h e g

Générateur d'applications en marque blanche : le cas des jeux web

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Nathanaël KHODL

Conseiller au travail de Bachelor :

Rolf HAURI, Chargé d'enseignement HES

Genève, le 12 juin 2013

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre de Bachelor of Science HES-SO en Informatique de gestion. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 12 juin 2013

Nathanaël KHODL

Remerciements

Merci avant tout à mon conseiller au travail de Bachelor, M. Rolf Hauri, pour m'avoir suivi, conseillé, et fait preuve de disponibilité tout au long de ce semestre.

Merci au PhD Philippe Dugerdil pour ses conseils avisés.

Merci à Régine et Jehan, pour avoir apporté leur touche artistique au prototype.

Merci à Brigitte, pour la relecture de ce travail.

Merci à tous ceux qui ont pu m'apporter leur soutien dans l'élaboration de ce travail.

Merci spécialement à toutes les personnes que j'ai eu le plaisir côtoyer durant ma scolarité à la HEG.

Résumé

Ce travail a pour objectif de déterminer quels sont les points nécessaires à la création d'un générateur suffisamment polyvalent pour proposer du contenu en marque blanche.

En préambule, il est expliqué quelles problématiques ont soulevé la nécessité d'un tel générateur, autour du thème des jeux web.

Puis, il sera fait l'analyse de différents secteurs pour lesquels la réutilisation d'une même application peut s'avérer utile, sans pour autant la dupliquer. Une attention particulière est apportée aux sites de e-commerce, ainsi qu'au secteur des jeux par navigateur.

Enfin, suite aux observations faites dans les applications déjà existantes, il est développé un prototype permettant de démontrer le fonctionnement de la génération de jeux. Cette réalisation débute par la création d'une solution permettant d'administrer plusieurs projets semblables, puis enchaîne avec la mise en œuvre de cette solution.

Au final, ce travail abouti à une solution générale et exemplifiée offrant la possibilité de proposer plusieurs variantes d'une même application.

Mots clés : générateur, application, récursivité, paramétrage, marque blanche, jeux, web, social, navigateur

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

1.1 Choix du sujet

Dès mes premiers pas dans le développement informatique, je me suis passionné pour les jeux, préférant les créer plutôt que de me divertir avec. De par mes diverses expériences dans le milieu de l'informatique ludique, j'aspire à pouvoir apporter une plus-value à ce secteur, en répondant à certaines problématiques pouvant être rencontrées par d'autres développeurs plus ou moins expérimentés.

La génération d'une application ne s'adresse pas uniquement aux jeux, et il m'a semblé intéressant de pouvoir aborder un sujet pour lequel chaque projet est potentiellement touché.

Ce travail est donc appuyé par des motivations personnelles me permettant, via la réalisation d'un prototype, de surmonter des obstacles rencontrés par le passé, et de les contourner de manière à ce qu'une problématique globale soit traitée.

1.2 Problématique traitée

Lors de la réalisation d'un nouveau projet, tout développeur se doit de penser à ses évolutions. La maintenabilité est un élément incontournable de chaque projet, et ne pas la prendre en compte est un choix peu recommandable, lorsque l'on sait que dans l'industrie par exemple, 80%[1] du budget IT est dédié à la maintenance. Ainsi, il n'est pas rare de développer une application en prenant en compte les nouvelles fonctionnalités que l'on pourrait être amené à implémenter par la suite. Cet aspect est nécessaire à la plupart des projets, quelle que soit leur taille.

Bien que l'on prévoie habituellement un système gérant parfaitement l'intégration de nouveaux modules, l'ajout de nouvelles interfaces, l'apparition de nouveaux rôles métier, ou d'autres changements, **il existe un type de modification qui est souvent mal préparé : celui de la réutilisation, sous une autre forme, d'une application déjà existante, sans la dupliquer**.

Ces variantes peuvent prendre différentes formes : la mise en place d'une nouvelle langue, l'utilisation d'un contenu différent en fonction de la provenance de l'utilisateur,

[1] « Modernisation des SI et maturité des entreprises », observ. Sapientis, 2011 : http://www.sapientis.fr/conseil/wp-content/uploads/2011/04/extrait1budget.pdf

l'utilisation d'interfaces différentes par rapport au public cible ou encore la mise à disposition de la plateforme à une marque commerciale.

La plupart des secteurs peuvent nécessiter de tels changements. Sur le marché existant, on peut citer notamment les plateformes de blogs, les comparateurs de prix, les forums, et bien entendu les sites de e-commerce. Chaque secteur a développé, en fonction de ses spécificités, ses propres méthodes d'adaptation de son offre. Ce qui fait la valeur de celle-ci est généralement conservé, tandis que ce qui gravite autour est adapté. Certains secteurs privilégieront ainsi le contenant plutôt que le contenu, ou vice-versa.

Dans le cadre de ce travail, c'est le secteur des jeux web qui sert de référence.

1.3 Générateur

Un générateur est associé à une notion de création, de production de résultat, comme le suggère cette définition :

> *« Générer : verbe transitif. Avoir pour conséquence, engendrer, produire. »*
>
> *(Le Petit Larousse Illustré 2001)*

La génération d'une application peut se faire de plusieurs façons : soit le résultat attendu est réalisé une fois, en créant tous les fichiers nécessaires à son obtention ; soit le résultat est recréé lors de chaque utilisation. La deuxième solution sera utilisée, afin de percevoir plus rapidement les différences entre les produits générés.

1.4 Marque blanche

Plus l'adaptation est poussée, plus le contenu/le support est personnalisable. A partir d'un certain niveau d'adaptation, il est possible de proposer à l'utilisateur une application pouvant être perçue comme différente d'une autre de ses variantes. Si celles-ci se distinguent par leur marque, on parle alors d'un outil en marque blanche.

Dans ce travail sont analysées, en les adaptant aux besoins du secteur du jeu par navigateur, les différentes méthodes permettant ce type de personnalisation.

1.5 Constat personnel

1.5.1 Historique

Ce travail de Bachelor se base sur un constat personnel, tiré d'une aventure qui a connu son petit succès. En décembre 2006 a commencé le développement - de

manière non structurée - d'un jeu par navigateur, « Sans Sursis »[2], qui a rapidement attiré plusieurs milliers de joueurs, dès son ouverture en mars 2007.

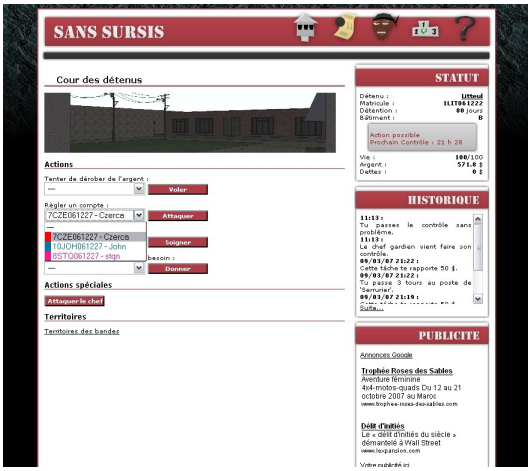

Figure 1: Sans Sursis : version 1

Figure 2: Sans Sursis : version 2

Après un an d'utilisation, il a commencé à s'essouffler pour plusieurs raisons : d'un côté, le jeu était relativement mal développé techniquement, la base de données était saturée et les pages mettaient plusieurs secondes pour se générer. De l'autre, il perdait de son attrait, car il était difficile d'apporter du nouveau contenu, par exemple dans les lieux à visiter, les objets à posséder et les actions à effectuer. C'est sur ce constat qu'a commencé le développement de la deuxième version de ce jeu, la première fermant en avril 2008, avec plus de 11'000 inscrits.

Après plusieurs mois de développement et d'essais en mode test, la version 2 a ouvert ses portes début août 2008. Cette version était dotée d'une plus grande flexibilité en termes de maniabilité de contenu :

- De nouveaux lieux sont faciles à ajouter sur la carte ;
- Les objets peuvent être ajoutés aisément et peuvent avoir des effets relativement variés ;
- Les actions sont, quant à elles, à développer au cas par cas. Elles varient en fonction des lieux, grâce à un système de modules que l'on ajoute et enlève, en dur, en fonction de l'emplacement du joueur.

Après environ de deux ans de combats acharnés et de luttes pour les premières places, la situation sur le jeu est problématique : les premiers du classement se reposent sur leur lauriers, sont de plus en plus forts et, par conséquent, presque inatteignables. Les joueurs commencent donc à se lasser. A ce stade, il est choisi d'autoriser la remise à zéro des comptes, permettant de recommencer une partie dès

[2] Sans Sursis : http://www.sans-sursis.com

le début, tout en conservant sa gloire passée, via l'historique de partie. C'est mis en place au mois de décembre 2010. A partir de cette date, on peut constater un nouveau souffle sur le jeu.

Figure 3: Évolution de la vente de biens virtuels

(Allopass)

Cependant, après quelques mois dans ce nouvel élan, la conjoncture est telle qu'une mise en place de parties parallèles est alors tentée. Après plusieurs semaines de développement, il devient évident que la mise en place d'un tel système est problématique : il est presque impossible de faire en sorte de permettre une "étanchéité" entre lesdites parties. En effet, un joueur d'une partie A pourrait vendre un objet à une partie B dans le cas où une vérification serait omise par le développeur. Le site a pris une telle ampleur qu'il est trop hasardeux de risquer une telle omission. Cette possibilité est donc écartée et une seule partie reste en en place.

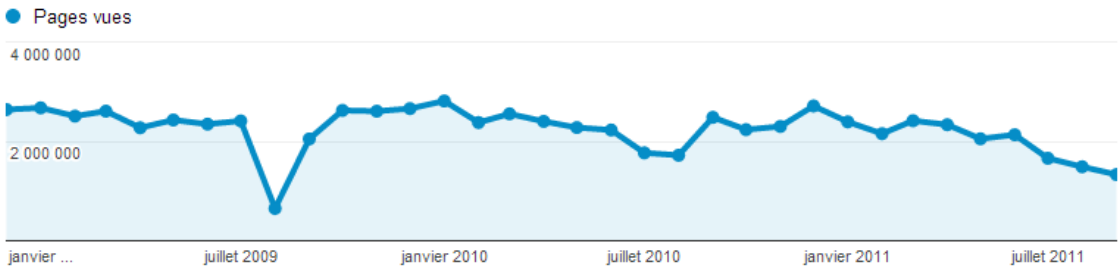

Figure 4: Pages vues mensuelles

(Google Analytics)

Fin 2011, ayant vraiment perdu de sa vigueur, Sans Sursis ferme ses portes. Cette deuxième version a connu, sur un peu plus de 3 ans, plus de 1'000'000 visites et près de 100'000'000 de pages vues, pour un taux de rebond de 12% (source : Google Analytics). A sa fermeture, elle compte plus de 30'000 membres.

Les joueurs sont toujours prêts à jouer et certains vont régulièrement aux nouvelles via les réseaux sociaux. La troisième version leur est promise depuis plusieurs mois et les premiers essais ne se montrent pas satisfaisants.

A la mi-février 2013, une version réinitialisée du jeu est remise en ligne pour occuper un petit groupe de joueurs et permettre de pouvoir montrer le jeu à quelques personnes curieuses. Dans les jours qui suivent, on peut observer jusqu'à une centaine de joueurs connectés simultanément, uniquement via le bouche à oreille et quelques publications sur Facebook et Twitter. Aucune information n'a été envoyée aux dizaines de milliers de membres. Après un mois en ligne (11 février-10 mars), c'est 3.5 millions de pages qui ont été visitées[3].

1.5.2 Problématiques rencontrées

Durant toutes ces années, plusieurs évolutions ont été envisagées voir même tentées, et ont été sources de problèmes :

1.5.2.1 Version 1 : Ajouter de nouveaux biens

Les objets créés sont notés en dur dans le code. Il y a moins d'une dizaine d'artefacts à obtenir et ils ont une utilité bien spécifique (entrer dans la cour, servir d'arme, etc.). Il n'est pas possible d'ajouter facilement des objets sans toucher au code.

Dans la version 2, il est possible d'ajouter des biens en les entrant dans la base de données. Pour chacun d'eux, l'administrateur peut lister les caractéristiques du joueur qui sont impactées lors de son utilisation ainsi que les objets qui peuvent être obtenus (par exemple en ouvrant un paquet surprise, en démontant une paire de ciseaux, etc).

Certains objets peuvent aussi être manipulables en dur, par exemple : la carte de travailleur débloque certains travaux au sein du pénitencier. Les missions intègrent également les objets, certains étant requis pour avoir la possibilité d'effectuer une action.

1.5.2.2 Version 1 : Création de nouveaux lieux

Chaque lieu est représenté par une page et une carte doit être mise à jour manuellement pour permettre l'accès aux différents emplacements de la prison. Il n'y a pas de déplacement à faire, un clic suffit pour se mouvoir n'importe où ailleurs.

[3] Source : Google Analytics

La version 2 voit la mise sur pied de plusieurs lieux, disponibles sur une carte en deux dimensions générée par le jeu. Il est donc possible de mettre certains lieux semblables dans toute la prison, et ce depuis l'administration. Chaque lieu est par contre l'inclusion simple d'un autre module. Il y a également, pour chaque type de lieu, la possibilité d'intégrer certains autres modules « spéciaux » comme les interactions avec les joueurs présents à cet endroit, qui sont toujours les mêmes. Le joueur peut se déplacer dans la prison comme il le souhaite, avec certains lieux interdits en fonction de son niveau. Suite à une évasion, le joueur est transféré dans un autre pénitencier.

Plus tard, des souterrains voient le jour, apportant une dimension supplémentaire au jeu. De nouvelles fonctionnalités sont alors ajoutées dans cette nouvelle dimension.

1.5.2.3 Version 1 : Nouvelles actions / Missions

Dans la première version, les actions sont décrites sur chaque page. Ainsi, une action n'est disponible que dans un seul lieu. Chaque action a également son propre fonctionnement : certaines permettent des interactions entre les joueurs, d'autres font des tirages au sort, d'autres servent à obtenir de l'argent, etc.

La deuxième version, quant à elle, contient des missions en plus des actions que le joueur peut effectuer dans les différents lieux, comme indiqué précédemment. Celles-ci sont un enchaînement de pré-requis et d'objets débloqués qui permettent de vivre des petites aventures au milieu de l'aventure principale.

Ces actions sont très limitées : elle ne peuvent être faites qu'une fois, et lorsqu'une nouvelle étape est franchie, les précédentes ne sont plus accessibles. Il s'agit d'un aller simple vers un objectif. Les missions pourraient par exemple, dans une meilleure version du jeu, être effectuables plusieurs fois.

1.5.2.4 Version 1 et 2 : Parties parallèles

Comme expliqué précédemment, il avait été souhaité de mettre en place des parties parallèles pour permettre de lancer de nouveaux défis aux joueurs blasés d'un classement qui n'évolue plus. Cette solution a été tentée mais abandonnée après avoir été forcé de constater que le code n'était pas adapté à cette solution.

1.5.2.5 Version 1 et 2 : Variantes du jeu

Lors d'une réunion avec plusieurs administrateurs du jeu, il avait été envisagé de créer des variantes de Sans-Sursis, permettant de proposer aux joueurs des alternatives

plus plaisantes, qui correspondrait plus à d'autres modes de jeu. Parmi les idées mentionnées, il y avait notamment la création d'une version où les règles anti-triche étaient plus souples, d'une version sans possibilité d'acheter des bonus, et d'une version où le rythme de jeu serait plus lent, pour les joueurs occasionnels. Le contenu du jeu, lui, ne changerait pas.

1.5.2.6 Version 1 et 2 : Multilingue

Les membres proviennent principalement des pays francophones avec, sans surprise, la France en premier :

- France : 72,72 %
- Tunisie : 6,66 %
- Canada : 6,40 %
- Belgique : 6,16 %
- Suisse: 2,37 %
- Algérie : 1,16 %
- Luxembourg : 0,92 %
- Réunion : 0,90%
- États-Unis : 0,86 %
- Mali : 0,29 %

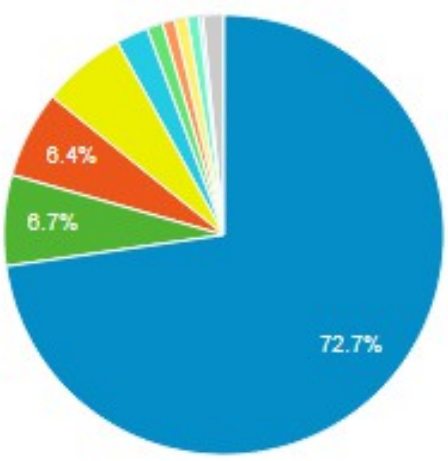

Figure 5: Visites : répartition des pays (1/12/2008 – 1/10/2011)

(Google Analytics)

Dans les pays anglophones existent d'autres jeux du même type (Exemple : Prison Block[4] par navigateur ou Prison Tycoon[5] sur PC). L'univers assez original de Sans-Sursis peut être un sérieux concurrent à proposer aux amateurs de ce genre de jeux.

Il pourrait être judicieux de penser le jeu pour le rendre multilingue afin de pouvoir rencontrer un certain succès hors des pays francophones. Attention toutefois à prendre en compte le fait que lorsque des messages sont envoyés entre joueurs, il n'est pas possible de traduire les messages. Il faut donc adapter le jeu aux langages multiples, en proposant par exemple une partie par langue ou en faisant que toute personne du même camp linguistique devienne alliée avec ses semblables.

