

Analyse de l'arbre Git pour P_GestProj426

Ce rapport présente un aperçu détaillé de l'historique des commits Git pour le dépôt `AESTML/P_GestProj426`. Vous trouverez également des informations sur le processus de développement, ainsi que sur le problème de suppression des branches fusionnées. L'arbre Git illustre un projet collaboratif où plusieurs contributeurs travaillent sur des tâches frontend, backend et de configuration.

Aperçu du projet

Le dépôt `P_GestProj426` est un projet d'application web avec des composants frontend (utilisant Vue.js) et backend. Les fonctionnalités clés incluent l'authentification des utilisateurs, une visualisation sous forme de globe, des fonctionnalités de cartographie et une base de données (liée à l'UNESCO, d'après `db_unesco.sql`). Le projet comporte de nombreuses fusions, pull requests et branches, indiquant un effort d'équipe avec une synchronisation fréquente.

Jalons clés

- **Configuration initiale** : Création de la structure du projet, du README et des fichiers liés à la base de données (ex. : `Add(MCD)`, `Create README.md`, `Ajouter le docker base de données`).
- **Développement backend** : Implémentation des modèles, routes, contrôleurs et scripts de base de données (ex. : `SCRIPT DB TERMINÉ !!!!!!!!`, `feature(seeder)`).
- **Développement frontend** : Adoption de Vue.js pour l'interface, création de vues, routes et composants comme les en-têtes, pieds de page et un globe (ex. : `Create routes accueil + map in VueJs`, `Ajout d'un globe`).
- **Authentification** : Travail sur les pages de connexion, d'inscription et de compte (ex. : `Feat(Register)`, `Start(Auth) WIP`).
- **Design et actifs** : Ajout de logos, sauvegardes Figma et stylisation CSS (ex. : `Logo Final`, `css_global`).
- **Fusions et pull requests** : Fusions fréquentes de branches comme `dev`, `backend`, `frontend-VueJs` et `main`, avec plus de 50 pull requests.

Décomposition de l'arbre Git

L'arbre Git montre un historique complexe avec de nombreuses fusions et branches. Voici un résumé des activités notables :

Début du développement

- **Commits** : Les premiers commits incluent la mise en place de la structure du projet, l'ajout d'une configuration Docker pour la base de données et la création d'un modèle conceptuel de données (MCD).
- **Exemple** : `Create README.md`, `Add(MCD)`, `Création du projet`.

Progrès backend

- **Commits** : Développement des scripts de base de données, modèles et routes (ex. : `Create(ScriptSQL)`, `Update(model)` ajout de bcp de models, `Finish(/lieu/ routes)`).

- **Fusions** : La branche `backend` a été fusionnée plusieurs fois dans `main` (ex. : `Merge branch 'backend'`).

Développement frontend

- **Commits** : Introduction de Vue.js, création de vues (ex. : `page de details(html/css)`), et fonctionnalité de globe (ex. : `Ajout d'un globe`, `Le globe marche`).
- **Branches** : La branche `frontend-VueJs` a été largement utilisée et fusionnée dans `main` via des pull requests (ex. : `Merge pull request #50 from ASETML/frontend-VueJs`).

Authentification et fonctionnalités

- **Commits** : Travail sur l'authentification (ex. : `Start(Register) [WIP]`, `update(fonction auth)`), les cartes (ex. : `carte sur la page home avec les points : OK`), et les pages de compte (ex. : `AccountPage(Ajout de la zone IN DANGER)`).
- **Branches** : Des branches dédiées comme `auth` et `globe` ont été créées (ex. : `(origin/auth) Feat(Register)`).

Design et documentation

- **Commits** : Ajout de logos, sauvegardes Figma et notes de revue de sprint (ex. : `Logo Final, backup figma 1 created, NOTES sprint review - cliente`).
- **Pull Requests** : Exportation des journaux et arbres (ex. : `Export du jdt et de l'arbre`).

Activité récente (mars 2025)

- **Commits** : Mises à jour de `Header.vue`, suppression de données fictives et corrections de bugs (ex. : `Update Header.vue, 🎯 remove fake Data, Bugfix(dependencies)`).
- **Référence de date** : Le dernier commit mentionne "JDT du 24.03" (24 mars 2025), en accord avec la date actuelle du 31 mars 2025.
- **Suite** : à continuer petit à petit.

Oubli de suppression des branches fusionnées

Dans Git, il est courant de supprimer les branches après leur fusion dans la branche principale (par exemple via des pull requests) pour garder le dépôt propre. Cependant, dans ce projet, nous avons oublié de supprimer plusieurs branches après leur fusion. Voici comment cela s'est produit et ses implications :

Preuves dans l'arbre

- **Étiquettes de branches persistantes** : Des branches comme `frontend-VueJs`, `backend`, `globe`, `auth` et `dev` apparaissent à plusieurs reprises dans l'arbre, même après avoir été fusionnées dans `main`. Par exemple :
 - (`frontend-VueJs`) `Merge branch 'frontend-VueJs' of https://github.com/ASETML/P_GestProj426 into frontend-VueJs` montre que la branche persiste après fusion.
 - (`origin/auth`) `Feat(Register)` reste visible malgré les fusions.
- **Pull Requests** : Plus de 50 pull requests (ex. : `#50, #49, #42`) indiquent des fusions réussies, mais les branches correspondantes subsistent dans l'historique.

