The NeceMoon : L'Album

Le Rêve à Temps Plus Que Partiel, le Travail à Temps Carrément Complet

Chapitre 6
Génie Logiciel : Quelques Notions Remarquables

Avec la participation de Darren Mart
et Jean-Patrick Ehouman

Par **Necemon Yai**

Première édition

Album.NeceMoon.com

Evasium ®

The NeceMoon : L'Album

Le Rêve à Temps Plus Que Partiel, le Travail à Temps Carrément Complet

Par Necemon Yai

Première édition

Publié par Evasium ®

Avril 2018

Londres, UK

Album.NeceMoon.com

© Necemon Yai

necemon@gmail.com

www.necemonyai.com

Version 1.0.7.186

A celles que j'ai perdues, à celles que j'ai retrouvées

Une de perdue, dix de retournées

Table des Matières

Introduction

A propos de l'Album "The NeceMoon" : ça c'est quoi ça encore ?

The NeceMoon™ est un Blog à propos de Technologie et de Stratégie. L'Album est le Best-Of, une compilation des articles les plus populaires sur The NeceMoon™.

L'objectif principal de The NeceMoon™, c'est de partager des analyses judicieuses sur divers sujets, pour permettre aux autres d'apprendre de mes erreurs et succès, et j'ose espérer, faciliter la tâche à celui ou celle qui entreprend des choses similaires.
L'objectif principal de l'Album est de favoriser un accès optimal à ce contenu. Le format Blog ne rend pas toujours justice au contenu techno-stratégique. A la base, les Blogs ont été conçus dans un esprit journalistique et sont mieux adaptés aux évènements chronologiques et aux discussions (plus ou moins banales) sur l'actualité. Même si l'utilité et l'importance d'une analyse perdurent dans le temps, l'empilement progressif des articles la rend quasiment introuvable et difficilement consultable.
C'est pourquoi les meilleurs articles ont été repris, revus et agencés dans un ordre logique correspondant mieux à celui d'un livre. L'Album est gratuit.

The NeceMoon™ est disponible sur NeceMoon.com (ou necemonyai.com/blog)
L'Album peut être téléchargé en entier dans différents formats à l'adresse Album.NeceMoon.com (ou necemonyai.com/blog/page/L-Album.aspx). Les formats PDF, EPUB, MOBI/AZW3/KF8 (Amazon Kindle) et MP3 y sont disponibles. Aussi, les différents chapitres et volumes peuvent être téléchargés indépendamment/séparément, selon tes intérêts.

A propos de l'Auteur : qui est Necemon Yai ?

Je suis un ingénieur logiciel à fond dans les technologies Microsoft .NET. Développeur à temps complet. Artiste digital, stratégiste, essayiste et entreprenerd à temps partiel. Je me suis spécialisé en informatique depuis NIIT, Christ University et Swansea University (Master en Génie Informatique). A l'heure où j'écris ces mots, j'ai travaillé pour l'une des principales entreprises de E-Commerce en Europe, pour l'un des groupes financiers les plus importants d'Angleterre, pour la multinationale General Electric et pour quelques startups technologiques dont tu n'as probablement jamais entendu parler.
Depuis une dizaine d'années , je tiens le blog "The NeceMoon", où je décris mes expériences, mes observations et mes réflexions. Je parle surtout de Technologie et de Stratégie. Je partage ici mes articles les plus populaires.

A propos des Contributeurs : qui est dans ton Conseil de Guerre ?

J'ai invité les meilleurs auteurs de mon réseau à inclure des contributions dans ce livre, notamment certaines de leurs pensées les plus pertinentes en termes de technologie et de stratégie. Il s'agit en l'occurrence de, Ahou l'Africaine, Antoine Mian, Cyriac Gbogou, Darren Mart, Edith Brou, Holty Sow, Israël Yoroba, Jean Luc Houédanou, Jean-Patrick Ehouman, Karen Kakou, Monty Oum, Nanda Seye, Nnenna Nwakanma, Olivier Madiba, Vanessa Lecosson et Yehni Djidji.
Au niveau de leurs textes respectifs, tu peux retrouver des liens vers leurs pages Web. De plus, la plupart de ces contributeurs et contributrices se révèlent et te livrent quelques tactiques dans des interviews exclusives que tu trouveras aussi dans ce livre.

A propos de Toi, cher lecteur, chère lectrice : à qui s'adresse ce livre ? Qu'est-ce que tu y trouves ?

Dans L'Album, il y a 9 chapitres organisés en 2 volumes. Chaque chapitre aborde un thème différent. Tu n'es pas obligé(e) de tout lire. Si tu es une personne qui s'intéresse de près ou de loin à un ou plusieurs de ces thèmes, tu apprécierais éventuellement le(s) chapitre(s) concerné(s) :

Volume 1 : Clair de Lune (softcore)
- Chapitre 1 : Stratégies et Tactiques
- Chapitre 2 : Marketing Digital et Visualisations Web
- Chapitre 3 : Carrières et Emergence
- Chapitre 4 : Bons Plans et Petites Victoires Faciles
- Chapitre 5 : Dégammage - Délires et Réflexions Rapides

Volume 2 : Pleine Lune (hardcore)
- Chapitre 6 : Génie Logiciel - Quelques Notions Remarquables
- Chapitre 7 : Programmation informatique avec C# .NET
- Chapitre 8 : Prototypes Epiques, Projets Classiques, Genre Historique
- Chapitre 9 : Recherches et Etudes de Cas

Si tu veux, tu peux télécharger et lire uniquement le(s) chapitres ou volume(s) qui t'intéressent.
Plusieurs formats sont disponibles sur Album.NeceMoon.com (ou necemonyai.com/blog/page/L-Album.aspx)
Tous les liens Web de ce document sont fonctionnels, n'hésite pas à cliquer dessus.

A Toutes Fins Utiles

Ce document contient le Chapitre 6 de l'Album : « Génie Logiciel - Quelques Notions Remarquables ». Si ça t'intéresse, 8 autres chapitres sont également disponibles. En fonction de tes intérêts, tu peux télécharger, (re)lire et partager chaque chapitre ou volume indépendamment/séparément. Les formats PDF, EPUB, MOBI/AZW3/KF8 (Amazon Kindle) et MP3 sont disponibles.

Il suffit de cliquer sur le(s) document(s) qui t'intéresse(nt) ci-dessous ou d'aller sur Album.NeceMoon.com (ou necemonyai.com/blog/page/L-Album.aspx).

