Conduite de projet et choix techniques

Master MIAGE 2 – UPJV

Théo BONTEMPS
Dorian DESCAMPS
Maël RHUIN

Sommaire

I. Introduction

Pour rappel, le département MIAGE de l'Université de Picardie Jules Verne (UPJV) est demandeur sur la mise en place d'une plateforme DevOps pour la gestion des projets des étudiants. Ce projet est conçu et réalisé par notre équipe d'étudiants en Master MIAGE 2, en collaboration avec les enseignants encadrants. Madame Lapujade, cliente et future utilisatrice de la plateforme nous a expressément exprimé ses besoins lors de différentes réunions tout en nous présentant l'existant.

Actuellement, les projets des étudiants des Master MIAGE sont gérés de manière semi-manuelle. Cela ne répond pas aux cas spécifiques, tels que le passage en Master 2, ou la division d'une équipe. D'autre part, le système actuel développé par un ancien étudiant du Master MIAGE est une solution peu robuste qui nécessite une utilisation rigoureuse. La gestion s'effectue ainsi :

1. L'import d'un fichier Excel contenant les informations sur les équipes et les étudiants qui doit être structuré de manière rigoureuse et sans erreurs pour garantir la réussite de l'import. Toute anomalie dans la formation des données engendre des erreurs dans la création des équipes et des dépôts.
2. Des scripts sont ensuite exécutés pour créer les dépôts selon les informations importées. Pendant cette phase, une attention continue aux logs est requise pour s'assurer que le script s'exécute correctement et pour intervenir en cas d'anomalie.
3. Quand un étudiant passe en Master 2, il a déjà un projet associé à l'organisation GitHub à la suite de sa première année. Son dépôt pour la nouvelle année doit faire l'objet d'une configuration complètement manuelle car le script d'import actuel ne prend pas en charge la création de projets pour les étudiants déjà présents dans l'organisation.

Ces procédures imposent une charge de travail et un temps considérable pour les administrateurs. Les principales limitations de cette méthode incluent une solution peu modulaire, des scripts peu robustes pour une gestion durable, ainsi qu'une absence de solution intégrée pour la gestion des archivages.

Les administrateurs et enseignants qui utilisant GitHub, n'ont pas d'interface dédiée afin de simplifier la gestion des promotions et des projets de manière centralisée. De plus, l'absence de suivi des contributions et de supervision des projets rend difficile l'évaluation des projets d'étudiants sans passer par une période d'apprentissage de la plateforme GitHub, notamment pour les enseignants non-initiés à cet outil.

II. Cahier des charges

La mise en place de cette plateforme a pour ambition de répondre à tous les besoins actuels, à savoir simplifier et fiabiliser la mise en place des dépôts GitHub pour les étudiants du Master MIAGE. Nous devons donc mettre en place des traitements de masse pour toutes les actions répétitives, et laisser la possibilité d'effectuer ces actions manuellement. Un cas concret est que nous avons un import en début d'année effectué par un administrateur afin d'instancier une liste d'étudiants dans l'applicatif, mais il peut y avoir un nouvel étudiant qui arrive en cours d'année.

Le client nous a donc demandé de trouver des solutions afin de simplifier toutes ces démarches. L'objectif final est donc d'avoir une plateforme qui centralise au maximum les actions en rapport avec la supervision et la gestion des dépôts GitHub. Les objectifs principaux de ce projet sont :

- L'automatisation des processus répétitifs
    - Traitement de masse pour l'import d'étudiants
    - Génération et configuration automatiques des projets
    - Archivage des dépôts

- La prise en charge des cas spécifiques
    - Un nouvel étudiant arrive en cours d'année
    - Un étudiant quitte le Master
    - Un groupe d'étudiants se sépare
    - Possibilité de modifier manuellement les données lorsque nécessaire

- Interface utilisateur intuitive
    - Une interface claire et hiérarchisée
    - Une interface adaptée aux besoins spécifiques de chaque type d'utilisateur (étudiants, enseignants et administrateurs)

- Supervision et évaluation des projets
    - Tableaux de bord pour visualiser les statistiques des projets facilitant la notation des projets par le biais de statistiques détaillées (pull requests, commits, etc. ...)

    - Centralisation des commentaires

- Fiabilité et sécurité
    - Mise en place de sécurités pour l'authentification (hachage salé des mots de passe, jetons d'authentification, etc. ...)
    - Services conteneurisés

- Valeur ajoutée pour les utilisateurs
    - Pour les administrateurs
        - Réduction de la charge de travail grâce à l'automatisation des processus
        - Suivi centralisé des dépôts, promotions et utilisateurs
        - Outils avancés de gestion
    - Pour les enseignants
        - Accès facilité aux projets des étudiants
        - Statistiques détaillées sur les contributions
        - Centralisation des commentaires
    - Pour les étudiants
        - Accès facilité à leurs projets
        - Statistiques détaillées sur leurs contributions

