Titre : Dev Web / Web Mobile (backend) – Promo : P23

API demande de prêt immobilier

*Pitch : la néo banque Ponionz développe son MVP. Elle souhaite proposer des prêts immobiliers aux particuliers. A cette fin, l’application doit disposer d’un formulaire de demande prêt. La partie frontend est développée au Groenland, la partie backend dans les locaux à Paris. L’équipe frontend a déjà bien avancé et a pu envoyer le wireframe de l’écran aux co-fondateurs qui ne s’en sont toujours pas remis. A toi de jouer si tu aimes le game ! (Mode start-up off, passons aux choses sérieuses).*



**Objectifs** : concevoir et implémenter la partie backend pour la persistance en base de données des demandes de prêts.

**Environnement** : un cerveau frais, un IDE, Java 8 ou 11, Spring boot 2.4.x\*, Postman, MySQL et Google.

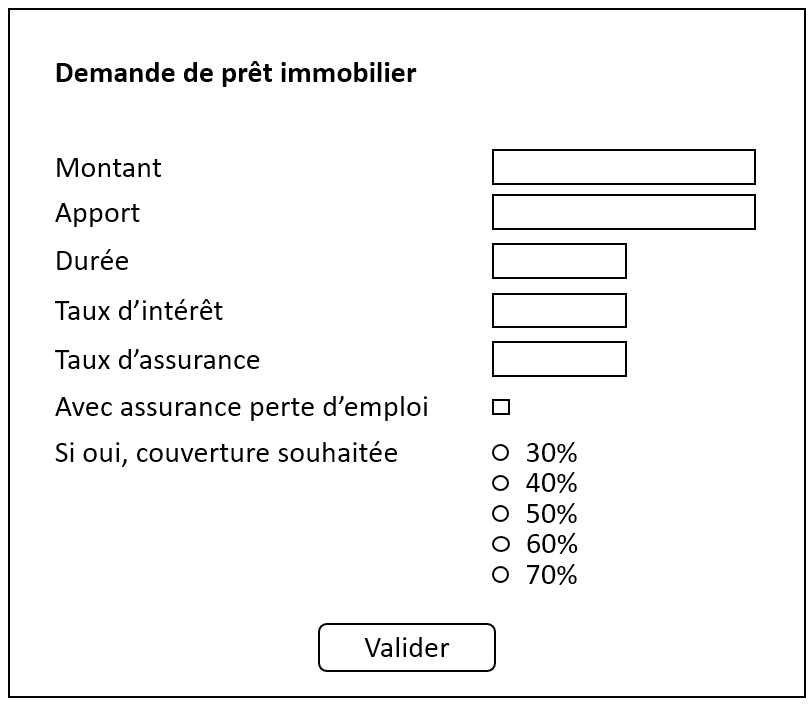
*\* Spring boot 2.5.0 vient tout juste de sortir, ne pas prendre de risque dans le cadre de cette évaluation*

**Rendu** : nouveau projet nommé « ponionz » dans un repository GitHub.

# Spécifications fonctionnelles

Une demande de prêt immobilier est composée d’un montant, d’un apport, d’une durée, d’un taux d’intérêt, d’un taux d’assurance, et d’une couverture perte d’emploi.

## Wireframe



## Règles de validation

Même si une validation des données saisies doit être implémentée dans la partie frontend, le backend doit procéder à la vérification de toutes les contraintes, détaillées ci-dessous, afin de garantir la qualité et la cohérence des demandes de prêts.

* **Montant** : obligatoire, entier, min. 100 000, max. 600 000
* **Apport**: facultatif, si renseigné doit être positif et max. 50% du montant emprunté
* **Durée** : obligatoire, en années pleines, min. 2, max. 30
* **Taux d’intérêt** : obligatoire, décimal positif
* **Taux d’assurance** : facultatif, si renseigné décimal positif
* **Assurance perte d’emploi** : une case à cocher, si pas cochée signifie que le demandeur ne veut pas se couvrir, si cochée alors le demandeur souhaite se couvrir
* **Couverture** : obligatoire si la case perte d’emploi est cochée, un seul choix possible parmi 5 taux (30, 40, 50, 60 ou 70%)

## Règles de gestion complémentaires

Si une demande passe la validation avec succès, nous souhaitons que chaque demande soit également datée. Il faut sauvegarder en base, pour chaque demande, la date et l’heure. Ce n’est pas l’utilisateur qui saisie cette information, elle est automatiquement ajoutée par l’application.

Enfin, nous souhaitons que les taux d’intérêt et d’assurance soit arrondis à 2 décimales avant la sauvegarde en base de données.

# Consignes et contraintes techniques

La conception et l’implémentation de l’API respectent les bonnes pratiques et les conventions. Notamment :

* Anglais
* Nommage
* Formatage du code
* Séparation des responsabilités
* Typage des données
* Tester différentes valeurs avant de croire que ça marche

Exceptionnellement, les tables peuvent être gérées directement par l’ORM et non par script SQL. Le schéma doit s’appeler « ponionz » comme le projet de l’API.

Ne pas chercher à tout faire d’un coup, procéder par étapes. Ne pas mettre le nez dans le code trop tôt. Ré-flé-chir ! La validation est prioritaire sur les règles de gestion complémentaires. Faire marcher son API petit à petit, tester régulièrement pour valider chaque étape de la réalisation.