L'Album est également disponible en Anglais à l'adresse TheAlbum.NeceMoon.com (ou necemonyai.com/blog/page/The-Album.aspx).

Chapitre 6

Génie Logiciel : Quelques Notions Remarquables

Avec la participation de Darren Mart et Jean-Patrick Ehouman

8 raisons pour lesquelles ça te ferait du bien d'être programmeur(euse)

Par Necemon

Si tu es programmeur(euse), il y a probablement beaucoup de raisons qui te motivent à faire ce que tu fais. J'espère que tu trouveras ici un peu plus de motivation.

Si tu veux être programmeur(euse), tu pourrais y trouver quelques raisons de plus pour opter pour cette voie extraordinaire.

Cependant, je suppose que j'écris ceci surtout pour ceux qui ne savent pas trop quoi faire dans leur vie. Voici quelques indices sur un job qui est amusant, utile et satisfaisant à plusieurs niveaux.

Voici les 8 raisons qui me viennent en tête et qui me font penser que tu aimerais être un programmeur(euse). J'espère que ceci t'inspire.

1. La programmation réalise tes rêves. Quand tu comprends la programmation, tu peux donner vie à tes idées en les appliquant à la vie réelle. tu peux littéralement créer des choses.

2. La programmation est la forme ultime d'art interactif. Tu peux faire des logiciels, sites web et des jeux pour que d'autres puissent jouer avec. Ainsi, tu peux leur parler de manière indirecte et ils peuvent te parler en retour. Aucune autre forme d'art n'est aussi interactive. Alors que le dessin, la peinture, le cinéma et la musique vont à l'audience (dans un sens), le code va dans les deux sens.

3. C'est le genre de job que tu peux faire de partout. Depuis ton canapé, de ta maison, du bureau, en voyage, quelque soit le pays ... Les seules choses dont tu as besoin sont un ordinateur et ton cerveau.

4. C'est facile à apprendre. Il y a des tonnes de ressources disponibles en ligne, beaucoup d'entre elles sont gratuites. Aussi, il y a beaucoup de communautés en ligne qui peuvent te soutenir à travers les forums, chats, courriels, etc.

5. Tu n'as à compter sur personne pour faire de la programmation. Tu peux avoir la chance de le faire pour une entreprise ou lors d'un programme de recherche. Même si ce n'est pas dans un travail d'entreprise, tu peux travailler avec une équipe. Et même si tu ne trouves pas une équipe qui te convient, tu peux toujours travailler par toi-même. Par ailleurs, tu n'as pas besoin de beaucoup d'argent pour commencer.

6. Tu ne peux pas t'en dégoûter. Tu n'as pas le temps de t'ennuyer. Les exigences de projets et la technologie évoluent si vite que tu auras toujours de nouveaux défis à relever.

7. C'est le domaine de la méritocratie. On sait que tu ne simules pas tes compétences. Tu sais ce que tu sais. Tu fais ce que tu peux et tu reçois une reconnaissance équitable pour ça.

8. C'est passionnant !

N.

Ecrire et Programmer, Finalement la Même Chose

Par Jean-Patrick Ehouman

Il y a 5 ans si l'on me disait que je pouvais tenir un blog, je ne l'aurai pas cru. En tant qu'ingénieur de développement, je passais plus de temps à concevoir et écrire des programmes. Mais de là à écrire des articles, à priori rien ne s'y prêtait.

Avec le recul, je peux dire que ces deux activités ont plus de points en commun que de points de différenciation. Lorsque vous écrivez, vous créez, vous remplissez un espace (le blanc de la feuille), vous donnez vie. De la même manière, lorsque vous concevez un programme, vous faites naitre un système qui sera utilisé au quotidien. Vous avez donc dans ces deux cas, le besoin de vous substituer au lecteur ou à l'utilisateur de vos programmes.

Ce jeu de substitution vous emmène à faire preuve d'imagination et de créativité de la même manière qu'un peintre ou un pianiste. Ainsi, lorsque l'on me demande le background pour apprendre à faire de bons programmes informatiques ou des logiciels, je demande à l'intéressé s'il a déjà écrit une lettre, une petite histoire, un article, etc. Si la réponse est « oui » alors la personne possède déjà des prérequis pour apprendre à faire des programmes. Sinon elle peut toujours essayer d'écrire une petite histoire pour juger elle-même de ses capacités à créer ou à innover.

Savoir s'orienter : 5 considérations fondamentales quand on débute en programmation

Par Necemon

Un Freshman (étudiant de première année) m'adresse la requête suivante :
Salut Necemon, merci pour l'acceptation. Je suis nouveau dans le domaine informatique et cela me ferait plaisir que tu m'orientes. Peux tu me dire ce que les sociétés recherchent chez un informaticien, toi qui as de l'expérience professionnelle ?
J'ai entendu parler de toi par les seniors de Christ University, je suis à Bangalore et j'aime l'informatique mais je ne sais pas quoi apprendre et par quoi commencer.
Bon j'aime la programmation mais on me dit que le langage C n'est plus utile, on me parle de Ruby, de C#, de Python, etc. Je suis confus.

Il ne faut pas être confus. La technologie est juste un moyen de résoudre un problème ou d'atteindre un objectif. Quel est ton objectif ?
Créer des applis ? Quelles applis tu veux créer et pourquoi ?
C'est comme si tu viens me voir pour me demander quel véhicule tu devrais emprunter. Si je te demande c'est pour faire quoi, tu ne me dirais pas que c'est pour te déplacer, n'est ce pas ? Je sais que c'est pour te déplacer... Ma question c'est où tu vas ?

Ce que les compagnies recherchent ? Ok, je comprends parfaitement ce que tu es en train de demander. Tu veux t'assurer que la formation que tu suivras te garantira un emploi intéressant plus tard dans le développement informatique. Evidemment, je pourrais te dire qu'une Technologie T est très demandée en ce moment, mais ce n'est pas si simple. Il y a d'autres éléments à considérer :

1. La demande change avec l'endroit (le pays ou la région). Les emplois les plus populaires aux USA ne sont pas forcément les plus populaires en Inde. Donc à moins que tu saches déjà où tu vas travailler, ce n'est pas facile de cibler.

2. La demande change avec le temps. Ce qui est populaire aujourd'hui pourrait ne pas être (aussi) populaire demain. Les technologies évoluent et se remplacent. Donc ce qui est populaire actuellement pourrait être différent de ce qui sera populaire au moment où tu finis tes études.

