Références

- https://www.gnu.org/ (système d'exploitation et mouvement du logiciel libre)
- https://producingoss.com (livre sur le développement open source)
- https://openhatch.org (archive: organisation aidant à contribuer dans l'open source)
- http://teachingopensource.org/ (actualité / blog posts sur l'enseignement open source)
- http://open-advice.org/ (conseils de développeurs open source)
- http://oss-watch.ac.uk (conseil sur l'utilisation, le développement et les licences open source)

I – Démarrer un projet

II.0 - Ressusciter un projet abandonné

II.1 - Un petit point avant de commencer

II.2 - Démarrage d'un nouveau projet

II.3 - Le choix de la licence

II.4 - Instaurer une ambiance collaborative

II.5 - Gérer l'ouverture du projet

II.0 - Ressusciter un projet abandonné

❓ Comment peut-on ressusciter un projet abandonné ?

❓ Pourquoi ressusciter un projet ?

Contacter le mainteneur d'origine

Le mainteneur d'origine peut (peut-être) s'engager à nouveau
💭 Il a pû être découragé par le manque d'intérêt perçu
Exemple: https://github.com/rsmpi/rsmpi/issues/109

Si vous arrivez à le contacter, il pourra :

- Vous donner accès au dépôt, traqueur de bogues, site web d'origine, ...
- Vous expliquer la raison de l'arrêt du développement du projet :
  💭 Problème personnel ? lié à la communauté ? obstacle non surmonté ?
  ⚠️ Si la raison est un obstacle, cela doit être votre priorité pour poursuivre le développement du projet !
- Vous détailler l'endroit où il s'est arrêté et les options déjà explorées

Rechercher des Forks du projet

⚠️ Il est possible que d'autres personnes aient eu la même idée que vous ! Vous pouvez chercher cela sur Google ou sur le forum / liste de diffusion associé(e) au projet !

S'il n'en n'existe pas (ou pas de satisfaisant), démarrez votre propre Fork ! Démarrez un nouveau dépôt et préparez tout ce qu'il faut :

- système de suivi de bogue;
- liste de diffusion;
- un wiki pour organiser une communauté;
- un nouveau site pour annoncer les nouveaux développements, ..

Démarquez vous des autres Fork!

Déterminez **pourquoi** les gens devraient **utiliser votre Fork** !

- Est ce que la liste de diffusion de votre projet est vivante quand celle du projet d'origine est morte ?
- Quelles fonctionnalités comptez vous ajouter / bug comptez vous réparer ?
- Est ce que vous comptez fusionner avec d'autres Fork ? Si oui, lesquels?

💭 Présentez tout ce qui se rapport à "Pourquoi le votre et pas le projet d'origine ou un autre Fork ?"

❓ Quelles actions à entreprendre en priorité ?

Redémarrer les développements !

Prenez les développements en main vous-mêmes

Communiquez sur les mises à jours et les progrès réalisés

Pourquoi ?

- Personne ne trouverait cela sérieux !
- Personne ne voudrait rejoindre un projet où le développeur principale ne prend pas le temps de poster des mises à jours !

💭 Prenez l'initiative, n'attendez pas que d'autres fasses le travail à votre place ! Ces simples actions vous permettront de trouver plus facilement des développeurs prêts à vous aider !

Faites vous de la publicité

Ce n'est pas aussi difficile que cela en a l'air, cela peut commencer par :

- discuter du projet sur la liste de diffusion
- ajouter des commentaires aux articles du projet d'origine
- écrire un article sur la version que vous développez

Faire passer le message que votre Fork existe et supplante le projet d'origine
Objectif : faire que les personnes qui cherchent le projet d'origine comprennent que les développements se font maintenant sur vôtre projet !

⚠️ Ne démarrez pas trop tôt la communication sur votre projet ! Ne faites pas de publicité sur vos **intentions**, mais sur vos **réalisations** !

Créez votre communauté

La dernière étape consiste à construire la communauté du projet :

- publiez fréquemment des articles sur votre site / blog;
- corrigez régulièrement les bugs;
- impliquez votre communauté dans le projet;
  💭 demandez de l'aide lorsque vous bloquez sur quelque chose
- répondez au mieux aux questions sur la liste de diffusion / irc;
- répondez aux commentaires sur votre blog.

=> *N'attendez pas que les autres fassent le travail pour vous !*

II.1 - Un petit point avant de commencer

Les apparences comptent !

