SaaS Guidelines

By Seed4Soft

Retour d'expériences et Best Practices

Novembre 2017

seed4soft®
mentors for your software story

Préface

Face à la multiplication des offres logicielles en mode SaaS –*Software as a Service*– et l'adoption massive du Cloud par les entreprises, les membres de Seed4soft, club d'investisseurs regroupant 20 experts du logiciel B2B (www.seed4soft.com), ont décidé de publier ces *Saas guidelines* à partir de leurs expériences au sein d'éditeurs SaaS de différentes tailles ou en tant qu'investisseurs.

Ce document est à destination des startups SaaS en phase d'amorçage (*Seed*) ou de décollage (*Series A*), aux éditeurs de logiciel en transition vers le modèle du SaaS, ainsi qu'aux investisseurs souhaitant approfondir leur connaissance du sujet et mieux analyser les projets d'investissements dans ce domaine, et par la suite mieux les accompagner.

Ces *SaaS guidelines* sont des pistes de réflexion et non un modèle fixe à appliquer, notamment compte-tenu de la diversité des éditeurs SaaS, entre ceux qui sont des *pure players* avec un produit en self-service disposant d'un outil de configuration et ceux qui sont sur un modèle *custom* avec un niveau plus ou moins élevé de customisation et/ou d'intégration. Ce dernier modèle implique des ressources de type *Professional Services*, qui au début seront fournies par l'éditeur lui-même, puis qui seront progressivement externalisées auprès de consultants ou d'intégrateurs lorsque l'offre sera suffisamment reconnue sur le marché.

Enfin nous nous excusons par avance de la terminologie « franglaise » utilisée dans ce document, mais la plupart des termes et des acronymes sont d'origine anglo-saxonne et universellement reconnus et adoptés.

Dans ce guide nous traiterons des aspects suivants :

1. Le pricing/packaging et le business model
2. L'organisation marketing, commerciale, services et le support client
3. Les process R&D
4. L'organisation produit / R&D
5. Les choix technologiques et l'infrastructure Cloud
6. Les aspects contractuels et les indicateurs financiers
7. Les éléments de valorisation

N.B.
- il existe de nombreux sites, newsletters et blogs qui traitent de ces sujets : nous recommandons particulièrement le groupe de discussion de Mark Briercliffe sur Linkedin : https://fr.linkedin.com/company/the-saas-group. Concernant les indicateurs financiers un excellent document a été publié par l'investisseur américain Andreessen-Horowitz https://a16z.com/2015/08/21/16-metrics/
- Pour les éditeurs SaaS nous leur recommandons fortement d'adhérer à TECH'IN France http://www.techinfrance.fr/ : cette association créée en 2005 qui regroupe plus de 400 entreprises du secteur du numérique dont un très grand nombre d'éditeurs SaaS de toutes tailles, organise régulièrement des évènements de networking entre ses membres et des ateliers thématiques à travers ses commissions R&D, marketing, finance, investisseur, international, etc.

I. Pricing/packaging et business model

Considérations sur le pricing et le packaging

Il convient de se poser très tôt les bonnes questions concernant le *pricing* et le *packaging* de l'offre et son modèle économique en analysant la typologie de ses clients et des utilisateurs chez ces clients. Un fois cet exercice terminé, il est important de comprendre comment fonctionne l'achat chez ses différents clients et d'adopter une stratégie adaptée pour chaque type de client. Par exemple dans les grandes entreprises, il peut être judicieux de définir un packaging qui permet de ne pas passer par le département achat mais par les opérationnels dans les départements concernés.

Le *pricing* du logiciel doit être simple (« *KISS = Keep It Straight & Simple »)* avec au maximum quelques paramètres : *per user ; per volume of data ; per transaction ; etc.*

Il est également nécessaire d'intégrer la notion d'*upsell* dès le début de la réflexion sur le *pricing/packaging*. Il y a généralement 2 axes :

- *Upsell* avec des *features* supplémentaires (par exemple séparer les fonctions d'Analytics des fonctionnalités de base du logiciel).
- *Upsell* en volume supplémentaire (par exemple accroissement de la base des utilisateurs ou du volume de stockage chez un client).

Enfin il peut être intéressant de packager des offres pour une meilleure lisibilité et en fonction des segments de marché que l'on vise. C'est ce qu'on appelle le « *tiered pricing* ». Par exemple :

- Le pack Pro est une offre jusqu'à 10 utilisateurs, a 1.000 € par mois, sans le module Analytics.
- Le pack Enterprise est une offre jusqu'à 100 utilisateurs, a 5.000 € par mois, avec le module Analytics inclus.

Subscription based !

En SaaS, le modèle de l'abonnement est à privilégier. L'abonnement sera au minimum annuel, avec la possibilité de souscrire à des abonnements pluriannuels (24 ou 36 mois).

Outre l'aspect récurrent inhérent à ce modèle économique, il permet aux entreprises d'anticiper le coût d'usage, ce que ne permet un modèle par *unités de crédit* ou *pay-as-you-go* : lorsque l'on vend un logiciel d'entreprise en SaaS à des grands comptes le décideur ou le responsable financier préfèrera connaitre le coût à l'avance afin de ne pas sortir de son enveloppe budgétaire.

Credit-based ou Pay-as-you-go

Certains éditeurs fonctionnent sur la base d'unités de crédit : par exemple un logiciel d'analyse de vidéos permettra de traiter jusqu'à 100 vidéos. Le client va ainsi acheter un pack avec un certaine quantité de crédit, et rachètera « probablement » de nouvelles unités lorsque son quota sera atteint.

L'approche *Pay-as-you-go* est une variante, et permet à l'utilisateur de payer « à l'acte » ou pour une courte période, sans s'engager sur le long-terme. Cela signifie souvent que l'usage de la solution est ponctuel : par exemple un logiciel d'optimisation des ressources ou de la performance sera utilisé au début ou de temps en temps, et une fois les résultats obtenus et l'optimisation atteinte, le client ne l'utilisera plus.

