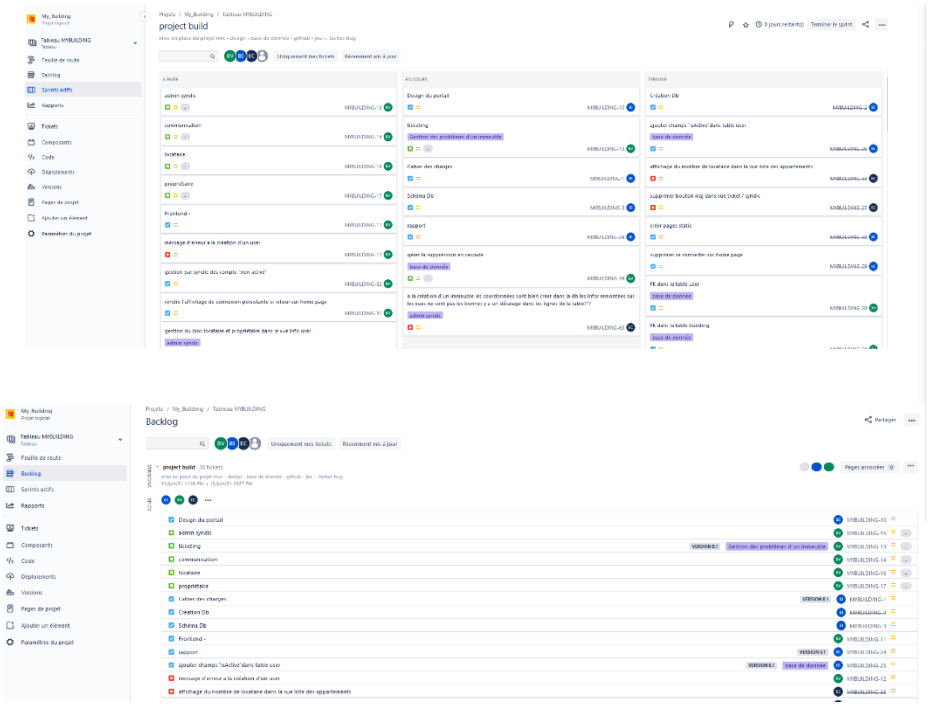
6. suivi de projet / Design

Design : xd adobe





Gestion de projet : Jira



7. Code

Notifications

Développement d'un système de notifications accessible sur toute l'application.

Lorsqu'on veut envoyer une notification on appelle cette fonction et une notification apparait sur l'écran avec le message qu'on veut faire passer.

Requêtes ajax/XHR

INSCRIPTION

- requête GET pour récupérer le menu déroulant des résidents lors de l'inscription d'un utilisateur

- requête GET pour récupérer le menu déroulant des appartements lors de l'inscription d'un utilisateur

COMPTE UTILISATEUR

- requête POST pour mettre à jour les informations d'un utilisateur

RESIDENCE

- requête POST pour mettre à jour un immeuble

- requête POST pour mettre à jour l'adresse d'un immeuble

COMMUNICATION

- requête POST pour mettre à jour une communication

TICKET

- requête POST pour mettre à jour un ticket

- requête POST pour mettre à jour le statut d'un ticket après détection du changement dans le menu déroulant

ADRESSE

- requête POST pour mettre à jour une adresse

8. Routes

Authentification

GET /

Réponse	Type	Status	Description	Condition
response : "Vue auth.login"	Text/HTML	/	Renvoie la page de connexion	Aucune condition

GET / auth/registerView

Response	Type	Status	Description
response : "Vue auth.register"	Text/HTML	/	Renvoie la page d'inscription

Communications

GET /communications

Réponse	Type	Status	Description	Condition
response : "Vue communications.list"	Text/HTML	/	Renvoie une page avec la liste des communications en fonction du rôle de l'utilisateur	Être authentifié

GET /communications/show/:id

Paramètre	Type	Status	Description	Être authentifié
id	int	Obligatoire	Id correspondant à la communication à montrer	Oui

Response	Type	Status	Description
response : "Vue communications.one"	Text/HTML	/	Réponse reçue si la requête s'est exécutée correctement. Les champs sont vides si l'utilisateur n'a pas été trouvé.

GET / communications /createView

Response	Type	Status	Description	Condition
response : "Vue communications.create-form"	Text/HTML	/	Formulaire permettant de créer une nouvelle communication.	Être authentifié et syndic.

Tickets

GET /tickets

Réponse	Type	Status	Description	Condition
response : "Vue tickets.list"	Text/HTML	/	Renvoie soit tous les tickets, si l'utilisateur est un syndic, soit les tickets de l'immeuble auquel appartient l'utilisateur.	Être authentifié

GET / tickets /show/:id

Paramètre	Type	Status	Description	Condition
id	int	Obligatoire	Id correspondant à la communication à montrer	Être authentifié

Response	Type	Status	Description
response : "Vue ticket.one"	Text/HTML	/	Réponse reçue si la requête a été exécuté sans erreur

GET / tickets /createView

Response	Type	Status	Description	Condition
response : "Vue tickets.create-form"	Text/HTML	/	Formulaire permettant de créer un nouveau ticket. Le ticket sera associé automatiquement à l'immeuble de l'utilisateur.	Être authentifié et résident.