Un logiciel libre doit acquérir des développeurs et des utilisateurs !
Il est donc important de :

- gérer correctement l'interaction entre ces deux profils;
- proposer les informations utiles au bon public (le degré de détail proposé, la documentation utilisateur / développeurs, ...)

💭 L'apparence est souvent dénigrée par les développeurs (amour du fond sur la forme)

La présentation d'un projet fait partie des situations où la forme compte !
Exemple : l'apparence du site Web d'un projet

Les apparences comptent !

C'est la première impression qu'aura un nouveau visiteur sur le projet :

- avant que le contenu du site ne soit compris;
- avant que le texte ne soit lu;
- avant que les liens ne soient suivis !

En quelques seconde, le visiteur détermine l'investissement dans la présentation du projet
-> **impression qui se répercute sur le reste du projet par association**

Dégager une aura de préparation => *"Votre temps ne sera pas perdu"*

https://github.com/Sci-Hook

II.2 - Démarrage d'un nouveau projet

Exprimer votre objectif au monde

Une des premières tâches à réaliser : transformer une vision privée en vision publique compréhensible !

Écriture d'une note avec la mission du projet (README) :

- qu'est ce qu'il fera / ne fera pas ?
- quelles seront ses limitations ?

Tâche usuellement simple à traiter et qui permet de :

- **clarifier** la vision du projet;
- **relever des hypothèses tacites**;
- **relever des désaccords** sur le projet.

Exprimer votre objectif au monde

Tâche souvent perçu comme laborieuse par les développeurs car :

- documente ce qu'ils savent déjà (README, document de conception, ...)
- n'ont pas besoin d'une arborescence de code soigneusement organisée

Tâche nécessaire pour les nouveaux arrivants, qu'ils soient utilisateurs ou développeurs (manuel d'utilisation, fonctionnalités prévues, ...)

⚠️ Il n'est pas nécessaire d'avoir tout cela en même temps !
-> tout faire *"aux petits oignons"* prendrait un temps prohibitif
-> d’autres personnes peuvent vous aider à réaliser ces tâches

Exprimer votre objectif au monde

Mais il est **nécessaire** d'investir suffisamment de temps dans la présentation
💭 Permet aux nouveaux venus de passer l'obstacle initial de la méconnaissance

Objectif : minimiser la différence entre :
– la quantité d'énergie investie;
– le fait de recevoir quelque chose en retour.
Plus la différence est faible, mieux c'est !

=> Première tâche pour votre projet : amener cette différence à un niveau qui encourage à s'impliquer !

Nous allons maintenant voir des Aspects important du démarrage d'un projet open source

💭 Présentés dans un ordre proche duquel un nouvel utilisateur pourrait les rencontre

💭 L'ordre dans lequel vous traitez ces points ne doit pas nécessairement suivre celui présenté. L'important est de couvrir chacun des points que nous allons voir ou alors vous devez accepter les conséquences associées !

Choisir un bon nom de projet

Se mettre à la place de quelqu'un qui vient d'apprendre l'existence du projet !

Les points auxquels il faut être sensible sur le nom:

- donne une idée sur ce que fait le projet;
- facile à retenir (privilégier l'anglais);
- ne partage pas le nom d'un autre projet;
- nom de domaine disponible, e.g. www.myproject.com, www.myproject.net, ...
- nom d'utilisateur disponible sur site de blog, ...

💭 Il est intéressant d'avoir le même nom de projet sur plusieurs espaces pertinents, e.g. Twitter, Github, Gitlab, ... car cela facilite la recherche

Avoir une description claire

Une fois le site trouvé, les nouveaux arrivants vont rechercher une description rapide du projet / feuille de mission

⚠️ **Détermine en ~30 secondes si le projet intéresse ou pas !**

Cette information doit être accessible depuis la première page du projet

Un bon exemple de description est le projet Git (https://git-scm.com/):
"Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency."

💭 Une phrase et beaucoup de points essentiels sont listés.
💭 Faire appel au connaissances préalables du lecteur !

