GITCLOUT DEV DOC

DEBATS-LY_IENG

Projet de java avancé 2023-2024

TECHNOLOGIES

- SPRING 6
  - backend du projet
- JPA
  - persistance du projet
- DERBY
  - Base de donnée du proejt
- VUEJS
  - Frontend du projet
- SEMANTIC
  - L'ui lié au frontend tu projet

Rest API

La Rest Api se trouve dans la classe `MessageController.java` et à pour route:

- `/app/rest`
  - `/hello` permet d'afficher le titre du projet
  - `/toTheBack` permet d'envoyer les données saisis dans la barre de recherche vers le backend du projet et renvoi vrai si c'est bien un git
  - `/percentFinished` permet d'afficher en temps reel l'avancement des taches soumise
  - `/getlink` permet d'afficher les liens des git soumis qui fonctionnent
  - `/getTags` permet d'afficher les tags du git soumis

> *La route `hello` est l'essai que on a fait pour la transmission des informations vers le front depuis le back je voulais voir a quoi ca ressemblait et comme c'etait sympas, on a laissé on a préferé laisser la classe de la rest api plutot épuré et elle fait appel a des fonctions se trouvant dans une classe s'appellant `BackApplication` qui travaille avec le backend et la base de donnée*
> *Toute les méthodes de la rest api sauf `/hello` sont en Post, au début on pensait que l'on pouvait les écrire en RequestMapping mais on a compris que c'était pas bon même si cela marchait. On l'utilse car, on veut demander les information pour un projet précis et il est possible que si on ne précise pas le projet cela nous donne un autre projet qu'on voulait pas comme on etait pas sur de comment cela fonctionnait vraiment.*

BACKEND

- Blame est le coeur du projet c'est la ou on analyse le projet
- Contributor est un record contenant le nom et le mail d'un contributeur
- ContributorLanguage permet de spécifier un contributeur avec un langage pour une utilisation plus simple lors d'une analyse
- Data est une classe qui stocke les données des contributeurs
- Extensions est un énum des extensions de langage
- FileExtension est un record qui permet de garder des extensions par rapport au fichier
- GitTools sert pour tout ce qui est en rapport avec l'utilisation de git, le clonnage, les differences, les contributeurs ...
- JGitBlame sert a préparer le blame
- StringWork sert a faire des manipulation sur les fichier ou le chaines de caracteres, par exemple convertir un lien en un localpath
- UtilsMethods sert pour garder les methodes annexes tels que isGitRepo qui permet de savoir si le lien entré est un git ou non.

- *L'entrée de l'analyse se fait avec JGitBlame, la dedans on clone et on analyse, on fait une analyse par tag. On a pensé que pour chaque tag different il etait préférable de de le faire dans une classe qui est ici Blame.*
- *On a crée une classe contributor parce que on savait qu'il etait possible que des personnes utilisent le même prénom et parfois le même mail, mais il y a moins de chance pour quelle utilisent les deux en meme temps et si c'est le cas c'est probablement la même personne*
- *ContributorLanguage me permettait de simplifier la differenciation dans l'analyse.*
- *Pour ce qui est de Extensions, on a pensé au début d'utiliser un HashSet, cependant on a pensé que utiliser un Enum qu'on appelerait a chaque fois serait plus correcte et faire nos choix via des switch sur des enums.*
- *Au début, il n'y avait pas de GitTools, mais on pensé que créer une classe qui me permettrais de faire tout ce qui est en rapport avec le git serait plus simple de compréhension, on a eu la même réaction pour StringWork et UtilsMethods, utilsMethods qui servait pour les méthodes dont on savait pas trop ou les mettre.*