[4] Prison Block : http://www.prisonblock.com
[5] Prison Tycoon : http://www.jeuxvideo.com/jeux/pc/00015029-prison-tycoon.htm

1.5.2.7 Autre : Autres jeux du même type

Depuis fin 2010, la possibilité de réaliser un jeu d'aventure se déroulant dans un monde post-apocalyptique à sortir le 21 décembre 2012 était envisagée. Cependant le manque de temps pour développer un nouveau jeu a empêché la réalisation de ce projet, alors que dans sa logique, il était très proche de ce qui existait déjà (des lieux, des objets, des actions, des interactions entre joueurs).

Lors de la réalisation d'un jeu, il est important de pouvoir se concentrer sur ce qui fait sa valeur : son scénario, ses illustrations, son design, ses objets, ses actions, ses lieux, sa philosophie, son support, etc. Trop de temps est consacré au développement de fonctionnalités qui ne peuvent pas être perçues par l'utilisateur, notamment le back-office.

1.5.2.8 Autre : Marque blanche

Plusieurs fois, des opportunités se sont présentées pour développer des jeux au nom d'autres personnes ou de sociétés. Voici donc un aspect à garder en tête : pouvoir permettre à d'autres personnes d'administrer leurs jeux sans impacter les autres projets existant d'autres utilisateurs.

1.5.3 Synthèse des problématiques

Ces problématiques se situent au niveau des possibilités d'évolution d'un jeu, ainsi que dans les variantes que l'on peut offrir aux joueurs. Un outil mieux conçu et un peu plus élaboré permettrait de se concentrer sur la partie métier, qui fait bien évidemment le succès d'un bon jeu.

2. Analyse de l'existant

2.1 Niveaux de modifiabilité

Voici une liste d'exemples des différents niveaux de modifiabilité souhaités, et dont les équivalents vont être cherchés dans les services existants :

- Éditeurs de jeux différents : client 1, client 2
- Jeux différents : jeu en prison, jeu post-apocalyptique
- Paramétrages différents : pour joueurs occasionnels, joueurs assidus
- Parties parallèles : partie lancée à l'ouverture, l'autre 6 mois après
- Multilingue : les joueurs francophones jouent avec les anglophones

Un exemple est disponible au chapitre 3.2.1 (Hiérarchisation des variantes), à la page 33.

Ensuite, dans le moteur même du jeu, certains points devraient être configurables à différents niveaux (en fonction du jeu, des variantes, des parties), comme par exemple :

- Caractéristiques d'un joueur
- Actions et missions
- Lieux et carte
- Objets et armes

2.2 Générateurs existants

Il existe différents types de sites qui proposent du contenu réutilisable et qui permettent à leurs utilisateurs d'avoir leur contenu personnalisé, voire en marque blanche. Pour chacune des rubriques trouvées, il a été choisi d'analyser plusieurs sites et d'observer quels niveaux de personnalisation ils proposent.

Il est important de noter que la plupart des sites présentés ici sont francophones. Il existe probablement des services semblables dans d'autres langues.

2.2.1 Blogs

Sites analysés : WordPress[6] (la plateforme de blogs, pas les versions téléchargeables), Skyrock Blog[7] (anciennement Skyblog) et Blogger[8] (service Google).

[6] WordPress : http://www.wordpress.com
[7] Skyrock Blog : http://www.skyrock.com/blog
[8] Blogger : http://www.blogger.com

Il va de soi que les blogs se différencient par leur contenu. Il est donc aisé de comprendre que ce n'est, à priori, pas d'usage de proposer du contenu similaire mais que la force de réplicabilité se situe dans autre chose que les textes publiés, et donc dans le contenant plutôt que dans le contenu.

Tous les blogs sont fortement personnalisables au niveau du thème et la plupart proposent un catalogue offrant différents designs sélectionnables, gratuitement ou contre paiement.

Les services de WordPress et de Blogger permettent à un utilisateur d'avoir plusieurs blogs et de les gérer indépendamment. Ce n'est pas le cas de Skyrock, qui associe clairement un utilisateur à son blog. Il s'agit là de renforcer l'aspect personnel des publications, puisque tout est mis au nom de son propriétaire. On n'y retrouve donc pas de blogs de sociétés, ou de médias à part entière.

WordPress et Blogger permettent également de choisir la langue et de modifier les textes. Il en est de même pour l'interface d'administrations, qui peut avoir une langue différente du site public. Chaque blog est en revanche doté d'une seule langue. Il est toutefois possible d'utiliser un plugin WordPress, WPML[9], moyennant paiement, pour proposer un contenu multilingue.

Concernant l'édition par plusieurs personnes, ces deux sites proposent d'ajouter des auteurs (Blogger) ou de donner plus de droits à un membre (WordPress). Il est donc possible d'avoir une équipe de rédaction derrière chaque blog.

Un compte Skyrock, Blogger ou WordPress permet d'être identifié sous une seule facette pour ajouter des commentaires sur d'autres blogs du même réseau. Il existe même un « Skyrock Connect »[10] pour utiliser cette identité sur d'autres sites que ceux du réseau Skyrock.

Enfin, il existe un service externe, blogEdit, qui propose de rédiger un même contenu pour plusieurs blogs WordPress distincts[11].

[9] WPML : http://wpml.org
[10] Skyrock Developer : http://www.skyrock.com/developer
[11] BlogEdit : http://www.blog-edit.com

2.2.2 Forums

Deux ténors ont été analysés pour ce qui est de l'observation des forums : Forumactif[12] et Xooit[13]. Ils offrent le même panel de fonctionnalités.

Premièrement, on peut constater que le contenu, à l'instar des blogs, est ce qui fait la plus-value d'un espace de discussions et doit donc être mis en place par les utilisateurs. Les rubriques ne sont donc pas préfabriquées, exception faite des exemples mis en place par défaut.

La personnalisation s'offre aux propriétaires de forums de plusieurs façons, à commencer par le thème utilisé et la personnalisation du design. Cette situation s'apparente très fortement aux blogs.

Ce qui fait la spécificité d'un forum est son aspect communautaire et qu'il contient plusieurs canaux de discussion, contrairement aux blogs qui sont généralement réalisés autour d'un seul flux d'informations, et qu'une seule personne peut animer de A à Z.

Chaque communauté, au même titre que chaque groupe dans la vie courante, comporte généralement une hiérarchie clairement établie. Les forums n'échappent pas à la règle en intégrant la mise en place de différents rôles, qui sont personnalisables de façon illimitée. Ces rôles régissent les accès aux discussions, ainsi qu'aux outils d'administration.

Étant donné que plusieurs discussions doivent être possibles au sein d'une même communauté, la catégorisation est un élément clé de l'organisation. Il est possible de restreindre l'accès à certaines discussions à certains groupes (seulement pour les administrateurs, seulement pour les graphistes, seulement pour les membres,...), et d'avoir des contenus communs à tous.

En revanche il n'y a aucun lien entre les différents forums, ce qui leur confère une indépendance les uns par rapport aux autres. Il n'y a pas non plus de fonctionnalité « Facebook Connect »[14] ou « Twitter Connect »[15] permettant d'avoir un compte commun.

[12] Forumactif : http://ww.forumactif.com
[13] Xooit : http://www.xooit.com/fr/
[14] Facebook Connect : https://www.facebook.com/about/login/
[15] Twitter Auth : https://dev.twitter.com/docs/auth/sign-twitter

2.2.3 Groupes de musique

Deux types de services ont été analysés pour cette catégorie, et proposent des contenus très différents. Le premier d'entre eux est MX3.ch[16]. Il peut s'apparenter à MySpace[17] ou d'autres sites préfabriqués répondant à un besoin spécifique, celui des musiciens.

Sur MX3, une page de groupe[18] intègre notamment la biographie du groupe, ses morceaux, sa tournée, ou encore les commentaires de ses fans. Le reste du contenu est relativement similaire aux autres sites générés sur cette plateforme.

Sur MySpace, ces fonctionnalités se retrouvent également, mais la possibilité de personnaliser certains éléments (images, fond et couleurs) est cette fois-ci laissée à l'administrateur.

Une personne ayant un compte sur l'un de ces deux réseaux peut/doit se servir de celui-ci en servir pour s'identifier et poster du contenu sur d'autres sites de la même plateforme.

Le deuxième type de service proposé aux groupes de musique est celui de l'intégration dans une autre plateforme. Dans le cas étudié, il s'agit de tracks.by[19], qui s'intègre dans Facebook[20].

Un onglet est créé sur la page fan de l'artiste, par exemple celle de Lil Wayne[21], qui compte plus de 40 millions de fans[22], et incite à la communication en intégrant du contenu disponible sur d'autres plateformes et en encourageant les fans à partager le contenu qu'ils aiment.

La personnalisation est faible mais s'intègre comme il se doit fans la charte graphique de Facebook. L'adaptation est uniquement dans le contenu.

2.2.4 Shops en ligne

Les shops en ligne peuvent être trouvés sous deux formes :

16 MX3 : http://www.mx3.ch
17 MySpace : http://www.myspace.com
18 Page de Groupe, The Postmen : http://mx3.ch/artist/thepostmen
19 Tracks.by : http://tracks.by
20 Facebook : http://www.facebook.com
21 Page Facebook de Lil Wayne : https://www.facebook.com/LilWayne/app_225931450787822
22 En date du 8 juin 2013

D'un côté il y a les services servant de support pour la vente de ses propres produits sur un site web, comme ShopApplication[23] ou ShopMachine[24], ou sur d'autres plateformes, comme l'onglet Facebook "ShopTab"[25].

De l'autre côté, il y a les sites proposant déjà un contenu. Ce système est mis en place pour du contenu difficile à gérer ou à obtenir, comme par exemple du contenu pour adulte sur Busyx[26], Dreamstore[27] ou Affiliation Lingerie[28]. Une autre raison de l'existence de boutiques avec contenu est l'uniformité d'une offre, où de plus petits organismes (tels que des artistes, des musiciens, des associations) font office de revendeurs : c'est le cas de Spreadshirt[29], qui se charge de l'impression de t-shirts personnalisés et de toute la logistique d'envoi, de la facturation ou du service client.

Cette catégorie regroupant de nombreuses variations, notamment en termes de réutilisation, de personnalisation, de paramétrage, de mise en place de thèmes, d'adaptation du contenu, il en sera fait une analyse précise dans la suite de ce travail.

2.2.5 Revente d'hébergements

Certains hébergeurs proposent de revendre leurs offres. C'est notamment le cas de Oxito[30], de OVH[31] et d'E-Clicking[32]. Ce ne sont cependant que des affiliations et la marque blanche intervient hors du cadre de l'hébergement : chacun est libre de revendre les offres de la manière dont il le souhaite.

En revanche il est plus intéressant de lire l'offre proposée par un outil permettant la revente, Parallels[33]. Celui-ci est utilisé par les hébergeurs (notamment OVH et KreativMedia[34]) pour gérer leur clientèle et proposer un service de revente à d'autres hébergeurs, qui utiliseraient l'infrastructure sans utiliser le nom.

Chaque titulaire d'un compte chez un hébergeur a, à son tour, un panneau de contrôle en ligne qui lui permet de gérer son compte, que ce soit sur un serveur mutualisé ou

23 Shop Application : http://www.shop-application.com
24 ShopMachine : http://www.shopmachine.com
25 ShopTab : http://support.shoptab.net/home
26 Busyx : http://www.busyx.com
27 Dreamstore : http://www.dreamstore.ch/affiliate.htm
28 Affiliation Lingerie : http://www.affiliation-lingerie.ch
29 Spreadshirt : http://www.spreadshirt.com
30 Oxito : http://www.oxito.ch/revendeurs
31 OVH : http://ovh.biz/fr/revendeurs
32 E-Clicking : http://fr.e-clicking.com/revendeur/
33 Parallels : http://parallels.com/fr/spp/resellerhosting/
34 Kreativmedia : http://kreativmedia.ch

dédié. Dans le ce deuxième cas, il est même possible pour une personne possédant un serveur de créer des comptes d'administration pour d'autres hébergements.

Ce panneau de contrôle permet notamment également de gérer le renouvellement, la facturation, la gestion des droits, des quotas d'utilisation des ressources serveur.

2.2.6 Contenu en marque blanche

Le contenu en marque blanche est généralement l'intégration d'un module spécifique, qui se place de façon discrète. Il est mis en place sur un site web déjà existant, contrairement aux sites complètement en marque blanche (cette catégorie sera passée en revue ultérieurement).

Avec les services proposé en marque blanche, la valeur est dans le contenu proposé à la clientèle. En règle générale, il s'agit d'un produit que le revendeur peine à proposer lui-même, que ce soit pour des raisons financières ou pour des raisons de connaissances.

2.2.7 Comparateurs de prix

La catégorie des comparateurs est un excellent exemple de ce qui ne peut pas être réalisé sans de bonnes connaissances techniques. En effet, il est nécessaire de pouvoir récolter des données, puis de les traiter pour les proposer au public. La rémunération se fait généralement par une affiliation avec les sites sur lesquels les visiteurs sont envoyés.

Parmi les sites proposant une marque blanche dans les comparateurs, on peut nommer AdFever[35] et vShop[36] qui proposent tous deux des comparatifs de produits. Certains comparateurs sont réalisés pour des secteurs spécifiques, comme par exemple Illico Travel[37], qui cible le secteur du voyage, avec des comparateurs de vols, d'hôtels, de location de voitures, et autres.

2.2.8 Contenu mobile

Le secteur du mobile est en pleine mue depuis l'avènement des smartphones et des répertoires d'applications proposés par les fabricants (avec en tête App Store et Google Play[38]). Toutefois, il existe encore des sociétés comme MagikBiz[39], qui

35 AdFever : http://www.adfever.com/marque-blanche-comparateur-de-prix.html
36 Vshop : http://www.vshop.fr/whitelabel
37 Illico Travel : http://www.illicotravel.com/info/affiliation.xhtml
38 Source : Comstore (cf bibliographie)
39 MagikBiz : http://www.magikbiz.com

proposent un service unique, mais qui permettent des reversements à chaque affilié. Il est difficile ici de parler de personnalisation.

Il existe également des services d'envoi de SMS via une API, mais c'est un simple service proposé et non pas un contenu proposé à l'utilisateur. Parmi les sites proposant cette offre, on peut nommer « sms envoi »[40].

Dans les applications de service auditel, on peut retrouver des affiliations de voyance, avec Francobiz[41], Gora Cash[42], ou le programme d'affiliation de voyance.fr[43] Il s'agit principalement, là encore, d'affiliation et l'intégration se fait uniquement via l'insertion d'un formulaire affichant les instructions d'appel à intégrer sur un site déjà existant.

2.2.9 Agence de voyage

Les voyages semblent également bénéficier d'une forte tendance à la mise en place du contenu en marque blanche, et pas uniquement via des systèmes de comparateurs comme mentionné précédemment.

Une petite société, par exemple un magasin de matériel de voyages, pourra se tourner vers un service comme « Voyages en Direct »[44], qui proposera une solution complète dont seul le logo, quelques textes et les couleurs sont éditables. Il s'agit toujours du même contenu.

A l'inverse, il existe des modules à intégrer, mais proposent, eux aussi, le même contenu. C'est notamment le cas de Expedia[45] et Traveldoo[46], qui s'adressent principalement aux agences de voyages ou autres prestataires souhaitant un outil puissant et fortement intégrable.

2.2.10 Sites web en marque blanche

Cette vaste catégorie de sites mise non seulement sur la mise en forme d'un contenu préfabriqué, car celui-ci est difficilement réalisable par les personnes souhaitant le proposer, mais également sur la mise en forme et la personnalisation de l'apparence.

2.2.10.1 Annonces

40 Sms envoi : http://www.smsenvoi.com/api-sms/http/envoi-sms-marque-blanche/
41 Francobiz : http://www.francobiz.com/index.aspx
42 Gora Cash : http://www.goracash.com
43 Voyance.fr : http://www.affiliation.voyance.fr/services-affiliation-voyance.html
44 Voyages en Direct : http://www.voyages-en-direct.com/affiliation.cfm
45 Expedia : http://www.expedia.fr/daily/privatelabel/default.aspx
46 Traveldoo : http://www.traveldoo.com/fr/home.html

Le secteur des annonces a fait ses preuves avant même l'avènement de l'informatique. Le déploiement d'une publication pouvait déjà se faire sur papier : si une compagnie possédait plusieurs journaux, elle pouvait mettre les mêmes annonces dans chacun de ses journaux et ainsi augmenter la visibilité des annonces.

Aujourd'hui, on n'échappe pas à une augmentation de la force de frappe avec des services de mutualisation du contenu et un système d'affiliation permettant de reverser une partie des gains générés par la vente d'annonces, comme par exemple sur Club Annonces[47] ou Euroaffiliate[48], qui proposent la personnalisation d'un site d'annonces complet, en adaptant le thème et en gérant tout ce qui touche à l'administratif. « Le Cube Affiliation »[49] propose quant à lui des annonces dans l'immobilier via un site clé en main et propose une traduction dans plusieurs langues.

Certains services se spécialisent et s'adaptent aux systèmes de communications récents, proposant des applications mobiles. C'est le cas de net1avenue[50], qui propose exclusivement ce contenu.