3. Tu pourrais ne pas aimer la technologie en vogue ou les usages de cette technologie. Si je te dis qu'une technologie T est très demandée, ça permet de repérer et réparer des bugs/erreurs dans un système super ennuyeux/énorme/compliqué de transactions bancaires, il y a plein de calculs... Mais si toi tu n'aimes pas les calculs, est ce que tu vas quand même adopter cette technologie et accepter cette voie pour le reste de ta carrière ?

4. Même si on considère une seule ville à un moment donné, différentes compagnies recherchent différentes choses, tout dépend de ce que la compagnie fait. Il n'y a pas une technologie parfaite qui est meilleure que toutes les autres dans tous les domaines. Chaque technologie a ses avantages et ses inconvénients. Il y a certaines choses que C et C++ font mieux que Ruby (et vice versa), il ya certaines choses que Python fait mieux que C# (et vice versa), etc.

5. Comme je le disais plus haut, la technologie, c'est juste un moyen pour arriver quelque part. Quand tu considères Facebook par exemple, la plupart des utilisateurs s'en fichent, si ça été construit avec PHP, C, Java, Perl, C# ou Python. Ce qui est important pour les gens, c'est comment le site ou l'application peut les aider dans leur vie.

C'est par là que tu devrais commencer, je crois. Quelles sont les atouts que tu as déjà ? (ne me dis pas que tu n'en as pas), Quelle contribution tu comptes apporter à ta famille, à tes amis, à ta communauté, à ton pays, au monde ? Et qu'est ce que tu attends en retour ?

Si tu ne sais pas que faire de ta vie, j'ai écris un article il y a quelques temps qui pourrait peut être t'inspirer : Qu'est ce que tu vas faire dans la vie ? Prends le temps de réfléchir sur tes ambitions et on pourra parler des ressources dont tu auras besoin.

Si tu sais CE QUE tu veux faire, ce serait plus facile pour moi de t'expliquer COMMENT le faire.

A bientôt,

N.

Quand les judokas se mettent à la programmation

Necemon:
mais je maintiens ce que j'ai dit
la programmation c'est comme le judo
Banglet:
soit plus explicite
Necemon:
ok
donne moi 2 techniques de judo
2 techniques différentes
ok
laisse moi choisir
Banglet:
tu m'envoies loin la
Necemon:
disons o soto gari
Banglet:
je sais pas comment on écrit
Necemon:
et o goshi
tu te souviens de ces techniques ?
Banglet:
t'as les vrais souvenirs !
Necemon:
?
Banglet:
humm
pas vraiment
Necemon:
osoto gari
Banglet:
le nom me dit quelque chose en tout cas
Necemon:

voici o soto gari
o goshi c'est ce qu'on fait avec la hanche en se retournant
alright ?

Banglet:
ok
Necemon:

voici o goshi
Banglet:
o soto gari
j'aimais bien ça
Necemon:
ok
donc c’est ça qu’on va prendre comme exemple
Banglet:
ok
Necemon:
quelqu'un qui maitrise son o soto gari
il peut frapper des gens qui connaissent juste un peu de toutes les autres techniques
à force de pratiquer ça, le geste devient de plus en plus précis
il devient un expert
Banglet:
ouais
Necemon:
mais ça ne veut pas dire que o soto gari est la meilleure technique du monde
o soto gari c’est mieux si le gars recule
si il fonce sur toi, c’est clair que o goshi est mieux
donc celui qui maitrise le o soto gari et qui veut te battre avec
va se débrouiller pour toujours te faire reculer
c'est pareil pour les langages de programmation
celui qui est fort en C sera plus à l'aise dans les projets où C est le langage adéquat
par exemple C est connu pour échanger plus directement avec les périphériques et la mémoire
C est aussi très bon pour les applications graphiques
si le mec passe des années à faire ce genre d’applications avec C, quand tu vas voir de quoi il est capable
tu vas te dire, "wow c’est un champion…"
mais ça ne veut pas dire que C est le meilleur langage du monde
il y a des applications ou Java peut être plus adéquat que C
et puis quand je dis que c’est comme au judo
ce n'est pas que le langage qui compte, il y a aussi l'expérience
un grand maitre peut te clouer au sol avec un bras
de la même manière, un expert de C qui a pratiqué le langage pendant des décennies peut faire des merveilles

que toi tu ne peux pas faire même si tu utilises une librairie graphique de C++ genre Qt
tu comprends ?

Banglet:
un peu un peu

Necemon:
pour compléter avec ce que je viens de dire aujourd'hui
le programmeur parfait utilise la bonne technique au bon moment
de la même façon, le judoka parfait utilise la bonne technique au bon moment
exemple simple : si tu avances, il te fait o goshi
tu recules, il te fait o soto gari

Banglet:
et c'est IPPON !

Necemon:
lol
j'ai dit "le programmeur parfait utilise la bonne technique au bon moment"
je voulais surtout dire "le programmeur parfait utilise le bon langage au bon moment"

Banglet:
ok ok
mais comme la perfection n'est pas de ce monde...
il vaut mieux maitriser un langage
et la question c'est de savoir lequel maitriser

Necemon:
la perfection n'est pas de ce monde donc pourquoi tu cherches le langage parfait ?

Banglet:
looooooooool
bon disons le langage qui tend vers la perfection

Necemon:
la façon pour toi de tendre vers la perfection c'est de choisir au mieux ton langage en fonction de la situation
en fonction de tes besoins
et surtout en fonction de tes gouts
c'est important que tu fasses quelque chose que tu aimes
comme au judo ou tu as une technique que tu aimes bien
dans laquelle tu te sens à l'aise
en programming ce que tu dois te demander c'est
qu'est-ce que je veux faire au fait ?
et ensuite, tu te donnes les moyens de faire ce que tu veux faire de façon optimale
si tu construis un logiciel pour un client
il s'en fiche de savoir si ton langage était orienté objet
si tu as utilisé des pointeurs
des librairies graphiques
des génériques
etc.