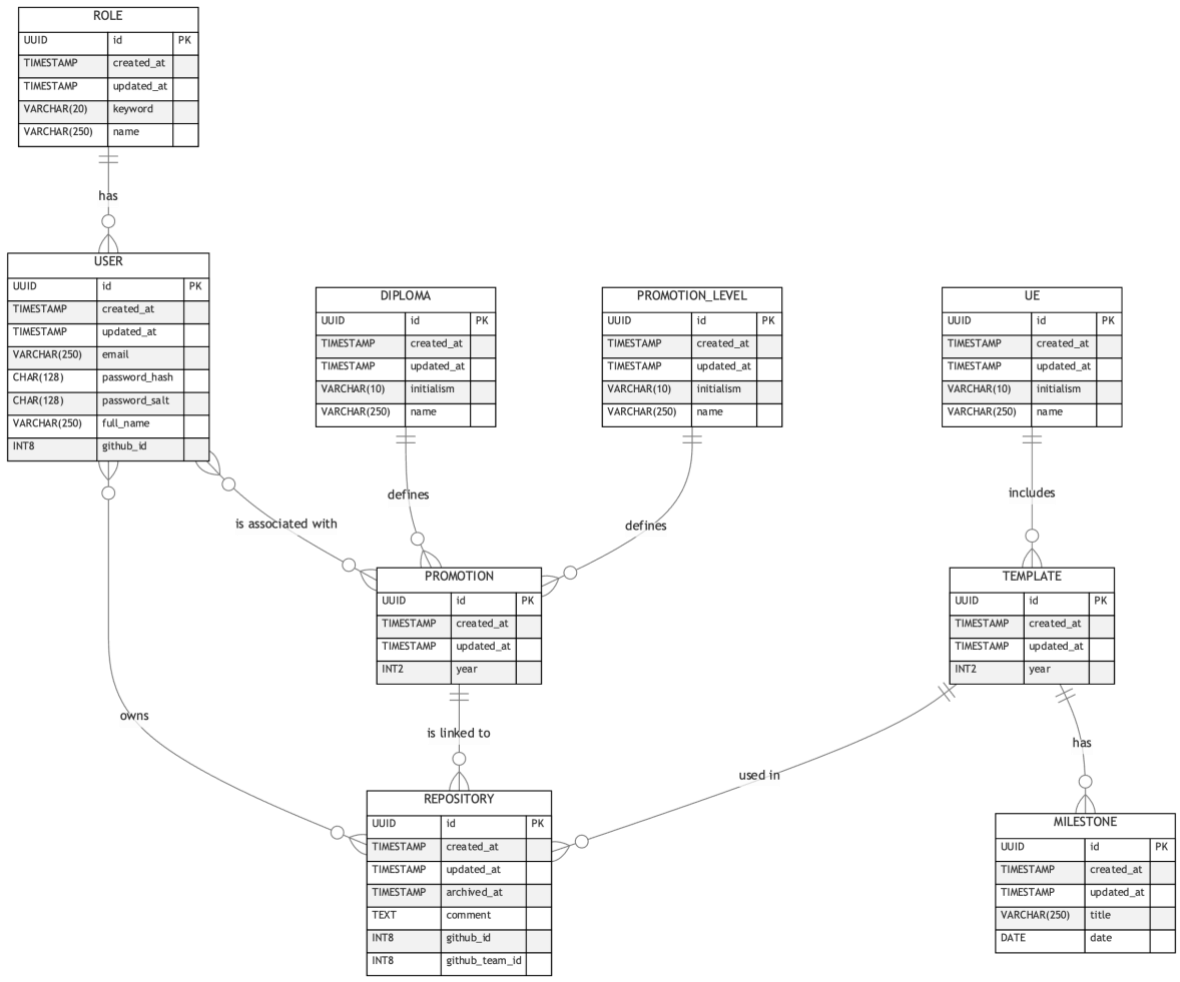
III. Modèle de données

A. Modèle Conceptuel de Données

Le MCD suivant s'articule autour de la table « repository » qui correspond aux dépôts des étudiants. Chaque entité a une table qui lui ai dédié afin de rendre le projet le plus évolutif possible.

Par exemple, le besoin n'a pas été exprimé de s'adapter à plusieurs filières, pourtant le modèle possède une table « diploma » (diplôme) qui pourra être modifiée à l'aide d'un explorateur de SGBD tel que DBeaver et rendra alors immédiatement le site compatible avec les filières ajoutées.

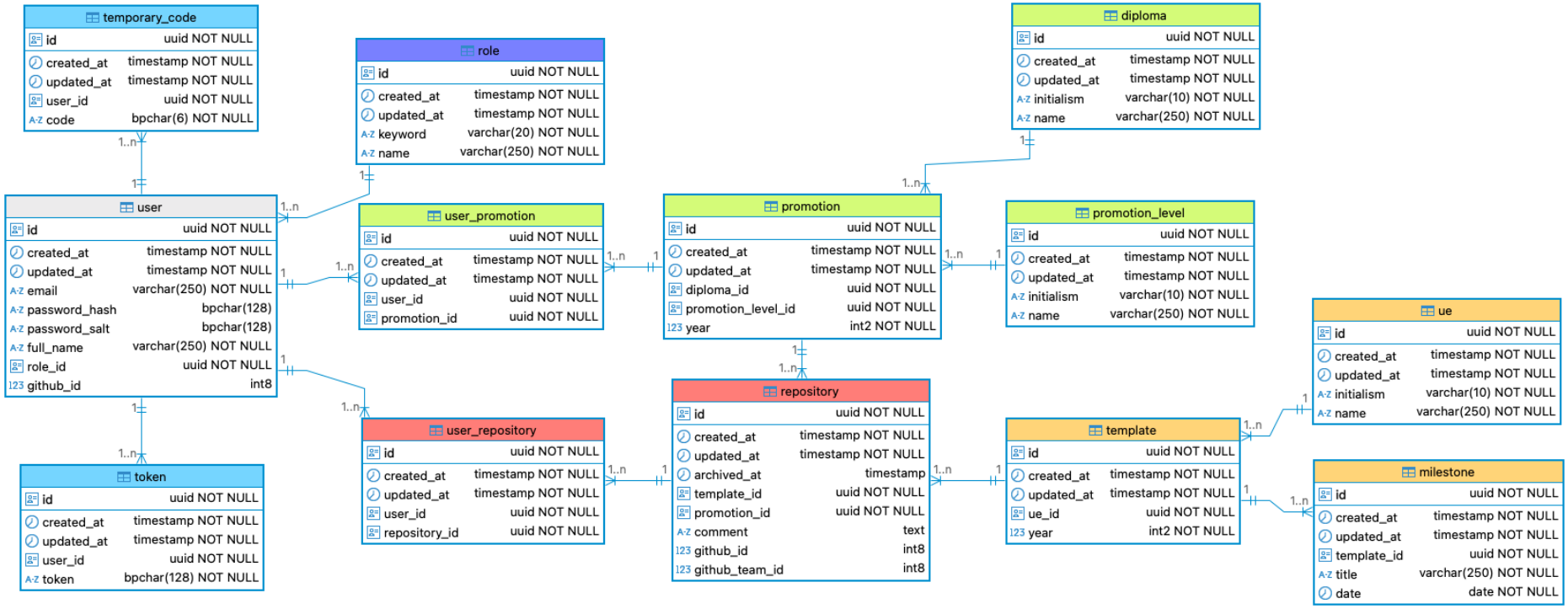
Enfin, chaque table possède un champ « created_at » et « updated_at » qui permettra d'enregistrer et d'afficher respectivement les dates de création et de modification de chaque entité.



B. Modèle Logique de Données

Ce MLD nous permet de mettre en lumière les tables techniques effacées du MCD.

On remarque les tables « temporary_code » (code temporaire) et « token » (jeton). Ces tables sont toutes deux liés au processus de connexion au site. Les tables permettant les liaisons N-N sont également visibles.



C. Dictionnaire de données

Le tableau suivant détaille les types et tailles des données ainsi que leurs règles de gestion associées, de manière exhaustive.