Si l'intérêt de ces approches est d'abaisser le ticket d'entrée pour le client, la difficulté pour l'éditeur est de prévoir le « *repeat business* » pour un client donné.

Freemium et Try&Buy

Contrairement à ce que les grandes success story (Slack, Trello, Dropbox…) pourraient laisser penser le *Freemium* et *Try&Buy* sont des modèles peu fréquemment utilisés ; en outre ils sont réservés au cas des éditeurs ayant un produit utilisable facilement, sans formation préalable et sans *set-up*, avec un « onboarding » en self-service :

- Le modèle *Freemium* consiste à mettre à disposition de manière gratuite une version limitée du logiciel ; la limitation peut se baser soit sur des fonctionnalités, soit sur un niveau d'utilisation (jusqu'à tel niveau de stockage ou pour un nombre de projet déterminé) ; pour utiliser le logiciel au-delà de ces limitations le client passera sur une version *premium* (payante) ; étant donné que le taux de transformation des utilisateurs gratuits en utilisateurs payants est en général <5%, il faut donc investir des sommes très importantes en marketing pour acquérir des milliers d'utilisateurs gratuits afin d'avoir une centaine d'utilisateur payants .
- L'approche *Try&Buy* permet de mettre à disposition le produit dans une forme complète ou limitée sur une période limitée (1 à 3 mois).

Si ces modèles ne permettent pas d'acquérir des clients automatiquement, ils peuvent en revanche servir à générer une base de leads qu'il faudra ensuite qualifier et exploiter avec des moyens marketing traditionnels.

Les éditeurs développent généralement une approche de *Proof of Concept* (POCs) : ces POCs doivent être généralement payants, sauf exceptionnellement dans les phases de lancement afin de valider l'usage du logiciel auprès des premiers clients (*early adopters*). N.B. La multiplication des POCs non transformés en abonnement dans un délai raisonnable (<3 mois) doit amener l'éditeur à réfléchir sur son offre et son produit…

II. Customer-oriented organization

Dans le SaaS, les clients doivent être au cœur de l'organisation de l'entreprise en ne sous-estimant pas le service client compte-tenu du revenu récurrent attendu de la base installée (*Customer LifeTimeValue – CLTV*) au regard du coût d'acquisition des clients (*Customer Acquisition Cost – CAC*) souvent élevé.

Dans le cas général des acteurs SaaS B2B, et notamment ceux qui ont un modèle avec une part de customisation et d'intégration importante, on retrouvera, outre le service client (support/hotline), dans l'ordre les 4 piliers organisationnels suivants : le marketing, les ventes, les *Professional Services*, et une organisation « *Customer Success* ».

N.B. Il est à noter que dans les premiers stades de développement de la société ces différents postes sont amenés à être réalisés par les mêmes personnes. Par exemple un ingénieur avant-vente pourra intervenir sur la qualification des leads, et également au niveau de la mise en route une fois le client signé. De même un télévendeur (*Inside Sales*) pourra s'occuper de la qualification des leads. Il est néanmoins essentiel de planifier très en avance l'organisation future dans le plan de recrutement, avec les bons ratios (% d'avant-vente par commercial, % de *Professional Services* vs. *Customer Success*)

Génération du pipeline : l'équipe Marketing

Outre la dimension Marketing Produit qui couvre les *marketing materials* et une stratégie de contenu adaptée au produit, les activités de l'équipe marketing sont généralement divisées en deux parties :

- Identification des leads (*Lead generation*) grâce aux multiples méthodes d'identification de leads, via des canaux digitaux (adwords, blog, newsletter, webinars, white papers, etc) ou physiques (salons, conférences, évènements de type petit-déjeuner, PR/media traditionnels, etc)
- Qualification des leads (« *LeaD➔ DeaL* ») grâce aux télévendeurs ou aux ingénieurs avant-vente; ces derniers doivent connaitre le produit et avoir un focus sur les besoins du lead ;

N.B. Un outil de *marketing automatio*n est fortement recommandé, comme Marketo par exemple.

Transmission à l'équipe Sales

L'équipe des commerciaux intervient une fois le lead qualifié, et est en charge de la négociation et de la signature du contrat (closing).

En général les commerciaux sont incentivés sur l'ARR (*Annualized Recurring Revenue*, donc la part logicielle du contrat sur 12 mois). Le pourcentage peut atteindre jusqu'à 8 à 10% de l'ARR. Certains éditeurs ajoutent une commission sur les *Professional Services*, de l'ordre de 4 à 5%. A titre d'exemple les objectifs annuels / quotas d'un commercial à objectifs atteints (*On-Target Earnings* -OTE 100%) peuvent varier en fonction de la taille des contrats :

- Pour des contrats sur des entreprises de taille intermédiaire (ETI) : ~500k€
- Pour des contrats sur des grands comptes / Fortune 500 : ~800k€ a plusieurs millions

Transmission à l'équipe Professional Services

L'équipe *Professional Services* prend le relai à la clôture du deal pour la mise en route effective des clients. Ils accompagnent la phase de *set-up* –en mode projet typiquement– et aident le client à formuler au mieux ses besoins pour adapter la solution à leur environnement, notamment

lorsqu'il y a des intégrations à réaliser avec les autres systèmes d'information via les API ou Web Services disponibles. Ils sont également en charge de la formation des utilisateurs. Leur travail est clé car il conditionne l'adoption rapide de la solution par un maximum d'utilisateurs.

Si possible il faut séparer l'offre Service Professionnel de l'offre Formation au produit surtout si les clients ciblés sont les grandes entreprises. Celles-ci disposent souvent d'un budget spécifique pour la formation et sont habituées à payer des prix de 1.000 à 1.500€ par jour de formation. Il faut donc penser à mettre en place un catalogue de formation détaillé pour maximiser ce revenu. Les clients bien formés, c'est aussi la garantie d'une meilleure utilisation du produit et éviter les frustrations de utilisateurs qui se plaignent de la qualité du produit souvent par méconnaissance des fonctionnalités disponibles.