2.2.10.2 Sites pour adulte

Pour certains sites pour adulte, il est difficile de déterminer la puissance du paramétrage ou le niveau de personnalisation. C'est le cas de secteurs où la plupart des sites se ressemblent, et où un seul éditeur se retrouve à la tête de plusieurs sites web concurrents en apparence. On peut notamment citer les sites de casino, où il est nécessaire d'avoir une licence d'exploitation.

Dans ce secteur, plusieurs prestataires vantent les mérites de leur technologie. C'est notamment le cas de Microgaming[51], de RealTime Gaming[52] ou du groupe B3W[53]. Aucun d'eux ne propose de marque blanche, vraisemblablement pour des raisons légales, et de responsabilités.

En revanche, d'autres types de sites pour adulte, également en concurrence entre eux, se servent de la marque blanche pour proposer le même contenu et ainsi proposer des données plus complètes à leurs utilisateurs, c'est le cas des sites de rencontre, comme

47 Club Annonces : http://webmasters.clubannonces.com
48 Euroaffiliate : http://www.euroaffiliate.com
49 Le Cube Affiliation : http://www.lecube-affiliation.com/detail-offre.html
50 Net1avenue : http://www.net1avenue.com
51 Microgaming : http://www.microgaming.co.uk/technology.aspx
52 RealTime Gaming : http://www.realtimegaming.com
53 B3W : http://www.b3wgroup.com

EasyFlirt[54] ou Dating Factory[55]. D'autres, comme Célibataire.ch[56] proposent simplement une affiliation.

Enfin, on retrouve également la mise à disposition de contenu pour adulte, pour lequel on devine aisément que toute personne souhaitant proposer une telle offre n'a pas nécessairement les ressources humaines permettant l'élaboration d'un tel projet. Parmi les sites observés, à savoir XCams[57], xponsor[58] et Carpe Diem[59], il existe des offres allant de l'intégration de publicités sur son site à la création d'un site personnalisé, en passant par la mise à disposition de contenu en marque blanche.

2.3 Le monde des jeux web

2.3.1 Types de jeux

Dans les jeux web, on distingue plusieurs catégories :

Figure 6: Jeu social alliant flash et persistance des données : Farmville

(Farmville[60])

D'un côté, se trouvent les jeux côté client, tels que les jeux flash. On y retrouve des applications permettant de se divertir sur des parties de courte durée, en misant généralement sur l'habileté de l'utilisateur. On peut trouver un certain nombre de jeux

54 EasyFirt : http://www.easyflirt-partners.biz/u_oursites.php
55 Dating Factory : http://www.datingfactoryfrance.com
56 Célibataire.ch : http://www.celibataire.ch/affiliation.php
57 XCams : http://www.xcams-partners.com/fr/home/promotools.php
58 Xponsor : http://v4.xponsor.com
59 Carpe Diem : http://fr.carpediem.fr
60 Farmville : http://farmville.com/

de ce type sur MiniClip[61], sur T45,[62] ou encore sur des plateformes mises en place par d'autres sites, comme le GameZone de "2001Jeux"[63] Ils sont généralement développé en Flash, en JavaScript ou en HTML 5. Cette catégorie de jeux ne sera pas analysée, car lorsque le contenu est réutilisé, il est simplement répliqué ou affiché sur plusieurs sites en n'étant hébergé que sur un seul site.

D'un autre côté, il existe les jeux côté serveur. Ils constituent ce que l'on appelle jeux par navigateur. Ces jeux misent plutôt sur la réflexion, la gestion ou la stratégie, et permettent de mettre en place un scénario sur le long terme. Cette catégorie sera observée plus en détail, car c'est ce type de jeu qui sera développé par la suite.

Enfin, entre deux, se trouvent les applications telles que celles que l'on peut retrouver sur Facebook, que l'on nomme "Jeux sociaux", ou "Social Games" dans la langue de Shakespeare. Il s'agit généralement de jeux mêlant une partie principalement côté client, où le joueur a la possibilité d'utiliser un bon nombre d'options pour interagir avec son environnement, avec une aventure côté serveur, assurant la persistance des données, et encourageant le joueur à revenir continuer ce qu'il a commencé, et à se mesurer à ses amis, voire à collaborer avec eux. C'est le cas, par exemple, de FarmVille, ou encore de l'adaptation des Sims "The Sims Social"[64]. Cette catégorie de jeux ne sera pas retenue, car la valeur de ce travail de bachelor se situe dans la partie serveur des social games, et se retrouve donc dans les jeux par navigateur.

2.3.2 Jeux par navigateur

Les jeux par navigateur sont très variés, car ils s'adressent à des publics très différents. Il n'y a qu'à observer les thèmes présents sur les sites spécialisés, comme par exemple BrowserGames[65], où les jeux vont de la gestion de ferme et de zoo aux batailles moyenâgeuses et spatiales, en passant par les jeux de rôles où le joueur incarne un cowboy, un pirate, ou encore les jeux de management d'équipe sportive, d'armée contemporaine ou de compagnie aérienne, allant jusqu'à l'élevage de chevaux et aux combats de catch.

Il est intéressant d'observer quels jeux sont semblables pour un même éditeur , s'ils proposent plusieurs parties, s'ils sont disponibles en plusieurs langues ou s'il existe des variantes d'un même jeu.

61 Miniclip : http://miniclip.com/games/fr/
62 T45 : http://t45ol.com
63 GameZone : http://gamezone.2001jeux.com
64 The Sims Social : http://facebook.com/TheSimsSocial
65 BrowserGames : http://www.browsergames.fr

2.3.2.1 Éditeur Gamovation

L'éditeur Gamovation[66] ne donne que très peu d'informations. Il est difficile de déterminer quels jeux il a réalisé, en revanche une recherche dans l'autre sens a été possible : c'est en cherchant l'éditeur des jeux « MaMafia »[67] que l'on peut tomber sur Gamovation.

Les jeux de la gamme « MaMafia » sont particuliers : ils permettent à tout particulier qui le souhaite de créer son propre jeu en ligne et de l'administrer. Les revenus sont partagés entre l'éditeur et l'administrateur du site. Ces jeux sont en partie personnalisables, l'administrateur pouvant sélectionner les thèmes dans une liste prédéfinie, et en adapter certains éléments.

Figure 7: Statistiques des jeux Gamovation

(MaMafia.fr, 14 avril 2013)

En français il existe la version « MaMafia », qui se décline également en anglais avec « MafiaCreator »[68] et d'autres versions comme le hollandais avec « MijnMaffia »[69]. 10 langues sont annoncées par l'éditeur et totalisent plus de 500'000 de jeux (Figure 7: Statistiques des jeux Gamovation). Chaque langue a son propre site, mais les jeux sont exactement identiques. On peut constater que lorsqu'une news n'est pas traduite

66 Gamovation : http://www.gamovation.com
67 MaMafia : http://www.mamafia.fr/
68 MafiaCreator : http://www.mafiacreator.com/
69 MijnMaffia : http://www.mijnmaffia.nl/

sur la version turque, elle apparaît en anglais. La “maison mère” se charge donc d’alimenter tous les sites et ils n’agissent donc pas de manière indépendante.

Chaque jeu est créé dans une langue spécifique. Il n’y a donc que deux niveaux de paramétrage : la langue de la plateforme sur laquelle on s’inscrit, et le paramétrage fait par l’administrateur sur son jeu.

2.3.2.2 Éditeur Travian Games

L'éditeur Travian Games[70] propose les jeux de plusieurs studios de création. En effet, ils mettent en avant les développeurs de chaque jeu :

Tableau 1 : Jeux Travian Games

Nom	URL	Développeur	Langues
Battlemons	http://play.battlemons.com	Travian Games GmbH (DE)	1 (2)
Imperion	http://play.imperion.com	Travian Games GmbH (DE)	1
Remanum	http://play.remanum.com	Travian Games GmbH (DE)	4
Roger & Out	http://play.rogerandout.com	Travian Games GmbH (DE)	1
Travians	http://play.travians.com	Travian Games GmbH (DE)	40
Miramagia	http://play.miramagia.com	Bright Future GmbH (DE)	8
Rail Nation	http://play.rail-nation.com	Bright Future GmbH (DE)	1 (2)
Goalunited	http://play.goalunited.org	northworks GmbH (DE)	28
Rise of Europe	http://play.riseofeurope.com	Perfect World (CN)	1

(Travian Games, avril 2013)

Concernant les jeux développés par Travian Games GmbH, seuls Imperion et Remanum semblent se baser sur le même scénario. Dès le départ, l’utilisateur a le choix entre trois types de personnages. Le mode de jeu est relativement similaire : il s’agit de choisir différents lieux sur une carte, puis de naviguer sur le lieu en question. La thématique abordée est toutefois très différente (Figure 8 et Figure 9).

[70] Travian Games : http://www.traviangames.com/en/products.html

Figure 8: Le jeu Remanium

Figure 9: Le jeu Imperion

(Travian Games, avril 2013)

L'autre jeu de ce développeur qui peut être intéressant à analyser est Travian. Celui-ci est disponible en 40 langues selon l'éditeur, mais toutes sont similaires. La seule particularité est son adaptation pour les langues dont l'écriture se fait de droite à gauche (Figure 10 et Figure 11).

Figure 10: Travian dans sa version francophone

Figure 11: Travian dans sa version arabophone

(Travian, avril 2013)

En dehors de cette particularité, tout est identique. Il n'y a pas d'adaptation de contenu, et chaque langue possède les mêmes scénarios que les autres.

Une particularité des serveurs peut toutefois être relevée : chaque langue possède plusieurs "mondes parallèles", qui peuvent être ouverts en fonction de la demande et de la saturation des autres mondes. Chaque univers a également la particularité de pouvoir proposer quelques variantes. En l'occurrence, la possibilité d'un jeu trois fois plus rapide est proposée aux joueurs, tout en conservant les caractéristiques

habituelles (Figure 12). Cela permet aux utilisateurs d'avoir un mode de jeu plus poussé, sans dénaturer l'univers qui leur plaît.

Figure 12: Travian : parties différentes pour une même langue

(Travian, avril 2013)

2.3.2.3 Similaire à Travian Games

D'autres studios d'édition de jeux proposent des jeux différents, et leur logique de mise en place de nouveaux produits est relativement proche de Travian Games.

2.3.2.3.1 BigPoint

Le studio BigPoint[71] possède une trentaine de jeux et ne se limite pas aux jeux par navigateur traditionnels. En effet, il propose également des MMORPG. Les jeux sont relativement semblables à l'approche faite par Travian Games.

2.3.2.3.2 Innogames

De leur côté, les responsables de chez Innogames[72] ont également opté pour des jeux sur des thèmes différents, et proposent un seul jeu par thème.

2.3.2.3.3 Looki

L'éditeur Looki[73] a une vingtaine de jeux dans son catalogue, et les thèmes sont variés. Seuls les jeux "Empire Universe II" et "Empire Universe III" ont une approche complémentaire à ce qui a pu être observé pour Travian Games : un jeu en version 2 a

[71] BigPoint : http://fr.bigpoint.com/
[72] InnoGames : http://www.innogames.com
[73] Looki : http://www.looki.fr/jeux/

vu une suite arriver, sous la nomination de la version 3, et les deux versions se déroulent en parallèle. Il est probable que la deuxième version soit encore utilisée par un bon nombre de joueurs ne la délaissant pas pour la troisième mouture.

2.3.2.3.4 Gameforge

Le studio Gameforge[74] est basé sur un jeu créé en 2002[75], Ogame[76]. Il est intéressant de relever qu'un jeu datant d'il y a plus de dix ans possède le même mécanisme que les autres éditeurs.

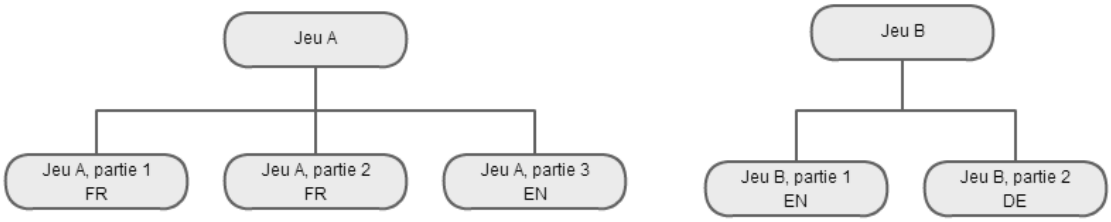

Figure 13: Déploiement des jeux par les éditeurs

Ces studios mettent tous en place, pour certains de leurs jeux, une logique de variante basée sur une nouvelle partie du même jeu, tout en gardant le même contenu. Seule la langue peut varier. Il est possible de représenter cette logique grâce à un diagramme (Figure 13).

2.3.2.4 Éditeur Feerik

Les jeux réalisés par Feerik[77] s'approchent également de la logique des éditeurs précédemment mentionnés.

En revanche, il est intéressant de s'attarder sur un de leurs jeux, Mafiarox[78] : celui-ci propose, pour une seule inscription, une succession de jeux à durée de vie limitée (Figure 14). En effet, le jeu est réinitialisé tous les 15 jours, redonnant à tous la possibilité de tenter d'accéder à la première place deux semaines plus tard.

Ce mode de fonctionnement est à double tranchant : si la lutte pour la première place est toujours de mise, et incite ainsi les joueurs à s'impliquer plus durant la durée limitée d'une partie, elle donne également la possibilité à chaque joueur d'arrêter le jeu tous les 15 jours. En effet, une personne étant proche de la première place aura plus

74 GameForge : http://fr.gameforge.com/home/index
75 Source : http://corporate.gameforge.com/en/company/history/ 9 juin 2013
76 Ogame : http://www.ogame.fr
77 Feerik : http://www.feerik.com
78 Mafiarox : http://www.mafiarox.com

facilement l'envie de continuer si la partie ne s'achève pas, contrairement à une personne perdant deux semaines d'efforts lors de la remise à zéro.

Dans le cas de Mafiarox, un intérêt financier est en jeu, car chaque vainqueur d'une partie remporte un prix.

Figure 14: Mafiarox : déroulement des parties dans le temps

2.3.2.5 Worlds Oriented Objects

Il aurait pu sembler exister un projet de générateur de jeux, sous l'appellation « Worlds Oriented Objects »[79]. Celui-ci peut toutefois paraître suspendu, les dernières actualités sur le site datant de 2009. Cependant, certaines pages annoncent une sortie pour « Juillet 2013 ».

Après quelques rapides vérifications, il s'avère qu'il s'agit d'un projet factice visant à démontrer l'utopie de la création d'un générateur permettant de créer des jeux variés.

L'offre alléchante n'est donc qu'une manipulation pour tenter de mettre en avant la difficulté à créer un générateur aussi complet, comme annoncé sur un forum[80] que gère le responsable de cette plaisanterie :

> *« Je vous rassure tout de suite, [Worlds Oriented Objects] n'est qu'un buzz... Néanmoins, pensez-vous que cela soit réalisable ?! »*
>
> *(Prélude, « créateur » de WOO, 5 Mars 2010)*

La problématique soulevée par World Oriented Objects a alimenté la discussion entre les membres de ce forum. L'un d'eux met le doigt sur une problématique retrouvée lors de la création d'un générateur de jeux :

> *« Tous les jeux de cartes ne sont pas identiques car les systèmes sont différents. [...] Un moteur de jeu te permettra de faire des variantes sur un SYSTEME donné, mais pas plus. [...] Ce n'est pas parce que tu JOUES à des jeux différents, dans un contexte différent, que les jeux SONT différents. [..] Un moteur de jeu ne permet pas l'originalité et uniformisera forcément les jeux »*
>
> *(Kalan, membre du forum, 22 mars 2012)*

[79] World Oriented Objects : http://woo.pbem.be/

[80] Discussion « Un logiciel pour créer vos jeux automatiquement » : http://forum.jeux-web.com/index.php?topic=283.0

Lors de la réalisation d'un prototype, il est donc nécessaire de fixer les limites des jeux que l'on souhaite pouvoir générer. La différence entre plusieurs jeux se fait lors de la création de l'ambiance dans laquelle évolueront les joueurs.

2.3.3 Synthèse jeux web

Les éditeurs de jeux web semblent miser sur des jeux les plus différents possibles. Parmi tous les sites observés, les thèmes sont souvent similaires entre les éditeurs : il est fréquent de retrouver par exemple des jeux historiques, comme des jeux de pirates, de cow-boys, de guerre, de chevaliers.

Cependant, chaque éditeur n'a généralement qu'un seul jeu par thème, et lorsque ce n'est pas le cas, les différences entre les jeux proches ont un mécanisme de jeu relativement différent. De ce fait, il est possible de se rendre compte qu'un éditeur ne décline pas un jeu à succès, par exemple en proposant une version avec des objectifs différents, se contentant de le répliquer par le biais de nouvelles parties d'un même jeu, parfois traduites, et pouvant avoir toutefois quelques nuances (vitesse de jeu par exemple).

2.4 E-commerce

2.4.1 Solutions analysées

Pour l'analyse approfondie des sites de e-commerce, les 5 CMS les plus répandus ont été sélectionnés. Pour obtenir ce top, la référence choisie a été Meanbee, via le Journal du Net[81]. Il s'agit de Magento (23% de part de marché en octobre 2012), Zen Cart (12,9%), VirtueMart (10,4%), osCommerce (8,2%) et PrestaShop (7,4%).

