RAPPORT
DE
PROJET PHP

Présenté et soutenu le 09 juin 2006 à l'Institut Charles Cros, Serris

Jonathan **BABY**
Julie **BILLARD**
Thomas **BOURGOIN**
François **BRAUD**
Catherine **CHANTHAVONG**
Sandra **DESROUSSEAUX**
Clément **DONY**
Séverine **FAYOLLE**
Anne-Laure **FOUQUE**
Stéphanie **JUILLARD**
Lucie **MOINEAU**

Tuteur enseignant : **M. Patrice BOUVIER**
M. Olivier DERPIERRE

IMAC – Promotion 2007
Institut Charles Cros
Université de Marne-la-Vallée

RAPPORT
DE
PROJET PHP

Présenté et soutenu le 09 juin 2006 à l'Institut Charles Cros, Serris

Jonathan **BABY**
Julie **BILLARD**
Thomas **BOURGOIN**
François **BRAUD**
Catherine **CHANTHAVONG**
Sandra **DESROUSSEAUX**
Clément **DONY**
Séverine **FAYOLLE**
Anne-Laure **FOUQUE**
Stéphanie **JUILLARD**
Lucie **MOINEAU**

Tuteur enseignant : **M. Patrice BOUVIER**
M. Olivier DERPIERRE

IMAC – Promotion 2007
Institut Charles Cros
Université de Marne-la-Vallée

Remerciements

Nous remercions Patrice Bouvier et Olivier Derpierre, plus connu sous le nom de Bud, de nous avoir donné les enseignements nécessaires à la réalisation du projet, d'avoir été à notre écoute et de nous avoir donné de leur temps .

Nous remercions aussi les deux autres groupes avec qui nous avons partagé ce projet.

Remerciement spécial au CRI qui nous a bien dépanné lorsque le serveur etud a malencontreusement planté.

Nous remercions aussi tous nos enseignants qui nous ont aidé lors de ce projet.

Table des matières

Introduction

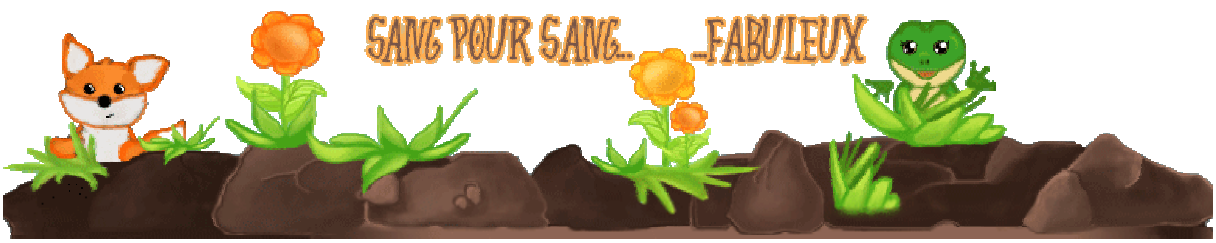
Dans le cadre du cours de développement Php donné par M. Bouvier et M. Derpierre, un projet de jeu multi joueurs, accessible via une interface Web ergonomique, graphiquement attractive, et conforme aux normes XHTML1.0 strict et CSS1.0, a été développé.

L'histoire est la suivante. Dans un monde lointain et merveilleux, des chercheurs testent virus et vaccins qui donnent parfois naissance à des bizarreries de la nature. Ces derniers, étranges êtres génétiquement modifiés, vivent paisiblement jusqu'au jour où certains d'entre eux s'organisent au sein du collectif ADH (Anti Domination Humaine) à la tête duquel règne Elcaro, un rat de laboratoire. Grâce à un plan machiavélique, les membres de ce mouvement parviennent à prendre la clef des champs et se font alors appeler les « *Mimichants* ».

En effet, la légende dit que certains sont mignons, d'autres méchants, alors que d'autres encore cachent bien leur jeu…Ainsi, le caractère et les capacités psychiques et physiques de chacun sont bien distincts.