Pourquoi nous avons oublié

1. **Priorité au développement** : L'équipe s'est concentrée sur l'implémentation des fonctionnalités (ex. : globe, auth, routes) plutôt que sur l'entretien du dépôt.
2. **Fusions fréquentes** : Avec des fusions rapides entre branches (ex. : `Merge branch 'main' of https://github.com/ASETML/P_GestProj426` apparaît souvent), le nettoyage des branches a été négligé.
3. **Absence d'automatisation** : Absence de workflows automatisés (GitHub Actions) pour supprimer les branches après fusion des pull requests.
4. **Chaos d'ensemble** : Plusieurs contributeurs fusionnant des branches dans `main`.

Implications

- **Dépôt encombré** : Les branches inutilisées gonflent l'arbre Git, rendant la navigation plus difficile.
- **Confusion** : Les développeurs pourraient travailler par erreur sur des branches obsolètes, pensant qu'elles sont encore actives.
- **Surcharge** : Les branches supplémentaires augmentent la taille et la complexité du dépôt.

Comment nous avons corrigé la situation

Finalement, nous avons remédié à la situation en procédant à des suppressions importantes dans le dépôt, comme en témoignent plusieurs commits dans l'arbre Git. Voici comment cela s'est déroulé :

1. **Suppression de données inutiles** : Nous avons éliminé des éléments obsolètes ou fictifs qui encombraient le projet, par exemple avec le commit `Remove fake Data`, qui a retiré des données fictives utilisées lors des phases initiales de développement.
2. **Nettoyage des fonctionnalités redondantes** : Certaines fonctionnalités ou champs ont été supprimés pour simplifier le code, comme illustré par `Remove Email Field`, montrant une volonté de réduire les éléments non essentiels.
3. **Réorganisation et consolidation** : Des commits comme `Nettoyer le git` et `Petite réorganisation du repo` indiquent des efforts pour restructurer le dépôt, en supprimant des branches ou fichiers inutiles après les fusions massives, et d'autres branches (ex. : `Merge de toutes les branches`) ont conduit à un historique encombré.
4. **Correction des dépendances** : Le commit `Bugfix(dependencies)` suggère que nous avons également nettoyé les dépendances inutiles ou problématiques, contribuant à alléger le projet.

Ces suppressions majeures ont permis de réduire l'encombrement accumulé par les nombreuses branches fusionnées non supprimées. Bien que l'arbre ne montre pas explicitement la suppression systématique des branches avec des commandes comme `git branch -d`, les actions de nettoyage et de retrait d'éléments superflus ont servi de solution pratique pour remettre de l'ordre dans le dépôt.

Dispute pour technologies utilisés

Au cours de la phase de développement, il y a eu des désaccords et disputes entre membres de la ScrumTeam au sujet des technologies et des méthodes de développement utilisées durant le projet. Notamment: au sujet de l'ORM Sequelize par rapport a JS traditionnel sans ORM

- Toute la scrum team sauf M. Denis était d'accord pour utiliser Sequelize avec notamment des modèles etc... pour les datas.

- M. Denis était contre cette idée et préférait tout coder en vanillaJS sans les méthodes, contrôleur et modèles apportés par Sequelize et c'est pourquoi il a codé toutes les routes sans Sequelize.
- Puis M. Piguet à notifié qu'aucune des routes n'utilisaient les beaux modèles Sequelize qu'il avait codé avec le plus grand soin. Une joute verbale entre Denis et Piguet pris alors lieu débattant l'utilité des modèles et l'importance d'un ORM.
- La semaine suivante, lors de l'absence de M. Piguet, M. Denis a réussi à endoctriner la suppression de Sequelize dans les esprits de la ScrumTeam et ainsi fut la suppression de Sequelize dans le main.
- A son retour, M. Piguet découvrit alors la décision de la ScrumTeam et il confronta le membre et médiateur de la ScrumTeam (M. Segalen) en lui demandant pourquoi cette décision fut prise. M. Segalen lui répondit que "ça marche sans." et cela suffit amplement comme raison pour M. Piguet.
- Après ceci, M. Piguet tenta de réintroduire discrètement Sequelize mais M. Denis flaira l'entourloupe et supprima la branche Sequelize, réduisant les efforts de M. Piguet à néant.

Conclusion du conflit

Après une longue discussion entre tous les membres de la ScrumTeam, ils arrivèrent à la conclusion que l'adaptation du code pour y introduire Sequelize serait trop long par rapport à l'importance de l'ORM.

