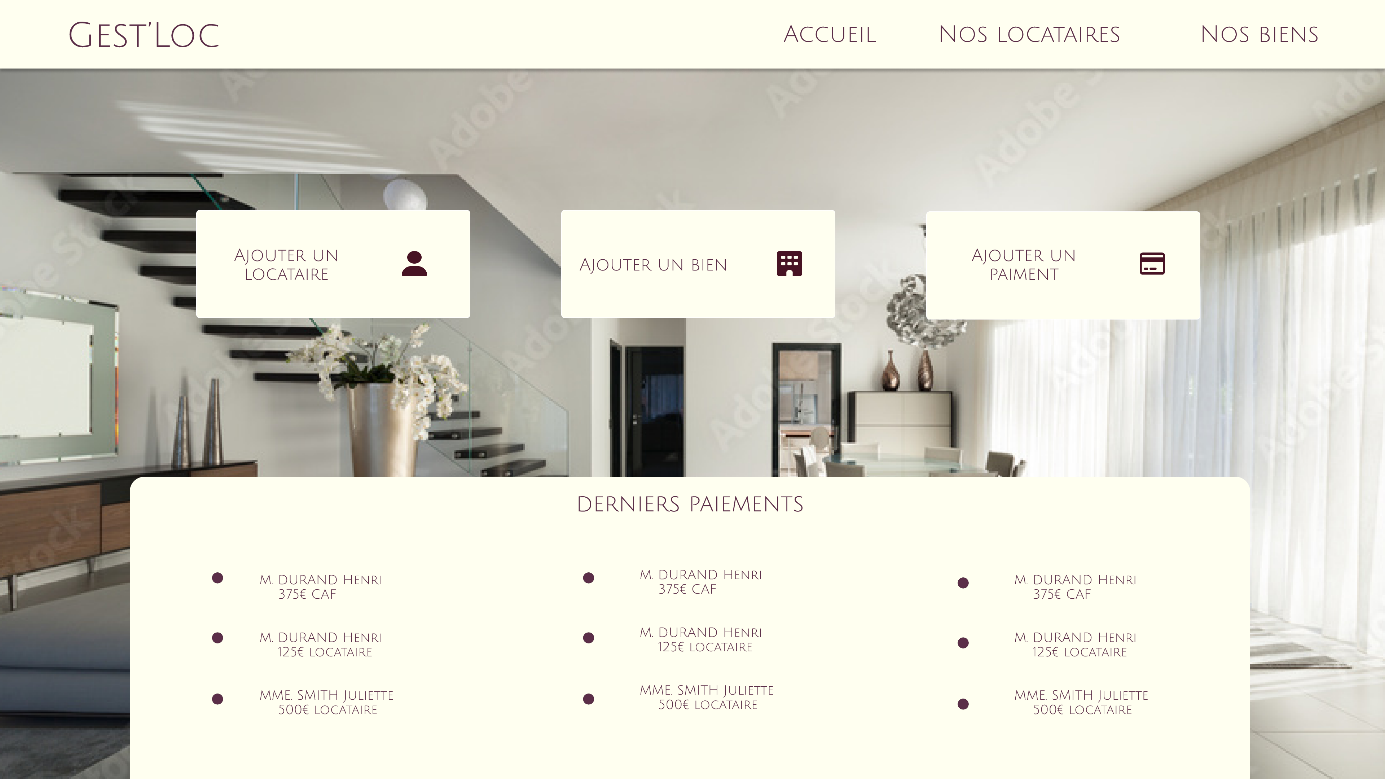
POC – Spécifications fonctionnelles et techniques du site de gestion de location



**SOMMAIRE**

[**Architecture du projet** 2](#_Toc129515875)

[**Planification de tests** 4](#_Toc129515876)

[**Informations complémentaires** 4](#_Toc129515877)

[**Technologies utilisées** 4](#_Toc129515878)

[**Spécifications de l’API** 5](#_Toc129515879)

[**Versionning** 8](#_Toc129515880)

[**Gestion de projet** 8](#_Toc129515881)

**Contexte**

Plusieurs agences immobilières ont créé un appel d’offres pour le développement d’un outil de gestion de location immobilière.

Notre entreprise va donc réaliser un POC (Proof of Concept) qui est une démonstration de faisabilité à destination du client qui doit prendre en compte tous ses besoins.