Cette communauté a pour monnaie d'échange le Chocopépite et ses membres passent le plus clair de leur temps à (gentiment ?) se battre et s'envoyer en l'air au sens propre comme au sens figuré.

Malgré leur fuite, les *Mimichants* restent attachés à la race humaine, et ne seraient pas contre leur faire partager leur jeux…La règle est la suivante : En tant que joueur humain, vous êtes à la tête d'un groupe de *Mimichants*. Votre but sera d'être le plus influent de tous les joueurs inscrits. Pour cela, il vous faudra toujours plus de terrain, de *Chocopépites* et de *Mimichants* sous vos ordres…



I. Cahier des charges

1. Idée de départ

Le but du projet est de « créer un jeu en ligne multijoueurs, tour par tour, points/actions ou autres, se déroulant dans un monde utilisant une carte ». Nous avions entière liberté sur le but du jeu et sur les règles de celui-ci. Les règles ont donc été fixées par trois représentants de chaque groupe.

Voici le but du jeu défini par ces représentants :

En tant que joueur, vous dirigez un ou plusieurs *Mimichants*. Pour obtenir plusieurs *Mimichants*, plusieurs possibilités : en recruter des tout prêts dans un terrier, mordre des animaux normaux pour les faire muter ou copuler avec un *Mimichant* de sexe opposé.
Le nombre maximum de *Mimichants* sous vos ordres dépend directement de votre espace vital qui est augmentable en prenant possession de plus de terrain. Vous pouvez ainsi devenir plus influent, plus riche, plus fort et de plus en plus gentil (la gentillesse n'étant pas perçue de la même façon chez les *Mimichants* que chez les humains).
En fonction de votre stratégie, vous pouvez récolter plus ou moins rapidement des *Chocopépites* qui permettent de monnayer l'équipement et des Mimichants.

2. Les contraintes à respecter

Lors de la création des règles du jeu, le groupe « création » a dû respecter certaines conditions imposées par le sujet du projet. Ces contraintes concernaient tout d'abord la carte tactique, élément clé du jeu, qui devait être représentée en miniature et créée dynamiquement à l'aide de la bibliothèque GD.

Le projet devait aussi contenir plusieurs modules obligatoires comme la messagerie interne, le système de gestion des personnages non-joueurs, un système d'annonces, de back-up, ainsi qu'un système de suppression automatique des comptes inactifs.

Concernant les parties du site, certaines étaient obligatoires comme la page de création de compte, une page de login (sécurisé), une page de résumé concernant les évènements survenus depuis la dernière connexion de l'utilisateur, une page de description de l'utilisateur et des personnages, une aide en ligne, une FAQ…

Ces contraintes nous ont permis de définir clairement des modules (Php ou simple XHTML) que nous nous sommes ensuite répartis dans le groupe.

3. Retro planning

Voir annexe.

II. Organisation du projet

1. Organisation du groupe

a) Contraintes de travail

Composition du groupe

La classe a été scindée en trois groupes de onze personnes. Ce nombre, ainsi que les personnes constituant chaque groupe, a été fixé par les deux professeurs encadrant le projet : M. Bouvier, et M. Derpierre.

En réalité, il s'agissait du premier projet, dans le cadre de notre scolarité à l'IMAC, où nous devions composer avec un groupe d'une telle envergure, et dont nous n'avions pas le choix des membres.

Les contraintes présentes dans le sujet

Le projet comporte donc une importante composante de gestion de groupe et d'organisation générale du travail, d'autant que certaines contraintes étaient fixées par le sujet. En effet, chaque membre du groupe devait participer au développement en Php et une page Web devait permettre le suivi du projet.

b) Les spécificités de notre situation

Chaque groupe a dû s'organiser en fonction des membres qui le composent. En ce qui nous concerne, nous possédions quelques spécificités.

En effet, certains des membres connaissaient et maîtrisaient déjà le langage Php avant son enseignement cette année à l'IMAC. Ils étaient donc relativement à l'aise dans ce domaine, ce qui a constitué une force.