Banglet:
ça c'est pas faux

Necemon:
bref, toutes les technologies de ton langage préféré dont tu es si fier, ton client s'en fiche

Banglet:
lool

Necemon:
la question c'est, est ce que ton logiciel correspond à ses besoins, ses objectifs ?

est ce que tu as marque' IPPON ?
c'est ça qui compte
maintenant comme je t'ai dit
tu peux t'orienter
comme je sais que tu aimes O Soto gari, ça veut dire que tu dois aimer faire reculer l'adversaire
de cette même façon, si tu aimes Java, tu dois savoir en quoi Java est bon
et qu'est ce que tu peux faire de beau avec Java
si tu veux je peux t'expliquer en quoi C# est bon
et pourquoi j'aime C#
mais je ne prétends pas que C# est le meilleur langage du monde
ce que je sais
c'est qu'il me convient
et qu'il me rend très efficace
je ne peux pas garantir qu'il te convient
parce que tout dépend de ce que toi même tu veux
et de ce qu'on attend de toi

Banglet:

ok ok

Création De Jeu Vidéo : 4 Tactiques Substantielles Qui Vont Forcément Booster Tes Compétences

Par Necemon

Un utilisateur de l'une de mes applications web m'a fait part de sa préoccupation:
Bonjour, je suis T. j'ai 15 ans, quand je serai grand j'aimerais faire partie de l'industrie du jeu vidéo et je me demandais si vous pouviez m'aider. Pouvez-vous me donner quelques conseils et partager votre expérience avec moi ?
Est-ce que le fait de publier sur votre plateforme pourrait m'aider dans cette situation? J'aimerais beaucoup lire votre réponse, ça m'aidera vraiment.

Telle fut ma réponse:

Il y a plein de chose à dire sur le sujet mais aujourd'hui j'essaierai de faire simple en te donnant quelques conseils pratiques que tu pourrais utiliser dès maintenant.

1. **Lis beaucoup sur les sujets qui t'intéressent.** Lis chaque jour si tu peux. Il y a des tonnes de ressources gratuites en ligne. Pour ce qui est de la création de jeux vidéo, certains de mes blogs et sites favoris incluent:
gdne.ws
lostgarden.com
procworld.blogspot.co.uk
whatgamesare.com
gamestudies.org
designer-notes.com
higherorderfun.com
gamecareerguide.com
webwargaming.org
raphkoster.com/gaming

gamedevelopment.tutsplus.com
nwn.blogs.com

2. **Etablis des objectifs précis.** Tu remarqueras que dans le premier point, je t'ai simplement donné une petite partie des informations disponibles, mais c'est déjà relativement étendu. Nous vivons des temps exponentiels, tu ne pourrais pas tout faire et tout apprendre à la fois. Tu dois donc être précis dans ce que tu veux. Les projets vagues provoquent des résultats vagues. Qu'est ce que tu aimerais faire dans l'industrie des jeux vidéo? Il existe plusieurs options sur lesquelles tu pourrais faire quelques recherches. Nous vivons des temps exponentiels, tu ne pourrais pas tout faire et tout apprendre à la fois. Tu dois donc être précis dans ce que tu veux. Les projets vagues provoquent des résultats vagues. Qu'est ce que tu aimerais faire dans l'industrie des jeux vidéo? Il existe plusieurs options sur lesquelles tu pourrais faire quelques recherches.

3. **Choisis tes étoiles.** Pour rebondir sur le point précédent, après avoir fait une liste des compétences que tu voudrais acquérir, fais quelques recherches sur les personnes qui les maitrisent déjà. Trouves toi des modèles de référence, vois comment ils ont commencé et comment ils ont procédé. Fais la combinaison de leurs méthodes et ajoute ta touche personnelle pour développer tes propres techniques. Cela peut prendre des mois, mais si tu t'y mets de façon constante, tu deviendras aussi très bon. Quelques personnes sur qui tu pourrais t'informer:

Développement de jeux:
- Shigeru Miyamoto
- Hideo Kojima
- Notch (Markus Persson)

Artistes digitaux:
- Akira Toriyama
- Masashi Kishimoto
- Monty Oum

Scénaristes:
- J.K. Rowling
- Stephen King
- Tom Clancy

4. **Construis un jeu.** Non, tu n'es pas trop jeune pour commencer. Pas besoin de débuter avec quelque chose d'immense dès le départ, tu peux déjà commencer à apprendre. Plus tu t'y mets tôt, plus tu as le temps de t'exercer et plus tu pourras devenir excellent rapidement. Tu peux débuter avec quelque chose de vraiment basique ou au moins lire des documents sur la procédure. Si tu veux des conseils sur comment débuter, comment trouver des tutoriels et documents, je peux t'aider. A ce propos, j'ai crée un jeu: babifraya.com Il n'est pas extraordinaire et je travaille présentement sur une version un peu plus améliorée. Mais le plus important, c'est de commencer et de continuer à pratiquer :-)

Pour résumer, approfondis tes connaissances continuellement.

Quant à savoir si le fait de poster sur degammage.com t'aidera, ma réponse honnête serait: peut-être. Tu vois, c'est un projet qui en est à ses débuts et c'est trop tôt pour évaluer à quel point il aura du succès. Toutefois, il y aura du contenu frais chaque jour, donc tu auras de nouvelles choses à apprendre, et tu pourrais aussi visiter flux.evasium.com qui est plus orienté sur le monde virtuel, l'éducation et la technologie. Ton compte Evasium fonctionne là également.

En plus, tu pourrais aussi rejoindre d'autres communautés et forums de jeux comme ceux que j'ai listé plus haut. Apprends autant que possible!

10 Ans De Programmation

Par Necemon

J'ai commencé la programmation quand j'étais au lycée. On nous apprenait, entre autres, Pascal et HTML. Entre temps, pendant les vacances d'été, je prenais des cours de programmation dans un institut TIC. C'est là-bas que j'ai appris Visual Basic et la conception de logiciels et de bases de données (A ce temps là, on utilisait MERISE (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise)). Un jour, j'ai pu obtenir le CD d'installation de Visual Basic, donc je l'ai installé sur mon PC et j'ai commencé à pratiquer tout seul, avec des bouquins.
Mais, ce n'est qu'après le bac que j'ai commencé à apprendre et à appliquer C# presque à plein temps pendant les études supérieures, et c'est ça que je considère comme mon véritable début en programmation et ingénierie logicielle.

J'ai appris quelques leçons au cours de la dernière décennie, et je me suis dit que je prendrais un moment pour recueillir mes réflexions sur ces sujets. Il m'a fallu environ dix ans et beaucoup d'expérimentation pour assimiler certaines de ces choses.

