

Welcome to OpenClassrooms! By continuing on the site, you are agreeing to our use of cookies. Read more

OK

Sign up

Sign in

[Home](#) ▶ [Course](#) ▶ [Concevez votre site web avec PHP et MySQL](#) ▶ Votre rôle de développeur web PHP

Concevez votre site web avec PHP et MySQL



30 hours



Medium

License



Votre rôle de développeur web PHP

[Log in](#) or [subscribe](#) to enjoy all this course has to offer!

Avant de commencer avec de la technique, il est important de bien cerner **votre rôle** dans l'ensemble des projets web dans lesquels vous prendrez part.

En effet, il est de notoriété publique que les projets informatiques sont voués à l'échec avant même qu'une ligne de code ne soit écrite. Il est donc vital de garder à l'esprit que votre but (et celui de l'ensemble de l'équipe) est de tout faire pour que le projet voie le jour. Pour y arriver, la production de code est nécessaire mais pas suffisante 😬!



Tu nous donnes l'impression que tous les projets se font en équipe, mais depuis le début de ma formation, tous les projets que je fais, eh bien je les fais seul !

Comment envisager les choses sereinement si je ne sais pas si je vais être intégré-e à une équipe ?

Je vous propose d'envisager les situations de manière assez ludique. De manière générale, les projets informatiques sont assimilables à une quête dont vous ne verriez pas forcément la fin. Au début du mode histoire, il est fort possible que vous soyez seul, **mais** au fur et à mesure de l'avancée du projet, l'équipe pourrait s'agrandir, ou se réduire. De la même manière, vous pourriez intégrer une équipe déjà constituée. Nous n'en savons rien et ce n'est pas grave ! Le tout est de garder le cap et d'y aller par étape. Dans tous les cas, l'une de vos premières missions est d'améliorer encore et toujours plus votre adaptabilité. Plus vous vous adapterez à une situation, plus il vous sera facile d'évoluer dans n'importe quel type de projet 😊!

Pour pouvoir avancer intelligemment, il faut établir des jalons au projet afin de ne pas voir ce dernier comme une montagne infranchissable. L'une de vos premières attributions est de faciliter l'établissement de ces étapes avec toute l'équipe (qu'il s'agisse de développeurs comme vous ou non).

Passons un peu en revue les quêtes annexes à votre mission de développement avant d'attaquer la technique dans le dur 🧐.

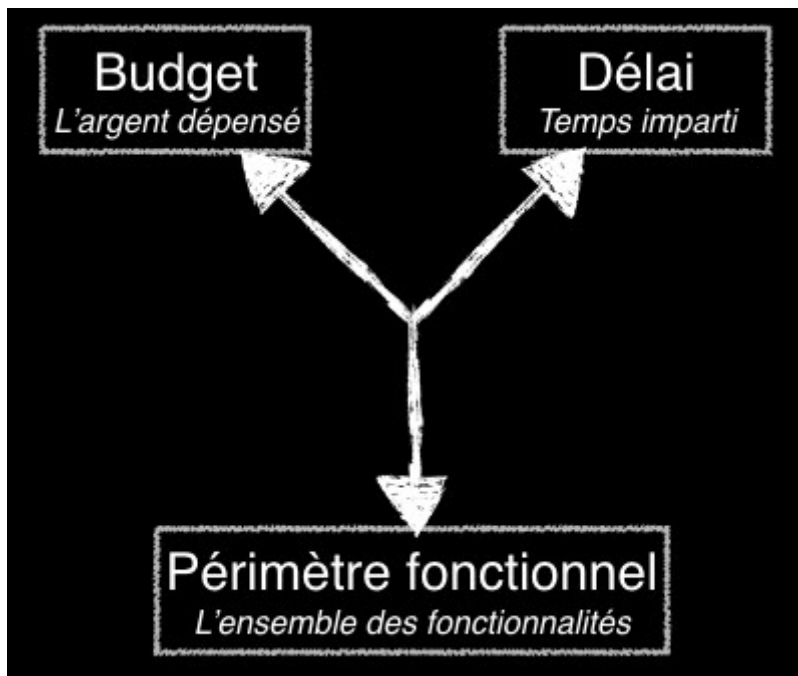
Du besoin client à la production du code



Comme je l'évoquais plus haut, le métier de développeur ne se cantonne pas uniquement à la production de code. Le recueil du besoin et la communication avec le (ou les) commanditaires est vital. La réussite d'un projet passe par le fait de se comprendre ; Ce serait dommage de développer une fonctionnalité qui ne ressemblerait en rien à ce que l'on aurait demandé 🙄...