Ensuite, Julie étant absente pour des raisons de santé, elle a dû effectuer du télétravail et nous avons donc dû faire un effort particulier dans le domaine de la communication au sein groupe.

Enfin, Thomas Bourgoin, mis à part une présence lors des deux premières réunions, n'est plus réapparu par la suite, devenant par là même un membre inactif de notre groupe.

c) Notre fonctionnement

Répartition du travail

- *Quantité*

Globalement, même si elle est difficile à mesurer, la quantité de travail effectuée par chacun des membres du groupe a été relativement équivalente. Toutefois, elle a été répartie différemment au cours du temps pour chacun, car des contraintes extérieures comme les

projets tutorés, donnant lieu à des rendus à des dates différentes, sont intervenues. De plus, la participation au groupe de base de données voire à celui de définition des règles du jeu a accru le travail de certains membres à des périodes données.

Pour finir sur l'aspect quantitatif de notre travail, il semblerait que Julie ait davantage travaillé que les autres membres. Elle fut un véritable moteur pour l'avancée du site.

- *Contenu*

Nous avons respecté la contrainte du projet selon laquelle chaque membre du groupe devait développer un module en Php. Ainsi, chacun a eu en charge un ou plusieurs des développements suivants : la conception et la construction de la base de données, la création des pages Php implémentant les requêtes ou opérations nécessaires à la liaison entre notre site et la base de données, ou l'écriture des procédures.

Le reste du travail a effectuer ne relevant pas directement du Php a été réparti selon les désirs, les acquis et les compétences de chacun. Nous avions l'avantage d'avoir chacun des profils assez divers, et chacun a donc pu prendre un module qui lui plaisait, et cela sans conflit interne. Ainsi, les profils plus « graphisme » (Julie, Séverine) ont pu s'exprimer lors de la création du site et des différents visuels. Les profils plus « conception et développement » (Jonathan, Clément) ont eux pu travailler sur la carte ou l'implémentation de technologies comme le httprequest. Ceux possédant des profils « mixtes » (Catherine, Sandra, Stéphanie, Lucie, Anne-Laure, François, Séverine) ou souhaitant travailler sur plusieurs modules successifs ont participé au développement des interfaces d'échange ou de la messagerie.

Lieu et groupes de travail

Tous les membres du groupe ne logeant pas à proximité de l'université et étant donné que nous étions amenés à avancer notre travail le soir, environ la moitié du développement s'est effectué dans les locaux de l'institut, et l'autre moitié au domicile de chacun.

Ceux du groupe se sentant plus à l'aise pour développer à deux ont pu le faire, notamment Stéphanie et Lucie ou Sandra et Catherine. Séverine, Anne-Laure et François ont davantage développé de façon autonome. Toutefois, ils ont été présents lors de l'intégration de leur module à ceux des autres. Jonathan et Clément, eux, ont travaillé en autonome tout en s'aidant et se conseillant mutuellement.

Julie, elle, a développé chez elle, mais était en dialogue avec chacun des membres du groupe, avec qui elle partageait certaines tâches comme la graphisme ou l'affichage des informations sur l'utilisateur ou le *Mimichant*. Elle a notamment beaucoup travaillé à distance avec François.

Suivi des avancées de chacun

Le travail à onze a nécessité une organisation et une communication efficaces.

- *Les réunions*

Pour cela, nous effectuions des réunions hebdomadaires. Etant prévues à un moment invariant de la semaine, elles permettaient d'organiser nos projets ou activités annexes en conséquence. Ainsi, à chaque réunion, la quasi-totalité des membres était présente.

Les réunions duraient environ deux heures. En début de séance, un tour de table était fait concernant les avancées et ce qui restait à faire pour chacun des membres. Chacun

expliquait également les problèmes auxquels il se heurtait en vue de trouver une solution à plusieurs, ou lorsque l'accord de tous les membres du groupe était nécessaire à une décision. En fin de séance, nous fixions les tâches à réaliser d'ici la prochaine séance, et plus globalement d'ici la fin du projet, afin de se donner un aperçu général de notre progression.