Transmission à l'équipe Customer Success

Dans le cas d'un client présentant un fort potentiel de développement, et une fois celui-ci « onboardé » et opérationnel, il est suivi par l'équipe *Customer Success* qui est en charge de la satisfaction client et notamment de renouveler les abonnements arrivés à échéance et de développer *l'upsell.*

Un des grands avantages du SaaS, souvent sous-estimé, est que l'éditeur peut « voir » l'usage que les clients font du logiciel, contrairement aux éditeurs « *on-premise* » dont le produit est installé sur les ordinateurs ou les serveurs de ses clients et qui sont donc incapables de savoir si le logiciel est réellement utilisé et comment il est utilisé.

L'éditeur SaaS doit donc mettre en place très tôt dans son développement les outils d'administration et d'analyse (« *usage metrics* ») pour mesurer l'activité par client (Daily Active Users-DAU, Weekly Active Users-WAU, Monthly Active Users-MAU) avant l'échéance et déterminer ainsi si le client risque de renouveler ou pas. Ces métriques sont également clés pour un investisseur car elle lui permette de déterminer si le produit est *must have* ou *nice to have* et s'il y a une certaine *stickyness* au produit.

L'équipe *Customer Success* est typiquement incentivée sur le renouvellement des contrats et sur l'*upsell*. Leur part variable est en général de l'ordre de 20% du salaire fixe, et se calcule sur la base d'un pourcentage de l'ARR du contrat renouvelé.

III. Les process R&D

Les process sont la clef de voûte d'une organisation R&D en SaaS. Nous en recommandons deux qui nous semblent particulièrement pertinents à mettre en place : les méthodologies « Agile » et « DevOps »

La méthodologie Agile

La méthodologie Agile se rapporte essentiellement aux personnes de l'entreprise chargées de la partie Produits, de leurs définitions à leurs livraisons aux équipes d'installation ou de production. Elle prend tout son sens à partir du moment où l'entreprise emploie quelques

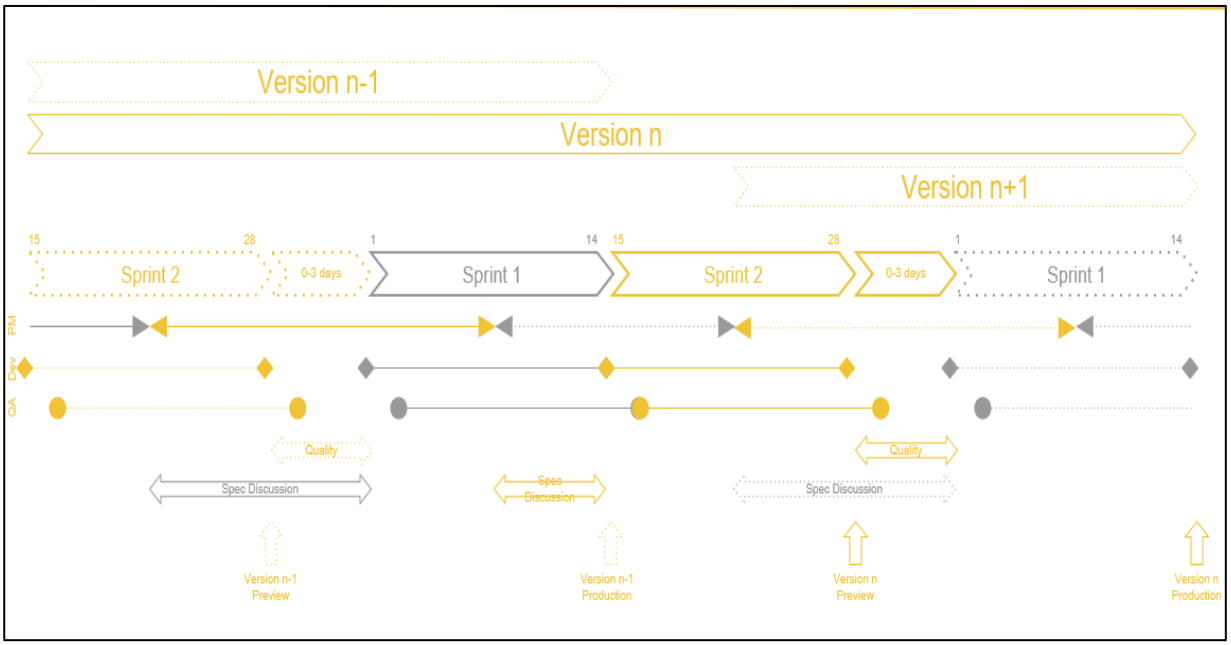
développeurs. Il est à noter que cette méthode ne doit pas être appliquée de manière dogmatique mais de manière pragmatique, c'est-à-dire en se posant les bonnes questions. Il existe de nombreux ouvrages et *frameworks* d'implémentation de cette méthodologie (Scrum, KanBan, SaFE, ...). Nous en rappellerons seulement les grands principes ci-après.

La méthode Agile allie bon sens et quatre points de formalisme aussi appelés « *ceremonies* ». Ces points sont à appliquer sur une unité de temps – le *sprint*- qui dure généralement quinze jours et qui vise à livrer du code qui fonctionne (*working deliverable*), visible par le Client.

Planning	→	Commitment
Review	→	Verify engagement
Retrospective	→	Feedback loop
Daily Standup	→	Communication

N.B. Pour le *daily standup* (réunion quotidienne), il est d'usage d'accorder 1mn30 par personne.

Cette méthode est souvent couplée à du déploiement continu pour les entreprises ayant adopté la méthode tôt dans leur développement. Pour les entreprises qui migreraient vers l'Agile, utiliser un cycle de release produit d'un mois divisé en deux sprints de deux semaines est un bon compromis.