1. Apprendre un langage de programmation, c'est la partie la plus facile: attention aux plateformes
Prenons C# par exemple. Apprendre le langage C# n'est pas difficile. Si tu as déjà une bonne compréhension des fondamentaux de langues informatiques, et si tu as un peu d'expérience dans d'autres langues orientées objet, tu peux devenir un programmeur C# compétent en quelques jours, du moins en ce qui concerne le langage en lui-même. Cependant, le vrai prix à payer de l'apprentissage n'est pas dans le langage, c'est dans la plateforme. Pour développer avec C# sur .NET, tu dois connaître:

- le Framework .NET
- une ou plusieurs technologies .NET telles que ASP.NET ou WPF
- et l'environnement de développement Visual Studio.

Le temps requis pour devenir compétent dans le développement sous .NET se mesure généralement en mois, même pour un développeur expérimenté. Apprendre une plateforme est toujours plus coûteux que d'apprendre un langage spécifique, donc choisir la plateforme est la décision la plus cruciale.
L'apprentissage a toujours un coût et ce coût est l'un des facteurs clés à prendre en considération lors du choix de la technologie que tu souhaites apprendre. Le coût réel de l'apprentissage est dans le temps, l'apprentissage prend toujours du temps. Puisque tu n'as pas le temps de tout apprendre, il est important de penser stratégiquement à ce que tu veux apprendre. Et puisque les langages sont faciles, c'est surtout aux plateformes qu'il faut faire attention : les technologies associées au langage, les outils de développement et de déploiement, les systèmes d'exploitation, et autres infrastructures.

2. Je répète, apprendre un langage de programmation est la partie la plus facile: à la rencontre des concepts fondamentaux de l'ingénierie logicielle
La syntaxe en elle-même, les mots que tu utilises lorsque tu utilises le langage sont relativement simples et tu peux les apprendre facilement. Cependant, c'est loin d'être suffisant pour produire du code de qualité, qui implique souvent des principes OOP, TDD, BDD et SOLID, des tests unitaires, des patrons de conception et d'autres concepts techniques qui dépassent le cadre de cet article. Enfin, bref.

3. En fait, écrire du code n'est qu'une (petite) partie du travail
Un ingénieur logiciel est souvent impliqué dans la recherche technologique, la configuration d'outils et de projets, les tâches d'administration et de déploiement, les procédures de débogage et d'essai, la documentation et la dette technique (correction et refactorisation du code existant). De plus, il doit réfléchir à des solutions et concevoir des systèmes: parfois, le travail le plus important se fait pendant qu'on est loin du clavier.

4. Recettes éprouvées et efficaces : les vieilles techniques ennuyeuses sont parfois les meilleures.
Ce n'est pas vraiment "vieux contre nouveau", ni même "cool vs ennuyeux", mais plutôt la technique avec laquelle tu as le plus d'expérience. Comme disait l'autre, je ne compte pas sur le codeur qui a pratiqué 1000 technologies une seule fois mais sur celui qui a pratiqué la technologie adéquate 1000 fois. Si l'objectif est de construire un truc de manière aussi effective et rapide que possible, ce serait plus productif d'utiliser les technologies que tu maitrises le mieux.
Par exemple, un de mes contacts se fait 25 000 $ par mois avec un SaaS qu'il a construit avec une combinaison ennuyeuse : ASP.NET + SQL Server + Angular 1, parce que c'est ce qu'il connaissait. Il l'héberge sur Windows, parce qu'il sait comment rendre Windows rapide et sécurisé. Il a réussi parce qu'il consacre tout son temps à construire des fonctionnalités que ses clients réclament, plutôt que d'apprendre les technologies les plus en vogue.
Il est important de se rendre compte que le tapis roulant de la technologie ne s'arrête jamais, il y a toujours de nouvelles choses à apprendre. En fait, un nouveau Framework JavaScript est probablement en train d'être lancé pendant que tu es en train de lire cet article. Les technologies de pointe les plus populaires aujourd'hui n'existaient même pas lorsque je débutais (EF Code First, Xamarin, ASP.NET Core, Razor), ce qui nous conduit aux 2 points qui suivent.

5. Concentre-toi sur les technologies durables
La seule constante dans le monde, c'est le changement. La gestion du temps et des actions est une compétence importante chez les développeurs, en particulier, parce que nous sommes sur un tapis roulant technologique qui ne cesse de bouger, voire d'accélérer.
Par exemple, les technologies Web qui étaient populaires vers l'an 2000 (Flash, ASP Classic et Java Applets) deviennent quasiment obsolètes et de moins en moins professionalisantes. Aujourd'hui, on parle de ASP.NET Core, SignalR, Angular2, React et VueJS. Aucune de ses technologies n'existaient en l'an 2000, et ces nouvelles technologies seront probablement obsolètes d'ici 10 ans.
Qu'est ce qui n'a pas vraiment change' ? Les fondamentaux de langages tels que C++/C#, leurs implémentations d'algorithmes et leurs principes sont toujours aussi pertinents sur plusieurs dizaines d'années. Si tu maitrises les bases d'un système stable, tu pourrais mieux t'adapter au changement, tu pourrais l'apprécier et t'en servir pour évoluer.

6. Équilibre entre exploration et exploitation
L'exploration consiste à apprendre de nouvelles choses, à étudier de nouvelles techniques, à lire des livres, à regarder des tutoriels, à pratiquer et à améliorer ses compétences. L'exploitation, au contraire, consiste à tirer parti de ce que nous savons déjà pour régler de vrais problèmes. Il s'agit de penser créativement à des façons d'utiliser les connaissances que nous avons déjà pour créer de la valeur pour les autres.
Donc, oui, ces deux tâches sont à la fois nécessaires et importantes. Le risque, c'est d'être trop fixé sur l'une ou l'autre de ces activités.
Trop d'exploration, et tu ne pourras jamais atteindre un niveau d'expertise utile dans une technologie donnée. Il y a un coût d'opportunité énorme avec cette sorte d'apprentissage léger, puisque, bien que ça t'élargisse l'esprit, le temps que ca nécessite implique que tu perfectionnes moins les compétences que tu as déjà acquises.
En revanche, une trop grande exploitation peut t'empêcher d'évoluer dans les nouvelles technologies, et peut limiter tes opportunités d'emploi.

