10 méthodes pour rendre heureux un développeur en entreprise, la 7ième vous étonnera

Speaker : Cyril Lakech (Axa), Romain Linsolas (SG)
Format : Conférence
Date : 6 avril 2017
Slides : https://linsolas.github.io/devoxx-france-2017/index.html#/

Aujourd'hui, les grandes sociétés sont capables de créer un endroit propice aux développeurs où ils vont pouvoir s'épanouir.
D'après stackOverflow, 62% des dévs sont prêts à quitter leur entreprise.

1. Investir dans le recrutement

Le problème de recrutement : 2ième cause d'échec d'un projet (la 1ière étant que le projet ne sert à rien).

- Partage de la vision
- Description du cadre
- Parler salaire. Il existe un marcher. Le salaire proposé doit être en adéquation avec le marcher.
- Parler du code : les outils, les compétences en termes de skill, les pratiques (pair programming). Nécessite qu'un développeur soit présent au recrutement, même si perte du temps sur le projet

Offre à éviter : recherche de profils impossibles car trop skillés
A éviter également : faire coder les candidats sur du tableau blanc à la virgule prêt.
Il existe des outils : exemple de Coding Game utilisé chez Axa.
Le tableau blanc peut être utilisé pour discuter avec le candidat à partir d'une problématique donnée.
Le plus simple : proposer au développeur de s'immerger dans l'équipe pendant ½ journée, l'impliquer dans une séance de code review ou de pair programming.
Autre possibilité : boire une bière de manière informelle afin de mieux le connaître

2. Avoir une perspective de carrière

Faire rester le développeur. Lui proposer une expérience durable.
Evolution possible : expertise technique, leadership, architecture, multi-technos, multi-métiers, management.

La quête d'une expérience extra ordinaire.
Faire varier les plaisirs. Les faire travailler sur des projets stratégies, innovants.
« Développeurs, apprendre est notre métier », Damien Cavaillès
Donner du temps pour que les développeurs puissent se former.
Encourager les développeurs à donner des cours et des présentations. Permet aux développeurs de savoir transmettre leur savoir aux autres développeurs.

3. L'environnement du développeur

L'Open Space n'est pas adapté aux développeurs. Déconseillé à cause du bruit et des distractions. Pas facile à faire. A défaut, on peut personnaliser les Open Spaces.
Quid des bureaux flexibles avec des endroits plus détendus ? Approuvés par les Speakers.
Espaces de détente autre que la machine à café.
Un bureau adapté : bureau debout ou pas + chaise de qualité
Disposer d'un bon matériel. Ordinateurs puissants. Mauvais matériel = 1h/j de productivité en moins ⇔ 30 jours / an.
Avoir de bons logiciels. De même, on peut perdre 1h par jour à cause de mauvais logiciel. Ne pas imposer les logiciels aux développeurs. IntelliJ vs VSCode. S'adapter au développeur.
Référence aux 12 règles du test de Joël
Donner confiance au développeur : ne pas lui retirer les droits d'admin (exemple du frein à main sur la Ferrari).
Bien maitriser le télétravail : séparer le privé du pro, éviter les distractions, se mettre des limites.

4. Organisation du travail

Feature teams répandues par Spotify. Une équipe durable aux talents complémentaires chargée d'implémenter une feature dans son ensemble.
Création de plus petites équipes, les Pizza Team.
Mouvement Devops : casser les murs entre Développeur et Opérateur. Le développeur doit être responsable de la mise en production de son code.

5. Software Craftmanship FTW

Le développeur veut toujours apprendre et s'améliorer.
Le manifeste Softwaire Craftmanship vise à patcher le manifeste Agile : Ok pour des logiciels qui fonctionnent, mais seulement s'ils sont bien conçus.
Pour mettre en place le Software Craftmanship : code review, entrainement, pattern d'architecture (CQRS/Event Sourcing) …
Demande de la persévérance.

6. L'ouverture technologique

Image du développeur qui veut essayer toutes les technos. D'où la création de murs pour limiter la propagation des technos. Les développeurs s'en vont.
Radar techno de Thoughwork avec 4 catégories : Excel, Reinforce, Assess, Deprecated.
Les développeurs souhaitant utiliser une nouvelle techno vient la présenter à une assemblée : analyse force / faiblesse, ROI, débats …. Puis projet pilote pour tester la techno.
A la fin du POC, feedback devant la même assemblée. Soit on la déprécie soit on la passe en Reinforce afin de la déployer sur d'autres projets.

7. Ne pas subjectiver la propédeutique liée à la transformation ~~digitale~~ numérique

Faire plaisir aux dévs, ce n'est pas que les pizzas, le café gratuit et le baby foot

8. S'organiser en communautés

Comment faire pour qu'une feature team puisse apprendre d'autres technos ? Développer des communautés transverses / internes. Permet de trouver de l'aide, de se rendre utile.
Exemple de communautés internes : Java, .NET, Crafts(wo)men, NodeJS …
Nécessité d'allouer du temps au développeur + mise à disposition d'outils collaboratifs

9. Contribuer à l'Open Source

Tous les développeurs l'utilisent. Les grandes sociétés cherchent de plus en plus à contribuer à l'Open Source, voir à reverser du code. Axa et SG en font partie.
Premier pas : inner source => rendre son code accessible en interne. Permet de la réutilisation.
Société Générale FIR (Fast Incident Response) : plateforme de gestion d'incident de cyber-sécurité

10. Participer à des évènements

Inviter un speaker à l'heure du déjeuner au cours de BBL.
Le Bagger s'exerce comme speaker, teste un nouveau sujet, se fait de la publicité et agrandit son réseau.
Participer aux Meetups : mini-conférence du soir sur une thématique donnée
Le hackathon : challenge de code pendant 1 à 3 jours sur une thématique donnée. Idées libres. Ouvert ou interne. En mode startup. Attention : le hackathon ne permet pas d'avoir de la main d'œuvre gratuite pendant 3 jours.
Conférences internes : organiser sa conférence, ouvert ou interne. Reste assez exceptionnel car pas facile à organiser.
Sponsoring : permet de se faire connaître. Sponsorer une conférence, des évènements récurrents (JUG) ou des évènements ponctuels (Meetup)