Les jours prévus entre le *sprint 2* d'une version et le *sprint 1* d'une nouvelle version sont dédiés à la vérification de la qualité du travail fourni. Les discussions sur les spécifications interviennent entre la fin d'un sprint et le début d'un autre. Dans l'exemple, ci-dessus la version n-1 est en preview à la fin du sprint 2 de la version n-1 et en production à la fin du sprint 1 de la version n. La version n est quant à elle en preview à la fin du sprint 2 de la version n et en production à la fin du sprint 1 de la version n+1

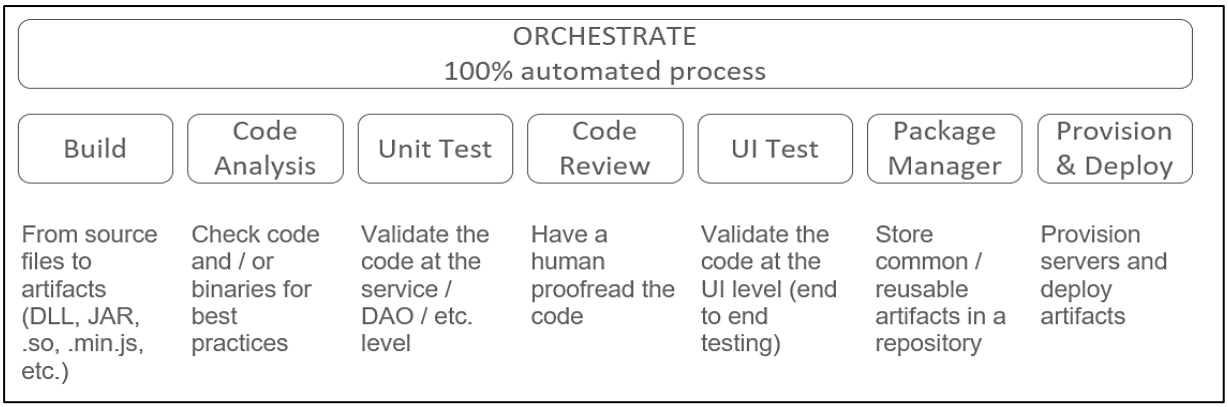
La méthodologie DevOps

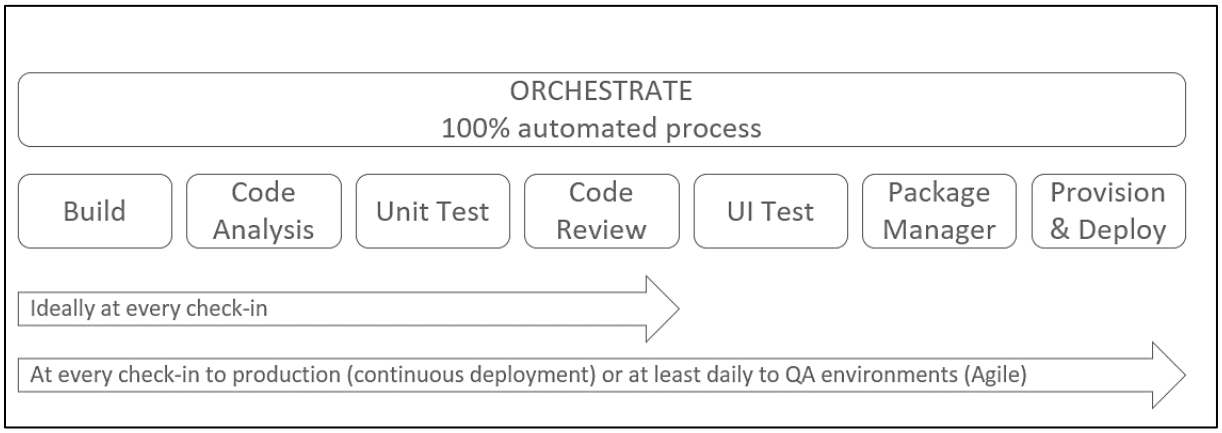
En complément d'Agile, la méthodologie DevOps se rapporte quant à elle davantage au cycle de développement technique. Elle vise à automatiser et rendre transparentes les transitions entre les différentes phases du développement et de la production (*Continuous Build, Integration, Deployment and Production*). Là encore, comme pour Agile, de nombreux sites et ouvrages décrivent cette approche (www.devops.com, « Continuous delivery » de Jez Humble et David farley, « DevOps for Developers » de Michael Huttermann, …)

Une version simplifiée du concept DevOps est *Build Pipeline*, notamment supportée par des modules du même nom chez Jenkins et Atlassian.

DevOps est à mettre en place dès le premier jour et avec le premier développeur. Certaines technologies poussent vers ce choix, et sont donc plus structurantes (.NET, Java), d'autres sont beaucoup plus laxistes, et peuvent entrainer des dérives (PHP, Ruby, etc.).

Les grandes étapes peuvent se schématiser ainsi





IV. L'organisation Produit

Le Product Management

Le Product Management regroupe en réalité deux missions celles du *Marketing Product Manager* et celles du *Product Owner*. Si ces deux rôles sont endossés par le même personne au début, il convient de préciser que le rôle de *Product Owner* est un rôle à temps plein qui nécessite donc de recruter quelqu'un pour ce poste à terme.

Le rôle du *Marketing Product Manager* est de construire une vision à 12, 18 ou 24 mois de la roadmap du produit et de son business plan. Il est, par exemple, présent sur les salons, et doit connaitre les concurrents et les besoins des clients. C'est également lui qui est chargé de parler aux analystes comme Gartner et Forrester.

Le *Product Owner* est lui responsable de l'interfaçage entre le *Marketing Product Manager* et l'équipe de R&D. Il est en contact directement avec les équipes de R&D et transforme la roadmap en user stories. Il représente le Client (au sens large, qui peut être un marché) et s'assure que les livrables correspondent à ses attentes et au business plan.