Notre but est de convaincre le client de travailler avec nous afin de réaliser l’application de manière complète, ce qui inclura la maintenance et l’évolution.

Il a été convenu que :

* Nous devons être en mesure de stocker les coordonnées du locataire
* Le locataire peut louer un ou plusieurs appartements, mais il doit verser chaque fois un dépôt de garantie lors de l’entrée.
* Un appartement possède plusieurs propriétés, dont une adresse et un complément ainsi qu’une ville et un code postal. De même, un prix des charges ainsi que du loyer et du dépôt de garantie doivent être modifiables à tout moment.
* Tous les mois, le locataire doit payer un loyer et il y a nécessité de conserver la date de paiement. Le paiement émis peut provenir de deux origines, par le locataire et/ou par la caisse d’allocation familiale (car le propriétaire peut décider de recevoir directement les APL du locataire).
* Un état des lieux est obligatoire lors de l’entrée ou la sortie du locataire de l’appartement, la date ainsi que les remarques observées doivent être conservées.
* Il est nécessaire de connaitre le solde du locataire à tout moment afin de lui gérer facilement sa sortie.
* Les agences prélèvent des frais de 8% sur les loyers uniquement des appartements qu’elles gèrent

# **Architecture du projet**

Le client souhaite une interface intuitive, simple et agréable permettant d’effectuer un CRUD sur les appartements ainsi que leur gestion. Les fonctionnalités seront les suivantes :

* Ajouter / modifier / supprimer un appartement ainsi qu’un état des lieux
* Ajouter un locataire et changer ses informations
* Affecter le locataire à un appartement et gérer le paiement de son dépôt de garantie
* Ajouter / modifier / supprimer les paiements des locataires concernant les loyers
* Génération de la quittance de loyer sur une période donnée. Cette génération sera effective que si le locataire est en règle des loyers sur la période.
* Faire un bilan des comptes des loyers émis par les locataires lorsqu’il quitte le logement.

Bien que non demandée par le client, la possibilité de supprimer un locataire sera implémentée puisque selon la loi sur les libertés informatiques, un locataire est en droit de demander la suppression de ses informations personnelles.

L’application web sera composé de 7 pages :

* Une page d’accueil montrant les derniers paiements reçus par l’agence et trois liens vers les pages d’ajout d’un locataire, d’ajout d’un bien et d’ajout d’un paiement
* Une page « locataires » affichant la liste des locataires enregistrés dans l’agence.
* Une page « locataire » affichant tous les détails d’un locataire, ses coordonnées ainsi que les biens qu’il loue et son solde actuel.
* Une page « biens » affichant la liste des logements enregistrés dans l’agence.
* Une page « logement » affichant tous les détails d’un logement, son adresse, le coût du loyer, des charges et de la caution ainsi que les états des lieux associés.
* Une page « locations » affichant toutes les locations en cours, le nom et prénom du locataire, l’adresse du bien loué ainsi que l’état de la caution (déposée ou non).
* Une page de modification avec un formulaire qui s’adapte à la modification demandée (locataire, location, paiement, logement et état des lieux)

# **Planification de tests**

La rédaction d’un plan de test fonctionnel et la mise en place d’un TDD (Test Driven Development) est attendue, ce qui forcera l’application à répondre à un certain niveau d’exigence.

# **Informations complémentaires**

La page biens

Cette page devra contenir les informations essentielles du logement ainsi qu’une photo et son état (disponible ou non disponible) permettant une vue d’ensemble et une rapidité de recherche de l’information.

La page logement

Pour des raisons de lisibilité et pour ne pas encombrer la page, seuls les quatre derniers états des lieux seront affichés.

La page locataire

Pour des raisons de lisibilité et pour ne pas encombrer la page, seuls les paiements des trois derniers mois seront affichés dans la partie « solde ».

# **Technologies utilisées**

Le backend sera réalisé en Java avec son framework Spring Boot. La librairie IText sera utilisée pour la génération de documents PDF.

Le frontend quant à lui sera réalisé en Javascript avec son framework React.

Enfin, la base de données choisie pour le projet est PostgreSQL. Pour le développement de PoC, l’hébergement sera effectué via le site alwaysData.