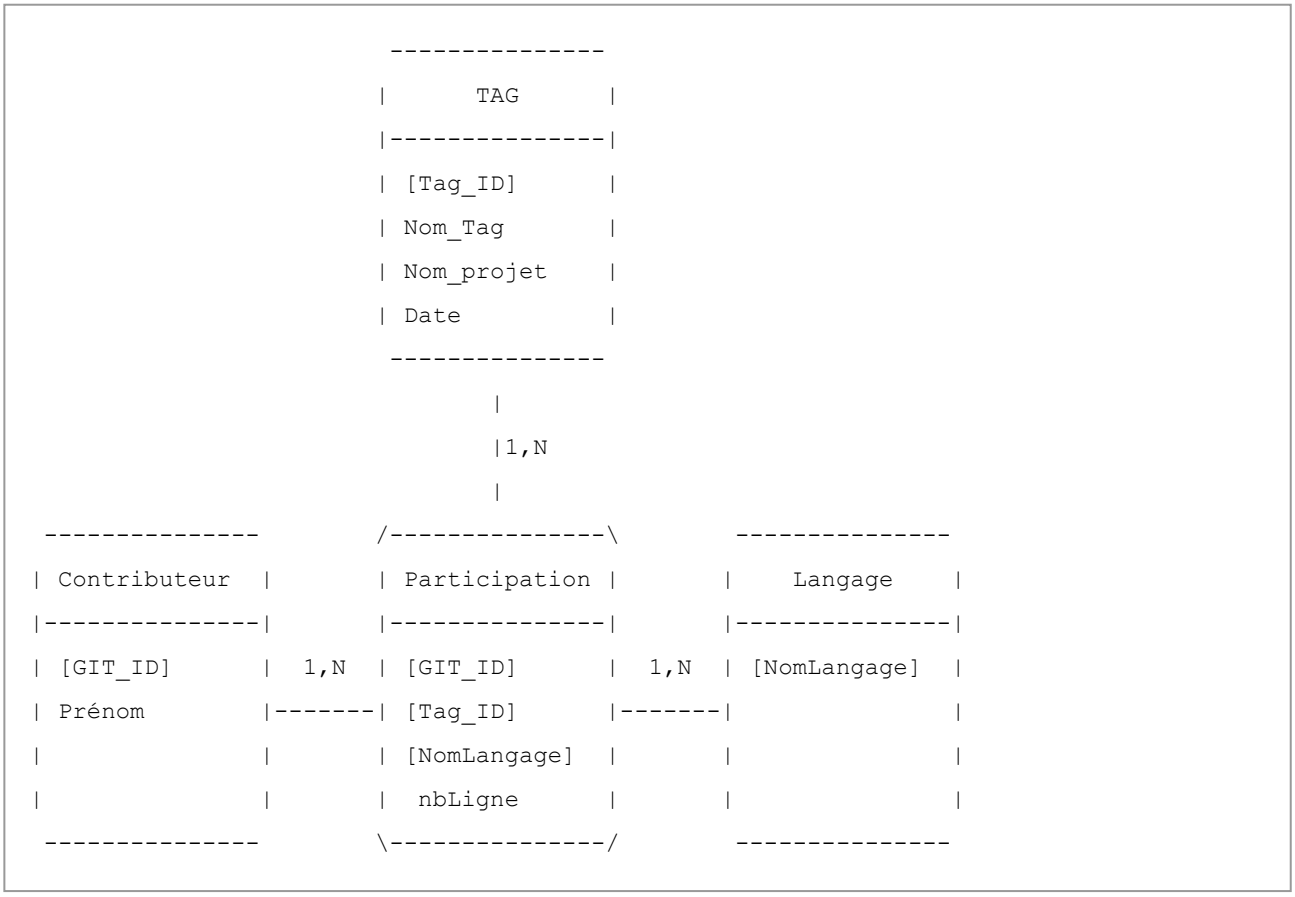
- *Pour le blame, on donne toutes les informations qui sont utile pour l'analyse avec le constructeur, par exemple, le GitDiff du tag actuel ce qui nous permet de ne pas faire une nouvelle analyse des fichiers déja présents et non modifiés.*
  On utilise des regex pour les differents énums qu'on a pour verifier si se sont des commentaires, pour les eviter. Les lignes n'etant pas de commentaires sont stockés avec la personne qui les à écrite dans la classe Data et peuvent s'incrémenter si la personne à écrit dans deux fichiers differents.
  Pour essayer d'accélerer l'analyse nous avons éssayé d'ajouter un executorService au Blame.

FRONTEND

- Homepage est le composant principale qui est appelé par App.vue
- ChartPage est un composant enfant de Homepage et affiche les Graphiques en baton (cela affiche les contribution pour un tag actuel)
- GitPage est un composant enfant de Homepage et affiche les leins git deja réalisé
- RadarChartPage est un composant enfant de Homepage et affiche les graphique en radar (cela affiche la moyenne des contributions par contributeur)

> *Apres avoir essayé pendant une semaine à ouvrir notre page dans un nouvel onglet sans succes (c'est pour cette raison que le composant principal se nomme Homepage), nous avons décidé de partir sur une page principale qui ferait tout, l'affichage du projet actuel et la liste des anciens projets. Au début on à tout écrit dans HomePage, mais on a pensé que ca serait plus propre de separer en plusieurs composants quitte a faire passer des information entre composants comme les tags*

Description de la base de donnée



- La table Langage stockera les langages (les 10 plus utilisés) sur lesquels on va faire l'analyse
- La table contributeur permettra de stocker les données des contributeurs c'est a dire leurs nom et leurs Email qui servira de clé primaire

- La table tag sert a stocker tout les tags de chaque projet, on pourra les differencier en stockant aussi le nom du projet dedans, pour la clé primaire, on utilisera le shawan du tag.
- La table participation est une table associative permettant de regrouper ces 3 tables et nous travaillerons majoritairement sur cette table.

> *A vrai dire notre base de donnée à été modifié plusieurs fois tout au long du projet, ce n'est que vers fin novembre que l'on à vraiment décidé qu'elle ressemblerait à ça, mais même si elle changeait assez souvent, la base de donnée restait assez similaire à cette base de donnée. On voulait par exemple au lieu d'avoir une table Langage, une table Fichier qui nous aurait probablement donné une piste pour une futur amélioration.*

> *Comme On l'a dit dans la partie backend, on ne faisait l'analyse des fichiers d'un tag uniquement si le fichier faisait partie de la liste des differences avec le tag précédent. Cela pouvait poser problème car, pour le tag actuelle ca voulait dire que le fichier n'etait pas dans la base de donnée.*
> *Ce qu'on a voulus faire c'est donc que pour chaque nouveau tag, on fait une copie image de la base de donnée puis on insert ou met à jours les nouvelles informations.*