Le CTO et le VP Software Engineering

Le CTO est responsable des choix technologiques. Le VP Software Engineering est en charge du management et du recrutement des équipes de R&D et de la mise en place des process.

Dans les entreprises ayant moins de 25 salariés, c'est-à-dire avec une R&D de 10 développeurs environ, le développeur qui a commencé le développement du produit est souvent à la fois CTO et VP Engineering. A un certain stade, cette personne se retrouve confrontée à un choix. Il lui faut choisir entre le poste de CTO et celui de VP Engineering ou assumer de faire les deux et avoir une double charge de travail.

L'équipe des Développeurs, les QA et les DevOps

Pendant une grande partie du temps, l'équipe des développeurs devra se charger non seulement du développement du logiciel mais également de la liaison avec les infrastructures Cloud – ce qui deviendra ensuite le rôle de personnes dédiées, que par extension de langage on appellera *DevOps.* Elle devra également endosser le rôle de QA (*Quality Assurance*) jusqu'à ce que ces fonctions critiques pour l'entreprise nécessitent des personnes à temps plein.

Il est important à ce stade de communiquer sur les quatre paramètres qui façonnent la vie des équipes Produit : date de livraison, budget alloué, fonctionnalité à livrer, et qualité du résultat. Tous ces paramètres sont des variables… sauf la qualité, qui ne doit pas être négociable ! Comme souvent la date est fixée d'avance, que le budget est contraint, la seule vraie variable d'ajustement du planning est donc la fonctionnalité. Cela peut paraitre contre-intuitif, surtout pour une start-up, mais l'expérience montre que l'on perd rarement un client par manque de fonctionnalité, mais régulièrement par manque de qualité.

Si les développeurs ont donc la qualité en tête de leurs préoccupations tout au long du cycle Produits, et s'il y a de très bons process de développement en place, on peut repousser l'arrivée d'une équipe *Quality Assurance*. Cette équipe s'assure que la qualité du produit final correspond au cahier des charges et aux attentes du Client. Il est donc possible de s'en passer dans les premiers stades de la société, encore une fois à la seule condition que les développeurs mettent la qualité en priorité maximale, et réalisent notamment eux-mêmes les tests unitaires et les test systèmes.

Les *DevOps* sont le lien entre les équipes techniques de développeurs et les infrastructures (Cloud Service Provider). Ils sont responsables de l'installation en production des logiciels (passage des environnements de pré-production à production par exemple), du provisioning, du monitoring, de la performance. Leur rôle consiste, entre autres, à investiguer les problèmes, les attribuer aux développeurs selon leur origine, et à modifier les process et les outils pour que ces incidents ne se reproduisent plus.

V. Les technologies R&D et l'infrastructure Cloud

Le choix des technologies – Back-end & Front-end

Traditionnellement on trouve 2 types de technologies chez les éditeurs SaaS :

- Les technologies « *back-end* » qui se focalisent sur les traitements applicatifs au niveau du serveur sur le Cloud et des bases de données.
- Les technologies « *front-end* » qui traitent des aspects d'interface utilisateur (pour l'utilisateur final et l'administrateur), que ce soit sur le navigateur et/ou sur une application mobile native.

Concernant le choix de la technologie *back-end*, il y a deux catégories :

- Celles, structurantes, qui imposent intrinsèquement un process : Java, .NET, …
- Celles, peu ou moins structurantes, qui nécessitent de mettre en place des process bien établis : Ruby, PHP, Python…

Pour les technologies ci-dessus, il faudra leur associer une technologie différente pour le *front-end* avec du JavaScript /HTML5 / CSS3 ou des technologies mobiles

Il est à noter que Node.js est une technologie Javascript pour les back-end, qui peut permettre d'uniformiser la technologie utilisée pour le *back-end* et le *front-end* autour de Javascript. Il ne faut pas surestimer cette uniformisation car la versatilité de Javascript fait que chaque *framework* est quasiment un langage en soit, et surtout parce que le développement front-end et back-end sont intrinsèquement très différents. Néanmoins node.js est une technologie structurante qui peut s'avérer intéressante pour minimiser la stack technologique dans un projet jeune et ainsi favoriser les échanges de connaissance et l'évolution au sein de l'équipe R&D.

Autre point clé sur les choix technologiques : celui des licences open source. Certaines licences comme GPL sont dites « invasives » et peuvent avoir des impacts directs sur la licence du logiciel

qui les embarque, en fonction en particulier de son mode de déploiement. Elles peuvent être considérées comme de réels « *showstoppers* » lors de la due diligence de fonds d'investissement ou d'acquéreur dans un process de M&A. Il importe donc :

- de connaitre raisonnablement précisément les licences des composants que l'on emploie,
- de maintenir une liste de tous les composants tiers employés et de leur licence,
- d'éviter les composants aux licences invasives.

Enfin une attention particulière doit désormais être portée au *framework* react.js, très populaire sur le front end, mais dont la nouvelle licence définie par Facebook interdit toute action en violation de propriété intellectuelle à l'encontre de Facebook. Selon les secteurs cela peut être un risque significatif pour un acquéreur…

L'infrastructure Cloud

Concernant l'infrastructure il y a deux questions à se poser impérativement :

- Est-ce que je choisis un PaaS ou un IaaS ?
    - Un IaaS - *Infrastructure as a Service*- correspond à la partie infrastructure de cloud (machines virtuelles et stockage chez un hébergeur par exemple Amazon Web Services, Microsoft Azure, Google Compute Engine ou OVH). Si on décide d'utiliser un IaaS on doit gérer toute la partie middleware ainsi que le déploiement et la scalabilité. Cela permet une plus grande liberté mais l'inconvénient est qu'il faut gérer beaucoup plus de choses par soi-même.
    - Un PaaS - *Platform as a Service* - est une plateforme cloud qui fournit des services techniques de haut niveau tels qu'une plate-forme d'API, des composants *serverless* pour des microservices, du Hadoop as a service, de l'orchestration ou du workflow, des files asynchrones, des services de machine learning, etc. Typiquement le PaaS impose un modèle de développement mais prend en charge une grande partie des problématiques de déploiement, de scalabilité et de sécurité (DDoS) : un bon PaaS peut permettre de concentrer la R&D sur le fonctionnel à valeur ajouté. En revanche, il n'est pas facile de changer de PaaS : les principaux PaaS –Amazon AWS, Microsoft Azure, Force.com (Salesforce) et Google Cloud Platform– offrent souvent des services de natures similaires, mais incompatibles dans les faits.