# **Spécifications de l’API**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Point d'accès | Request body | Type de réponse attendue | Fonction |
| GET | /api/property |  | Array of properties | Renvoie un tableau de tous les logements de la base de données |
| GET | /api/property/:id |  | Single property | Renvoie le logement associé à l'id fourni |
| POST | /api/property | { address : String, additionnal : String, postal\_code : String, city : String, rent : String, charges : String, deposit : String } | { message : String } | Récupère les données du nouveau logement et les enregistre dans la base de données en générant un id. Initialise l'état du bien comme étant disponible à la location (available : true) |
| PUT | /api/property/:id | { address : String, additionnal : String, postal\_code : String, city : String, rent : String, charges : String, deposit : String, available : Boolean } | { message : String } | Met à jour le logement associé à l'id fourni |
| DELETE | /api/property/:id |  | { message : String } | Supprime le logement associé à l'id fourni |
| GET | /api/tenant |  | Array of tenants | Renvoie un tableau de tous les locataires de la base de données |
| GET | /api/tenant/:id |  | Single tenant | Renvoie le locataire associé à l'id fourni |
| POST | /api/tenant | { firstname : String, lastname : String, birthday : String, phone : String, email : String } | { message : String } | Récupère les données du nouveau locataire et les enregistre dans la base de données en générant un id. |
| PUT | /api/tenant/:id | { firstname : String, lastname : String, birthday : String, phone : String, email : String } | { message : String } | Met à jour le locataire associé à l'id fourni |
| DELETE | /api/tenant/:id |  | { message : String } | Supprime le locataire associé à l'id fourni |
| GET | /api/rental |  | Array of rentals | Renvoie un tableau de toutes les locations de la base de données |
| GET | /api/rental/:id |  | Single rental | Renvoie la location associée à l'id fourni |
| POST | /api/rental | { property : Number, tenant : Number, deposit : Boolean } | { message : String } | Récupère les données de la nouvelle location et les enregistre dans la base de données en générant un id. |
| PUT | /api/rental/:id | { deposit : Boolean } | { message : String } | Met à jour la location associée à l'id fourni |
| DELETE | /api/rental/:id |  | { message : String } | Supprime la location associée à l'id fourni |
| GET | /api/payment |  | Array of payments | Renvoie un tableau des paiements effectués sur les trois derniers mois depuis la base de données. |
| GET | /api/payment/:id |  | Single payment | Renvoie le paiement associé à l'id fourni |
| POST | /api/payment | { rental : Number, date : String, origin : String, amount : String } | { message : String } | Récupère les données du nouveau paiement et les enregistre dans la base de données en générant un id. |
| PUT | /api/payment/:id | { date : String, origin : String, amount : String } | { message : String } | Met à jour le paiement associé à l'id fourni |
| DELETE | /api/payment/:id |  | { message : String } | Supprime le paiement associé à l'id fourni |
| GET | /api/inventory |  | Array of inventories | Renvoie un tableau des états des lieux effectués sur les trois derniers mois depuis la base de données. |
| GET | /api/inventory/:id |  | Array of inventories | Renvoie les états des lieux associés à l'id du logement |
| POST | /api/inventory | { property : Number, status : String, date, String, notes : String } | { message : String } | Récupère les données du nouvel état des lieux et les enregistre dans la base de données en générant un id. |
| PUT | /api/inventory/:id | { status : String, date, String, notes : String } | { message : String } | Met à jour l'état des lieux associé à l'id fourni |
| DELETE | /api/inventory/:id |  | { message : String } | Supprime l'état des lieux associé à l'id fourni |

# **Versionning**

Le projet sera versionné sous Github. L’architecture des branches sera la suivantes : deux branches permanentes « main » et « develop » auxquelles seront ajoutées trois types de branches temporaires « feature », « release » et « hotfix ».   
Pour le développement du PoC, l’utilisation des features et des hotfix n’est pas attendue.

# **Gestion de projet**

Le suivi du projet sera fait grâce à Jira.

Une méthodologie SCRUM a été choisie avec des sprints de deux semaines afin de permettre le développement de toutes les fonctionnalités frontend et backend par un seul développeur.

Le sprint 0 décrira les diverses étapes de la préparation du projet et durera quatre semaines afin de mettre en place toutes les spécificités du projet et préparer au mieux la phase de développement.