[81] Journal du Net, « E-commerce : Magento et PrestaShop gagnent en popularité » : http://www.journaldunet.com/solutions/saas-logiciel/parts-de-marche-logiciels-e-commerce-octobre-2012-1112.shtml

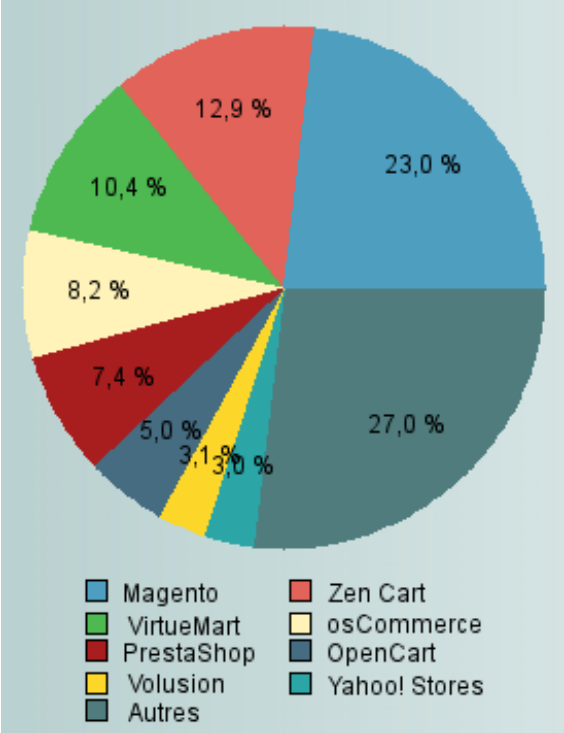

Figure 15: Parts de marché des solutions e-commerce

(Meanbee, octobre 2012)

Deux d'entre eux ont été écartés d'emblée. En premier, osCommerce est un "vieux" CMS[82]: Sa première version a été publiée en l'an 2000 . Cependant, ses développeurs ont continué son développement pendant plus de 10 ans. Sa troisième version (3.0) a vu le jour le 31 mars 2011, mais sa dernière mise à jour (3.0.2) date également de 2011, le 6 août. La deuxième version (2.3.3) a été mise à jour plus récemment, le 15 août 2012, mais se base sur du code publié le 2 décembre 2000, pour la version 2.0. Les développeurs préfèrent donc continuer de faire évoluer un système archaïque mais étant en place sur de nombreux sites plutôt que de s'attarder sur l'évolution de leur dernier système. osCommerce n'est donc pas analysé.

La deuxième solution de e-commerce écartée est VirtueMart. En effet, ce n'est pas un CMS à proprement parler, mais une extension pour un autre CMS, à savoir "Joomla!". Étudier la base de données d'un tel outil devient alors fastidieux, car les schémas de modélisation ne sont pas globaux, et font référence à d'autres éléments qui ne font pas partie de VirtueMart. Ce dernier n'est donc pas analysé.

[82] OsCommerce sur Wikipedia : http://en.wikipedia.org/wiki/OsCommerce

2.4.2 Analyse des sites de e-Commerce

Seront donc analysées ici les structures des solutions suivantes : Magento, Zen Cart et PrestaShop. Les observations seront faites au niveau de la structure de la liste des prix, l'hypothèse de départ étant la suivante : un produit peut voir son prix varier en fonction de plusieurs paramètres : la boutique, la provenance de l'acheteur, la devise, l'acheteur (clientèle commerciale ou privée), une réduction commerciale, etc. Il s'agit ensuite d'évaluer s'il est facile d'ajouter une nouvelle réduction à priori peu usuelle : si le client a plus de 95 ans, une réduction de 50% est faite.

2.4.2.1 ZenCart

products	
PK	products_id
PK,FK1,I1	manufacturers_id
	products_type
	products_quantity
I5	products_model
	products_image
	products_price
	products_virtual
I3	products_date_added
	products_last_modified
I4	products_date_available
	products_weight
I9	products_status
	products_tax_class_id
I6	products_ordered
	products_quantity_order_min
	products_quantity_order_units
	products_priced_by_attribute
	product_is_free
	product_is_call
	products_quantity_mixed
	product_is_always_free_shipping
	products_qty_box_status
	products_quantity_order_max
I8	products_sort_order
	products_discount_type
	products_discount_type_from
I7	products_price_sorter
I2	master_categories_id
	products_mixed_discount_quantity
	metatags_title_status
	metatags_products_name_status
	metatags_model_status
	metatags_price_status
	metatags_title_tagline_status

Figure 16: ZenCart : table des produits

products_attributes	
PK	products_attributes_id
FK3	options_values_id
FK2,I1	products_id
FK1,I2,I1	options_id
	options_values_price
	price_prefix
I4	products_options_sort_order
	product_attribute_is_free
	products_attributes_weight
	products_attributes_weight_prefix
	attributes_display_only
	attributes_default
	attributes_discounted
	attributes_image
	attributes_price_base_included
	attributes_price_onetime
	attributes_price_factor
	attributes_price_factor_offset
	attributes_price_factor_onetime
	attributes_price_factor_onetime_offset
	attributes_qty_prices
	attributes_qty_prices_onetime
	attributes_price_words
	attributes_price_words_free
	attributes_price_letters
	attributes_price_letters_free
	attributes_required
FK2	manufacturers_id
FK1,FK3	languages_id

Figure 17: ZenCart : table des attributs

products_discount_quantity	
FK1	products_id
	discount_qty
	discount_id
	discount_price

Figure 18: ZenCart : table des réductions

(Schéma complet : Wikimedia[83], 7 mai 2007)

[83] Schéma de la base : http://commons.wikimedia.org/wiki/File:ZenCart_DB_Schema.jpg

Sont ici analysées trois classes significatives de l'organisation des prix : « products_discount_quantity » (Figure 18) et « products_attributes » (Figure 17), qui sont chacune reliées à « products » (Figure 16). Dans un premier temps, il peut être constaté que les rabais (*discounts*) sont attribués en fonction du nombre de produits commandés (*discount_qty*, *qty* étant *quantity*, donc quantité). Il s'agit donc d'une réduction simple.

Dans un second temps, on observe que les produits sont composés d'attributs (*products_attributes*), qui comportent des valeurs permettant de calculer le prix d'un produit (*price_factor*, *price_factor_offset*, …).

Ces éléments montrent que le prix est donc calculé lors de l'exécution de l'application, en intégrant les éléments provenant de la base de données.

2.4.2.2 Magento

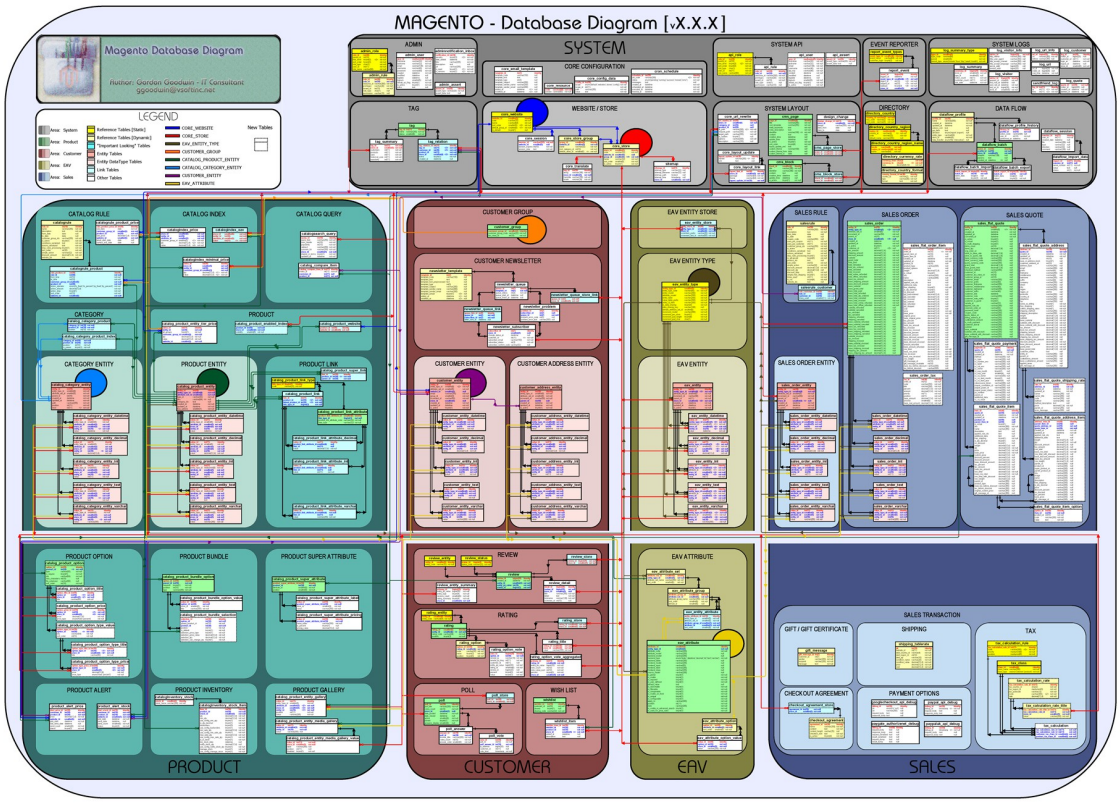

Figure 19: Schéma de la base de données de Magento

(Magentite[84])

84 Magentite : http://www.magentite.com/221/magento-shopping-cart/magento-database-diagram-table-list/ (consulté le 10/6/2013)

Le schéma global (Figure 19) de la base de données de Magento démontre une séparation bien distincte des groupes de tables. Il semble donc ne pas exister de système de paramétrage centralisé permettant d'ajouter des réductions liées à un élément non prévu lors de la réalisation de la base de données. Les réductions hors des habitudes commerciales classiques sont donc à mettre en œuvre au niveau du code source.

2.4.2.3 PrestaShop

Selon le schéma de la base de données (Figure 20) PrestaShop semble être logé à la même enseigne que Magento. En effet, une catégorisation des tables a été effectuée et, à nouveau, aucun élément « central » ne semble présent.

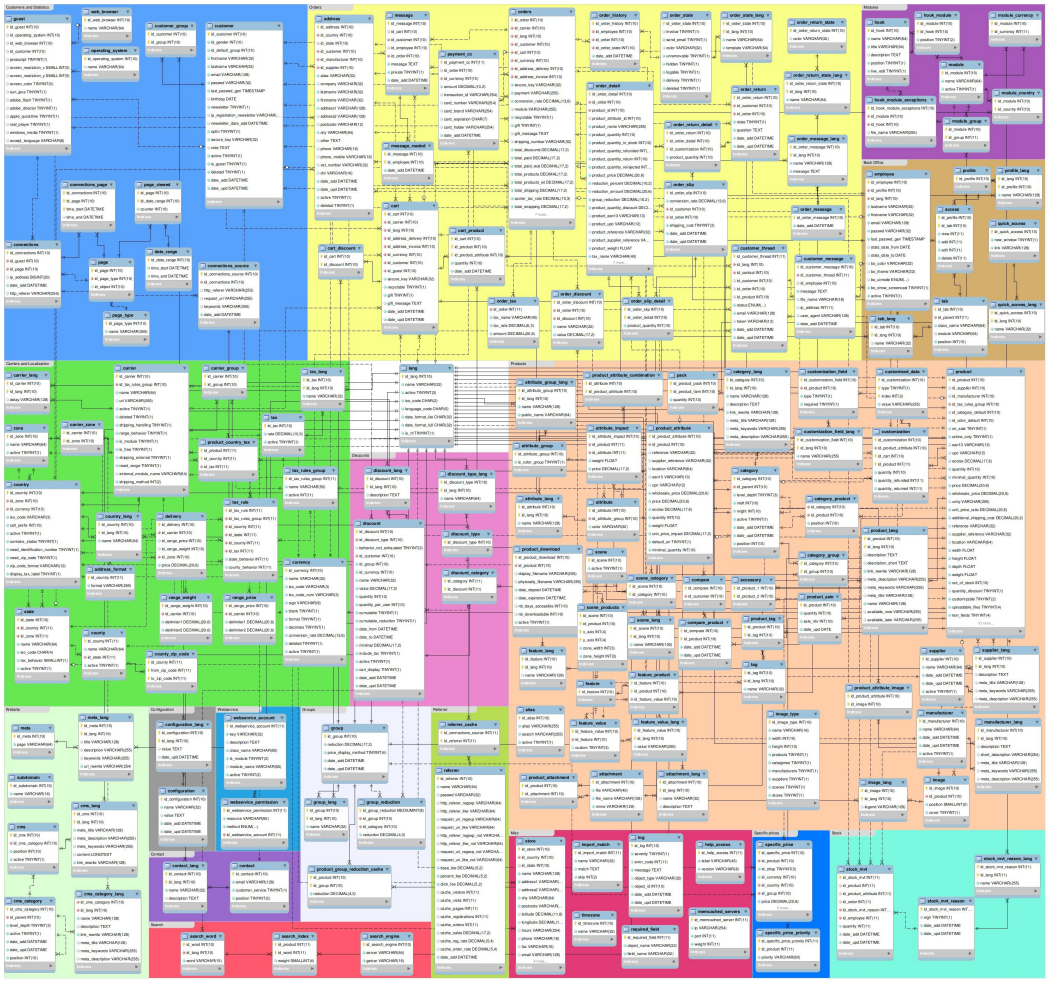

Figure 20 : Schéma de la base de données de PrestaShop

(Dave Egerton[85])

85 Dave Egerton : http://www.daveegerton.com/prestashop-guides/Prestashop-Developers-Guide.html

2.4.3 Synthèse des sites de e-commerce

Comme cela a pu être relevé précédemment, les sites de e-commerce ont des besoins très spécifiques. De ce fait, ils ont été pensés de manière à combler lesdits besoins, en cherchant à couvrir de base l'intégralité des besoins requis par un marchand.

L'hypothèse suggérant que ce type de site, de par ses besoins de flexibilité, avait probablement un système central permettant de mettre le même type de paramètres attachés à plus ou moins n'importe quelle table (un prix en fonction du lieu, du client, etc...), s'avère donc être fausse, du moins pour les solutions analysées.

2.5 Synthèse de l'existant

Parmi tous les éléments analysés, tous secteurs confondus, aucune solution ne semble prendre en compte une centralisation du paramétrage. Cette observation est quelque peu surprenante étant donné la diversité des types de sites parcourus, certains proposant d'être des contenants, d'autres du contenu, et d'autres alliant les deux.

Toutefois, ceci est explicable par les besoins métiers très ciblés. En effet, en faisant le choix de parcourir des secteurs ciblés, cela a pu fermer des portes vers des solutions plus polyvalentes.

Lorsqu'un besoin est clairement défini d'entrée de jeu, il est amplement suffisant de mettre en place les spécifications requises. Ainsi, le paramétrage est établi dès le départ, et chaque modification est faite au cas par cas, au fil des maintenances.

Les sites de e-commerce ont ainsi beaucoup de paramétrages possibles, mais ceux-ci sont étroitement liés à une partie du site, rendant son utilisation dans un autre contexte relativement difficile.

Le monde du jeu exploite systématiquement le même schéma : réalisation d'un jeu, traduction, et différentes parties, éventuellement paramétrées légèrement différemment.

Enfin, il existe probablement des situations dans lesquelles le code est dupliqué, afin de proposer des applications similaires, mais elles restent indépendantes.

Le prototype est donc réalisé sans pouvoir se baser sur une solution existante.

3. Développement d'un prototype

3.1 Éléments du prototype

3.1.1 Le prototype

Le prototype réalisé a pour nom de code « Gameway ». Il comprend une administration, une API, ainsi que des pages pour les exemples d'intégration d'un jeu.

3.1.2 Type de jeux

Afin de réaliser un générateur de jeux, il est nécessaire d'en déterminer les caractéristiques principales. Un jeu d'exploration en 3D n'a pas du tout la même logique qu'un jeu de casse-tête ou qu'un jeu d'élevage au clic par clic.

Dans le cas du prototype réalisé, il est développé un jeu faisant appel au maximum à la logique de génération, en limitant l'aspect graphique, celui-ci variant fortement en fonction du type de jeu mis en place.

Cette restriction fait notamment suite à la remarque faite au point 2.3.2.5 par « Kalan », membre d'un forum de discussion autour de World Oriented Objects.

C'est donc un jeu de type « point and click », donc au clic par clic : le joueur effectue une requête au serveur en cliquant sur un bouton/un lien/une image pour effectuer une action. Dans d'autres catégories de jeu il existe, par exemple, des jeux dans lesquels il est nécessaire de faire appel au clavier, d'utiliser les fonctions tactiles d'un écran multi-points, de prendre en compte les données du gyroscope d'un smartphone, ou d'autres modes d'interaction permettant à l'utilisateur de communiquer avec le jeu.

Le mode de jeu retenu, « point and click », est utilisé par exemple dans les jeux d'aventure : le joueur clique sur le lieu qu'il souhaite visiter, puis clique sur l'action qu'il souhaite effectuer. C'est donc dans l'optique de pouvoir réaliser un jeu d'aventure que ce prototype est mis en place.