    Une idée courante est que l'on peut changer de fournisseur IaaS ce qui donne une plus grande liberté, une plus grande portabilité et rend moins dépendant qu'une solution PaaS. Si un PaaS crée clairement une dépendance à cause du manque de compatibilité entre les plateformes, il faut nuancer la liberté effective fournie par un IaaS. La vraie difficulté de la portabilité ce n'est pas le code mais le volume de données : une fois que le volume de données est important, que la solution SaaS a des clients en production et tourne 24x7, toute migration est excessivement complexe. En revanche si un PaaS peut permettre d'augmenter la vitesse d'exécution, il comporte un risque d'être inacceptable pour tel ou tel acquéreur potentiel.

Est-ce que je manipule directement ou indirectement des « données personnelles » ?

- Pour rappel, le 25 mai 2018 deux règlements européens (GDPR et probablement « *e-privacy* » en cours d'adoption par l'UE) viennent encadrer très sévèrement tous les traitements de données personnelles de citoyens européens, ainsi que les traitements de données personnelles de non-européens effectuées sur le territoire de l'Union. Les sanctions « administratives" qui y sont liées sont extrêmement dissuasives (jusqu'à 4% du chiffre d'affaires mondial).
- En substance ces règlements donnent une définition large de la donnée personnelle (ex : une adresse IP dynamique est une donnée personnelle) et fixent le principe d'un quasi propriété d'une personne sur ses données. En conséquence tout traitement ou échange est en pratique interdit sans un véritable consentement, régulièrement révocable. En parallèle le « device » est propriété de l'individu, et donc ses capacités de calcul ou de stockage ne sauraient plus être utilisées sans consentement : ce qui inclut du code exécuté dans le navigateur ou des cookies par exemple (Règlement « e-Privacy »).
- En cas de manipulation de données personnelles directe ou indirecte, il ne s'agit plus seulement de les stocker sur le territoire européen : si les transferts hors Union deviennent quasiment interdits, les obligations sont plus vastes. En particulier le principe de « *security by design* » force pratiquement l'anonymisation et le chiffrement de toutes les données personnelles.
- Ces règlements alignent enfin la responsabilité du donneur d'ordre et celle du sous-traitant quant aux données personnelles : le prestataire SaaS devient pleinement responsable des traitements qui lui sont confiés et doit toujours agir conformément à la GDPR et à « *e-Privacy* ». L'éditeur SaaS doit garantir à son client (et aux utilisateurs finaux, clients de son client) qu'il ne procède à aucun autre traitement sur sa plate-forme.

Multi-tenant ou single-tenant ?

Le *multi-tenant* consiste à imbriquer les données de plusieurs clients dans une seule plateforme. Cela n'a d'intérêt que pour l'éditeur car c'est avant tout une réponse économique : si les éditeurs pouvaient héberger et opérer 100 clients sur 100 plateformes dédiées pour un coût moins élevé et avec autant de vélocité qu'une seule plateforme, alors tous les acteurs du SaaS seraient *single tenant*.

Il est à noter qu'en B2C, le *multi-tenant* est quasiment incontournable. En B2B, notamment avec des grands comptes cela n'est pas toujours pertinent.

Enfin, l'étanchéité des données est primordiale quand on choisit du *multi-tenant* pour éviter que les clients perdent confiance dans la solution.

Questions à se poser concernant la production

L'équipe R&D doit se poser impérativement les questions suivantes :

- Ai-je la capacité à maintenir une version unique pour tout le monde ?
  Il est important de s'interroger sur ce point car cela joue sur le modèle économique notamment au niveau de la structure et de la croissance du coût de maintenance et de support

- Quel est le degré d'industrialisation de la chaine de production logicielle, du développement aux tests, à l'intégration continue, à la mise en production ?
  Cela se traduit par une mesure simple : Combien d'heures au minimum entre la fin de correction d'un bug critique en production et sa release dans un patch de production, tous tests compris ? Ceci permet de mesurer la performance « d'opérateur » du fournisseur, sa capacité à agir en urgence avec sérénité.

- Quel est la fréquence de release et mise en production ?
  Pour pouvoir opérer en mode SaaS il faut pouvoir mettre en production ou restaurer intégralement son logiciel n'importe quand. Un éditeur moderne utilisant Agile et DevOps alignera son cycle de release sur son cycle de sprint, c'est à dire tous les 15 jours, et prévoira un rythme de 4 releases majeures par an, et au moins autant de mineures.

VI. SaaS metrics & financials

N.B. ce chapitre s'adresse aux éditeurs qui ont choisi un business model sous forme d'abonnement.

Les métriques importantes : booking, MRR, churn

Une des caractéristiques principales du SaaS dans un modèle d'abonnement –et sa principale vertu– est que l'éditeur de logiciel devient « opérateur » avec à terme un revenu récurrent important au fur et à mesure que sa base installée de clients et d'utilisateurs augmente. Les éditeurs doivent donc s'inspirer des opérateurs télécom qui mesurent le churn, la croissance du revenu et autres métriques en priorité par rapport aux concepts traditionnels de chiffre d'affaires et de résultat.

En *Early Stage*, les métriques clefs sont, dans l'ordre d'importance :

- Le *Booking* (prise de commande) : il est essentiel de distinguer la partie logicielle (usage/abonnement) de la partie *professional services* (*set-up*, services de mise en route)

- Le *MRR (Monthly Recurring Revenue)* : c'est le revenu mensuel lié à l'utilisation du logiciel par les clients dans le cadre des abonnements signés, et qui permet de déduire l'ARR (*Annualized Recurring Revenue*, ARR = 12 x MRR).