Chaque réunion donnait lieu à un compte-rendu rédigé par Stéphanie qui prenait des notes durant nos séances. Ainsi, Julie pouvait obtenir un détail des points qui avaient été abordés, ainsi que des responsabilités de chacun, ce qui permettait à tous de s'adresser à la bonne personne en cas d'interrogation sur un des points du projet. Parfois, nous téléphonions à Julie afin de savoir si le module qui lui a été attribué lui convenait. Souvent, au cours des séances, nous répondions également collectivement à ses mails posant des questions sur le projet.

- *La « communication numérique »*

Stéphanie et Lucie ont créé un blog dont l'adresse est la suivante : http://lesptitsmalins.free.fr/

Cela a permis de tenir au courant les membres du groupe ainsi que les professeurs des avancées de chacun, et ceci de façon détaillée. En effet, les comptes rendus de réunions étaient plus concis. De plus, à l'aide du blog, nous pouvions retrouver le post qui nous intéressait, sans parcourir l'intégralité de notre boîte mail, ce qui a constitué un gain de temps appréciable…

d) Analyse de notre gestion du groupe : points positifs et négatifs, leçons pour l'avenir.

<u>Communication au sein du groupe :</u>

- *Points positifs*

Globalement, la communication ne fut pas le point faible de notre groupe. Nous sommes tous capables de dire avec précision qui fut en charge de quelle tâche.

En effet, les réunions régulières ainsi que la rédaction de compte-rendus de fin de séance et l'utilisation active du blog ont permis une bonne visibilité du travail de chacun et des avancées du groupe.

- *Points négatifs*

La communication par mail a bien fonctionné, même si l'envoi au groupe a parfois posé problème : en effet, face à la réception de dizaines de messages journaliers, certains restaient sans réponse. Heureusement, les réunions permettaient d'aborder les problèmes non résolus.

Par ailleurs, lors des réunions, des conflits ont parfois eu lieu. Il aurait sûrement été utile d'élire un « chef de projet » chargé de recadrer les discussions quand il le fallait, et de décider en dernier lieu en cas de problème. Nos réunions auraient sûrement été plus efficaces de par leur gestion du temps, et moins conflictuelles.

Gestion du télétravail :

- *De notre coté*

Travailler avec Julie ne fut en aucun cas une contrainte. En effet, cela a obligé à la mise en place de comptes-rendus et d'une communication efficace, ce qui fut profitable à l'intégralité du groupe. De plus, Julie nous communiquait très régulièrement ses avancées. Ainsi, nous n'avions par l'impression de travailler à distance, car elle fut extrêmement présente sur le projet.

- *Du côté de Julie*

« De mon coté je trouve que tout s'est plutôt bien passé et aucun point négatif n'est réellement ressorti de ce travail à distance forcé. La communication s'est bien passée même si quelques personnes du groupe ne répondaient pas tout le temps à mes mails. En effet le mail était pour moi le moyen le plus rapide de poser des questions et de montrer mon travail et j'en envoyais donc beaucoup. La mise en place du blog m'a été très profitable car j'ai pu mettre plus facilement à disposition des autres mon travail. Le travail avec François lors de l'élaboration du début du site s'est très bien passé et nous avons réussi à nous relayer efficacement sans perdre de temps. »

2. Déroulement de la création des règles

La définition des règles du jeu devait se réaliser au sein d'une équipe affectée de trois membres de chaque groupe sous forme d'une ou plusieurs réunions. Ainsi, l'assemblée fut composée de Jonathan Baby, François Braud, Flavien Clermont, Clément Dony, Alexis Granger, Alexis Martin, Thomas Prigent, Antoine Trébuchet et Arnaud Trouve.