Déclarer que le projet est libre ou open source

Les lecteurs intéressés voudront avoir plus de détails, mais avant tout, être sûre que le projet est libre ou open source !
💭 Indiquez dès le début que le projet est libre ou open source

Donnez la Licence exacte du projet

Exemple : l'éditeur d'image GIMP (https://www.gimp.org/)

Arrivez là, le nouveau visiteur a déterminé en ~ 1 minute s'il veut consacrer plus de temps à découvrir votre projet ou non

Fonctionnalités et dépendances

Les lecteurs intéressés voudront savoir:

- la liste (claire) des fonctionnalités supportées
  💭 si ce n'est pas encore complet, signalé le comme *WIP* ou *PLANNED*
- l'environnement nécessaire pour exécuter le logiciel

Exemple : projet Git
*" Git is **easy to learn** and has a **tiny footprint with lightning fast performance**.*
*... with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**"*

💭 Certains projet listent aussi es dépendances de dev / test / doc / ...
=> De nouveau le lecteur se fait une idée rapide *"ça peut marcher pour moi ?"*

État du développement

Une fois ferré, l'utilisateur voudra savoir comment le projet se porte:

- nouveau projet : écart entre la promesse faite et la réalité
- vieux projet: activité de maintenance / release / bugfix

Pour répondre à ces questions on peut avoir :

- Une page de statuts avec :
  - la liste des objectifs et besoins à court terme;
  - l'historique des précédentes releases;
  - la planification des releases futures.

État du développement

- Des compteurs / indicateurs embarqués dans la page du projet :
  - un panneau "annonces";
  - un flux twitter;
  - un historique des dernières releases;
  - l'activité autours de la résolution de bogues / issues.

💭 Les indicateurs servent de passerelle à plus d'informations !

⚠️ Ne pas céder à la tentation d’exagérer ou de gonfler l'état du développement
⚠️Ne pas avoir peur de ne pas être *" prêt"* / qu’il faille *"essuyer les plâtres"*

État du développement

Pourquoi ne faut-il pas craindre de dire que le projet est en version *alpha* et qu'il contient des bugs connus ?

État du développement

Pourquoi ne faut-il pas craindre de dire que le projet est en version *alpha* et qu'il contient des bugs connus ?

- N'effrayera pas les développeurs dont vous pourriez avoir besoin à cette étape !
- N'attirera pas des utilisateurs avant que le projet ne soit prêt !

⚠️ Une réputation de logiciel instable / « bogué » est difficile à retirer une fois acquise !

💭 L'honnêteté est rentable sur le long terme

État du développement

Noms de version utilisés pour qualifier l'état du développement:

Version alpha:

- usuellement la première version mise à disposition aux utilisateurs;
- contient toutes les fonctionnalités attendues;
- contient des bugs connus;
- n'a pas une API déjà stabilisée !

Objectif: avoir des retours utilisateurs pour savoir sur quoi travailler

État du développement

Noms de version utilisés pour qualifier l'état du développement:

Version bêta:

- API stabilisée (pas de changement sauf si absolument nécessaire);
- bug sérieux corrigés;
- pas encore assez testée pour garantir un passage en production.

Objectif : devenir la version officielle ou tirer des retours utilisateurs pour atteindre rapidement une version officielle

Téléchargement / Compilation / Installation du projet

Logiciel téléchargeable sous forme de code source dans un format standard !

Pas besoin d'un exécutable au début
💭 Si c'est le cas à cause de la procédure de compilation (beaucoup de dépendances, prérequis de compilation, ...) c'est mal parti ...

La procédure de compilation et d'installation doit être la plus standard possible
⚠️ Sinon, risque de perdre des utilisateurs / développeurs potentiels

💭 Besoin de standardiser les procédures de téléchargement, compilation et installation ! Action pénible mais qui est très rentable sur le long terme

Version des sources misent à disposition

Le téléchargement des sources est :

- **suffisant** pour les **utilisateurs** (et encore ...);
- **insuffisant** pour les **développeurs** : déboguer, ajouter des fonctionnalités, ...

Solution possible : utilisation de *nightly snapshot* (mais pas assez fin)

Besoins de pouvoir :

- accéder aux dernières sources en temps réel;
- soumettre des changements basés sur ces sources !

Solution : mise à disposition d'un dépôt en ligne avec visibilité publique

Gestionnaire de bogues

Qu'est ce qu'un gestionnaire de bogue ?

Gestionnaire de bogues

Qu'est ce qu'un gestionnaire de bogue ?

- utile pour l'interaction utilisateurs / développeurs
- preuve que le projet est pris au sérieux

Le nombre de bugs rapportés est proportionnel à :

- le nombre de défauts du logiciel;
- le nombre de personnes utilisant le logiciel;
- la facilité de signaler des bugs.