- Le *churn* (taux d'attrition, ou à l'inverse le taux de rétention). Le *churn* se calcule sur une période donnée (par mois, par trimestre, par année) et ne peut être complètement modélisé qu'après 2-3 ans de commercialisation, lorsqu'un nombre suffisant de contrats échus seront arrivés à renouvellement. N.B. Le churn sur le MRR se calcule de 2 manières :
  - *churn* brut : c'est le taux de diminution du MRR du fait de réduction (en valeur) ou des non-renouvellements de contrats
  - *churn* net : c'est le churn brut augmenté de l'*upsell* sur les contrats renouvelés ou de la réactivation de contrats résiliés

- La croissance du MRR : MRR month N+1 = MRR month N + MRR from new accounts + MRR from *upsells* - MRR from *churn*; idéalement dans le modèle SaaS la croissance du MRR doit être > 15% par mois dans les premières phases de développement de la startup ; par la suite on visera une croissance annuelle forte : Mark Benioff, le fondateur de Salesforce, aime à répéter qu'avoir 35% de croissance annuelle de l'ARR revient à être flat !

Comme on le verra à la fin le MRR (ou l'ARR) et la croissance du MRR (ou de l'ARR) sont les indicateurs clés pour la valorisation de l'éditeur.

Une fois les premiers stades de développement passés, il devient primordial de calculer également :

- La *CLTV* (*Customer Lifetime Value*) : la valeur économique d'un client sur la durée de vie du produit ; de manière simplifiée la LTV = l'ARR divisé par le *churn*.
- Le *CAC* (*Customer Acquisition Cost*) : le coût de toutes les activités marketing et commerciales sur une période divisé par le nombre de nouveaux clients acquis sur la période en relation avec ces investissements marketing et commerciaux. Idéalement le *CAC* doit être inférieur à 3 à 4 x *LTV.*

Eléments contractuels : durée des contrats et mode de facturation

Par défaut, la durée des contrats est d'un an. Il n'est pas judicieux de faire des contrats par mois car il est très difficile ensuite de passer à des contrats annuels. Plus on attend plus c'est douloureux de passer du mois à l'année. Salesforce a failli faire faillite à cause de cela au début de son développement.

Il est d'usage de prévoir des contrats renouvelables avec une reconduction tacite, incluant un préavis d'1 ou 2 mois.

Par défaut, la facturation se fait en une fois un an à l'avance. Il est également possible pour des contrats de taille importante qu'elle soit semestrielle ou trimestrielle, à la demande des clients.

Budgéter ses bookings, son MRR, son P&L et son cash

Pour le prévisionnel (business plan/budget annuel) et le suivi du budget (reporting), l'éditeur SaaS doit se baser sur les indicateurs suivants :

- La taille moyenne de ses contrats (*Average Deal Size – ADS*) exprimé en ARR.
- Le cycle de vente moyen (*Average Sales Cycle – ADS*) exprimé en mois.
- Le cout d'acquisition client (*CAC*).
- Le taux d'attrition (*Churn*).

Pour budgéter son *booking* on distingue deux approches :

- A partir de ses forces de vente : Nombre de vendeurs x Sales *ramp-up* x Quota de vente avec une marge de sécurité de 20 à 30% ; cette approche est bien adaptée pour un éditeur établi sur un marché de renouvellement avec un *pain point* important. N.B. Une approche intéressante pour estimer rapidement le revenu qu'un commercial peut

atteindre sur une année en fonction de son objectif mensuel est celle de la « règle des 78 » très bien décrite dans le document http://www.intelliverse.com/blog/what-is-the-rule-of-78-and-how-does-it-apply-to-sales/

- A partir de ses clients existants : nombre de clients - churn + upsell (provenant d'une cohorte client sur la période considérée) + nouveaux clients x ADS ; cette approche fonctionne dans tous les cas, mais elle est plus pertinente dans le cas d'un marché nouveau (évangélisation).

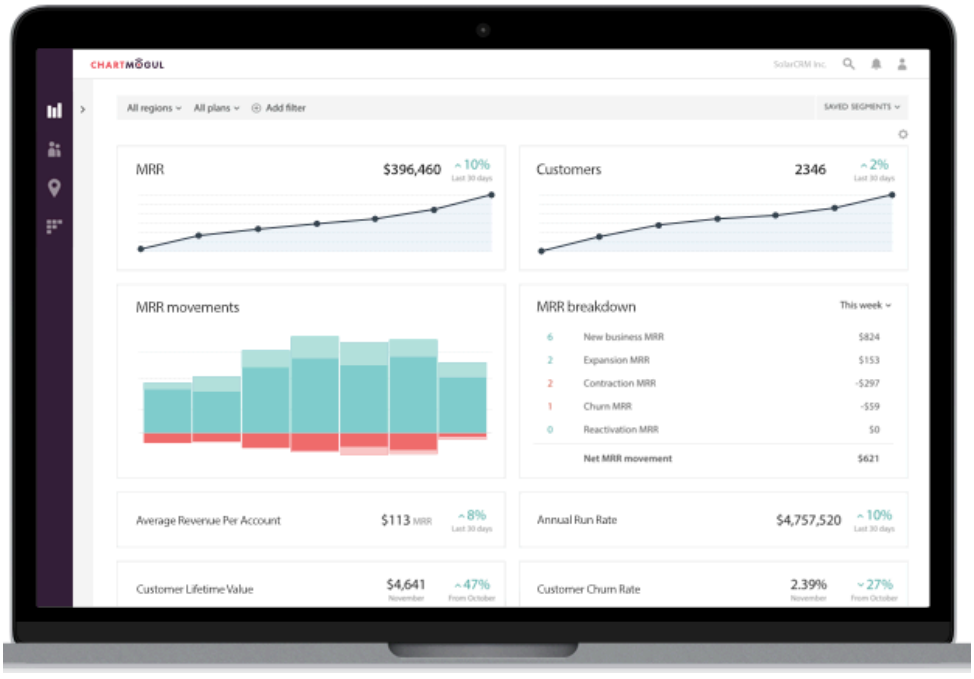
Enfin pour construire son compte de résultat (*P&L*) et son plan de trésorerie (*cash forecast*), il faut partir du *booking* et en déduire, de manière mensuelle, trimestrielle et annuelle :