Colonne	Désignation	Type	Taille	Règles de gestion
role.id	Identifiant unique du rôle	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
role.created_at	Date de création du rôle	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
role.updated_at	Date de mise à jour du rôle	Timestamp	-	Champ obligatoire, mis à jour automatiquement lors de la modification.
role.keyword	Mot-clé du rôle	Varchar	20	Champ obligatoire, valeur unique. Doit respecter l'expression régulière ^[a-z]+$.
role.name	Nom complet du rôle	Varchar	250	Champ obligatoire, valeur unique, ne peut pas être vide.
user.id	Identifiant unique de l'utilisateur	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
user.created_at	Date de création de l'utilisateur	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
user.updated_at	Date de mise à jour de l'utilisateur	Timestamp	-	Champ obligatoire, mis à jour automatiquement.
user.email	Adresse email de l'utilisateur	Varchar	250	Champ obligatoire, valeur unique. Doit correspondre à un email se terminant par @u-picardie.fr ou @etud.u-picardie.fr.
user.password_hash	Hash du mot de passe de l'utilisateur	Char	128	Optionnel, NULL par défaut, valeur unique si non nulle. NULL si user.password_salt est NULL. Obligatoire si user.password_salt est non NULL. Doit respecter l'expression ^[a-f0-9]*$.
user.password_salt	Sel utilisé pour le mot de passe	Char	128	Optionnel, NULL par défaut, valeur unique si non nulle. NULL si user.password_hash est NULL. Obligatoire si user.password_hash est non NULL. Doit respecter l'expression ^[a-f0-9]*$.
user.full_name	Nom complet de l'utilisateur	Varchar	250	Champ obligatoire, ne peut pas être vide.
user.role_id	Référence au rôle de l'utilisateur	UUID	-	Champ obligatoire, doit exister dans role.
user.github_id	Identifiant GitHub de l'utilisateur	Int8	-	Optionnel, NULL par défaut, valeur unique si non nulle. Doit être >= 1.
temporary_code.id	Identifiant unique du code	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
temporary_code.created_at	Date de création du code	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
temporary_code.updated_at	Date de mise à jour du code	Timestamp	-	Champ obligatoire, mis à jour automatiquement.
temporary_code.user_id	Référence à l'utilisateur	UUID	-	Champ obligatoire, doit exister dans user, suppression en cascade si l'utilisateur est supprimé.
temporary_code.code	Code temporaire	Char	6	Champ obligatoire, longueur fixe à 6 caractères. Doit respecter l'expression ^[0-9]*$.
token.id	Identifiant unique du jeton	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
token.created_at	Date de création du jeton	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
token.updated_at	Date de mise à jour du jeton	Timestamp	-	Champ obligatoire, mis à jour automatiquement lors de la modification.
token.user_id	Référence à l'utilisateur	UUID	-	Champ obligatoire, doit exister dans user, suppression en cascade si l'utilisateur est supprimé.
token.token	Jeton d'authentification	Char	128	Champ obligatoire, unique. Doit respecter l'expression ^[a-f0-9]*$.
diploma.id	Identifiant unique du diplôme	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
diploma.created_at	Date de création du diplôme	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
diploma.updated_at	Date de mise à jour du diplôme	Timestamp	-	Champ obligatoire, mis à jour automatiquement lors de la modification.
diploma.initialism	Sigle du diplôme	Varchar	10	Champ obligatoire, unique. Doit respecter l'expression ^[A-Z0-9_]+$.

diploma.name	Nom du diplôme	Varchar	250	Champ obligatoire, unique, ne peut pas être vide.
promotion_level.id	Identifiant unique du niveau	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
promotion_level.created_at	Date de création du niveau	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
promotion_level.updated_at	Date de mise à jour du niveau	Timestamp	-	Champ obligatoire, mis à jour automatiquement.
promotion_level.initialism	Sigle du niveau	Varchar	10	Champ obligatoire, unique. Doit respecter l'expression ^[A-Z0-9_]+$.
promotion_level.name	Nom du niveau	Varchar	250	Champ obligatoire, unique, ne peut pas être vide.
promotion.id	Identifiant unique de la promotion	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
promotion.created_at	Date de création de la promotion	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
promotion.updated_at	Date de mise à jour de la promotion	Timestamp	-	Champ obligatoire, mis à jour automatiquement.
promotion.diploma_id	Référence au diplôme	UUID	-	Champ obligatoire, doit exister dans diploma.
promotion.promotion_level_id	Référence au niveau de promotion	UUID	-	Champ obligatoire, doit exister dans promotion_level.
promotion.year	Année de la promotion	Int2	-	Champ obligatoire, entre 2000 et 2099, valeur par défaut l'année actuelle.
user_promotion.id	Identifiant unique de l'association	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
user_promotion.created_at	Date de création de l'association	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
user_promotion.updated_at	Date de mise à jour de l'association	Timestamp	-	Champ obligatoire, mis à jour automatiquement.
user_promotion.user_id	Référence à l'utilisateur	UUID	-	Champ obligatoire, doit exister dans user. L'utilisateur doit être un étudiant. Suppression en cascade si l'utilisateur est supprimé.
user_promotion.promotion_id	Référence à la promotion	UUID	-	Champ obligatoire, doit exister dans promotion. Suppression en cascade si la promotion est supprimée.
ue.id	Identifiant unique de l'UE	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
ue.created_at	Date de création de l'UE	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
ue.updated_at	Date de mise à jour de l'UE	Timestamp	-	Champ obligatoire, mis à jour automatiquement.
ue.initialism	Sigle de l'UE	Varchar	10	Champ obligatoire, unique, doit respecter l'expression ^[A-Z0-9_]+$.
ue.name	Nom de l'UE	Varchar	250	Champ obligatoire, unique, ne peut pas être vide.
template.id	Identifiant unique du template	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
template.created_at	Date de création du template	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
template.updated_at	Date de mise à jour du template	Timestamp	-	Champ obligatoire, mis à jour automatiquement.
template.ue_id	Référence à l'UE	UUID	-	Champ obligatoire, doit exister dans ue.
template.year	Année du template	Int2	-	Champ obligatoire, entre 2000 et 2099, par défaut année actuelle.
repository.id	Identifiant unique du dépôt	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()). Ne peut pas être modifié si repository.archived_at est non NULL.
repository.created_at	Date de création du dépôt	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle. Ne peut pas être modifié si repository.archived_at est non NULL.
repository.updated_at	Date de mise à jour du dépôt	Timestamp	-	Champ obligatoire, mis à jour automatiquement. Ne peut pas être modifié si repository.archived_at est non NULL.
repository.archived_at	Date d'archivage du dépôt	Timestamp	-	Optionnel, null par défaut.