❓ Pourquoi ces facteurs sont importants ? lesquels le sont le plus ?

Gestionnaire de bogues

Tout logiciel de taille / complexité suffisante a un nombre arbitraire de bugs attendant d'être découverts ! **La vrai question est** : *"dans quelle mesure le projet réussira à enregistrer et hiérarchiser ces bugs ?"*

Avoir une base de données de bugs bien maintenue (i.e. bugs résolus rapidement, duplicatas unifiés, ...) et conséquente fait bonne impression !
⚠️ Ne s'applique pas aux nouveaux projets qui auront peu de bugs à leurs actifs

💭 Les gestionnaires de bugs sont souvent utilisés pour monitorer d'autres choses: tâches en attentes; proposition d'améliorations; refactoring; proposition de feature; ...
Exemple: https://github.com/python-gitlab/python-gitlab/issues/ (1447, 1602)
💭 Présentation de l'utilisation de templates dans gitlab / github

Moyens de communication

Quels moyens de communication pouvez-vous mettre à disposition ?

Moyens de communication

Quels moyens de communication pouvez-vous mettre à disposition ?

- liste de diffusion;
- sallon de discussion (discord, slack, matrix, talkspirit);
- sallon IRC.

Cela **rassure le nouveau visiteur**:

- possibilité de contacter facilement les développeurs;
- obtention de réponses (rapide) à ces questions !

⚠️ Différenciez les moyens de communication utilisateurs / développeurs (pas forcément nécessaire au démarrage d'un projet)

Guide de développement

Quelle est la première chose que fera un nouveau contributeur ?

Guide de développement

Une des premières choses que fera un nouveau contributeur : rechercher la procédure à suivre pour le développement

Objectif: Faciliter l'entente entre développeurs
💭 Explique l'interaction entre développeurs et avec les utilisateurs

Éléments basiques à retrouver:

- comment remonter des bogues ? proposer des patchs ?
- indications sur le développement; prise de décision au sein du projet (cf IV); pointeurs vers des forums d'interactions avec d'autres développeurs.

Exemple de fichier CONTRIBUTE.rst

Documentation

Besoin d'une documentation, même partielle, dès le début !

Tâche souvent mise de côté (surtout au début d'un projet) car :

- nécessite de repasser dessus régulièrement;
- utile pour les lecteurs et pas les écrivains;
- difficile de se mettre à la place d'un nouveau lecteur.

Contenu nécessaire de la documentation :

- comment configurer et faire fonctionner le logiciel ?
- comment réaliser quelques actions standards ?

Documentation

Besoin d'utiliser un format standard et facile à éditer, e.g. texte brut, rst, md, ...
Objectif : incorporer facilement et rapidement les retours des lecteurs

Une bonne documentation doit au minimum valider les critères suivants:

- préciser le niveau d'expertise que doit avoir le lecteur;
- décrire clairement et en détail comment configurer le logiciel;
- expliquer comment lancer les tests pour valider la configuration;
- donner des exemples (style tutoriels) pour réaliser quelques actions;
- préciser les endroits où la doc est encore incomplète;
- (extra) préciser les lacunes connues de l'outil.

Doit être **disponible en ligne** et dans la **distribution** téléchargeable du logiciel

Documentation développeur

Documentation développeur != Guide de développement

Objectif : Faciliter la compréhension du code
💭 Plus orientée technique que *"sociale"* !

❓ Si la quantité de temps à investir dans la documentation est limitée, qu'elle documentation doit être privilégiée ?

Documentation développeur

Documentation développeur != Guide de développement

Objectif : Faciliter la compréhension du code
💭 Plus orientée technique que *"sociale"* !

❓ Si la quantité de temps à investir dans la documentation est limitée, qu'elle documentation doit être privilégiée ?
💭 **Avis personnel**: la documentation utilisateur car, d'une certaine manière, c'est aussi une documentation développeur

https://github.com/pylint-dev/pylint/tree/main/doc/development_guide

Supports visuel

N’hésitez pas à fournir des supports visuel autours du projet !
💭 Une preuve que votre logiciel fonctionne !

A mettre en place pour les projets avec:

- une interface graphique (en anglais GUI "Graphical User Interface");
- production de graphiques;
- réalisation de sorties distinctes.

Types de supports : capture d'écran, vidéos de quelques minutes, fichiers exemples, ...