! Ceci dit, il est **vital** de s'assurer de la **faisabilité** de ce que l'on vous demande de développer.

Une idée reçue serait de se dire que la faisabilité ne dépend uniquement (ou grandement) de la technique. Eh bien non les amis. Regardons le triptyque présentant ce qui conduit un projet :



Les trois axes qui conduisent un projet

Il est impossible de tendre vers le 100% sur les trois axes présentés ci-dessus, sinon cela impliquerait d'être à budget et délai illimité (personne ne dépense d'argent sans compter voyons 🙄). En début de projet, il n'est pas rare d'avoir un délai imposé ainsi qu'un budget imparti, et pourtant cela n'empêche pas le (ou les) commanditaire(s) de vouloir la Lune 🌕. C'est là qu'il faut arriver à **étudier** cette fameuse **faisabilité**.

L'axe sur lequel vous aurez le plus d'amplitude (n'en déplaise aux commanditaires) est celui du **périmètre fonctionnel** : en effet, il est impossible d'imposer un budget (au risque de perdre le client), et les délais correspondent en général à

des dates qui permettent de lancer un business. Ainsi, l'une des cartes qu'il va falloir abattre est la **négociation des fonctionnalités / corrections que vous aurez à développer**. Parmi vos interlocuteurs, vous aurez des protagonistes non techniques et sans doute assez méfiants lorsque vous leur donnerez une estimation des délais pour mener à bien une tâche. Pour pouvoir négocier et gagner la confiance de vos interlocuteurs, il faut que vous soyez en mesure de bien argumenter vos estimations. Lorsqu'il s'agit de savoir en combien de temps il faut développer une fonctionnalité, la première personne avec laquelle il faut être honnête c'est... **vous** 😊!

Il n'y a pas de problème à dire que vous avez besoin de formation (même seul) pour pouvoir mener à bien une tâche, du moment que vous l'intégrez dans votre estimation et que vous êtes en mesure d'établir un plan précis de ce qu'il y a à apprendre. Un moyen de bien le vendre est de montrer que votre apprentissage peut-être déversé aux autres membres de l'équipe, et que cela vous sera nécessaire tout au long du projet (et peut-être des prochains).

Par ailleurs, une fonctionnalité peut vous paraître complexe a priori : la complexité c'est comme les goûts et les couleurs, cela reste propre à chacun et rattaché à son expérience personnelle. Néanmoins, il faut pouvoir exprimer son point de vue sur le niveau de complexité que l'on sent, et surtout proposer des solutions qui ne mettent pas en péril le projet.

Prenons un exemple : vous discutez avec votre chef de projet d'une fonctionnalité que vous trouvez un peu trop complexe pour l'utilisateur final. Voici comment vous pourriez lui exposer la problématique avec une solution a priori acceptable : *"Je trouve que le scénario utilisateur présente un peu trop d'étapes. Que pensez-vous de le réduire à 2 ou 3 pour commencer, en s'assurant que l'utilisateur puisse arriver à ses fins. Puis une fois développé et testé, nous pourrions ajouter les autres étapes dès la prochaine itération ?"*.



Attention aux mots que vous emploieriez lors de (ou des) discussion(s) que vous allez avoir lors des négociations pour modifier le périmètre fonctionnel. Évitez les mots tels que "complexe", "inadéquat", "infaisable"... Tous ces mots qui ne laissent pas d'ouverture(s) pour trouver une solution qui puisse convenir à tous et surtout au projet.