- Le *Billing* (facturation sur la période) : cf. le prévisionnel de trésorerie
- Le *Recognized Revenue* (Chiffres d'Affaires comptable reconnu) : cf. le (P&L) :
    - Chiffre d'affaire *« software »* en ne comptabilisant que la quote-part de chaque contrat sur la période considérée (*)
    - Chiffre d'affaire *« Professional Services »* en fonction des prestations effectuées

(*) Exemple : Le 1er aout 2017, l'éditeur a conclu un contrat annuel de 24k€ (100% logiciel, pas de *set-up fees*), soit 2k€ de MRR, avec une facturation en une fois à la signature du contrat et un délai de paiement de 45 jours (en H.T.) :

- *Booking* Août 2017 = 24k€
- Facturation Août 2017 = 24k€
- Encaissement Septembre 2017 = 24k€
- Chiffre d'Affaires reconnu sur l'année 2017 = 10k€
- Produits constatés d'avance (*backlog* de CA) au 31/12/2017 = 14k€

N.B. le logiciel Chartmogul constitue une bonne base pour mettre en place et suivre ses *Saas financial metrics & analytics* :



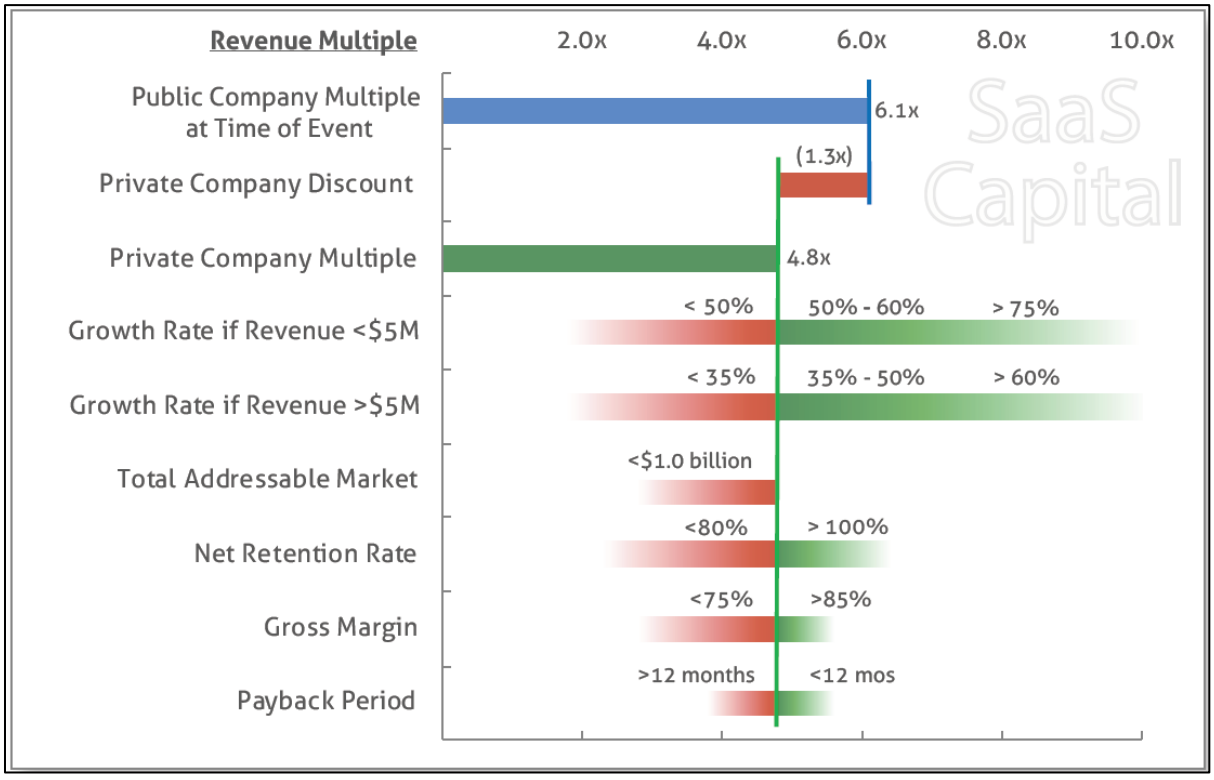
VII. SaaS valuation

Il existe de nombreuses études sur la valorisation des éditeurs SaaS disponibles sur Internet. Une des meilleures est celle de SaaS Capital (www.saas-capital.com) publiée fin 2016 :

http://www.saas-capital.com/resources/wp-saas-company-valuation-st?utm_source=SaaStr&utm_campaign=SaaStr_Post&utm_medium=post

En résumé on calcule la valorisation de l'éditeur sur la base d'un multiple de l'ARR corrigé (en + ou en -) à partir de :

- La croissance annuelle
- La taille de l'éditeur
- La taille du marché adressable (TAM)
- Le taux de rétention net
- La marge brute
- La période de *payback* (combien de temps cela prend pour recouvrir le CAC en moyenne)



Bien entendu ces multiples peuvent varier dans le temps -notamment compte-tenu de la fluctuation des index des sociétés SaaS cotées en bourse, et on s'attachera à rechercher les études les plus récentes, comme par exemple celle de Software Equity Group : http://softwareequity.com/seg-snapshot-2q17-saas-public-market-update/

Notes Personnelles