3.1.3 Éléments du jeu

Une fois le type de jeu choisi, il est nécessaire de déterminer ce qui va le constituer. Les éléments récurrents dans les jeux observés sont généralement les suivants : « Qui » fait « quoi », « où », et « avec quoi ». Une autre caractéristique est parfois « avec qui », mais n'est pas forcément nécessaire, certains jeux étant une aventure à

vivre en solitaire. Il est donc possible de séparer ces quatre aspects en les représentant par des groupes précis :

Tableau 2 : Composants d'un jeu

Nom	Description	Exemple
Qui	Caractéristiques d'un joueur : Cela représente ses talents, ses compétences et toutes ses possessions, celles-ci étant une simple association clé-valeur. Les caractéristiques ont une valeur par défaut.	niveau de vie argent force habileté points de classement
Avec quoi	Objets/artefacts : Les objets sont ce qu'un joueur peut stocker dans son inventaire et récolter tout au long de son aventure. Ils sont cumulables et peuvent servir à débloquer des actions par exemple.	pansement boisson énergétique barre en or planche de bois
Quoi	Actions : Ce sont les possibilités qu'a un joueur de faire évoluer des caractéristiques ou récolter des objets. Les actions font le lien entre les objets et les caractéristiques.	"se soigner" nécessite un pansement et augmente la vie "se déplacer au nord-est" modifie les coordonnées du joueur
Où	Lieux : Ce sont les endroits dans lesquels le joueur peut effectuer des actions. Les lieux constituent les endroits où les actions peuvent être placées. Un même lieu peut se situer à plusieurs endroits sur une carte. Il y aura alors les mêmes caractéristiques à chacun de ces endroits	l'infirmerie la rue principale un coin de la galaxie

Voici un exemple concret combinant ces quatre éléments :

« Un joueur possédant "un pansement" et se trouvant "à l'infirmerie" a la possibilité de "se soigner". S'il le fait, sa "vie" augmente de 10, un pansement est déduit de son inventaire. »

Dans notre cas, nous n'utiliserons que les actions pour le prototype, et les caractéristiques pour les exemples mentionnés dans ce travail, car c'est suffisant pour démontrer les possibilités de paramétrage. Il est par contre difficile de réaliser un jeu intéressant autour d'éléments aussi restreints.

3.2 Paramétrage récursif

3.2.1 Hiérarchisation des variantes

Il s'agit ici d'anticiper l'évolution d'un jeu, par exemple d'en proposer des variantes, d'ajouter des parties ou de réaliser d'autres jeux dans le même moule. Il n'est pas question de maintenance : cet aspect est commun à tous les jeux du générateur et l'ajout d'une nouvelle fonctionnalité ou la correction d'un bug s'applique partout, avec des options de paramétrage (et notamment le fait d'activer ou non une fonctionnalité) pouvant varier.

Une solution serait la mise en place de paramètres précis, en définissant à l'avance des niveaux, par exemple : éditeur du jeu, type du jeu, ambiance du jeu, partie du jeu. Une telle arborescence pourrait être représentée via un schéma (Figure 21)

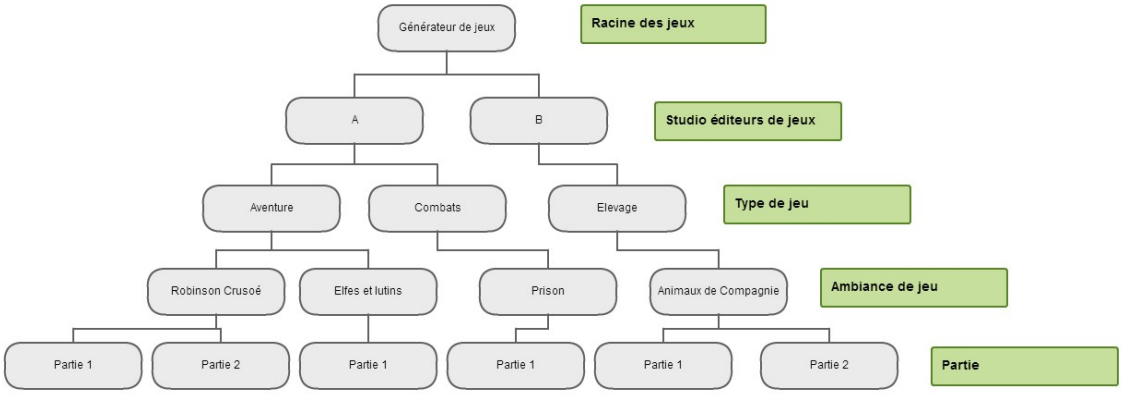

Figure 21: Arbre à niveaux prédéfinis

Sur ce schéma, il serait possible de déterminer que tous les jeux "A" ont le même règlement, que les jeux "Aventure" ont un fond rouge, que les jeux "Prison" sont interdits aux moins de 16 ans et que la "partie 1" (celle dans "animaux de compagnie") propose d'acheter des biens virtuels par SMS.

Toutefois, lorsque l'on souhaite ajouter un étage supplémentaire à cet arbre, par exemple pour proposer le jeu "Elfes et Lutins" en français et en anglais, cela peut vite devenir compliqué : la hiérarchie est clairement établie, et difficilement modifiable, étant donné que cela signifie ajouter un nouvel étage pour tous les jeux, et pas uniquement ceux concernés par la modification de hiérarchie à effectuer.

Dans le but de permettre un paramétrage plus facile pour chacun des jeux et d'en anticiper l'évolution, une des solutions est de faire en sorte que ledit paramétrage soit récursif.

Afin d'en comprendre la logique, un des parallèles possibles est une arborescence de dossiers sur la plupart des systèmes d'exploitation : lorsque l'administrateur modifie les paramètres d'un dossier, par exemple en changeant le mode d'affichage du contenu de celui-ci (voir schéma ci-dessous), la modification affecte tous les sous-dossiers. Or ces sous-dossiers n'ont pas été modifiés un à un : ils héritent du même mode d'affichage que leur parent (Images).

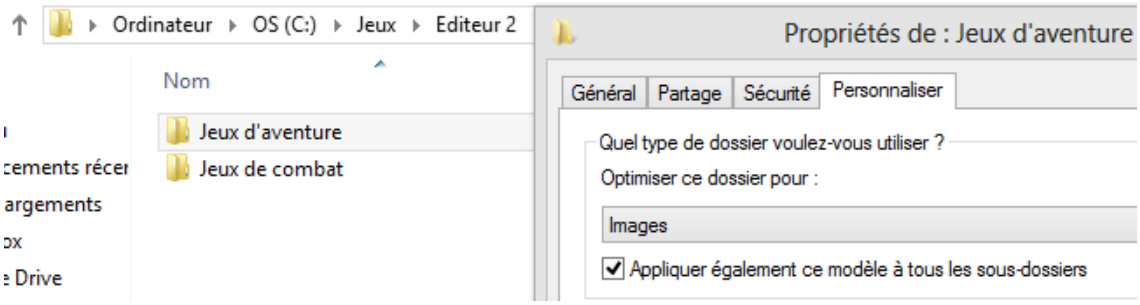

Figure 22: Dossiers Windows : changement des paramètres 1

Il est par la suite possible de modifier à nouveau le mode d'affichage (Vidéos) pour en choisir un spécifique appliqué à un groupe de jeux, voire un jeu seul :

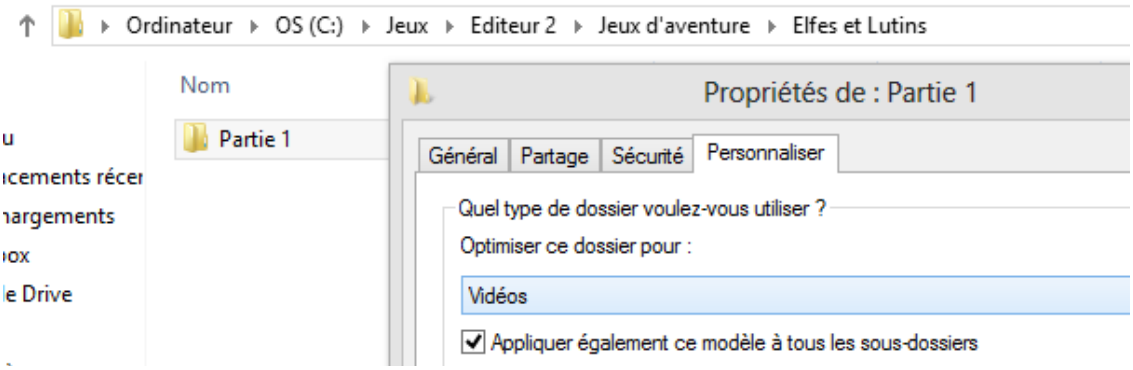

Figure 23: Dossiers Windows : changement des paramètres 2

Dans le cas d'un jeu, changer la couleur de fond d'un groupe de jeux doit changer le fond de tous les jeux de ce groupe. Si un sous-groupe change à son tour de couleur, les jeux qu'il contient prennent alors la deuxième couleur.

Il est donc possible de transformer l'arbre « Figure 21: Arbre à niveaux prédéfinis » comme sur la Figure 24.

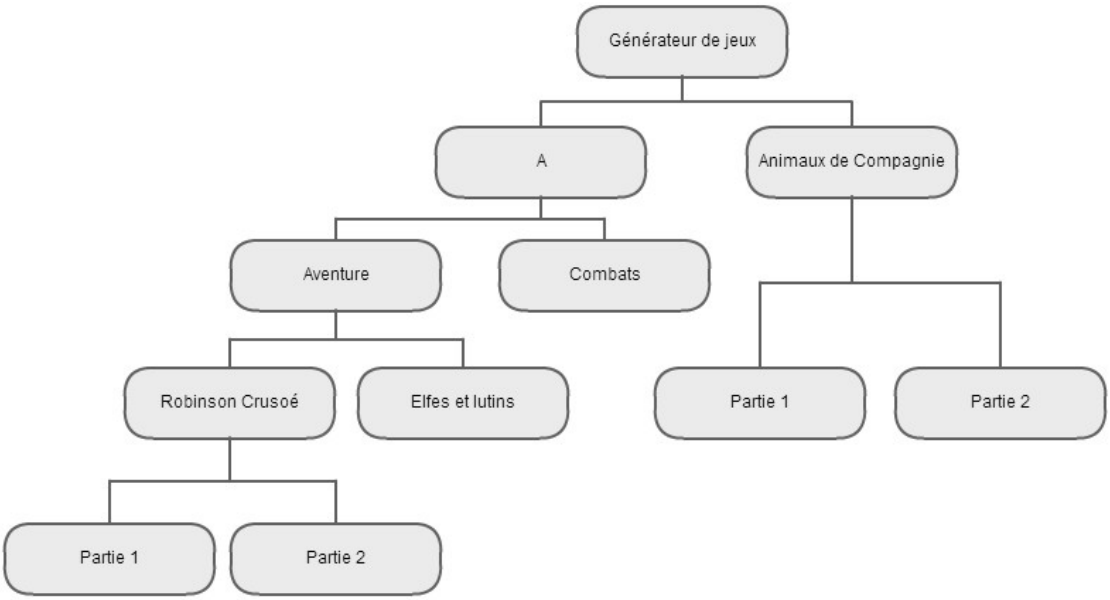

Figure 24: Paramétrage libre

Avec ce type de schéma, pour lequel il n'y a aucune restriction quant à la signification du niveau dans l'arbre, les nuances peuvent s'ajouter à n'importe quel endroit (Figure 24).

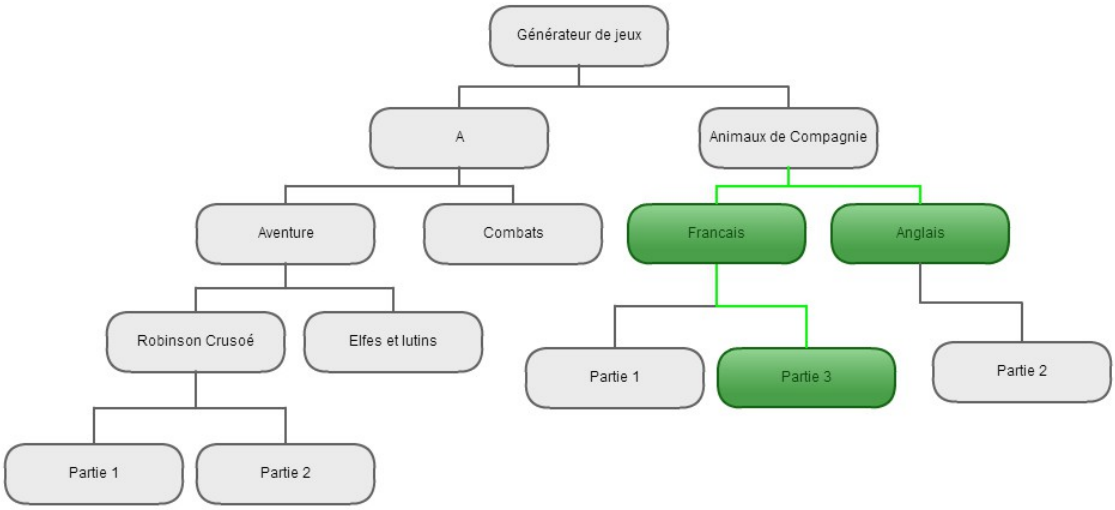

Figure 25: Paramétrage libre : ajout de variantes

Dans cet exemple, il a été choisi de différencier deux parties en raison d'un choix de langue différenciant la "Partie 1" de la "Partie 2". L'anglais aurait très bien pu être paramétré directement dans la "Partie 2", tandis que le français voit arriver une nouvelle partie, la "partie 3".

Pour permettre une manipulation aisée, il est intéressant de faire le parallèle avec les dossiers : un sous-dossier est lui-même un dossier parent pour ses propres sous-dossiers. C'est donc là que l'on observe la notion de récursivité.

Un premier diagramme de classes démontre le lien récursif qui existe entre les différentes variantes (Figure 26)

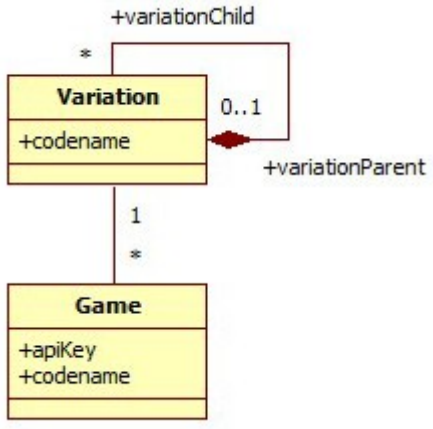

Figure 26: Diagramme de classes :
récursivité des variantes

Chaque variante (*variation*) est donc l'enfant d'une autre variante, exception faite de la variante du sommet de l'arbre, qui n'a pas de parent. Un jeu à proprement parler est une feuille attachée à une variante. Il est possible d'avoir deux feuilles sur la même variante, les deux jeux ayant ainsi exactement les mêmes paramètres.

A noter que le jeu est identifié par une clé spéciale (*apiKey*), qui sert à l'identifier depuis l'extérieur, sans pouvoir le parcourir facilement, par exemple en changeant de numéro dans le cas où les nombres se suivent.

Toutes les variantes et tous les jeux ont un nom de code (*codename*), qui est utilisé en interne pour permettre aux administrateurs de retrouver facilement leur hiérarchie.

3.2.2 Gestion des paramètres

Pour ce qui est des paramètres, la structure implique de les lier directement à une variante. Une instance d'un objet est accessible par des associations spécifiques à d'autres objets, qui réduisent son "environnement". Dans le cas de l'association de ses paramètres à une variante, cet environnement doit, en plus des associations mentionnées précédemment, être associé de manière systématique à une variante.

De ce fait, il est ajouté une classe de paramètre, qui est une classe abstraite (donc non instanciable), dont héritent les classes précises faisant office d'option paramétrable. Un cas particulier est que les paramètres ayant une seule valeur pour le jeu (c.f. 3.2.3, Cumul des paramètres) est représenté par une classe "SingleValue", comportant tous ces paramètres.

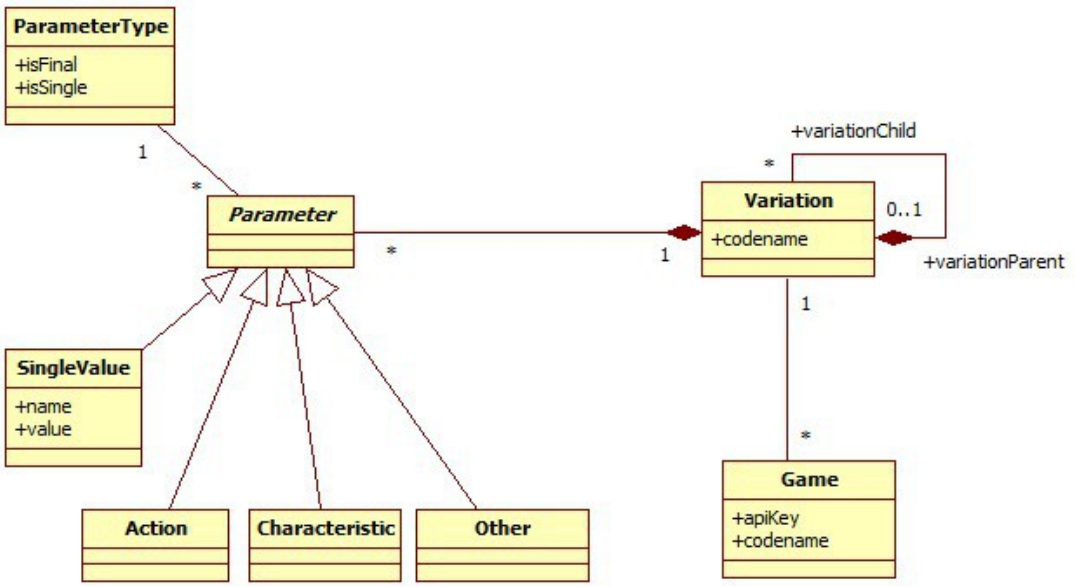

Figure 27: Diagramme de classes : ajout des paramètres aux variantes

Afin de faciliter la manipulation des données, un gestionnaire de paramètres est réalisé. Il se charge d'obtenir tous les paramètres pour un seul jeu, de les séparer par catégorie pour permettre une utilisation plus aisée par la suite : lorsque l'on a besoin d'ajouter un paramètre, de le lier à une autre variante ou toute autre action commune à tous les paramètres, peut importe leur nature, il est possible de faire appel à ce gestionnaire.