repository.template_id	Référence au template	UUID	-	Champ obligatoire, doit exister dans template. Ne peut pas être modifié si repository.archived_at est non NULL.
repository.promotion_id	Référence à la promotion	UUID	-	Champ obligatoire, doit exister dans promotion. Ne peut pas être modifié si repository.archived_at est non NULL.
repository.comment	Commentaires des enseignants	Text	-	Optionnel, null par défaut. Ne peut pas être modifié si repository.archived_at est non NULL.
repository.github_id	Identifiant GitHub du dépôt	Int8	-	Optionnel, unique si non nul, doit être >= 100000000. Ne peut pas être modifié si repository.archived_at est non NULL.
repository.github_team_id	Identifiant GitHub de l'équipe	Int8	-	Optionnel, unique si non nul, doit être >= 10000000. Ne peut pas être modifié si repository.archived_at est non NULL.
user_repository.id	Identifiant unique de l'association	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
user_repository.created_at	Date de création de l'association	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
user_repository.updated_at	Date de mise à jour de l'association	Timestamp	-	Champ obligatoire, mis à jour automatiquement.
user_repository.user_id	Référence à l'utilisateur	UUID	-	Champ obligatoire, doit exister dans user.
user_repository.repository_id	Référence au dépôt	UUID	-	Champ obligatoire, doit exister dans repository, suppression en cascade si le dépôt est supprimé.
milestone.id	Identifiant unique du jalon	UUID	-	Clé primaire, valeur par défaut générée automatiquement (gen_random_uuid()).
milestone.created_at	Date de création du jalon	Timestamp	-	Champ obligatoire, valeur par défaut date actuelle.
milestone.updated_at	Date de mise à jour du jalon	Timestamp	-	Champ obligatoire, mis à jour automatiquement.
milestone.template_id	Référence au template	UUID	-	Champ obligatoire, doit exister dans template, suppression en cascade si le template est supprimé.
milestone.title	Titre du jalon	Varchar	250	Champ obligatoire, ne peut pas être vide.
milestone.date	Date du jalon	Date	-	Champ obligatoire.

IV. Choix techniques

A. Plateforme

Docker est la plateforme imposée par le client, toutefois celle-ci est largement bénéfique.

En effet, Docker est une plateforme open source qui permet de créer, déployer et gérer des applications au sein de conteneurs légers et portables. Ces conteneurs encapsulent une application et toutes ses dépendances, assurant ainsi une exécution cohérente sur différents environnements. L'isolation des applications permet de réduire les conflits, et limite l'impact des vulnérabilités de sécurité. En partageant le noyau du système d'exploitation hôte, Docker permet de créer des conteneurs qui ne consomment pas excessivement les ressources de la machine.

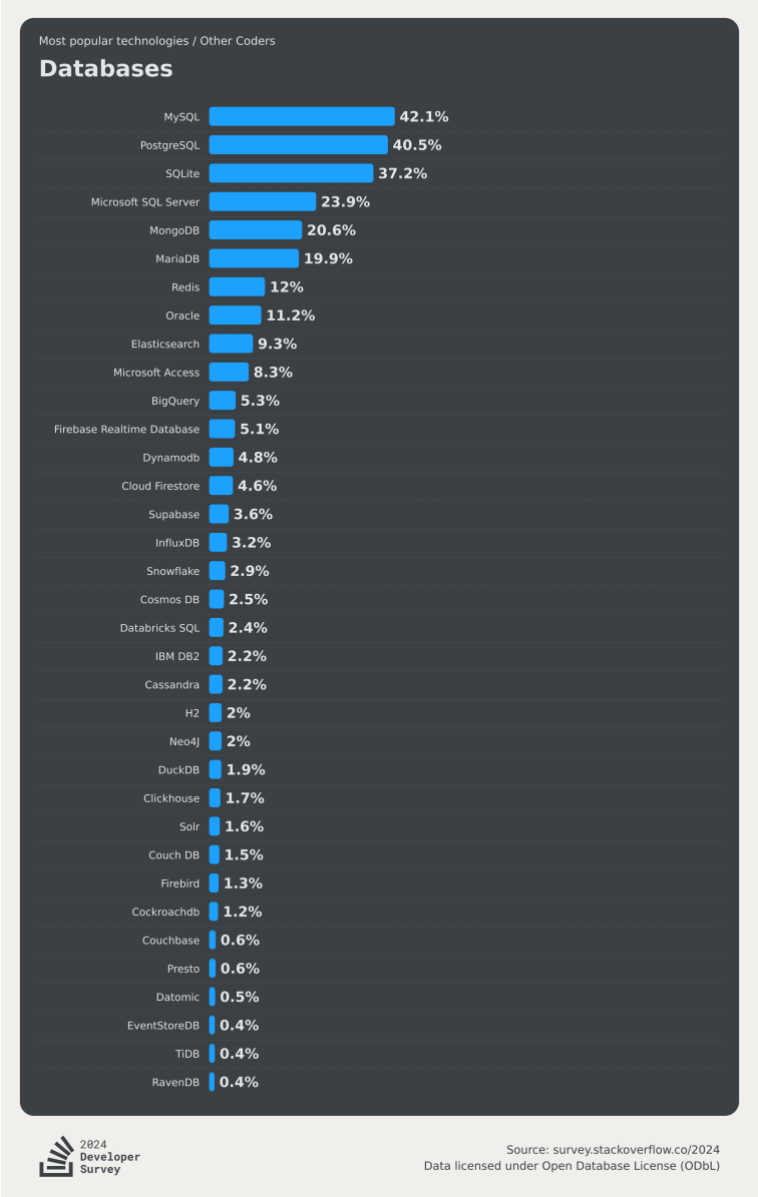
B. SGBD

1. Méthode de sélection pour le comparatif

Dans tous les SGBD qui répondent au besoin, nous avons choisis ceux qui excèdent 10% de parts de marché et dont le code est ouvert.

2. Statistiques de popularité

Les statistiques suivantes sont basés sur le « Developer Survey 2024 » de Stack Overflow. La catégorie de développeurs analysée concerne ceux qui ne sont ni professionnels, ni en phase de découverte du développement.



3. Comparatif

	PostgreSQL	MongoDB	SQLite	MariaDB
Open Source	Oui, licence PostgreSQL très permissive	Non officiellement, licence SSPL contraignante	Oui, domaine public très permissif	Oui, licence GPL modérément permissive
Maitrise basique par l'équipe	2/3	0/3	0/3	3/3
Simplicité	3/5	4/5	5/5	3.5/5
Structuration des données	Relationnel, strict	Non relationnel	Relationnel, typage dynamique	Relationnel, strict
Support des triggers	Oui, support avancé	Non	Limité	Oui
Performance	Performances excellentes grâce à son cache	Performances élevées pour des données non structurées	Performances élevées pour petites bases locales	Performances élevées
Nombre de plateformes supportés nativement par l'image Docker	8	2	6	4
Taille des images Docker	~82 Mo (postgres:17-alpine)	~236 Mo (mongo:7-jammy)	~4 Mo (alpine/sqlite:3.47.1)	~115 Mo (mariadb:11-noble)
Documentation	Exhaustive en français	Exhaustive en anglais	Exhaustive en anglais	Exhaustive en anglais, partielle en français
Indicateur Google Trends sur la semaine du 12 janvier	81	63	26	17

C. Cache

1. Méthode de sélection

À notre connaissance, il n'existe que 3 technologies qui répondent précisément à notre besoin.