Dans le cas où vous êtes face à un refus catégorique "d'alléger" une fonctionnalité (ce qui peut arriver), il est temps de négocier à un niveau plus large : une ou plusieurs autres fonctionnalités devront être réduites pour répondre aux impératifs de budget/délais.

Une estimation n'est pas uniquement qu'une question de temps. Il s'agit de prendre également en compte les éléments suivants :

- votre temps de **formation** sur une librairie / technologie nouvelle ;
- la **compréhension** du besoin business pour toutes les parties prenantes du projet ;
- le travail en amont à accomplir pour **mûrir le besoin** afin qu'il soit **prêt** à être effectivement développé ;
- la constitution / intégration au sein d'une **équipe**.



Ne craignez pas de vous tromper dans vos estimations ! La vision en début de projet ne permet pas d'avoir une idée parfaitement claire et ce n'est pas grave. Le but est avant tout d'être **transparent**.


Par ailleurs, il n'est pas évident de bien connaître sa capacité de production du premier coup, encore plus lorsque nous sommes débutants. Gardez simplement en tête que la clé pour que le projet ne soit pas mis en

péril est la **communication**.

Oh la communication, un sujet crucial ! La réputation des développeurs a la peau dure : souvent vus comme des collaborateurs très repliés sur eux-mêmes, très peu dans la discussion et surtout des personnes que l'on ne peut pas comprendre lorsqu'ils parlent de ce sur quoi ils travaillent... Bref, a priori des personnes inaccessibles. Il vous appartient de casser ces préjugés !



Comment faire pour parler de ce que je fais sans trop être technique alors ?

Tout est question d'adaptation. Les développeurs sont vus comme des magiciens  et pourtant il est dans votre intérêt de démystifier vos tâches quotidiennes afin que l'on ne vous voie surtout pas comme un faiseur de miracle. Cela facilitera aussi vos négociations et les interminables "*mais cela devrait être super simple pour toi !*". Expliquez ce que vous faites, partagez vos difficultés, expliquez (sans infantiliser), prenez le temps de montrer aussi ce que vous faites, encore plus si votre interlocuteur est demandeur.

Par ailleurs, aidez à mettre en place des outils efficaces de communication.

Pour finir sur la communication, je vous laisse jeter un œil sur ce proverbe de [Bernard Werber](#) et vous laisse méditer sur le sujet :

Entre ce que **je pense**,

ce que **je veux dire**,

ce que **je crois dire**,

ce que **j'ai dit**,

ce que **tu entends**,


et ce que **tu comprends**,

il se peut qu'on ait des **difficultés** à communiquer, mais ***essayons quand même !***



Si nous récapitulons les grands axes à explorer en plus de la production de code, nous avons :

- le triptyque `budget` , `délai` , `périmètre fonctionnel` ;
- vos estimations ;
- la communication.

Gardez-les en tête tout au long d'un projet et pour tous les projets que vous aurez à mener à bien. Conservez-les comme le *manuel* du développeur capable de s'adapter à toutes les situations comme un vrai ninja .

N'oublions pas le principal tout de même, la production de code. Ce dernier doit répondre à toutes sortes de préceptes qui à eux seuls sont en réalité des fonctionnalités à part entière. Le code que vous produirez devra être :

fonctionnel

maintenable

en adéquation avec le besoin exprimé

testable

testé

évolutif


performant

facilement compréhensible

respectueux des bonnes pratiques

produit dans les temps

bien architecturé ...

Toutes ces problématiques auxquelles il faut répondre, parfois seul...  Il va vous falloir adopter une stratégie, une méthodologie et utiliser les bons outils. Le travail en équipe ne facilite pas les choses si quelques règles ne sont pas observées.

Le travail en équipe



Il est important de déterminer des règles qui permettront à tous les membres de l'équipe d'apporter la bonne pierre à l'édifice.

Que ce soit dans le cadre d'une création d'équipe ou que vous intégriez une équipe, vous aurez à faire en sorte que votre travail soit vu et connu de tous, adoptez tous les outils qui vous permettront de communiquer / collaborer le plus facilement possible.