7. C'est facile d'être excellent... C'est difficile d'être constant.
C'est facile d'être excellent pendant 2 minutes. C'est difficile de l'être constamment, chaque jour.
Quand tu es habité par une bonne idée pour un nouveau projet, tu ressens une grande envie de commencer la recherche, le design et la programmation. Tu as hâte de transformer ton idée en quelque chose de réel et tu deviens super productif. Mais le problème est que cette motivation se fane avec le temps.
Oui, c'est marrant et c'est facile d'avoir de nouvelles idées et de commencer à y travailler. Mais ensuite, il y a les efforts à fournir, les ajustements, le lancement, la maintenance, les corrections, les améliorations, etc. Sur plusieurs mois. C'est là que ça devient dur. C'est dur de rester concentré sur une même idée, sur un même projet pendant des mois et des années. Ça demande beaucoup de discipline.
C'est facile d'être excellent. C'est difficile d'être constant.

8. Diversifie tes compétences
Ne sois pas seulement un programmeur, devient un Expert Qui Programme, un expert dans un autre domaine pertinent dont tu es passionné. Tu peux être un entrepreneur, un chef de projet, un scientifique en Big Data, un chercheur, un spécialiste de la sécurité, etc.
Si tu es un Expert Qui Programme, en plus de pouvoir programmer (peut-être à temps plein), tu as également une crédibilité supplémentaire qui est liée à d'autres domaines, au dela de l'ingénierie logicielle. D'où l'importance de poursuivre des études supérieures. Si tu vas à l'Université ou en Grande École alors que tu sais déjà programmer, tu n'apprendras probablement pas grand-chose à propos de la programmation. Mais ça ne veut pas dire que tu ne devrais pas aller dans ces écoles. Tu auras besoin d'une certaine culture, et les universités sont d'excellents endroits pour l'obtenir. Tu acquiers la culture en étudiant et en comprenant le monde que les humains ont créé, sous différents angles. Ce serait difficile d'acquérir ce genre de connaissances si tu ne fais rien d'autre qu'étudier la programmation.

9. Choisis des niches pour te démarquer
Plus la niche est petite, plus tu es perçu comme un agent exceptionnel dans ton domaine. Par exemple, il est très difficile pour des développeurs de se démarquer avec un titre tel que "Développeur Web PHP". Ils sont compétents, polyvalents, utiles, mais pas remarquables. Ils se sentent facilement remplaçables parce qu'il y en a tellement, avec un ensemble de compétences comparables. Le domaine est trop large pour que tu puisses te démarquer facilement du lot. Si, par contre, tu deviens reconnu dans une niche, comme Xamarins.Forms ou Visualisations JavaScript, tu serais beaucoup plus apte à te distinguer par ceux qui recherchent spécifiquement ces compétences.

10. L'âge des Compétences
L'information est la connaissance spécifique dont tu as besoin pour résoudre des problèmes. Les compétences représentent la capacité de mettre en œuvre des solutions en utilisant tes connaissances. Dans un monde où la plupart des connaissances et des outils sont quasiment gratuits, qu'est-ce qui fait la différence ? C'est la compétence, bien évidemment. Nous ne sommes plus une société fondée sur le savoir, nous sommes une société axée sur les compétences. Il fut un temps où presque tous les diplômes universitaires garantissaient un bon travail. Maintenant, ce n'est plus le cas. On s'en fiche de ce que tu sais. On s'intéresse à ce que tu sais FAIRE. On te paye pour faire des choses, pas pour connaître les choses.

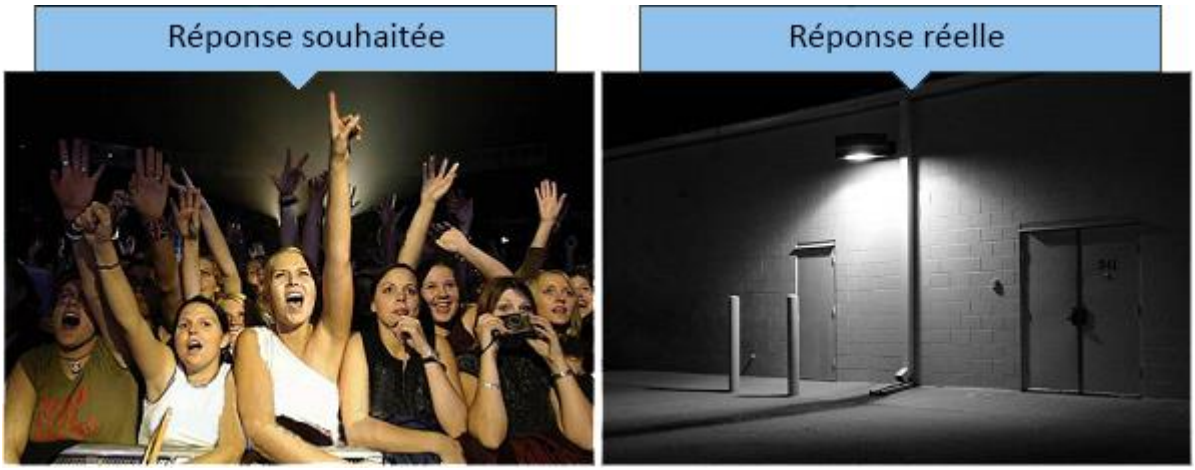
Apprendre à programmer en 10 ans.
Selon plusieurs recherches, il faut environ 10 ans (ou 10 000 heures) pour développer une expertise.
La solution, c'est la pratique réfléchie : il ne suffit pas de refaire les mêmes choses encore et encore, mais plutôt de se défier avec une tâche qui dépasse sa capacité actuelle, l'essayer, analyser ses performances pendant et après, et corriger toute erreur. Ensuite, répéter. Et recommencer. Il semble qu'il n'y ait pas vraiment de raccourcis. L'apprentissage par la lecture, c'est bien. Mettre la main à la pâte, c'est mieux. Le meilleur type d'apprentissage est l'apprentissage par la pratique.
Personnellement, les petits projets perso et les prototypes m'ont effectivement aidé à m'améliorer. Mais il y a encore beaucoup de choses intéressantes à maitriser donc continuons d'apprendre.

Faut-il construire ton site même si personne n'y vient ?

Par Darren Mart

Il n'y a pas si longtemps, j'ai reçu un e-mail d'un développeur énergique et talentueux qui s'attaquait à un projet logiciel ambitieux. Il m'a demandé si j'avais un conseil à lui donner.