2. Comparatif

	Nginx	Varnish	Squid
Open Source	Oui, licence BSD-like très permissive	Oui, licence BSD très permissive	Oui, licence GNU modérément permissive
Maitrise basique par l'équipe	1/3	0/3	0/3
Simplicité	4/5	3/5	5/5
Transparence	Proxy inverse / proxy transparent avec plus de configurations	Proxy inverse	Proxy transparent
Type de mise en cache	Durée / requêtes conditionnelles	Durée / requêtes conditionnelles	Durée / requêtes conditionnelles
Autonomie	Autonome	Dépend d'un serveur web (Nginx/Apache)	Autonome
Performances	Performances élevées	Performances excellentes	Performances élevées pour du statique, moins optimisé pour du dynamique
Nombre de plateformes supportés nativement par l'image Docker	8	6	4
Taille des images Docker	~5 Mo (nginx:1.26-alpine-slim)	~42 Mo (varnish:6.6-alpine)	~79 Mo (ubuntu/squid:6.6-24.04_beta)
Documentation	Exhaustive en anglais	Exhaustive en anglais	Exhaustive en anglais
Indicateur Google Trends sur la semaine du 12 janvier	57	31	16

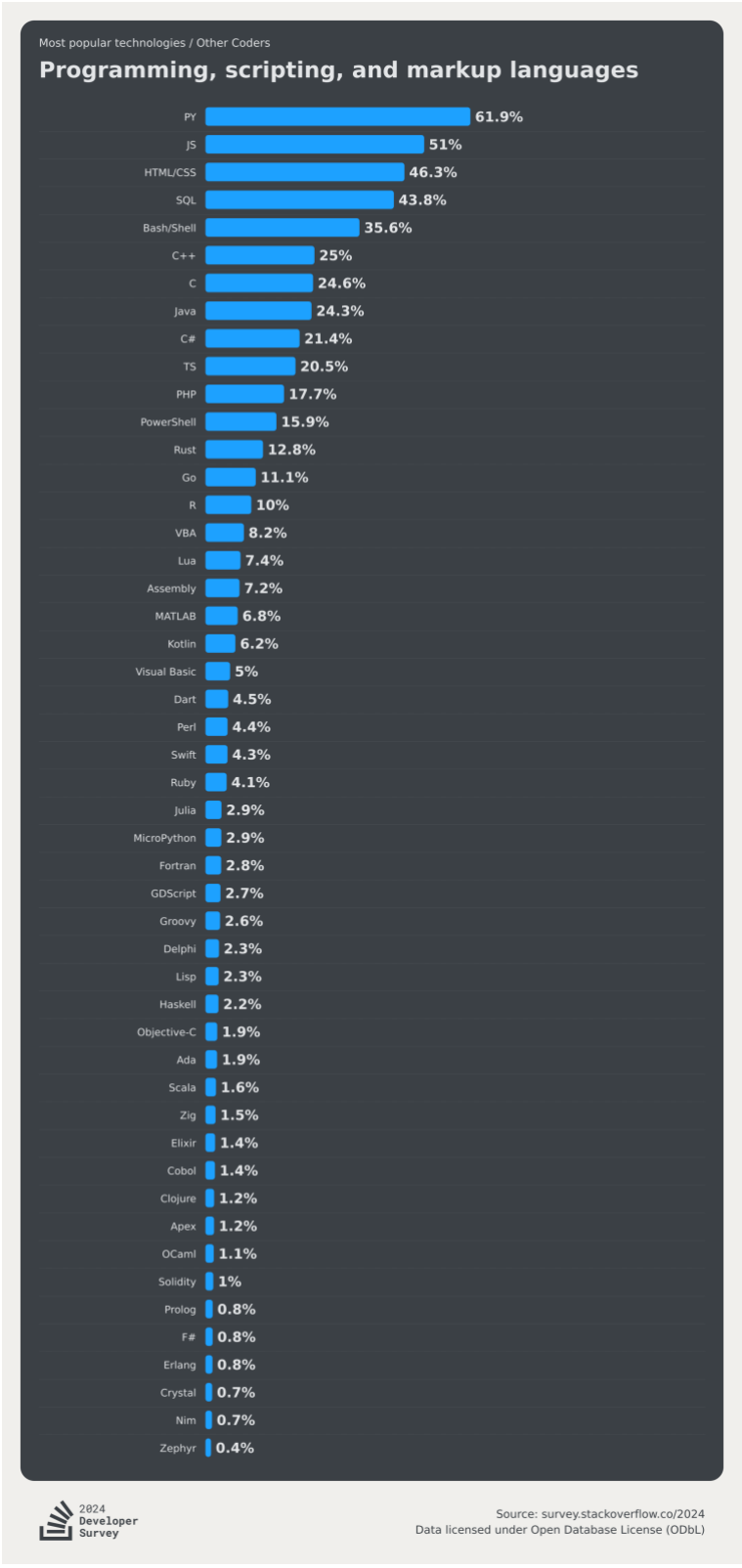
D. Langage de programmation

1. Méthode de sélection

Dans tous les langages qui ont au répondent au besoin aussi bien en backend qu'en frontend, nous avons retenu les langages qui excèdent 25% de parts de marché (Python et JavaScript). Enfin, le JavaScript a été choisis par préférence.

2. Statistiques

Les statistiques suivantes sont basés sur le « Developer Survey 2024 » de Stack Overflow. La catégorie de développeurs analysée concerne ceux qui ne sont ni professionnels, ni en phase de découverte du développement.


Most popular technologies / Other Coders
Programming, scripting, and markup languages
PY 61.9%
JS 51%
HTML/CSS 46.3%
SQL 43.8%
Bash/Shell 35.6%
C++ 25%
C 24.6%
Java 24.3%
C# 21.4%
TS 20.5%
PHP 17.7%
PowerShell 15.9%
Rust 12.8%
Go 11.1%
R 10%
VBA 8.2%
Lua 7.4%
Assembly 7.2%
MATLAB 6.8%
Kotlin 6.2%
Visual Basic 5%
Dart 4.5%
Perl 4.4%
Swift 4.3%
Ruby 4.1%
Julia 2.9%
MicroPython 2.9%
Fortran 2.8%
GDScript 2.7%
Groovy 2.6%
Delphi 2.3%
Lisp 2.3%
Haskell 2.2%
Objective-C 1.9%
Ada 1.9%
Scala 1.6%
Zig 1.5%
Elixir 1.4%
Cobol 1.4%
Clojure 1.2%
Apex 1.2%
OCaml 1.1%
Solidity 1%
Prolog 0.8%
F# 0.8%
Erlang 0.8%
Crystal 0.7%
Nim 0.7%
Zephyr 0.4%
2024 Developer Survey
Source: survey.stackoverflow.co/2024
Data licensed under Open Database License (ODbL)

E. Framework backend

1. Méthode de sélection

Dans tous les framework JavaScript qui répondent au besoin, nous avons retenu tous ceux qui excèdent 1 million de téléchargement par semaine sur npmjs.com qui est le gestionnaire de paquets pour le JavaScript.