Voici une liste d'outils intéressants à explorer et mettre en place :

- Pour proposer des patches de code pour de nouvelles fonctionnalités/corrections : [Github](#).
- Pour lister vos tâches / user stories : [Pivotal](#), [Jira](#), [Trello](#).
- Pour suivre votre activité : [Asana](#).
- Pour communiquer : [slack](#), [workplace](#).

Pour vous aider à faire votre choix et argumenter ceux-ci auprès de votre équipe, pensez que les outils à adopter devraient faciliter les problématiques suivantes :

- s'assurer que **le besoin est bien compris** par toutes les parties prenantes du projet ; ce que vous allez produire doit être validé, mais n'empêchera pas d'être modifié par la suite ;
- le **flux de travail** (plus communément appelé *workflow*) doit être bien établi et bien accepté de tous ; il s'agit de pouvoir sereinement ajouter / modifier du code, partager son avancement, faire relire son code et intégrer du code de manière continue ;
- assurer votre **montée en compétence** en continu ;
- être en mesure de maintenir votre code par vous-mêmes ou par n'importe qui d'autre (la clé du succès : la **lisibilité**).

L'une des méthodologies souvent employées est SCRUM (méthodologie agile), néanmoins selon les équipes son application diffère pour faire en sorte que son adoption se passe bien. Quelques conseils tirés de mon expérience concernant le déroulement d'un sprint :

Adoptez une définition du **"ready"** (qui signifie "prêt" en anglais) : seul ou en équipe, déterminer tous les critères qui vous permettront de dire si vous avez tous les éléments pour que cette user story soit prête à partir en développement.

Prenez le temps de lire les user stories avant les poker plannings afin de gagner du temps et avoir le recul nécessaire pour poser les bonnes questions et surtout déterminer si la user story est dite **"ready"**.

Adoptez une définition du **"done"** (qui signifie "terminé" en anglais) : une tâche et/ou une user story n'est pas forcément terminée lorsque votre [pull request](#) a été acceptée et que votre branche est enfin fusionnée dans la branche principale. Cette définition est primordiale afin de vous aider lors de vos estimations, et qu'ainsi, il n'y ait pas de quiproquo lors de la démonstration.

Plus que du code !



En tant que développeur, l'on attend de vous une production de code "qualitative". Mais il est important de faire un focus sur ce qu'est la **qualité**, d'autant qu'en fonction de votre interlocuteur, les définitions changent 😊. Pour nous faciliter un peu le travail, nous allons subdiviser les types d'interlocuteurs en deux : les membres de l'équipe qui ne sont pas techniques et ceux qui le sont. Dans les deux cas, il faut que la qualité dite "perçue" soit à la hauteur.

Commençons par vos *interlocuteurs non techniques* : il arrive souvent que l'on pense que leurs attentes ne s'arrêtent qu'au fait d'avoir une application qui fonctionne, que nenni ! La qualité perçue couvre également les éléments suivants :

- *la bonne compréhension du business* - lors de vos démonstrations il vous faudra montrer que la nouvelle fonctionnalité qui a été développée s'inscrit bien dans la continuité de l'objectif business global de l'application (développez donc vos talents de communicants 😊) ;
- *la vitesse d'exécution* - lorsque l'on vous demande d'effectuer une modification et que cela impacte du code que vous avez créé et que vos estimations atteignent des plafonds impressionnants, la qualité perçue de votre travail devient très mauvaise ;
- *l'anticipation* - cet aspect là est à double tranchant parce qu'il demande de faire un peu plus que prévu, or un adage bien connu dit "*le meilleur est l'ennemi du bien*". Là dessus, prenez simplement le temps d'être force de proposition et posez des questions aux commanditaires pour pouvoir vous projeter et penser votre code en adéquation.

Je pense qu'il est impossible de faire en sorte que son code soit prêt à être modifié sans difficulté. L'anticipation est un **bon réflexe**, mais n'est **pas une priorité** : en effet, vous pourriez vous retrouver à développer bien plus que prévu (donc exploser vos délais) et sans doute passer à côté de ce que l'on vous demande vraiment. Il faut trouver le juste milieu. Aidez vous de vos coéquipiers pour poser les bonnes barrières.