3.2.3 Cumul des paramètres

Certains paramètres peuvent être cumulables pour le même jeu :

- Caractéristiques d'un personnage (santé + argent + force + habileté = personnage)
- Lieux à découvrir (forêt + rivière + route + clairière = carte du jeu)
- Objets à posséder (armure + bouclier + épée + cheval = inventaire possible)

Certains paramètres écrasent le paramétrage précédent :

- Couleur d'un élément (fond du site rouge donc pas bleu)
- Nom du jeu, description (deux noms pour le même jeu ?)
- Date de fin de partie (lorsqu'il est fini, il ne peut pas finir à nouveau)

Dans le cadre de ce prototype, la classe *ParameterType* (Figure 27) n'est pas implémentée. Elle est là à titre informatif.

La gestion du cumul est toutefois gérée une fois, de manière ciblée, sur les paramètres de configuration (classe *SingleValue*), tels que le nom du site, l'email du responsable, qui sont des valeurs uniques.

L'interdiction de la redéfinition d'une valeur (*isFinal*) n'est pas implémentée non plus, mais est également présentée pour suggérer des évolutions sur la plateforme.

3.3 Code développé

3.3.1 Interface d'administration

L'interface d'administration a été réalisée à l'aide de différents éléments déjà existants pour permettre une navigation plus agréable, et se concentrer sur le développement du code :

- jQuery[86] est une librairie JavaScript permettant notamment de manipuler facilement les éléments DOM[87], et d'utiliser aisément certaines fonctionnalités standard, et de manière compatible sur tous les navigateurs. Cette librairie laisse également la possibilité de créer des composants.
- Bootstrap[88], de Twitter est une base de CSS et de JavaScript permettant de ne pas débuter la structure d'une page sans rien. Cette couche de base permet notamment la mise en forme de tableaux, de formulaires, et propose également des fonctionnalités permettant un responsive-design (design s'adaptant à la taille de l'écran)
- Le thème « Core Admin »[89] est basé sur Bootstrap. En plus des fonctionnalités proposées par ce dernier, il propose une réelle identité visuelle, et ajoute son lot de composants jQuery, permettant notamment la création de graphiques.
- Le logo « Gameway » a été réalisé par Régine Amichba[90], étudiante en graphisme au Centre de Formation Professionnelle Arts Appliqués.

86 jQquery : http://jquery.com/
87 DOM : Document Object Model
88 Bootstrap : http://twitter.github.io/bootstrap/
89 Core Admin : https://wrapbootstrap.com/theme/core-admin-WB0135486
90 Contacter Régine Amichba : regine.amichba@gmail.com

3.3.2 Variantes

3.3.2.1 Éditeur de variantes JavaScript

Afin de pouvoir traiter les variantes tout en représentant les données de manière intuitive, un éditeur JavaScript a été créé.

Dans un premier temps, les données sont récupérées via un fichier JSON, ne contenant aucune arborescence à proprement parler : chaque variante (nœud, *knot* en anglais) a quelques données à elle, telles que son nom, son nombre de jeux attachés, son identifiant, et comporte impérativement l'identifiant du nœud de son parent, exception faite du nœud « racine », qui contient l'identifiant 0 (zéro). Il est impératif que le JSON contienne des données déjà triées : lorsque l'enfant est traité, le parent doit être connu. Cela économise des traitements.

Un tableau prêt à recevoir les nœuds traités est créé :

```
$knotsArray = [] ;
```

Les nœuds sont parcourus les uns après les autres, afin d'être ajoutés correctement dans l'emplacement prévu.

```
$knots = $.parseJSON($jsonData) ;
jQuery.each($knots, function($indexKnot,$knot){
	// Traitement de chaque nœud ici
}) ;
```

Pour chaque nœud, il s'agit ensuite de l'ajouter dans le tableau, pour qu'il puisse être trouvé par ses nœuds enfants. Le nœud est également ajouté au contenu de son parent. Si le nœud n'a pas de parent (le cas de la racine), il est ajouté au bloc servant à l'affichage.

```
// Nouveau block ayant une classe précise (identifiant CSS)
$blockKnot = $('<div>',{'class':$blockVariationClass}) ;
// $blockKnot.append(...) ; // ajout d'informations si voulu

// Ajout à son parent ou à la racine
if($knot.parent in $knotsArray)
	$knotsArray[$knot.parent].content.append($blockKnot) ;
else
	$blockComponent.append($blockKnot) ;
```

Avec le sélecteur jQuery, une collection de « blocks » représentant les nœuds est créée, afin de pouvoir les traiter en groupe. Le préfixe du point représente les classes.

```
$blocks = $('.' + $blockVariationClass) ;
```

Les « blocks » sont ensuite définis pour du *drag & drop*, à nouveau en utilisant les fonctions de jQuery : *draggable* et *droppable*.

```
$blocks.draggable({
	// Paramètres pour tout ce qui est « glissable »
}) ;

$blocks.droppable({
	// Paramètres pour tout ce qui est «déposable»
	drop: function( event, ui ) {
		// Déplacement du block vers son nouveau parent
		$(this).append(ui.draggable);
		// Enregistrement du nouveau parent
		setNewParent(ui.draggable,$(this))
	}
}) ;
```

Pour sauvegarder les changements, il est envoyé au serveur un tableau contenant la liste des nœuds modifiés, et contenant leur nouveau parent :

```
// Tableau des parents changés
var $knotsParentsChanged = {} ;

// Fonction assignant le nouveau parent
function setNewParent($child,$newParent){

	// Nouvelles informations
	$idParent = getBlockId($newParent) ;
	$idChild = getBlockId($child) ;

	// Sauvegarde des modifications
	$knotsArray[$idChild].parent = $idParent ;
	$knotsParentsChanged[$idChild] = $idParent ;

	// Préparation des données à l'envoi
	$toSave = JSON.stringify($knotsParentsChanged) ;
	$('#input-newparents').attr('value',$toSave) ;
}
```

Le résultat est ensuite traité lors de la validation des modifications. Les données sont envoyées au serveur, qui se charge alors de modifier les parents dans la base de données.

Voici la mise en œuvre de ce script dans le prototype. Dans un premier temps, 4 parties sont créées, nommées de A à D (Figure 28).

Figure 28: Parties initiales

Le déplacement d'une variante dans une nouvelle variante se fait en glissant un bloc dans un autre bloc (Figure 29). Dans ce cas, deux nouvelles variantes ont été créées, « Contemporain » et « Aventure », pour représenter deux thèmes de jeux.

Figure 29: "Partie A" glissée dans la catégorie "Aventure"

En ajoutant des variantes, l'utilisation reste identique. Il n'est pas nécessaire de respecter une architecture préétablie, les « branches » peuvent être de longueur différente (Figure 30).

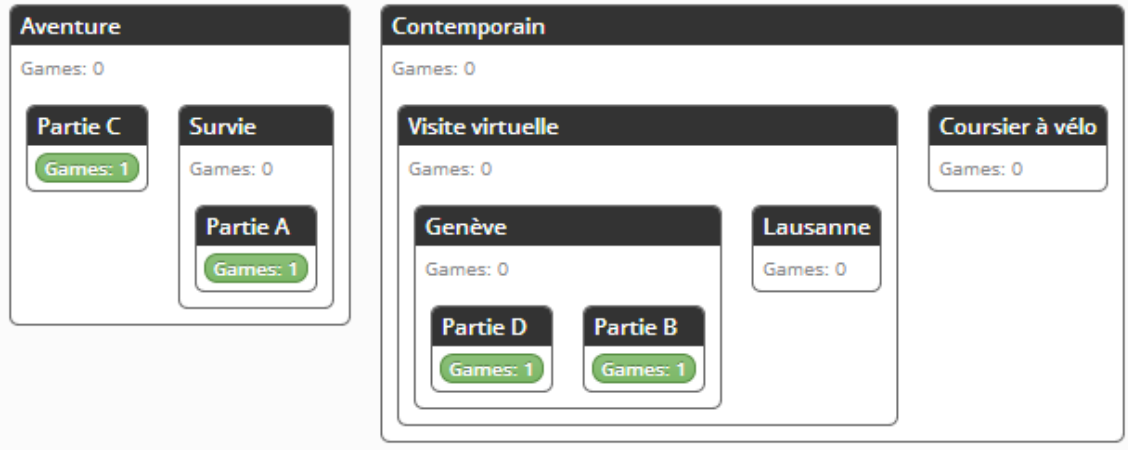

Figure 30: Variantes composées de variantes

Il est possible, par exemple, de décider que les personnes jouant à un jeu se trouvant dans une variante de « Genève » passent dans un jeu de « Survie », au lieu de faire de la « Visite virtuelle ».

Du côté serveur, la classe Variations.class.php (Annexe 1 : Variations.class.php, page 57) permet de gérer toutes les variations, quelle que soit la racine à traiter. Outre les éléments permettant de reconstituer la hiérarchie d'une variante, il est possible de vérifier différents éléments, comme le fait qu'un nœud fasse partie des enfants, ou d'obtenir le nom des parents au format texte.

3.3.3 Paramètres

Un paramètre est un élément lié à une variante. De ce fait, lorsqu'une variante est déplacée, ses paramètres l'accompagnent et se répercutent sur les jeux associés.

Tous les paramètres se basent sur des fonctionnalités et des attributs semblables, et des nuances sont ajoutées autour des données spécifiques de chacun d'entre eux : les paramètres liés aux actions comportement des attributs spécifiques à celles-ci. Il en est de même des caractéristiques d'un joueur, ou tout autre type de paramètre.

Lors de l'ajout d'un paramètre, il est possible pour l'utilisateur de sélectionner la variante autour de laquelle il souhaite travailler pour mettre en place ses modifications (Figure 31).

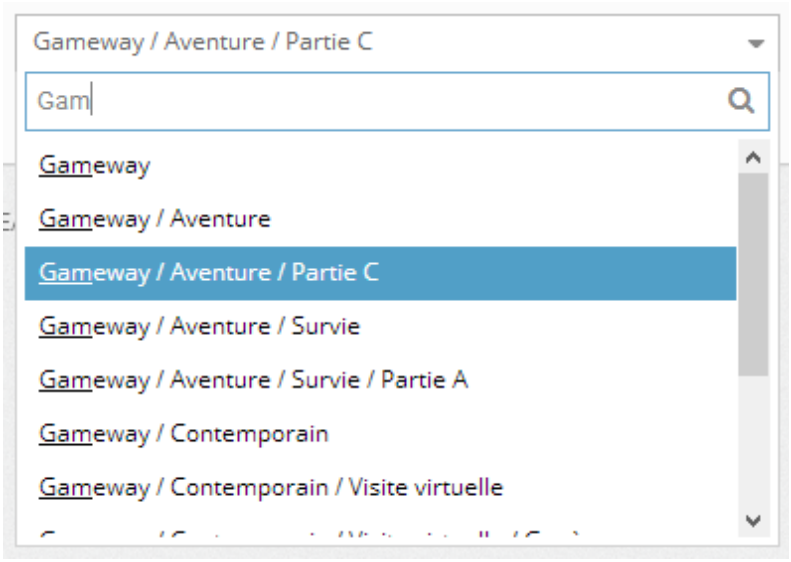

Figure 31: Choix de la variante

Un paramètre est basé sur la classe « Parameter.class.php » (Annexe 2 : Parameter.class.php, page 59) . Cette classe est abstraite, et ne peut donc pas être directement instanciée. Il faut donc, pour un paramètre précis, étendre ladite classe.

Pour détailler le fonctionnement, voici comment un paramètre de configuration, servant par exemple à choisir le titre du jeu, peut être mis en place :

```php
<?php
// Création d'une nouvelle configuration
$newConf = new Configuration();

// Ajout des données (dans Configuration.class.php)
$newConf->setName($name) ;
$newConf->setValue(stripslashes($value)) ;
```

```php
// Lier à une variante (dans Parameter.class.php)
$newConf->setVariation(getCurrentRoot()) ;

// Sauver l'instance (dans Parameter.class.php)
$newConf->saveInstance();
?>
```

Il fait donc appel à la classe « Configuration ». qui étend la classe « Parameter » :

```php
<?php
class Configuration extends Parameter {

	// Configuration attributes
	private $name = null;
	private $value = null;

	function __construct() {
		// Define database parameters
		$this->setTableName("configurations","con");
	}

	// Called when the new paramater is created
	function udpateInstance(){
		$this->queryUpdateInstance(array(
			"name"=>$this->name,
			"value"=>$this->value,
		)) ;
	}

	// Set the attributes
	function setName($n){ $this->name = $n ;}
	function setValue($v){ $this->value = $v ;}

}
?>
```

Le constructeur définit de quel type de paramètre il s'agit, en renseignant l'équivalent dans la base de données. Dans ce cas, il s'agit de la table « configurations », et les champs sont préfixés par l'abréviation du nom de la table, « con » : « con_id », « con_name », « con_value ».

Lorsque la fonction *saveInstance* est appelée, un nouveau paramètre est inséré dans la base de données via le code présent dans la classe Parameters. Son identifiant est enregistré, et la fonction *updateInstance* est alors appelée. Configuration possède le code à exécuter lors de la création de l'instance. Il est choisi de mettre à jour les données via la fonction updateInstance, en fournissant les champs correspondant.

Les champs ne sont pas renseignés par défaut, et une nouvelle entrée dans la table configurations comporte des champs vides attendant d'être complétés.

3.3.4 Gestionnaire de paramètres

Les identifiants des paramètres sont identiques aux identifiants de la table qu'ils lient. Ainsi, il est possible de récupérer toutes les données en une seule requête à la base de données.

La classe ParameterManager (Annexe 3 : ParameterManager.class.php page 61) se charge de récupérer, pour une variante donnée, les paramètres associés, répartis par catégorie. Pour ce faire, elle se charge dans un premier temps de récupérer les identifiants des variantes à l'aide de « Variations.class.php » :

```
// Call the load functions
private function loadParameters(){
	$this->loadVariationsIDs();
	$this->loadParametersFromBase();
}

// Get the parents IDs (included self id)
private function loadVariationsIDs(){
	$this->variationsIDs =
		self::$variations->getParentsKeys($this->finalId);
}
```

Une fois les identifiants créés, la fonction loadParametersFromBase est appelée. Elle se charge de construire la requête à l'aide de la fonction constructQuery, de l'exécuter, puis de passer les résultats à la fonction treatBaseResults.

```
// Generate query
private function constructQuery(){

	// JOIN ALL THE PARAMETERS TABLES
	$joins = array() ;
	foreach(self::$tables AS $tName => $tPrefix){
		$joins[] =
			"LEFT JOIN $tName ON ${tPrefix}_id = par_id" ;

	$joins = implode("\n",$joins) ;

	$where = implode(',',$this->variationsIDs) ;
	$where = "WHERE par_var_id IN ($where)" ;

	return "SELECT * FROM parameters $joins $where" ;

}
```

La requête est construire en ajoutant toutes les tables, en liant leur identifiant à l'identifiant du paramètre. Afin de récupérer les paramètres correspondant à la variation, la clause « WHERE » vérifie que la variante fasse partie d'une énumération de valeurs : la clé de la variante.

Le traitement des résultats est réalisé de la manière suivante : chaque résultat retourné par la base de données est parcouru. Ensuite la table, et donc le paramètre auquel elle correspond, est vérifiée : il est possible que d'anciens paramètres ne soient plus utilisés et tout de même présents dans la base.

Pour chaque résultat, tous les champs sont parcourus. Il ne faut pas oublier qu'en fonction du LEFT JOIN de la requête, les champs vides des autres paramètres sont présents. Il faut donc récupérer les champs correspondant à la table actuelle.

Lorsqu'un champ est un des champs spécifiques aux paramètres, il est ajouté à un tableau correspondant à ceux-ci.

```
// Init filds to use
$fields = array() ;

// Get table prefix
$prefix = self::$tables[$table];

// Each field from the current parameter
foreach($d AS $name => $value){

	$regex = '#(^'.$prefix.'_)#i' ;
	if(preg_match($regex,$name)){
		$fields[preg_replace($regex,'',$name)] = $value ;
	}elseif(preg_match('#par_#i',$name)){
		$fields['parameter'][$name] = $value ;
	}
}
```

Une fois les champs récoltés, ils sont ajoutés à un tableau comportant tous les paramètres de même type (avec le même nom de table). Dans le cas où le tableau n'existe pas, il est créé.

```
// If doesn't exists, create it
if(! isset($this->parameters[$table]))
	$this->parameters[$table] = array() ;

// Add a row in the table of this parameters
$this->parameters[$table][] = $fields ;
```

Avec ce tableau rempli, il est aisé de récupérer tous les paramètres d'un champ :

```
// Get parameters table from a specific type
public function getParametersOf($name){

	if(! isset($this->parameters[$name]))
		return array() ;

	return $this->parameters[$name] ;

}
```

Ce gestionnaire instancié, il est alors possible de le transmettre à d'autres gestionnaires servant à traiter des paramètres précis :

```
$this->pm = new ParameterManager();
$this->pm->setFinalId($this->gameVariationId) ;
$this->configManager = new ConfigurationManager($this->pm) ;
```

Le traitement des données est alors géré par le nouveau gestionnaire, mais les données n'ont été chargées qu'une seule fois depuis la base de données.