2. Comparatif

	Express	**Nest**	**Koa**	**Fastify**
Open Source	Oui, licence MIT très permissive	Oui, licence ISC très permissive	Oui, licence MIT très permissive	Oui, licence MIT très permissive
Maitrise basique par l'équipe	3/3	0/3	0/3	2/3
Simplicité	5/5	3/5	4/5	4.5/5
Validation des schémas	Avec une bibliothèque	Avec une bibliothèque	Avec une bibliothèque	Intégré
Gestion des erreurs	Manuelle	Centralisée et structurée	Asynchrone	Centralisée et structurée
Performances	Performances élevées	Performances excellentes avec une utilisation de Fastify	Performances élevées	Performances excellentes
Nombre de plateformes supportés nativement par l'image Docker	5	5	5	5
Taille des images Docker	~50 Mo (node:22-alpine)	~50 Mo (node:22-alpine)	~50 Mo (node:22-alpine)	~50 Mo (node:22-alpine)
Documentation	Exhaustive en anglais, partielle en français	Exhaustive en anglais, partielle en français	Exhaustive en anglais	Exhaustive en anglais
Nombre de téléchargements sur npm	34	4	3	2
Indicateur Google Trends sur la semaine du 12 janvier	69	36	1	2

F. Framework frontend

1. Méthode de sélection

Dans tous les framework JavaScript qui répondent au besoin, nous avons retenu tous ceux qui excèdent 1 million de téléchargement par semaine sur npmjs.com qui est le gestionnaire de paquets pour le JavaScript.

2. Comparatif

	React	**Vue**	**Angular**	**Svelte**
Open Source	Oui, licence MIT très permissive	Oui, licence MIT très permissive	Oui, licence MIT très permissive	Oui, licence MIT très permissive
Maitrise basique par l'équipe	2/3	1/3	3/3	0/3
Simplicité	4/5	5/5	3/5	4,5/5
Performances	Performances élevées mais inférieures à Vue	Performances élevées	Performances élevées mais inférieures à Vue	Performances excellentes
Nombre de plateformes supportés nativement par l'image Docker	5	5	5	5
Taille des images Docker	~50 Mo (node:22-alpine)	~50 Mo (node:22-alpine)	~50 Mo (node:22-alpine)	~50 Mo (node:22-alpine)
Documentation	Exhaustive en français	Exhaustive en français	Exhaustive en anglais	Exhaustive en français
Nombre de téléchargements sur npm (en millions)	28	6	4	2
Indicateur Google Trends sur la semaine du 12 janvier	60	16	9	1

G. Serveur web

Afin que le domaine puisse pointer sur le bon conteneur, un conteneur faisant office de serveur web d'entrée sur l'IP devra être utilisé. Le projet est en mesure de s'adapter à la configuration client.

Cependant, si ce conteneur n'est pas encore en place, l'équipe proposera un conteneur Nginx (ou Apache) afin de répondre à ce rôle.

H. DNS

Il conviendra d'utiliser un DNS qui se charge de servir le contenu en HTTPS. Cela évitera les étapes de création, mise en place, et remplacement des certificats au cours du temps.

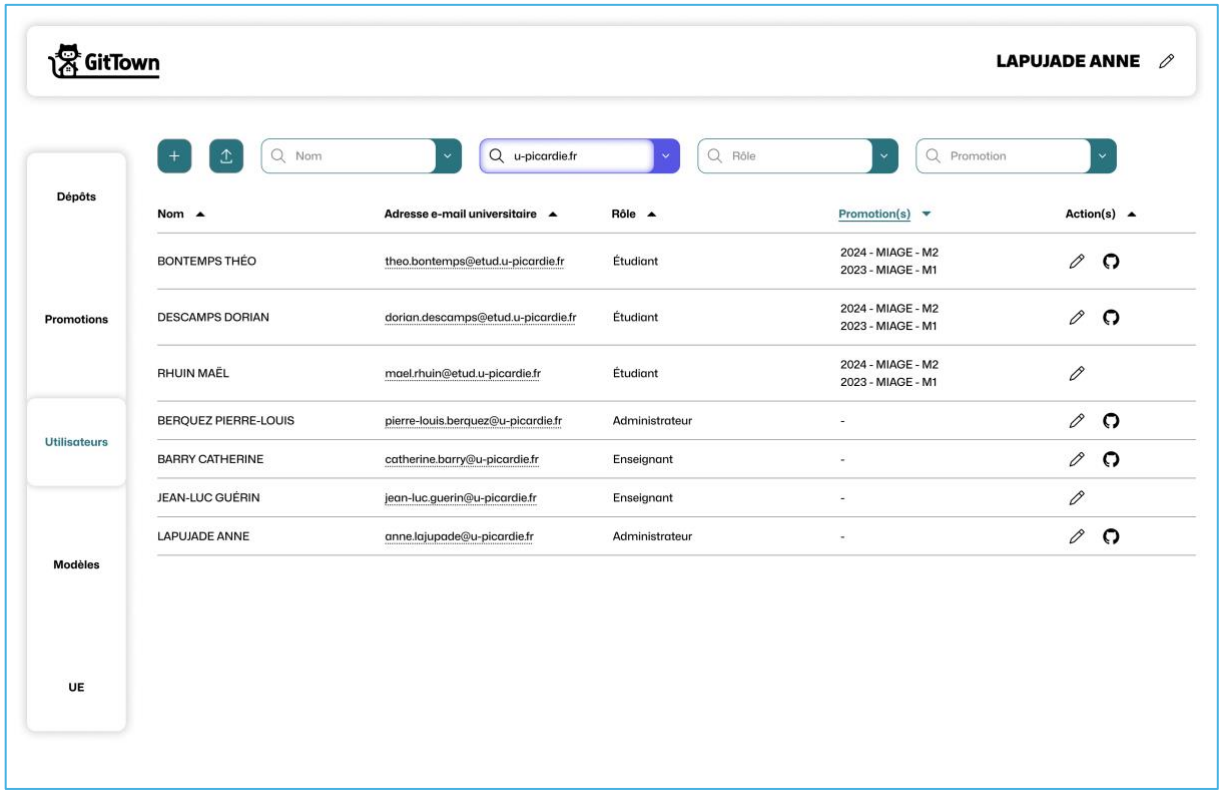
La migration des DNS de « Ionos » à « Cloudflare » est fortement recommandée par l'équipe. Cela n'impliquera aucun impact sur l'infrastructure actuelle.