II.3 - Le choix de la licence

Guide rapide pour le choix d'une licence

Le choix de la licence traduit des intérêts juridiques, techniques et commerciaux
⚠️ Adopter la licence la plus adaptée au projet

💭 Existence de centaines de licences (situations particulières les ayant justifiées) https://fr.wikipedia.org/wiki/Liste_de_licences_libres

Guide rapide pour le choix d'une licence

Obligations différenciable sur lesquelles se pencher :

- les clauses copyleft;
- les clauses relatives aux usages par la communauté;
- les clauses relatives aux brevets;
- les mesures techniques de protection;
- la tivoïsation : système incluant des logiciels libres mais qui utilise le matériel électronique pour interdire aux utilisateurs d'y exécuter des versions modifiées (e.g. TiVo, Tomtom);
- la langue utilisée, ..

Les licences *"fais ce que tu veux"*

Connaissez-vous de telles licences ?

Les licences *"fais ce que tu veux"*

Connaissez-vous de telles licences ?

La licence **MIT (https://mit-license.org/) et ses dérivées**:

- licences de logiciel permissives;
- affirment un droit d'auteur nominal;
- droit illimité d'utilisation, copie, modification, distribution, vente;
- compatiblent avec de nombreuses autres licences;
- spécification explicite de non garantie sur l'utilisation du logiciel.

Obligation : incorporer la notice de licence et de copyright dans toute copie

💭 N'assure pas la persistance des 4 libertés d'un logiciel libre et permet la cohabitation avec du code propriétaire

La licence GPL

Probablement la licence de logiciel libre / open source la plus reconnu de nos jours !

- familier pour beaucoup d'utilisateurs / développeurs
- limite le temps consacré à la lecture / compréhension de la licence

Le code peut être copié et modifié sans restriction

Les copies et les travaux dérivés, i.e. version modifiées, doivent être distribués sous la même licence ! Pas de restrictions en moins ou supplémentaires !
⚠️ Interdit l'utilisation du code dans des programmes propriétaires

💭 Assure la persistance des 4 libertés d’un logiciel libre et interdit la cohabitation avec du code propriétaire

Appliquer une licence au projet

Comment met-on en œuvre une licence sur un projet ?

Appliquer une licence au projet

Comment met-on en œuvre une licence sur un projet ?

Énoncer clairement la licence utilisée sur la page principale de votre projet et l'inclure **DANS** votre logiciel !
💭 Son nom et un lien vers le texte complet

- Écrire le texte complet de la licence dans un fichier LICENSE / COPYING
- Inclure un commentaire en début de chaque fichier où trouver :
  - la date du droit d'auteur;
  - le titulaire;
  - le nom de la licence;
  - un lien vers le texte complet de la licence.

Exemple d'une notice type GNU GPL

“ Copyright (C)

This program is free software: you can redistribute it and or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of FITNESS FOR A PARTICULAR PURPOSE or MERCHANTABILITY. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see https://www.gnu.org/licenses/. ”

II.4 - Instaurer une ambiance collaborative

❓ Comment feriez-vous pour instaurer une ambiance collaborative dans un projet libre / open source ?

❓ Comment feriez-vous pour instaurer une ambiance collaborative dans un projet libre / open source ?

💭 Une bonne approche consiste à avoir de *"bons"* précédents !

Pourquoi créer des bons précédents ?

Les personnes qui joignent un groupe unifié par un but commun recherchent des comportements / normes sociales qui les marqueront comme membres

Créer des précédents tôt permet de faire en sorte que ces comportements soient utiles au projet et auto-entretenus !

- Situation : Développeur travaillant seul chez lui
- Comportement : Faire en sorte que le développeur ait l'impression de travailler avec d'autres dans la même pièce
- Intérêt : Augmente le temps que le développeur passera sur le projet

💭 Google et ses cafétéria gratuite, salles de sports et de repos, ...

Exemple de bon comportement

❓ Comment traiter les questions difficiles relatives au projet ?

Exemple de bon comportement

❓ Comment traiter les questions difficiles relatives au projet ?

💭 De mon expérience, une mauvaise approche consiste à discuter en privé entre membres fondateurs. Alternative envisagée : la **discussion en publique** !

Quels sont les désavantages associés à la discussion en publique ?

Exemple de bon comportement

❓ Comment traiter les questions difficiles relatives au projet ?

💭 De mon expérience, une mauvaise approche consiste à discuter en privé entre membres fondateurs. Alternative envisagée : la **discussion en publique** !