Résidences

GET /buildings

Response	Type	Status	Description	Condition
response : "Vue tickets.list"	Text/HTML	/	Renvoie la liste de toutes les résidences de l'application.	Être syndic.

GET / buildings /show/:id

Paramètre	Type	Status	Description	Condition
id	int	Obligatoire	Id correspondant au ticket à montrer.	Être authentifié.

Response	Type	Status	Description
response : "Vue ticket.one"	Text/HTML	/	Réponse reçue lorsque la requête a été exécuté correctement.

GET / tickets /createView

Response	Type	Status	Description	Condition
response : "Vue buildings.create-form"	Text/HTML	/	Formulaire permettant de créer une nouvelle résidence.	Être authentifié et syndic.

Appartements

GET /apartments

Réponse	Type	Status	Description	Condition
response : "Vue apartments.list"	Text/HTML	/	Renvoie la liste de toutes les appartements.	Être syndic.

GET / apartments /show/:id

Paramètre	Type	Status	Description	Condition
id	int	Obligatoire	Id correspondant à l'appartement à montrer.	Être Syndic.

Response	Type	Status	Description
response : "Vue apartment.one"	Text/HTML	/	Réponse reçue lorsque la requête a été exécuté correctement

GET / apartments /createView

Response	Type	Status	Description	Condition
response : "Vue apartment.create-form"	Text/HTML	/	Formulaire permettant d'ajouter un nouvel appartement.	Être authentifié et syndic

Utilisateurs

GET /users

Réponse	Type	Status	Description	Condition
response : "Vue users.list"	Text/HTML	/	Renvoie la liste de toutes les utilisateurs.	Être syndic.

GET / users /show/:id

Paramètre	Type	Status	Description	Condition
id	int	Obligatoire	Id correspondant à l'utilisateur à montrer.	Être authentifié et syndic.

Response	Type	Status	Description
response : "Vue users.one"	Text/HTML	/	Réponse reçue lorsque la requête a été exécuté correctement

Requêtes

GET /requests

Réponse	Type	Status	Description	Condition
response : "Vue requests.list"	Text/HTML	/	Renvoie la liste des requêtes d'utilisateurs	Être authentifié et syndic.

9. Points d'attention d'une fonctionnalité

Nous sommes assez fiers de notre application, particulièrement au niveau de l'inscription qui est dynamique.

L'utilisateur de type « Propriétaire résident » peut choisir directement sa ou ses propriété(s) à l'enregistrement. En fonction de son choix, on génère un composant.

Le premier composant est un menu déroulant qui permet à l'utilisateur de sélectionner une résidence et après cette sélection, la liste des appartements de cette résidence sera affichée.

L'utilisateur devra ensuite choisir un appartement pour valider cette section du formulaire.

Si le rôle de l'utilisateur est "Propriétaire", il pourra ajouter plusieurs appartements dans sa liste de propriété.

En cliquant sur le bouton « ajouter », on peut voir visuellement la liste s'agrandir.

Ce code a été réalisé essentiellement en javascript et avec du php pour récupérer les menus déroulants.

Figure 2 ajout résidence et ajout propriété

10. Autocritique

Première problématique

Le développement des menus déroulants du formulaire d'inscription fut complexe au début, nous étions partis sur du Json pour récupérer les données et ensuite générer les menus déroulants

Cela a posé quelques problèmes mais finalement nous sommes partis sur une autre solution : Ecrire les menus déroulants dans des fichiers php et les renvoyer lors d'appels Ajax.

Seconde problématique

Ce n'était pas vraiment un problème mais il nous a fallu être assez avancé dans le projet pour pouvoir avoir une vue d'ensemble du rendu final au niveau du code. C'est seulement une fois qu'on a eu ce recul qu'on a pu refactoriser notre code convenablement, en le rendant le plus concis, lisible et propre possible.
Nous aurions aimé avoir ce résultat plus tôt mais c'est hélas compliqué sans recul.

Dans un prochain projet, nous saurons comment faire.

Troisième problématique

De manière globale nous n'avons pas eu de gros soucis au niveau du développement, juste des grains de sable dans le code qui provoquaientt des erreurs et des bugs. Mais cela fait partie du métier.

10. Conclusion

Le projet a été très enrichissant, il a été un réel défi en termes de délai et de charge de travail.

Durant le projet, nous avons toujours cherché à aller plus loin dans nos réflexions et démarches, que ce soit pour le code comme pour les fonctionnalités.

Nous sommes fiers du résultat, fiers de rendre un projet fonctionnel.

Bien entendu, nous aurions pu aller plus loin avec un peu plus de temps.

11. Remerciements

Nous remercions notre professeur Benjamin Delbar pour nous avoir supporter durant cette année académique.