V. Maquette

Le prototype Figma est disponible sur cette URL avec le mot de passe « git-town ».

Ce prototype a pour objectif de présenter la manière dont fonctionnera le projet, sans pour autant présenter celui-ci de manière réellement exhaustive. Aucun bouton d'action n'a par exemple été réalisé hors de la page « Dépôts ».

Ce prototype a été conçu en vue administrateur. En effet, notre cahier des charges fonctionnel indique que les rôles fonctionnent de manière hiérarchique. Ainsi, certaines possibilités n'apparaitront pas pour les autres utilisateurs. Pour connaitre les permissions de chacun utilisateur, il convient de se référer au cahier des charges fonctionnel.



Par exemple, un enseignant ne pourra avoir accès qu'aux onglets « Dépôts », « Promotions » et « Utilisateurs ». Dans ces onglets, il pourra effectuer toutes les actions de visualisation, filtrage et commentaires mais aucune action de modification, ni de duplication.

Un étudiant ne pourra lui avoir accès qu'à l'onglet « Dépôts » qui sera filtré inaltérablement sur ses propres dépôts. Il pourra effectuer des actions semblables à celle des enseignants, sans pouvoir accéder aux commentaires en visualisation, ni en modification.

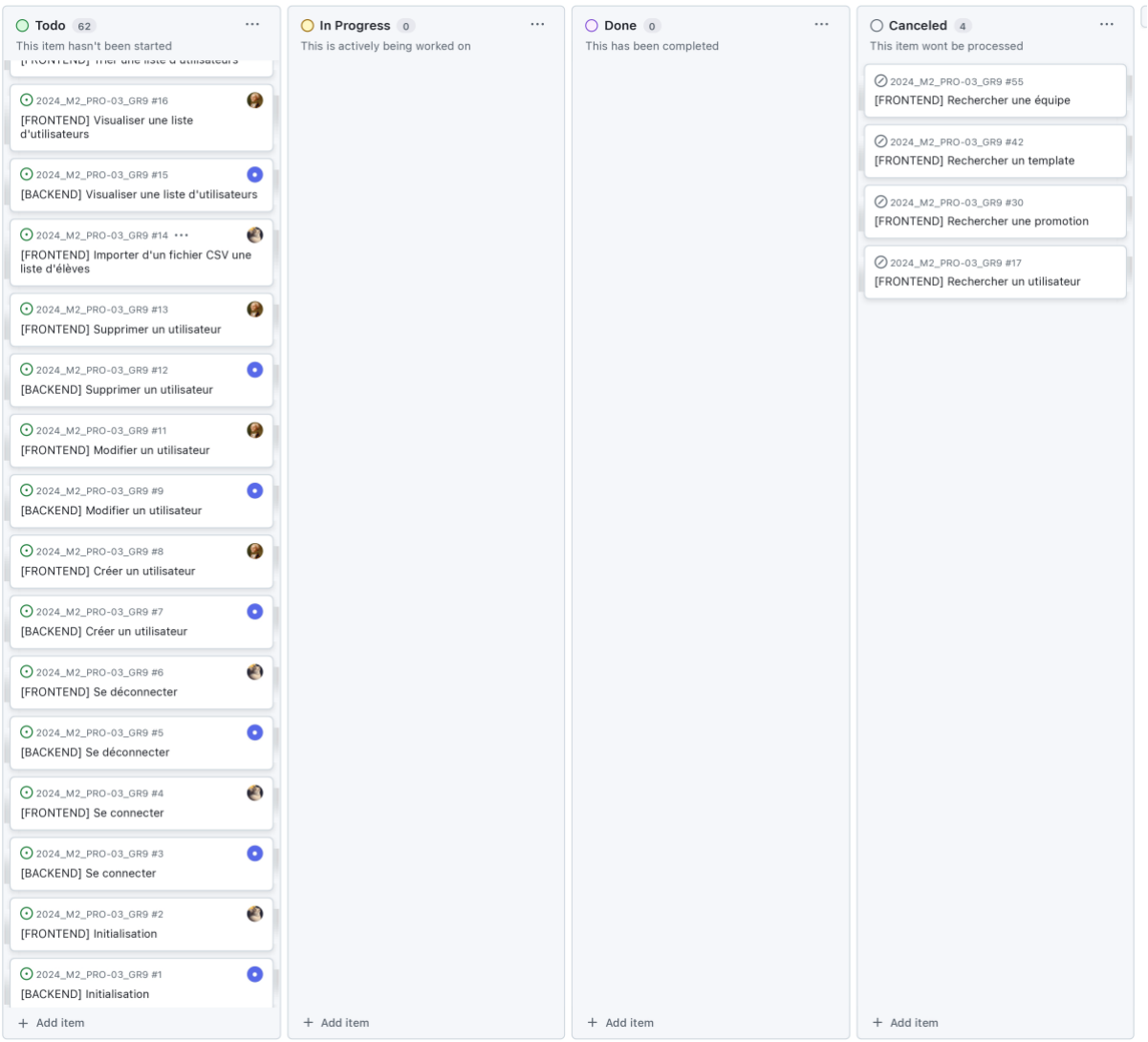
VI. Conduite de projet

A. Découpage en tâches

Il a été décidé par l'équipe de découper la plupart des fonctionnalités décrite dans notre diagramme de package en 2 tâches.

Lorsque la fonctionnalité est à traiter d'une part en frontend et d'une autre part en backend, chacune de ces parties représente une tâche. Lorsque ça n'est pas le cas, seule une tâche frontend est créée. En effet, il n'existe aucune tâche purement backend suivant notre diagramme de package.

Chaque tâche a un statut qui est indiqué dans notre tableau Kanban.



B. Répartition

Afin de profiter de l'expertise de chacun dans le domaine qu'il maitrise le plus, l'application sera développée de la manière suivante :

- Backend : Théo
- Frontend : Dorian & Maël

En effet, nous estimons également que la tâche de frontend sera plus importante en raison d'un nombre de composants conséquent à développer (barre de navigation, listes déroulantes, fenêtre de modification, etc. ...).

La tâche de backend se concentrera principalement sur la liaison avec l'API GitHub, tout en prenant soins d'inclure tous les contrôles d'accès nécessaires.

Cette configuration permettra de limiter au maximum les conflits lors de pull requests en séparant bien les développements de chacun.

Nous nous laissons cependant la possibilité de modifier la répartition au fil de l'eau si un retard majeur venait à arriver dans une des parties. Le but final étant bien entendu de livrer le projet complet en production avant la deadline.