Grâce à ce conflit, les membres de la ScrumTeam se rapprochés et ont appris à mieux communiquer pour favoriser un environnement de travail sain et agréable (et sans Sequelize)

Conclusion

L'arbre Git de [P_GestProj426](#) met en lumière un projet dynamique avec des avancées significatives dans le frontend (Vue.js), le backend et le design. L'oubli initial de supprimer les branches fusionnées avait conduit à un historique encombré, mais nous avons finalement corrigé cela en procédant à de grandes suppressions, comme le retrait de données fictives et de fonctionnalités inutiles. Ces efforts ont permis de rendre le dépôt plus clair et fonctionnel, bien qu'une suppression systématique des branches aurait pu compléter ce nettoyage. À l'avenir, adopter des pratiques proactives de gestion des branches renforcerait cette amélioration. `git push -u origin brancheDetails`
`git push -u origin brancheDetails`
`git push -u origin brancheDetails`

Conclusions Personnelles

- **Antoine Piguet**

J'ai apprécié travailler avec une "grande" équipe, cela a pu me donner une vraie vision du teamwork en entreprise et des conflits qu'il peut engendrer. C'est pourquoi nous avons beaucoup appris à s'organiser pour éviter le maximum de conflits, travail à double, travail pas en accord avec la DOD etc... J'ai aimé apprendre ces choses car je sais qu'elles me font cruellement défaut et donc elles me sont utiles même en dehors de l'ETML.

- **Romain Denis**

Je pense que ce projet me sera utile pour la suite de ma carrière. Il m'a permis d'acquérir de nombreuses compétences, notamment en communication et en gestion des conflits. Il m'a également donné un aperçu de ce à quoi ressemblera un environnement de travail type pour l'avenir. Malgré de

nombreux conflits de branches au début du projet, nous avons beaucoup progressé et notre utilisation globale de GitHub est devenue très fluide. Nous avons également rencontré quelques conflits de conventions, par exemple concernant Sequelize. Nous aurions dû être plus stricts et plus clairs sur les conventions. Cela aurait permis de gagner du temps à long terme. Nous aurions également dû consacrer moins de temps à la maquette de l'application. Nous y étions encore à la fin du deuxième sprint, ce qui nous a ralenti. À part cela, je pense que nous avons bien travaillé sur le projet et que chacun a joué son rôle.

- **Mateo Thode**

J'ai beaucoup aimé ce projet et module. J'ai trouvé que l'ambiance et les manières de fonctionner ressemblait beaucoup à ce que j'ai vu en entreprise. En plus, il n'est maintenant plus exclu pour moi d'être Scrum Master dans une équipe de développement. Je pense que nous avons tous beaucoup appris, sur le plan purement informatique, mais également sur le savoir-vivre en groupe, c'est ce qui m'a beaucoup plu.

- **Alban Segalen**

J'ai bien aimé que ce module/projet se soit concentré sur la pratique et non sur la théorie. J'ai trouvé compliqué et intéressant le travail en grande équipes. Ce projet m'a appris beaucoup sur la méthodologie Scrum. Malgré les quelques dissensions dans l'équipe, nous avons quand même réussi à réaliser une application fonctionnelle.

- **Antoine Fabre**

Ce projet m'a beaucoup apporté en matière de travail en équipe avec SCRUM et GitHub Project. Cela m'a donné un avant-goût du travail en entreprise et en équipe. Ce qui m'a bien plu !

- **Charles-Henri Moser**

J'ai bien aimé ce projet car il m'a permis d'avoir une réelle vision de ce qu'est le travail en équipe, il m'a apporté différents skills en communication et en organisation. Bien que le projet ait eu des complications, je pense avoir mené à bien mon travail, j'ai particulièrement vu à quel point la coordination, les règles communes (comme la DOD) et les conventions techniques sont essentielles pour éviter les conflits inutiles. Je pense que j'aurais pu faire beaucoup mieux quant à l'organisation. Mais ce que je retiens surtout, c'est l'apprentissage du fonctionnement en équipe, dans un cadre SCRUM, avec des outils comme GitHub Project. Ça m'a donné une image plus réaliste de ce qui m'attend en entreprise.

- **Yosef Nademo**

Ce projet m'a permis de mieux comprendre la gestion d'un dépôt Git collaboratif et les défis liés au suivi des branches et aux fusions. J'ai aussi vu l'importance de maintenir un dépôt propre pour éviter la confusion et les erreurs. L'analyse de l'arbre Git m'a montré combien la coordination entre les membres de l'équipe est essentielle, surtout quand il y a des désaccords techniques comme avec l'utilisation de Sequelize. Ce genre de conflit m'a appris qu'un bon compromis est souvent nécessaire pour avancer efficacement en équipe. J'ai aussi apprécié développer des fonctionnalités techniques en frontend, comme l'intégration de la carte interactive et l'affichage dynamique des sites UNESCO. J'ai trouvé intéressant de réfléchir à l'ergonomie pour rendre la plateforme intuitive et agréable. Travailler en équipe Scrum m'a appris à mieux structurer mon travail et à rester proactif face aux défis techniques.