3.3.5 Pages de démonstration

3.3.5.1 Traitement des données

La démonstration est constituée d'un composant jQuery réalisé dans le but de traiter le fichier JSON obtenu sur le serveur, ainsi que de plusieurs intégrations au sein de pages de test.

Les fichiers JSON sont générés à l'aide des gestionnaires de paramètres, de configurations, d'actions, de caractéristiques par exemple.

Ces fichiers ont la particularité d'être moins consommateurs en bande passante que leur alternative XML, qui privilégie la lisibilité des données. Ils ont également la caractéristique d'être standardisés, ce qui implique que la plupart des librairies, que ce soit côté client ou serveur, permettent de lire ce type de fichiers, et de les générer à partir de tableaux ou d'objets.

Le traitement de ces fichiers est réalisé par le composant « gameway_web.js » (Annexe 4 : gameway_web.js). Celui-ci peut servir de base pour tous les jeux de Gameway, et il suffit d'intégrer ce composant sur le nouveau jeu, et de le configurer en lui fournissant au minimum la clé correspondant au jeu :

```
$('#game').gamewayGame({
      key: "VALEUR_DE_LA_CLE"

}) ;
```

Étant donné qu'il s'agit d'un composant, il est nécessaire de charger au préalable jQuery, par exemple dans sa dernière version :

```
<script src="http://code.jquery.com/jquery-latest.js"></script>
```

```
{
   "GET":{
      "key":"apketed59ens9yiwukxb9ey66y7ke3xnzpklq430elicrrgqm4
      62269andfrvssd2pgjigbxkl16u24237acdl89kdasqgwtovyz"
   },
   "POST":[
   ],
   "api":{
      "version":0.1,
      "updated":"2013-06-03 12:00:00",
      "date":"2013-06-10 08:14:52"
   },
   "game":{
      "key":"apketed59ens9yiwukxb9ey66y7ke3xnzpklq430elicrrgqm4
      62269andfrvssd2pgjigbxkl16u24237acdl89kdasqgwtovyz",
      "id":"1",
      "name":"Made in Gameway",
      "description":"Ceci est un jeu r\u00e9alis\u00e9 \u00e0
      l'aide de Gameway, la solution de jeux en marque
      blanche.",
      "editor":"Company name",
      "members":14459318
   },
   "characteristics":[
      {
         "name":"Vie",
         "value":"50"
      },
      {
         "name":"Points",
         "value":"0"
      }
   ],
   "actions":[
   ]
}
```


Figure 32: Exemple de JSON généré

3.3.5.2 Répercutions des configurations

Les exemples sont basés sur la structure générée pour démontrer l'administration des variantes (Figure 30, page 41).

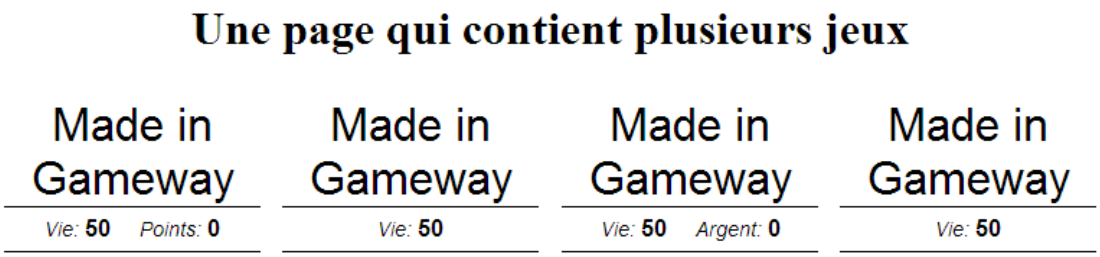

Figure 33: Les 4 jeux en version non configurée

Quatre jeux sont mis en place sur la même page, chacun faisant référence à un jeu différent. Cela permet également de démontrer « l'étanchéité » d'un composant. Il est possible de constater que ceux-ci ont le même nom par défaut (Figure 33).

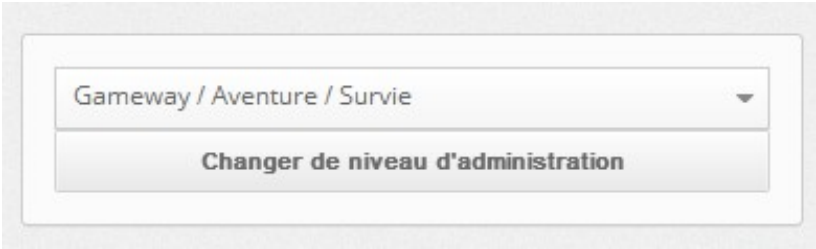

Figure 34: Administrer une variante

En se plaçant maintenant dans une nouvelle variante (Figure 34), il est possible de redéfinir, pour cette variante uniquement, le titre des jeux qu'elle contient (Figure 35).

Figure 35: Reparamétrage du titre

Le résultat est que seul le jeu dans la catégorie « Survie » a été impacté (Figure 36)

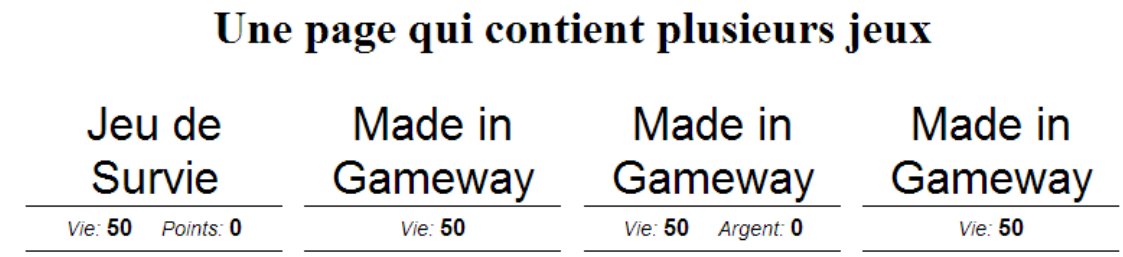

Figure 36: Un seul titre impacté

En éditant un titre au niveau « Aventure », qui englobe la variante « Survie », seul le titre de la partie ne faisant pas partie de cette dernière variante est édité (Figure 37).

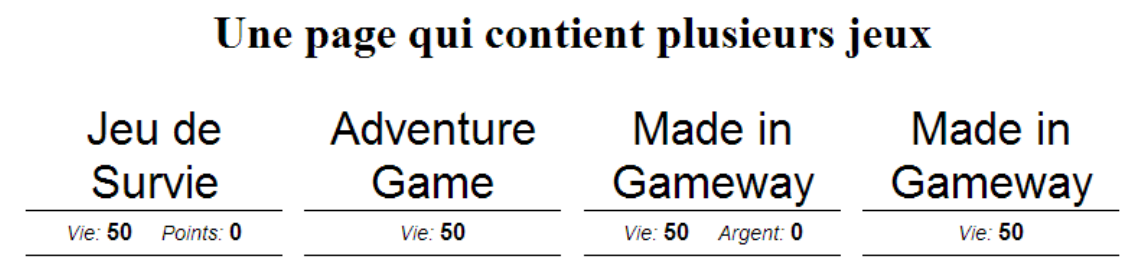

Figure 37: Un titre sur deux d'impacté

Bien entendu, lorsqu'une variante englobant plusieurs variantes est modifiée, toutes les variantes parmi ses enfants sont changées (Figure 38).

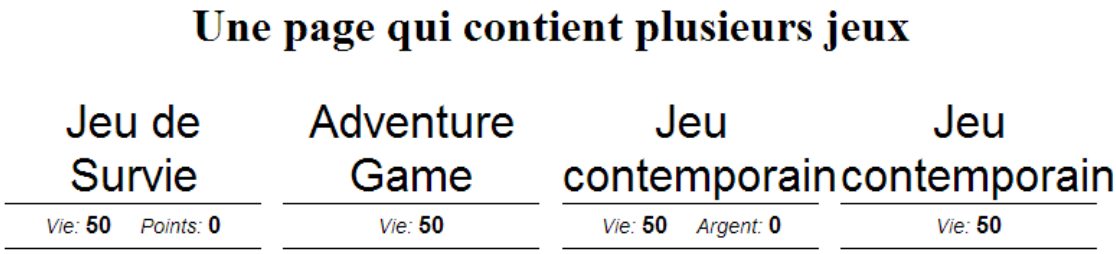

Figure 38: Modification de toutes les sous-variantes

Lors d'un déplacement d'une variation dans une autre (Figure 39), les paramètres éventuellement liés sont conservés, et un nouvel héritage de configuration est mis en place.

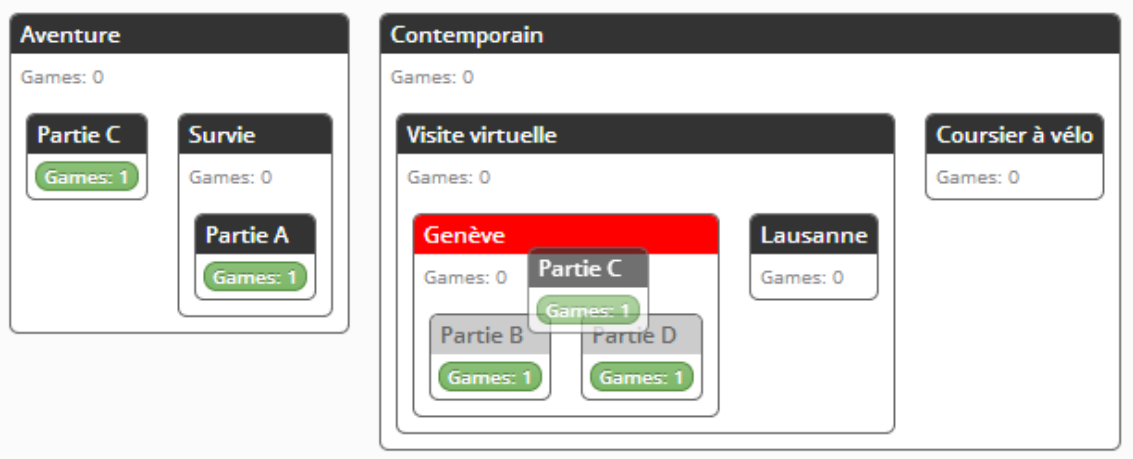

Figure 39: Déplacement d'une variante paramétrée

Les paramètres mis en place dans « Aventure » sont donc perdus au profit de ceux affectés aux jeux Contemporains, de Visite Virtuelle, de Genève (Figure 40).

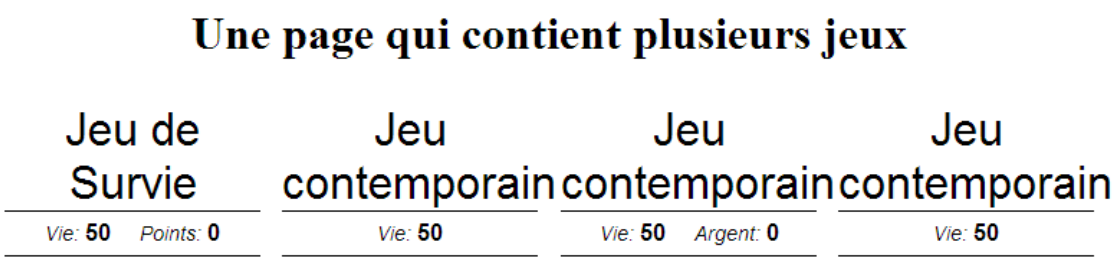

Figure 40: Nouveau titre affecté

3.3.5.3 Personnalisation de l'interface

L'interface est personnalisable à l'aide de CSS. Les classes CSS nommées à l'intérieur du composant comportent toutes le préfixe « gw- », afin de ne pas empiéter sur de mêmes éléments du même nom présents sur le site qui intègre le jeu.

Une mise en page initiale a été faite, afin de structurer l'information de base (Figure 41).

Figure 41: Jeu sans personnalisation

Il est possible de redéfinir le style en ajoutant une nouvelle feuille de style pour adapter, voire substituer, la feuille précédente (Figure 42). L'image de fond a été réalisée par Jehan K.

Figure 42: Jeu avec personnalisation

Dans le cas d'une application n'utilisant pas le fichier JavaScript « gameway_web.js » (Annexe 4 : gameway_web.js), les données peuvent bien entendu être manipulées différemment. Il est tout à fait envisageable de proposer de jouer via une interface totalement différente manipulant le fichier JSON.

Un exemple fantaisiste consisterait à utiliser une application lisant la liste des actions disponibles, et il serait ensuite nécessaire de secouer un smartphone pour effectuer une action, sans jamais avoir à utiliser d'interface graphique.

3.4 Aller plus loin

Ce prototype n'étant pas, comme son nom l'indique, une application complète, les possibilités d'évolution sont multiples. En effet, le prototype ouvre uniquement la voie vers d'autres possibilités, que l'on pourrait mettre en place en proposant du paramétrage supplémentaire.

Cela peut passer par les éléments déjà mentionnés dans ce travail, comme par exemple mettre des lieux en place, ou des interactions avec d'autres joueurs.

La gestion des administrateurs d'un jeu peut faire office de paramètre : relié à une variante en particulier, un utilisateur peut modifier seulement les autres paramètres correspondant à l'arborescence qui lui a été désignée. Cela est utile pour nommer des modérateurs, et séparer proprement les clients.

La fusion de paramètres est également une possibilité d'évolution : lorsqu'une arborescence est composée d'une lignée de variantes uniques, il pourrait être souhaitable de regrouper celles-ci, en rattachant tous les paramètres à une seule d'entre elles.

La prise en compte des conflits peut être nécessaire en fonction des types de paramètres choisis. Dans le cas concret de l'ajout de lieux sur une carte, l'ajout de nouveaux emplacements permet d'agrandir l'espace du joueur. Il reste toutefois toujours des lieux vides, en dehors des limites d'une carte. Un administrateur peut ainsi l'agrandir. Cependant, il est possible que la carte ait déjà été agrandie dans une sous-variante. Il faut donc être en mesure de résoudre ces conflits.

4. Conclusion

4.1 L'existant

Comme mentionné dans le chapitre « Synthèse de l'existant », en page 30, il est assez surprenant mais néanmoins explicable que les sites aussi importants que des sites de e-commerce ou des jeux web ne proposent pas de variantes plus nuancées.

Dans la plupart des cas, des variantes sur un seul niveau sont mises en place, à l'image des blogs et des forums, proposant systématiquement, pour un contenu différent, le même contenant : lorsqu'une modification est apportée, elle l'est soit sur l'ensemble des sites, soit sur un seul site en particulier.

Une solution plus polyvalente pourrait toutefois trouver une utilité dans bien des secteurs, notamment lorsqu'une segmentation territoriale est souhaitée : un site d'informations pourrait décider de la portée de sa publication en fonction de son importance et du public potentiellement impacté.

Existe-t-il des secteurs pour lesquels une telle solution est utilisée ? Peut-être dans de façon plus discrète, ou pour des applications utilisées à l'interne d'entreprises ?

4.2 Le prototype

La réalisation du prototype a, quant à elle, apporté beaucoup plus à la mise en place de variantes. Cette solution est relativement globale et peut s'appliquer à bon nombre d'applications, quel que soit leur secteur.

La logique, au final relativement simple, permet de proposer des variantes facilement, de les réorganiser, d'inverser des variantes et des sous-variantes, et d'attacher n'importe quel type de paramètre.

Le système du transfert d'informations en JSON offre un large choix d'opportunités, étant donné que ce format est utilisable dans bien des domaines : les applications web, les applications mobiles, les jeux flash, depuis la partie serveur d'un site web, une application en dur, une application pour de nouvelles lunettes tactiles, une caisse enregistreuse, ou un jouet électronique connecté au web.

Du côté du résultat obtenu, beaucoup de temps a été alloué à la recherche et il aurait été intéressant de pouvoir mettre en œuvre certaines fonctionnalités comme la

possibilité de définir si un paramètre et définissable à nouveau dans sa descendance, ou non.

Un aspect manquant au prototype est la connexion des joueurs. Dans ce qui a été réalisé, seul le contenu du jeu a été paramétré. Il aurait été appréciable de pouvoir s'attarder sur le lien entre un joueur attaché à un jeu, et dont les caractéristiques changent : si le joueur récolte du blé, et que sa partie est rattachée à un jeu de récolte de carottes, que se passe-t-il de son blé ?

Serait-il également possible de mettre en place une partie se déroulant sur plusieurs variantes ? Ou mettre une variante avec deux parents ?