Voyons maintenant ce qu'attendent en général vos *interlocuteurs techniques* lorsqu'il s'agit de qualité :

- *du code maintenable* - le code que vous allez produire doit être modifiable par n'importe quel développeur. Pour se faire, il faut que vous pensiez à la suite du projet et surtout au fait que vous n'écrivez pas du code pour vous faire plaisir uniquement. Pour se faire, pensez aux principes [SOLID](#) (nous les détaillerons un peu plus loin dans le cours), à rendre votre code le plus lisible (facile à lire, bien documenté) et suivre les règles imposées par les bonnes pratiques communes à l'équipe ;
- *pensez compatibilité* - "Nan mais ça fonctionne sur ma machine !" 💡. Si vous aviez l'intention à un moment donné de prononcer cette phrase, eh bien pensez simplement qu'une fois la mise en production de votre application (envoi en ligne), elle ne sera pas consultée sur votre machine (du moins pas uniquement) ! Ainsi,

pensez qu'il existe une multitude de navigateurs et de configurations machine. Avant de vous lancer dans le développement d'une fonctionnalité/une correction, pensez à établir les règles concernant la compatibilité lors de l'élaboration de cette tâche.

- *accessibilité* - C'est un sujet important qui est souvent laissé de côté. Prenez le temps de vous sensibiliser au sujet et surtout connaître les règles de base afin d'offrir une expérience utilisateur la plus agréable possible au plus grand nombre (le référentiel est [ici](#)).

Vous l'aurez compris, la maîtrise du langage ne fait pas de vous un bon développeur. Tous les sujets annexes propres au mode projet sont tout aussi importants et pourront faire de vous un développeur que l'on s'arrache 😎.

Apprentissage continu



Pour terminer ce premier chapitre, parlons un peu de votre montée en compétence. J'aime à dire qu'un bon développeur est un développeur curieux. Vous êtes sur le point de passer beaucoup de temps à apprendre un langage qui est (peut-être) encore complètement inconnu pour vous. Vous rencontrerez des difficultés, et c'est normal. Le tout est de persévérer et surtout de rester toujours aux aguets ! Des nouveautés il y en a tous les jours dans le monde du développement. Avec ce cours nous prenons les bases, mais nous apprenons également à apprendre toujours plus. Vous rencontrerez également beaucoup de succès qui vous mettront en confiance pour avancer au point que vous pourriez vous dire que vous avez tout vu. Attention ! Le monde du développement web regorge de nouveautés et sujets à apprendre.

Je vous propose d'adopter le mantra suivant :

Je sais que je ne sais pas.

- Socrate

Ne vous reposez pas sur vos acquis !

Pour vous y aider, vos amis seront :

- la documentation des technologies que vous devez prendre en main ;
- une petite liste de sites de référence pour pouvoir poser des questions, et trouver des réponses (et peut être en apporter aussi) ;
- la communauté autour de la technologie que vous apprenez/utilisez est un bon moyen de partager, réseauter et surtout se tenir au courant de ce qu'il se fait ;
- [Twitter](#) est un bon média pour suivre les personnes/organisations qui permettront de suivre les nouveautés.

En ce qui concerne PHP en France, la communauté gravite principalement autour de l'[AFUP](#) (Association Française des Utilisateurs de PHP). Des rendez-vous réguliers sont organisés (meetups et conférences), dans toute la France. Faites-y un saut 😊 !

Et maintenant, je vous propose de rentrer dans le vif du sujet avec un chapitre important et incontournable : la configuration de votre ordinateur afin de pouvoir commencer à travailler avec PHP !

[Concevez votre site web avec PHP et MySQL](#)[Fonctionnement d'un site écrit en PHP](#)

The author

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Check out this course via



eBook



Livre papier



PDF



Vidéo

OpenClassrooms

[Who we are](#)

[How our courses work](#)

[Jobs](#)

[Contact us](#)

Partnerships

[Affiliate program](#)

Learn more

[Terms of Use](#)

Follow us

[OpenClassrooms blog](#)



[Español](#)

[Français](#)