Quels sont les **désavantages associés à la discussion en publique** ?

- le délais inhérent à la communication par courriers
- le temps nécessaire à un consensus
- l'embarrât de traiter avec des nouveaux arrivants : questions naïves; auto-proclamés *"génie"* qui pensent comprendre; incompréhension du fait de souhaiter seulement résoudre un problème X qui leur semble appartenir au sous-ensemble Y; ...

Exemple de bon comportement

Au vu des inconvénients, pourquoi passer sur un mode de discussion publique ?

Exemple de bon comportement

Au vu des inconvénients, pourquoi passer sur un mode de discussion publique ?

- Prendre des décisions importantes en privé est un répulsif à contribution !
- Effet bénéfique principale : quelqu'un proposant une idée non envisagée
- Effets bénéfiques secondaires :
  - discussions et conclusions archivés pour toujours
    💭 évite au futur discussions de retracer les mêmes étapes
  - entraîner et éduquer les nouveaux développeurs
    💭 possibilité de lire la conversation sans avoir besoin d'y participer
  - amélioration de votre capacité à expliquer des problèmes techniques

Autre exemple de bon comportement

Avez vous une autre idée de bon comportement (relationnel) ?

Autre exemple de bon comportement

Avez vous une autre idée de bon comportement (relationnel) ?
⚠️ Ne pas tolérer la grossièreté !

Exemple de situation : un commentaire technique mixé avec une critique sur une autre développeur
💭 Mauvaise solution : ignorer la partie agressive du commentaire

Pourquoi être vigilant sur ce point ?

Autre exemple de bon comportement

Avez vous une autre idée de bon comportement (relationnel) ?
⚠️ Ne pas tolérer la grossièreté !

Exemple de situation : un commentaire technique mixé avec une critique sur une autre développeur
💭 Mauvaise solution : ignorer la partie agressive du commentaire

Pourquoi être vigilant sur ce point ?
⚠️ Transformation de débats techniques en conflits de personnes
💭 L'échange par courrier électronique / commentaire peut amener à des comportements que l'on aurait jamais en face à face !

Ne pas tolérer la grossièreté !

Si un mauvais comportement se produit, reprendre la personne et lui indiquer le comportement à tenir.
Exemple de bonne réponse à un comportement grossier :

> “ Tout d'abord, s'il vous plaît, limitons les commentaires ad hominem; par exemple, appeler la contribution de A. « naïve et ignorante ». C'est peut-être vrai ou pas, mais dans les deux cas, ce n'est pas un moyen d'avoir une discussion. Si la contribution de A. présente des défauts, signalez-les et ils seront corrigés ou nous étudierons une alternative. Je suis sûr que B. ne voulait pas insulter personnellement A., mais la formulation était malheureuse, et nous essayons de garder les choses constructives ici.
>
> Passons maintenant à la contribution ... ”

Ne pas tolérer la grossièreté !

💭 En dernier recours, il vous est possible de retirer la personne de la liste de diffusion / de lui retirer la possibilité de pousser des nouveaux patchs

Objectif : faire en sorte que cette bonne étiquette soit perçu comme l'un des comportement attendus de la communauté

Effet bénéfique principale : éviter que des développeurs soient chassés du projet

- Stratégie de survie sur le long terme
- Une fois adoptée, auto-entretenu par les contributeurs

Communication sur les bon comportements relationnels

Il existe d'autres mauvais comportement relationnels desquels il faut se prémunir !

Exemple de situation gênante : utilisation d'un langage ou d'images sexualisées

💭 L'objectif était de *"blaguer"* pour avoir une atmosphère bon enfant et accueillante

⚠️ N'a pas forcément l'effet escompté car peut être offensant

Communication sur les bon comportements relationnels

De quelle manière pouvez-vous simplement communiquer sur les bons comportements relationnels ?

Communication sur les bon comportements relationnels

De quelle manière pouvez-vous simplement communiquer sur les bons comportements relationnels ?

Un **code de conduite** :

- permet de créer un environnement positif
- aide à éviter / résoudre les problèmes interpersonnels

https://www.contributor-covenant.org/fr/version/2/0/code_of_conduct/code_of_conduct.txt
->Eclipse, Facebook, Git, Gitlab, Linux

Autre exemple de bon comportement (code)

Avez-vous des idées sur des bon comportements à retrouver concernant la production de code ?