C. Estimation

Chaque tâche a été estimé en complexité par l'ensemble de l'équipe en suivant ces possibilités :

- Basse
- Moyenne
- Complexe
- Très complexe

Ensuite, nous avons estimé chacune des complexités en temps. Voici respectivement les estimations correspondantes :

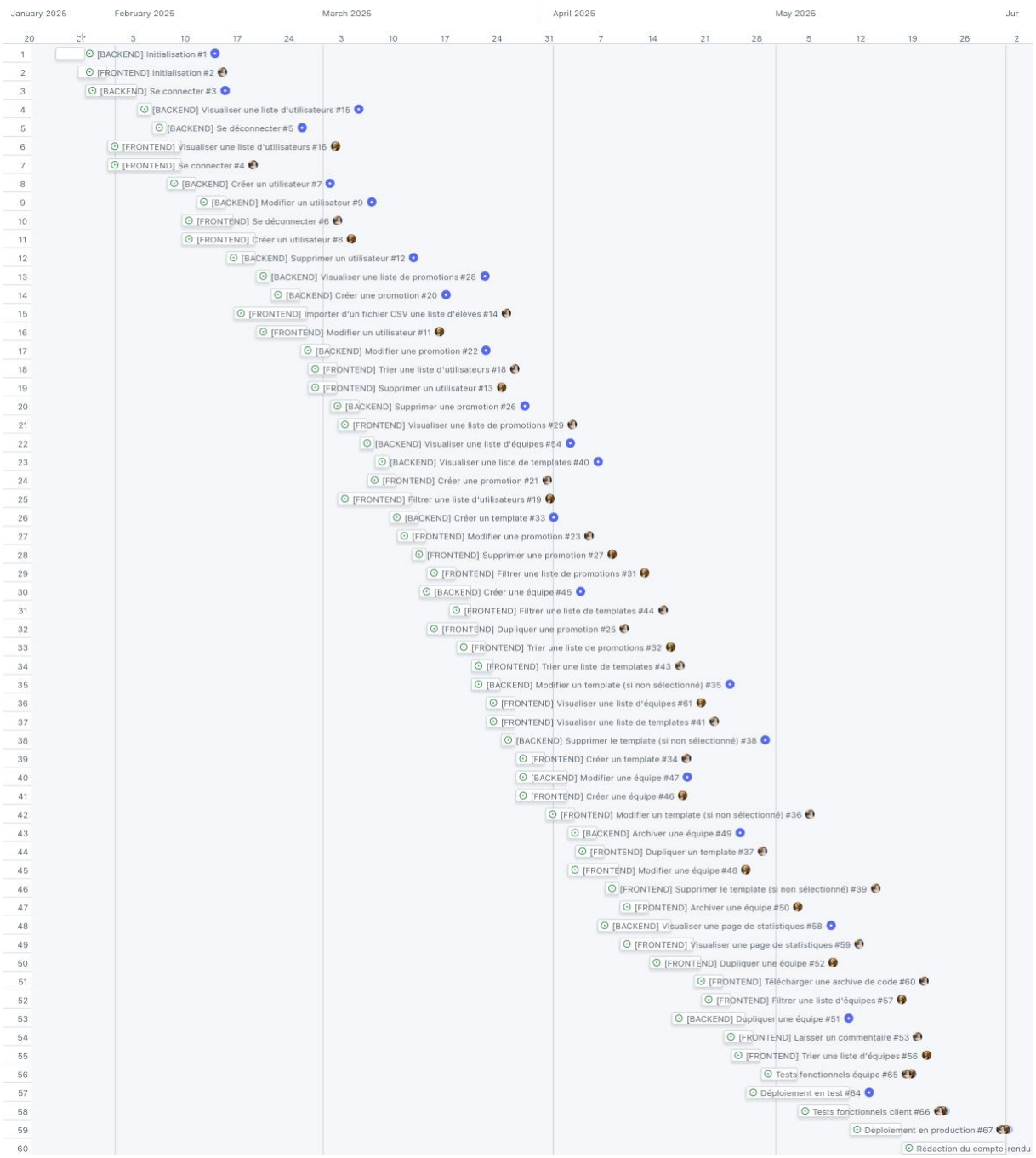
- 2 heures
- 4 heures
- 7 heures
- 10 heures

Enfin, nous avons dû choisir un rythme hebdomadaire en fonction du cumul des estimations et du temps qui nous est alloué. Nous avons décidé d'un rythme de 7 heures par semaine.

D. Planification

Nous avons établi un diagramme de Gantt en nous basant sur cette répartition et cette estimation.

Vous constaterez que celui-ci s'étale sur légèrement plus de 4 mois. D'autre part, nous avons fait le choix de nous laisser une vingtaine de jours de marge avant la deadline du troisième jalon afin d'anticiper tout éventuel retard.



E. Gestion des branches

Une branche sera créée pour chaque tâche à partir des issues mise en place sur GitHub. Lorsque la fonction sera terminée par le développeur en charge, elle pourra être revue par un autre développeur si ce premier l'estime nécessaire. Une fois ce processus terminé, la branche de tâche sera envoyée sur la branche principale avant d'être supprimée.

Afin de garder de la cohérence entre tous les développements, un linter sera utilisé et exécuté avant tout envoi sur la branche principal.

VII. Prise de recul

Au cours de ce jalon, nous avons été confrontés à plusieurs ajustements et constats importants.

A. GitHub Project

Malgré une analyse approfondie, l'intégration de GitHub Projects n'a pas pu être retenue. Cela s'explique par le fait que la fonctionnalité est toujours en phase de bêta et n'est pas pleinement supportée par l'API GitHub classique, tandis que l'API GraphQL disponible ne propose pas encore toutes les opérations nécessaires. Cette contrainte a donc orienté nos développements vers les fonctionnalités GitHub déjà stables et entièrement documentées.

B. Ajustement des estimations

Certaines estimations initialement prévues pour le backend ont dû être réévaluées à la baisse. En effet, il est primordial que les routes nécessaires soient disponibles dans des délais compatibles avec les tâches de développement frontend.

C. Anticipation des défis techniques

Nous avons certaines inquiétudes quant à la complexité de l'API GitHub et à l'assemblage de composants sur le frontend. Toutefois, nous comptons sur notre entraide et notre marge de retard afin de pouvoir livrer le projet complet à temps.

D. Validation de la maquette

La maquette Figma a été présentée au client et validée. Cette validation conforte nos choix de conception d'interface et nous permettra d'avancer en toute confiance vers la phase de développement.

E. Prochaine étape

La mise en place des environnements de développement en local pour chaque membre de l'équipe sera la prochaine étape et celle-ci commencera dès le 24 janvier pour Théo afin de préparer les premières routes. Cette initialisation est cruciale pour pouvoir travailler en parallèle dès la semaine suivante.