Tous les souvenirs ont commencé à me revenir en masse. Les rêveries. Sortir du lit au beau milieu de la nuit parce qu'une idée ne peut pas attendre. Des mois de travail minutieux qui poussent ce bébé à ne plus vaciller et à marcher seul. Tout ça pour un public enthousiaste. Sauf que...



Le cycle après le cycle

J'avais peut-être rédigé une demi-douzaine de réponses à son e-mail. J'ai admiré son enthousiasme et compris où son esprit en était. Mais je savais aussi ce qui l'attendait probablement après le cycle de développement ardu: un autre cycle de turbulence, une série d'affrontements entre les prévisions et les tristes réalités.

Qu'est ce qui n'a pas marché?

C'est une question dangereuse que nous nous posons en tant que développeurs. "Dangereuse" parce qu'elle présuppose que nous avons fait quelque chose de travers. Il est tout à fait possible que ce que tu as conçu soit assez bon, peut-être même brillant. Le problème n'a peut-être rien à voir avec la qualité de ton travail, mais plutôt avec tes critères de réussite.

Alors, comment un développeur peut-il mesurer précisément le succès d'un projet solo? J'aurai aimé avoir une réponse concrète à te donner. Par contre, je peux te donner quelques conseils pour ce qui est de comment *ne pas* le mesurer:

Astuce 1: Méfies-toi des sirènes des plateformes sociales

Ah, mais elles nous leurrent avec la promesse d'une acceptation et d'une consommation de masse. Nous réalisons rarement leur capacité à berner nos esprits. Il est possible que tu puisses passer des mois sur un projet, que tu le partages sur Facebook avec la vigueur et l'excitation d'un chien qui tient un jouet à mordiller, et que tu observes le nombre de "J'aime" monter en flèche jusqu'à... 3. L'un des "j'aime" vient de ta mère, un autre provient d'une page que tu as conçue pour promouvoir ton projet, et l'autre d'un tapotement hasardeux sur un téléphone portable.

C'est le même effet de refroidissement que te font les photos de parcs d'attractions abandonnés. Tu essaies d'imaginer des fous rires et une excitation effrénée, et tu as du mal à accepter ce qui se trouve juste sous tes yeux.

Pendant ce temps, le nouveau et très profond statut de ton pote disant "J'aime les raviolis!" vient d'obtenir 23 likes et 16 commentaires. La popularité et la substance sont deux choses différentes, et nous devons simplement faire avec.

Astuce 2: La réaction apathique n'a rien de personnel (elle pourrait même ne pas être apathique)

Tu es déjà conscient de ça, mais il est bon de s'en rappeler. La plupart des gens n'ont aucune notion de ce qu'il faut pour réussir ce que tu as accompli. Ils ne réalisent pas que tu as construit à toi seul quelque chose qui rivaliserait avec les efforts de toute une équipe. Tu ne peux pas, et tu ne vas pas, les entraîner à s'y intéresser sincèrement.

Les applications qui agitent le monde relèvent généralement de deux catégories: celles qui aident les utilisateurs à se promouvoir eux-mêmes et celles qui nécessitent peu de réflexion. Si ton projet nécessite un investissement mental supérieur à celui de Candy Crush Saga, voici la dure réalité: tu vas te retrouver avec beaucoup de restes de nourriture lors de ta soirée de lancement.

Astuce 3: Réfléchis à deux fois avant de solliciter les commentaires de tes confrères

Dans le film "Minuit à Paris", l'écrivain en herbe Gil demande à Ernest Hemingway de lire son roman et de donner son opinion.

- Hemingway: Mon opinion est que je le déteste.
- Gil: Ah bon? Mais vous ne l'avez même pas lu.
- Hemingway: Si c'est mauvais, je le détesterai parce que je déteste quand c'est mal écrit, et si c'est bon, je serai jaloux et je le détesterai d'autant plus. Tu ne veux pas l'opinion d'un autre écrivain. Les écrivains sont compétitifs.

Ne néglige pas la profondeur de cet échange. Ceci s'applique aussi aux développeurs.

Astuce 4: Qu'on le veuille ou non, tu es un artiste

Si tu te dévoues corps et âme à la réalisation de projets solo parce que tu aimes le défi créatif, alors tu es un artiste. Si tu le fais parce que tu essaies de résoudre un problème, c'est que tu es probablement un hybride artiste-ingénieur. Si tu le fais strictement pour l'argent, tu as sûrement abandonné cet article depuis longtemps, donc peu importe comment je t'appelle.

Tout comme les peintres, les écrivains, les décorateurs et les chefs gastronomes du monde entier, nous espérons secrètement que le public sera élevé et inspiré par nos créations.

Mais il y a une différence clé. Si quelqu'un n'est pas inspiré par une peinture, il serait quand même capable d'apprécier l'effort individuel. Avec un logiciel, il n'y a pas ce luxe. Peu importe si tu es le Picasso du monde logiciel, ton appli est très nulle par rapport à Office 365 ou Google Maps ou Skyrim, peu importe la flotte de ressources requises par ce dernier.

Donc... Faut-il continuer à construire ton projet même si personne n'y vient ?

Oui. Parce qu'en tant qu'artistes, nous sommes enrichis par le processus global quel que soit le résultat final. Aussi banal que cela puisse paraître, nous avons ce sentiment d'accomplissement et la fierté de savoir à quel point il a fallu s'autodiscipliner pour poursuivre le projet jusqu'au bout. Nous avons stimulé notre intellect, nous avons appris ce qui fonctionne et ce qui ne fonctionne pas, nous avons acquis une expérience pratique. Le prochain projet, qu'il soit personnel ou professionnel, en récoltera les fruits.

Certifications Microsoft pour Développeurs

Par Necemon

Microsoft offre une large gamme de programmes de certification conçus pour t'aider à développer tes compétences et ta carrière.

Cet article se concentrera sur les certifications de développeurs. MCSD (Microsoft Certified Solutions Developer) est une certification destinée aux professionnels de l'informatique qui cherchent à démontrer leur capacité à créer des solutions innovantes dans de multiples technologies. Par exemple, la certification MCSD App Builder confirme que tu as les compétences nécessaires pour créer des applications et des services mobiles modernes.