Autre exemple de bon comportement (code)

Avez-vous des idées sur des bon comportements à retrouver concernant la production de code ?

La revue de code est un pratique saine qui peut s'appliquer sur:

- du code déjà présent depuis un moment;
- du code récent issue d'un ou plusieurs commit

Objectif : rechercher des bugs et potentielles améliorations

Pourquoi se focaliser sur les apports récents plutôt que sur les anciens ?

Autre exemple de bon comportement (code)

Avez-vous des idées sur des bon comportements à retrouver concernant la production de code ?

La revue de code est un pratique saine qui peut s'appliquer sur:

- du code déjà présent depuis un moment;
- du code récent issue d'un ou plusieurs commit

Objectif : rechercher des bugs et potentielles améliorations

Pourquoi se focaliser sur les apports récents plutôt que sur les anciens ?
Social : intérêt max du contributeur; importance de la contrib; point d'entrée

☁️ La revue de code est attendue et régulière !

II.5 - Gérer l'ouverture du projet

Déclarez le projet comme libre dès le début!

⚠️ La difficulté pour ouvrir un code privé augmente avec le temps

Quelles peuvent être les raisons de cette difficulté ?

Déclarez le projet comme libre dès le début!

⚠️ La difficulté pour ouvrir un code privé augmente avec le temps

Quelles peuvent être les raisons de cette difficulté ?

- Contributeurs habitués à travailler en environnement **propriétaire**
  - nouvelles méthodes de travail;
  - questions distrayantes venant d'étrangers;
  - exposition du code à des étrangers qui vont *"juger"* le résultat final
  - documentation toujours inadéquate
  - manque d'assurance exacerbé

Déclarez le projet comme libre dès le début!

- Prise de décision(s) incompatible(s) avec le libre !
  - situations : correctif de bug (court terme) vs compatibilité libre / open source (future)
  - en production *"situation urgente"* >>> *"élégance"*

⚠️ Accumulation d'une dette technique qu'il faudra payer à l'ouverture !

Avez-vous des exemples d'une telle dette technique ?

Déclarez le projet comme libre dès le début!

- **Exemples de dette à payer** :
  - exemples basés sur des données issues d'informations confidentielles
  - commentaires exprimant des sentiments négatifs sur les requêtes utilisateurs
  - problème de licences dû à des dépendances sur d'autres bibliothèques
  - problème de compilation hors environnement interne
  - rapport de bug avec des informations critiques ne devant pas être publique
  - documentation écrite dans un format interne, ...

Déclarez le projet comme libre dès le début!

Comment pouvez-vous gérer les étapes de nettoyage ?

- Retravailler l'historique entier du projet ?
- Nettoyer la dernière version et la rendre libre / open source ?

Solution mécaniques :

- Données confidentielles -> Sortie d'une version écrémée du projet
- Dépendances propriétaires -> Rercherche d'alternative libre / open source
- ...

⚠️ Le coût de la dette technique peut être non négligeable et conduire vers une *"fausse"* ouverture du projet !

Annoncer le projet

Une fois le projet présentable : annoncez le !

Processus simple :

- soumettre le projet sur un site d'hébergement, e.g. https://sourceforge.net/
  💭 sites scrutés pour leurs projets open sources
- courrier électronique à une liste de diffusion pertinente

⚠️ Il n'est pas nécessaire d'avoir du code fonctionnel pour démarrer un projet libre / l'open source !

- un simple document de conception peut suffire;
- ceci dit, avoir du code est une bonne règle de lancement !

Exemple de courrier électronique

To: discuss@lists.example.org
Subject: [ANN] Elo un projet de gestion de versions
Reply-to: dev@elo.org

Ceci est un message unique d'annonce de création du projet Elo, un outil gestion de versions basé sur Git et Gitlab. Il est doté d'une API simple et conçu pour les utilisateurs de code de simulation désirant pouvoir versionner et partager simplement leurs études. Elo fonctionne, est toujours en développement actif, et recherche des développeurs et des testeurs !
Site du projet: https://www.elo.org

Fonctionnalités :

- configuration assistée (paramètrage git & gitlab);
- gestion simple de dépôt par snapshot;
- (prévu) gestion de collaboration autours du dépôt;
- (prévu) création de groupes et gestion des collaborations.

Pré-requis : python 3 ou supérieur; Git; accès à un serveur Gitlab.

Merci,
M. X