4.3 Conclusion personnelle

Au début de ce travail, je souhaitais, en analysant les sites existants, être en mesure d'élaborer un prototype permettant d'ajouter facilement des « niveaux de variations », à l'image de la figure « Arbre à niveaux prédéfinis » en page 33. Il s'agissait de répertorier différentes catégories de personnalisation, et de les hiérarchiser. La subtilité aurait consisté à pouvoir intercaler un nouveau niveau entre deux niveaux existants.

Cependant, la solution retenue avec les paramètres liés aux variantes récursives permet une manipulation encore plus aisée de l'information. Je ne regrette donc aucunement de ne pas avoir suivi l'idée de base, en trouvant une méthode plus efficace pour répondre à l'objectif souhaité.

Durant mes recherches, j'ai été surpris de ne pas trouver de solutions plus polyvalentes dans les propositions de variantes. Cela m'a aussi permis de me rendre compte que le monde du jeu web, que je connaissais en partie avant de commencer, répétait le même schéma pour proposer des parties différentes, indépendamment de l'éditeur.

J'ai également pu me rendre compte, outre l'aspect technique, que les thèmes abordés étaient récurrents : presque tous les éditeurs proposent de jeux de bataille médiévale, de pirates, de cow-boys, de combats spatiaux. Un générateur de jeux pourrait apporter des nuances à l'industrie du jeu, en proposant d'autres thèmes importants de l'Histoire : un jeu de pirates pourrait être joué du côté des marchands d'épices, un jeu au Moyen-Âge pourrait proposer de faire de l'élevage et non des batailles, et d'autres possibilités encore peu exploitées.

Ce fut intéressant de travailler sur les différentes parties du prototype, car elles étaient complémentaires. Pour les autres projets auxquels j'ai participé, il y avait en général une application, autour de laquelle venaient se greffer des éléments. Dans le cas de ce travail de Bachelor, l'efficacité de l'administration ne pouvait être testée qu'en réalisant la génération d'un JSON qui, lui-même, avait besoin d'un site web pour être interprété.

J'ai particulièrement apprécié de pouvoir mettre en exergue une de mes passions, les jeux par navigateur, durant mon travail de Bachelor, et ainsi apporter une solution aux différentes problématiques rencontrées durant mes expériences.

Le fait d'avoir pu être encadré durant la réalisation de ce prototype m'a permis de prendre du recul sur la finalité souhaitée, sur les pistes à explorer, et m'a contraint à me questionner plus qu'à mon habitude avant de m'atteler au développement.

A titre personnel, je pense continuer l'avancement du prototype, pour en trouver les nouvelles limites, et y apporter, je l'espère, de nouvelles solutions.

Bibliographie

Certaines citations présentes dans ce travail proviennent des documents suivants :

« Prélude ». *Un logiciel pour créer vos jeux automatiquement*. In : Forum Jeux-Web.com [forum]. 5 mars 2013 http://forum.jeux-web.com/index.php?topic=283.msg1196#msg1196 (consulté le 9 juin 2013)

« Kalan ». *Re : Un logiciel pour créer vos jeux automatiquement*. In : Forum Jeux-Web.com [forum]. 22 mars 2013 http://forum.jeux-web.com/index.php?topic=283.msg1227#msg1227 (consulté le 9 juin 2013)

comScore, *comScore Reports April 2013 U.S. Smartphone Subscriber Market Share* [en ligne]. 4 juin 2013 http://www.comscore.com/fre/Insights/Press_Releases/2013/6/comScore_Reports_April_2013_U.S._Smartphone_Subscriber_Market_Share (consulté le 9 juin 2013)

Sapientis. *modernisation des SI et maturité des entreprises* [en ligne]. 2011 http://www.sapientis.fr/conseil/wp-content/uploads/2011/04/extrait1budget.pdf (consulté le 9 juin 2013)

Google Analytics. *Rapports* [en ligne]. 2013 http://www.google.com/analytics/ (consulté le 9 juin 2013)

Travian Games. *Products* [en ligne]. 2013 http://www.traviangames.com/en/products.html (consulté le 9 juin 2013)

Gameforge. *A Company on the Road to Success* [en ligne]. 2012 http://corporate.gameforge.com/en/company/history/ (consulté le 9 juin 2013)

JDN. *E-commerce : Magento et PrestaShop gagnent en popularité* [en ligne]. 7 novembre 2012 http://www.journaldunet.com/solutions/saas-logiciel/parts-de-marche-logiciels-e-commerce-octobre-2012-1112.shtml (consulté le 9 juin 2013)

Wikipedia. *osCommerce* [en ligne]. 27 avril 2013 http://en.wikipedia.org/wiki/OsCommerce (consulté le 9 juin 2013)

Megentite. *Magento database diagram table list* [en ligne]. 22 mai 2010 http://www.magentite.com/221/magento-shopping-cart/magento-database-diagram-table-list/ (consulté le 9 juin 2013)

DaveEgerton. *Prestashop database schema* [en ligne]. http://www.daveegerton.com/prestashop-guides/Prestashop-Developers-Guide.html (consulté le 9 juin 2013)

LAROUSSE (éd.). *Le Petit Larousse Illustré*. Paris : Larousse / HER, 2000, ISBN 2-03-530201-3

Annexe 1 : Variations.class.php

```php
<?php
class Variations {

	protected static $variations = null ;

	// Load for the first time
	function __construct(){
		if(self::$variations == null)
			$this->loadVariations();
	}

	// Loading function
	protected static function loadVariations() {
		self::$variations = array();
		$sql = "SELECT * FROM variations ORDER BY var_id ASC" ;
		$r = query($sql) ;
		while($d = $r->fetch_assoc()){
			self::$variations[$d['var_id']] = array(
				'parent'=>$d['var_parent'],
				'name'=>$d['var_name']
			) ;
		}
	}

	// Get a specific variation
	public static function getVariationById($id){
		if(isset(self::$variations[$id]))
			return self::$variations[$id] ;

		return null ;
	}

	// Boolean, if is root of the tree
	public static function isRoot($id){
		return (count(self::getParents($id)) == 1) ;
	}

	// Get self + parents
	public static function getParents($id,$list=array()){

		$a = self::getVariationById($id) ;

		if($a == null)
			return $list ;

		$list[$id] = $a ;

		return self::getParents(
			self::$variations[$id]['parent'],$list
		) ;
	}

	// Get parents + self keys
	public static function getParentsKeys($id){
		return array_keys(self::getParents($id)) ;
	}
```

```
	// Get childs, but not self
	public static function getChildren($id,$list=array()){
		foreach(self::$variations AS $idChild=>$child){
			if($child['parent'] == $id){
				$list[$idChild] = $child ;
				$list = self::getChildren($idChild,$list) ;
			}
		}
		return $list ;
	}

	// Get childs keys
	public static function getChildrenKeys($id){
		return array_keys(self::getChildren($id)) ;
	}


	// Get keys of self + current ID
	public static function getChildrenID($root){
		return array_merge(
			array($root),self::getChildrenKeys($root)
		) ;
	}

	// Get specific name
	public static function getVariationName($id){
		$n = self::getVariationById($id) ;
		if($n != null)
			return $n['name'] ;

		return null ;

	}

	// Get a name for all the parents (hierarchy)
	public static function getParentsName($id) {
		$parents = array_reverse(self::getParentsKeys($id)) ;
		$n = array() ;
		foreach($parents AS $idC)
			$n[] = self::getVariationName($idC) ;

		return implode(' / ',$n) ;
	}
}
?>
```

Annexe 2 : Parameter.class.php

```php
<?php
abstract class Parameter {

	// Linked to a variation
	private $variationId = 0 ;

	// ID of the current parameter (0 when not created)

	private $parameterId = 0 ;

	// Database informations
	private $tableName = null ;
	private $tablePrefix = null ;

	private function __construct() {
		// NOTHING
	}

	// Set database informations
	protected function setTableName($tableName,$tablePrefix){
		$this->tableName = $tableName ;
		$this->tablePrefix = $tablePrefix ;
	}

	// Link to a variation
	public function setVariation($id){
		$this->variationId = $id ;
	}

	// Save instance
	public function saveInstance(){

		// Variation must be declared
		if(! $this->variationId)
			throw new Exception('Use setVariation( first');

		// Table must be declared
		if(! $this->tableName)
			throw new Exception('Use setTableName first');

		// Save instance
		$this->saveParameterInstance() ;
	}

	// Generate parameter instance, and save ID
	private function saveParameterInstance(){
		query("
			INSERT INTO
			parameters(par_table,par_var_id)
			VALUES(
				'". $this->tableName ."',
				'". $this->variationId ."'
			)"
		) ;

		$c = getDBLink() ;
```

```
            $this->parameterId = $c->insert_id ;
            $this->saveSpecificInstance() ;
      }

      // Save a new instance of the current class
      private function saveSpecificInstance(){

            query("
                  INSERT INTO
                  ". $this->tableName ."(". $this->tablePrefix ."_id)
                  VALUES(".$this->parameterId .")"
            ) ;

            $this->udpateInstance() ;
      }

      // Update instance (redeclare it when extends this class)
      protected function udpateInstance(){
            // NOTHING FOR THE MOMENT
      }

      // Generate and execute query to update fields on the current
      instance
      protected function queryUpdateInstance($fields){

            $sets = array() ;
            foreach($fields AS $k => $v)
                  $sets[] =
                        $this->tablePrefix.'_'.$k."='".secSQL($v)."'" ;
            if(count($sets)){
                  $sql =
                        "UPDATE ". $this->tableName .
                        " SET ".implode(",",    $sets).
                        " WHERE ". $this->tablePrefix ."_id = ".
                        $this->parameterId ;
                  query($sql) ;
            }
      }

      // Delete a specific instance from the defined table and the
      parameters table
      public function deleteOtherInstance($id) {
            query("DELETE FROM parameters WHERE par_id = ".$id) ;
            query("DELETE FROM ". $this->tableName .
                  " WHERE ". $this->tablePrefix ."_id = ".$id) ;
      }

}
?>
```

Annexe 3 : ParameterManager.class.php

```php
<?php
class ParameterManager {

	// Tables related to parameters
	private static function setTables(){
		self::addTable("actions","act"); // Actions
		self::addTable("characteristics","cha"); // Characteristics
		self::addTable("configurations","con"); // Configurations
	}

	// Variations Manager
	private static $variations = null ;

	// Tables names and prefix
	private static $tables = null ;

	// Current variation
	private $finalId ;
	private $variationsIDs ;
	private $parameters = null ;

	// Constructor
	function __construct(){
		self::setTables();
		self::$variations = new Variations() ;
		$this->parameters = array() ;
	}

	// Only load one time
	static function addTable($name,$prefix){
		if(self::$tables == null)
			self::$tables = array() ;
		self::$tables[$name] = $prefix ;
	}

	// Set the variation id for the parameters
	public function setFinalId($id){
		$this->finalId = $id ;
		$this->loadParameters() ;
	}

	// Call the load functions
	private function loadParameters(){
		$this->loadVariationsIDs();
		$this->loadParametersFromBase();
	}

	// Get the parents IDs (included self id)
	private function loadVariationsIDs(){
		$this->variationsIDs =
		self::$variations->getParentsKeys($this->finalId);
	}
```

```php
// Generate query
private function constructQuery(){
	// JOIN ALL THE PARAMETERS TABLE
	$joins = array() ;
	foreach(self::$tables AS $tName => $tPrefix)
		$joins[] =
			"LEFT JOIN $tName ON ${tPrefix}_id = par_id" ;
	$joins = implode("\n",$joins) ;

	return "SELECT * FROM parameters
			$joins WHERE par_var_id IN
			(". implode(',',$this->variationsIDs) .")" ;
}

// Load from table
private function loadParametersFromBase(){
	$sql = $this->constructQuery() ;
	$r = query($sql) ;
	$this->treatBaseResults($r) ;
}

// Treat the results
private function treatBaseResults($r){

	// For each parameter
	while($d = $r->fetch_assoc()){

		// Get table name, and if is a table from the Manager
		$table = $d['par_table'] ;
		if(isset(self::$tables[$table])){

			// Init filds to use
			$fields = array() ;
			// Get table prefix
			$prefix = self::$tables[$table];
			$regex = '#(^'.$prefix.'_)#i' ;

			// Each field from the current parameter
			foreach($d AS $name => $value){
				// Check if the field is from the table
				or from the parameter
				if(preg_match($regex,$name)){
					$fields[preg_replace(
						$regex,'',$name
					)] = $value ;
				}elseif(preg_match('#par_#i',$name)){
					$fields['parameter'][$name] =
						$value ;
				}
			}

			// If doesn exists, create it
			if(! isset($this->parameters[$table]))
				$this->parameters[$table] = array() ;

			// Add a row in the table of this parameters
			$this->parameters[$table][] = $fields ;
		}
```

```
            }
        }

        // Get parameters table from a specific type
        public function getParametersOf($name){
            if(! isset($this->parameters[$name]))
                return array() ;
            return $this->parameters[$name] ;
        }

}
?>
```

Annexe 4 : gameway_web.js

```
(function($){

	$.fn.gamewayGame = function(options){

		// Default parameters
		var defauts = {
			'key':'',
			'urlAPI':'http://gameway.me/api/',
			'loadingText':'Loading...',
			'callbackVersion':null,
			'changePageTitle':true,
		};

		// Add options to parameters
		var parameters = $.extend(defauts, options);
		return this.each(function()
		{

/* COMPONENT CODE */

	// Main component
	var $element = $(this) ;

	// Blocks
	var $blockGame ;
	var $blockContent ;
	var $blockInformations ;

	// Variables
	var $gameTitle = "unknown game" ;

	// Display loader
	function newLoad(){
		$blockLoading = $('<div>').addClass("gw-loading")
		.append(parameters.loadingText) ;
	}

	// Add the key to the data
	function appendKey(data){
		$.extend(data,{'key':parameters.key});
		return data ;
	}

	// Init blocks
	function initBlocks(){
		$blockGame = $('<div>').addClass("gw-container") ;
		$blockHeader = $('<div>').addClass("gw-header") ;
		$blockContent = $('<div>').addClass("gw-content") ;
		$blockInformations = $('<div>')
			.addClass("gw-informations");
		$element.html("") ;
		$element.append($blockGame);
		$blockGame.append($blockHeader)
			.append($blockInformations)
			.append($blockContent) ;
	}
```

```
// Return a text block
function textBlock(text){
	return $('<div>').addClass("gw-textblock")
			.addClass("gw-maintext").append(text) ;
}

// Display game from JSON
function displayGame(d){

	APIInformations(d.api);
	initBlocks() ;

	$gameTitle = d.game.name ;
	$blockHeader.append($gameTitle) ;

	if(parameters.changePageTitle)
		document.title = $gameTitle ;

	if(d.text)
		$blockContent.append(textBlock(d.text)) ;

	if(d.actions)
		displayActions(d.actions) ;

	if(d.characteristics)
		displayCharacteristics(d.characteristics) ;

}

// Call the function, if exists, to display the version
function APIInformations(d){
	if(parameters.callbackVersion)
		parameters.callbackVersion(
			d.version,
			d.updated,
			d.date
		);
}

// Parse JSON when called back
function parseJSON(json){
	if(json.error){
		alert(json.error);
		 // If game not found
		if(json.error == "GAME_NOT_FOUND") return ;
	}else{
		displayGame(json);
	}
	return ;
}
```

```
	function loadJSON(data){

		// Display loading text
		newLoad();

		// Add game key to the datas and call json
		data = appendKey(data) ;
		$.ajax({
			dataType: "json",
			url: parameters.urlAPI,
			data: data,
			type:"get",
			success: parseJSON
		});
	}

	// Init component
	function init(){

		// First load only with key
		loadJSON({}) ;
	}

/* SPECIFIC METHODS */

	/* CHARACTERISTICS */
	function displayCharacteristics(characteristics){

		$blockInformations.html() ;
		$chaContent = $('<ul>').addClass('gw-characteristics') ;

		$.each(characteristics, function(idCha,cha) {
			$blockCha = $('<li>').addClass('gw-characteristic') ;
			$blockCha
				.append($('<span>')
				.addClass('gw-characteristic-name')
				.append(cha.name)) ;
			$blockCha
				.append(' ').append($('<span>')
				.addClass('gw-characteristic-value')
				.append(cha.value)) ;
			$chaContent.append($blockCha) ;
		});

		$blockInformations.append($chaContent) ;

	}
```

```
	/* ACTIONS */
	function displayActions(actions){

		// Init block
		$blockActions = $('<div>').addClass("gw-actions");
		$blockContent.append($blockActions) ;

		// For each action
		$.each(actions, function(idAction,action) {

			$blockAction = $('<div>').addClass("gw-action") ;
			$blockActions.append($blockAction) ;
			$blockAction.append($('<div>')
				.addClass("gw-action-title")
				.append(action.title)) ;

			// Add buttons
			if(action.buttons){
				$blockButtons = $('<div>')
					.addClass("gw-action-buttons");

				$.each(action.buttons,
					function(idButton,button) {
						$blockButton = $('<div>').addClass(
							"gw-action-button"
						) ;
						$blockButton.append(button.text) ;
						$blockButton.click(function(a,b){
						loadJSON(
							{'doAction':button.value}
						);
					});

					$blockButtons.append($blockButton) ;

				});

				$blockAction.append($blockButtons) ;

			}
			// If there is a text
			if(action.text)
				$blockAction.append($('<div>')
					.addClass("gw-action-text")
					.append(action.text)) ;

		});
	}

/* / COMPONENT CODE */

			init() ; // AUTO INIT WHEN CALLED

		});
	};
})(jQuery);
```