Contents

À propos de ce document

Afin de pouvoir assurer une évolution la plus saine possible du projet, ce document a pour but de donner les comportements attendus lors d'une contribution à Re2o. Il s'inspire notamment de :

- A successful Git branching model (https://nvie.com/posts/a-successful-git-branching-model/);
- Les comptes rendus des différentes réunions autour du projets.

Développement

Code source

Le code source doit être en anglais, respecter la PEP8 (https://www.python.org/dev/peps/pep-0008/). Il existe des outils qui vous permettent de rendre votre code *PEP8-compliant* de manière presque autonome. Respecter la PEP20 (https://www.python.org/dev/peps/pep-0020/) est plus qu'une bonne idée.

Documentation

Il est impératif d'écrire de la documentation pour que le projet soit pérenne. La langue à privilégier est là encore l'anglais.

On se réfèrera à cette page : (https://gitlab.federez.net/federez/re2o/wikis/Dev%20Documentation/Documentation) pour plus de précisions sur la documentation.

Interface

L'interface utilisateur doit être intuitive. Cette notion étant difficile à évaluer, la meilleure façon de s'en assurer et de faire tester l'interface à un maximum de personnes, de préférence concernées par l'interface en question. Par exemple, un utilisateur lambda devrait tester la page de profil, et pas uniquement un administrateur ou un développeur chevronné.

La langue de l'interface doit être à minima en anglais, l'idéal étant d'ajouter la traduction le plus tôt possible.

Migrations

Afin de lutter contre la complexification de l'arbre des migrations, il est fortement conseillé de fusionner un maximum de migrations ensemble.

Utilisation du Gitlab

Le gitlab du projet se trouve ici : https://gitlab.federez.net/federez/re2o.

Structure des branches

Master

Master est la branche stable. ***Il est interdit de `push` directement sur master, à l'exeption des commits de merge***. Les *Merge requests* autorisées doivent venir d'une branche de *release* ou de *hotfix* et avoir été validées par 2 contributeurs (si possible non impliqués dans la phase de développement) pour être mergées.

Merge dans master

Afin de garder un historique des commits propre, l'utilisation de `merge` avec l'option `--no-ff` semble une bonne solution. Cela implique de ne pas passer par

le bouton `Merge` du Gitlab. Les commits de merge dans master doivent avoir la forme suivante :

`Release : <Numéro de release> [: <Nom de release>]`

ou

`Hotfix : <Description sommaire du hotfix>`

De cette manière on peut facilement annuler un merge introduisant un bug.

Dev

`Dev` est la branche de développement. Toutes les branches de *feature* et de *release* doivent partir de dev.

Lors du merge d'une *release* dans master, il faut également merger ladite release dans dev.

Merge dans dev

Le merge dans dev doit être précédé par la création d'une *Merge Request*. Soit il s'agit du merge dans master d'une *release*, soit il s'agit de l'intégration d'une *feature*. Dans le deuxième cas, il faut créer une MR sur le Gitlab, qui devra avoir été revue par deux contributeurs avant d'être fusionnée. On peut pour cela utiliser le bouton prévu à cet effet sur l'interface de la MR dans Gitlab.

Branches de *release*

Les branches de release préparent le merge dans master de la branche dev. On peut créer une branche de *release* dès lors que la *milestone* associée (https://gitlab.federez.net/federez/re2o/milestones) est complétée ou que les modifications à effectuer sont 'mineures' (traduction, retouche de design, etc...). Cela permet à la branche dev de continuer d'avancer, même si la *release* a besoin de quelques retouches.

Il semble bon de nommer ces branches selon la convention suivante :

`release-<numéro de release>`

Branches de *feature*

Dans l'idéal, une feature doit être précédée de la création d'une *issue* (https://gitlab.federez.net/federez/re2o/issues), avec les bons tags, notamment `To Do`, `Doing` et `Done`, afin de pouvoir suivre l'évolution du projet sur cette page : https://gitlab.federez.net/federez/re2o/boards.

Si votre *feature* nécessite certaines manipulations de la part de l'utilisateur, ou introduit des notions non triviales, il ne faut surtout pas hésiter à éditer le fichier `CHANGELOG.md` ainsi que créer les pages wiki nécessaires.

Le nom de la branche de *feature* devrait être explicite. Bien qu'il n'y ait pas de restriction sur le format des messages de commits, essayez de les garder le plus clair possible.

Les branches de *feature* ont vocation à être mergées dans dev. Pour cela utilisez des merge requests, en faisant bien attention à la branche cible. La merge request peut-être utilisée afin d'obtenir des retours. Pour cela préfixez le nom de la merge request par `WIP :`.

Branches de *hotfix*

Malgré la relecture attentive et les efforts de chacuns, des bugs peuvent réussir à se glisser en production. Afin de les corriger rapidement, il faut créer une branche de *hotfix*. Ces branches sont destinées à être mergées dans master et dans dev en suivant les règles établies pour le merge dans ces branches.

Une bonne manière de nommer ces branche est

`hotfix-<nom du bug ou de l issue si elle existe>`

Fusion de branches

Afin d'éviter des conflits lors de fusions il est préférable de rebaser la branche cible dans la branche source.

Utilisation des issues

Les issues permettent de remonter les problèmes et de proposer de nouvelles features. L'utilisation des tags à disposition est fortement encouragée.

Communication et vie du projet

Le projet dispose de plusieurs moyens de communication :

- Le Gitlab avec les issues pour les propositions détaillées de nouvelles *features* et les rapports de bugs;
- Le chan IRC (`irc.rezosup.org` sur le cannal `#re2o`), bridgé avec cette conversation Telegram;
- La mailing-list re2o@federez.net.

Pour tout ces canaux de discussion, les règles de savoir vivre usuelles sont en vigueur. En particulier, il est attendu des participants :

- Qu'ils respectent les idées des autres, et les laissent les exprimer;
- Qu'ils encouragent la participation des nouveaux venus au projet. Cela signifie en particulier qu'il n'existe pas de question stupide, que si une question revient de manière récurrente, alors il est nécessaire d'éclaircir la partie du projet concernée et que les nouveaux venus doivent être guidés s'ils le souhaitent;
- Qu'ils soient capables de se retenir de participer s'ils pensent que leur participation ne sera pas constructive. En effet ce projet nous tient tous à cœur, et il peut nous arriver de défendre une idée au delà du raisonnable car elle nous semble vitale.
- Qu'ils prennent du plaisir à participer à ce projet communautaire et instructif.