J'ai reçu ma première certification Microsoft en 2008. En progressant, j'en ai obtenu d'autres au fil des années et je suis maintenant un Développeur de Solutions Certifié. Est ce que ça en valait la peine ? Parfois, ça n'avait pas d'importance, parfois c'était très utile.
Selon mon expérience, il y a 5 avantages à obtenir une certification:
- Obtenir une certification professionnelle Microsoft ne garantit pas l'obtention d'un emploi, d'une augmentation ou d'une promotion, mais ça augmente les chances.
- Quand il s'agit d'embaucher, certaines entreprises demandent s'il y a des certifications, en plus de l'expérience. De nombreux recruteurs vérifient les certifications parmi les candidats à l'emploi et considèrent ça comme une partie de leurs critères d'embauche. Certains considèrent les certifications informatiques comme une priorité lors de l'embauche de postes informatiques.
- Obtenir une certification peut augmenter la confiance en soi, ta confiance en tes compétences.
- Cela montre du sérieux et de la passion (tu l'as fait tout parce que tu voulais, et non parce que tu devais).
- Le processus de préparation à la certification augmente tes compétences théoriques et pratiques.

Conseils et astuces pour la préparation des examens
Ma recommandation principale est de consacrer une période spécifique à la préparation de l'examen, quelques jours avant. Trouve des livres, participe à des cours (les cours virtuels sont une préférence personnelle, mais les cours en classe sont aussi disponibles) et entraine-toi sur les technologies concernées.
- Livres: il y a souvent des livres de préparation liés à l'examen donné. Par exemple : Examen 70-740 - Installation, gestion du stockage et des traitements sur Windows Server - Préparation à la certification MCSA
- Cours: les examens les plus populaires ont des cours courts disponibles en ligne. Par exemple : Le cours 20487B: developpement de Services Web
- Au-delà du matériel de formation, c'est bien de faire des recherches sur les sujets concernés
- S'exercer sur lesdites technologies
- Répondre aux questions à choix multiples: procéder par élimination. En cas de doute sur la bonne réponse, exclure les réponses qui n'ont pas de sens (ou qui ont moins de sens)

Chemins de carrières pour les développeurs
Tu devrais peut être vérifier cette partie. Ils mettent les offres à jour assez régulierement, en fonction des technologies les plus récentes, mais au moment ou j'écris ces mots, il existe 5 options pour devenir MCSD (Microsoft Certified Solutions Developer)
- MCSD: applications Web: expertise dans la création et le déploiement d'applications et de services Web modernes.
- MCSD: applications Windows Store: expertise dans la conception et le développement d'applications Windows rapides et fluides. Il existe deux moyens pour obtenir cette certification, en utilisant soit HTML5 ou C#.
- MCSD: applications SharePoint: expertise dans la conception et le développement d'applications de collaboration avec Microsoft SharePoint.
- MCSD: Azure Solutions Architect: expertise sur toute l'étendue de l'architecture, le développement et l'administration de solutions Azure.
- MCSD: gestion du cycle de vie des applications: expertise dans la gestion du cycle de vie complet du développement d'applications.

Planificateur de certification
Le Planificateur de certification est un outil pour t'aider à connaître les conditions requises pour ta prochaine certification. Il s'agit de planifier tes prochaines étapes (tes options, les examens à suivre, dans quel ordre, etc.).

Autres liens et références:
Certification MCSD
Certification WebApps
Page Wikipedia

C'est tout pour le moment. Maintenant retourne travailler.

Rappel Final

Avant de conclure, je voulais juste te rappeler que ce document contient le Chapitre 6 de L'Album et que 8 autres chapitres sont également disponibles. Tu peux télécharger, (re)lire et partager chaque chapitre ou volume indépendamment/séparément, en fonction de tes intérêts. Les formats PDF, EPUB, MOBI/AZW3/KF8 (Amazon Kindle) et MP3 sont disponibles.

Il suffit de cliquer sur le(s) document(s) qui t'intéresse(nt) ci-dessous ou d'aller sur Album.NeceMoon.com (ou necemonyai.com/blog/page/L-Album.aspx).

The NeceMoon, L'Album Complet

Volume 1 : Clair de Lune (softcore)

- Chapitre 1 : Stratégies et Tactiques
- Chapitre 2 : Marketing Digital et Visualisations Web
- Chapitre 3 : Carrières et Emergence
- Chapitre 4 : Bons Plans et Petites Victoires Faciles
- Chapitre 5 : Dégammage - Délires et Réflexions Rapides

Volume 2 : Pleine Lune (hardcore)

- Chapitre 6 : Génie Logiciel - Quelques Notions Remarquables
- Chapitre 7 : Programmation informatique avec C# .NET
- Chapitre 8 : Prototypes Epiques, Projets Classiques, Genre Historique
- Chapitre 9 : Recherches et Etudes de Cas

L'Album est également disponible en Anglais à l'adresse TheAlbum.NeceMoon.com (ou necemonyai.com/blog/page/The-Album.aspx).

Conclusion

C'est le moment de se quitter, mais on peut rester en contact. N'hésite pas à m'ajouter sur LinkedIn, Twitter ou Facebook. Si toi aussi, tu as des outils ou des techniques pour accomplir les choses efficacement, je voudrais bien que tu m'en parles. Mon adresse email, c'est necemon@gmail.com. N'hésite pas à m'écrire pour signaler d'éventuelles erreurs dans ce document, pour suggérer des améliorations ou pour me faire part de tes passages préférés. Par contre, si tu n'aimes pas L'Album, ce n'est pas la peine de m'écrire.

Dans tous les cas, bon courage pour la suite, et merci d'avoir lu. Et merci à Wikipédia, MSDN, IconFinder et FreeDigitalPhotos pour les clarifications et les ressources infographiques. Remerciements spéciaux à la Team Evasium. Merci à tous ceux et celles qui ont contribué, merci Ahou l'Africaine, Antoine Mian, Cyriac Gbogou, Darren Mart, Edith Brou, Holty Sow, Israël Yoroba, Jean Luc Houédanou, Jean-Patrick Ehouman, Karen Kakou, Nanda Seye, Nnenna Nwakanma, Olivier Madiba, Vanessa Lecosson et Yehni Djidji.
Merci Monty Oum, repose en paix.

Partage L'Album avec Tes Potes

Clique simplement sur l'icône qui te convient ci-dessous. Ça ne prend que quelques secondes et c'est gratuit pour tout le monde. Alternativement, tu peux partager ce lien sur toute Plateforme Web/Sociale ou l'envoyer par e-mail à tes contacts qui pourraient en bénéficier : http://Album.NeceMoon.com