Contenu des réunions

Il y eut plusieurs réunions. La première permit de définir les grandes caractéristiques du jeu à savoir être un jeu de gestion de personnages sur des territoires capturables. Le jeu comporte un plateau sur lequel les personnages peuvent se déplacer librement. Nous avions également défini l'univers du jeu, c'est-à-dire un univers de guerriers tout en étant gentil, joli et politiquement incorrect. Un monde du genre des « Happy Tree Friends » fut par conséquent imaginé pour ce jeu. Afin de pouvoir donner cours à notre imagination, nous avons considéré les personnages comme des animaux génétiquement modifiés pouvant ainsi faire à peu près tout ce que l'on voulait. Les deux réunions suivantes nous ont permis de peaufiner les règles, de définir les caractéristiques des personnages, leurs actions, …

Ambiance et communication au sein du groupe

La définition des règles du jeu était plutôt conflictuelle, car chaque groupe était représenté et voulait plus ou moins imposer les informations récoltées parmi ses membres. De plus, la mise en entente de neuf individus est une chose plutôt délicate, surtout quand l'équipe est composée de personnes de caractère, avec des expériences individuelles des jeux, qu'ils soient informatiques ou de société, et une vision personnelle de l'avenir du jeu. Il y eut d'ailleurs l'intervention de personnes extérieures afin de trancher sur certaines décisions difficiles et permettant ainsi de continuer l'avancement des règles.

Analyse du travail effectué

Aux vues des problèmes encourus par la suite, il semble que les règles ne furent pas toujours assez détaillées, certaines étant mentionnées sans plus d'explications. Cela devait

permettre de diminuer le travail d'harmonisation des valeurs de jeu, mais s'est vite transformé en défaut qui a porté préjudice à la conception et l'utilisation de la base de données. Les règles n'ont donc pas permis de cadrer entièrement la conception mais toutefois suffisamment pour que les problèmes en résultant soient corrigés rapidement. Enfin, les règles du jeu étaient très complexes et peut-être trop ambitieuses si l'on considère le temps qui nous était imparti pour la réalisation du projet.

3. Déroulement de la création de la Base De Données

Lorsque les règles ont été validées, nous avons de nouveau réuni les 3 groupes afin de créer la base de données du jeu. Le groupe « Base de données » était composé de Marie, Flavien, Nadia, Caroline, Nicolas, Maya, Jana, Catherine, Stéphanie et Lucie.

Les premières réunions pour la création de la base de données ont été très efficaces, nous avons repris les règles du jeu dans l'ordre pour être sur de ne pas oublier de détails. Les règles étaient assez compliquées à comprendre, heureusement Flavien qui était déjà aux réunions « règles du jeu » a assisté à toutes les réunions et a permis de nous éclairer sur les règles.

Lorsque la base a été créée, il ne nous restait plus qu'à lister les procédures afin de les répartir entre les groupes. Nous nous sommes alors rendus compte que nous ne savions pas exactement à quoi servait une procédure. Nous avons perdu du temps à essayer de comprendre ce qu'on pouvait considérer comme procédure. Pour finir, nous avons demandé à faire une mise au point avec M. Bouvier qui nous a remis de la documentation et qui nous a fait un topo sur les procédures. Cette mise au point nous a permis de lister les procédures à développer.

Ces procédures ont ensuite été réparties entre les 3 groupes. Il y avait une vingtaine de « thèmes » de procédures à créer. La répartition a été la plus équitable possible.

III. La gestion du site

1. La partie graphique du site

a) La charte graphique

Les *Mimichants* sont de petits animaux qui vivent sur et sous terre. Il nous paraissait donc logique que l'environnement du site ressemble au lieu de vie des *Mimichants*, c'est-à-dire la terre, l'herbe et le sous-sol et le ciel.

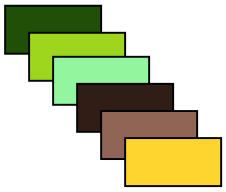
D'autre part pour rendre le site plus vivant nous avons ajouté des petits *Mimichants* dessinés un peu partout dans les pages du site. Leurs traits sont souvent malicieux et heureux car même si leur but est de s'entretuer, nous avons tout de même voulu rendre ce jeu plus humoristique que cruel. Ce choix apparaît d'ailleurs dans tous les choix graphiques que nous avons faits que cela soit au niveau du design du site, des couleurs, des *Mimichants* ou des armes.

b) Les couleurs

Les couleurs dominantes du site seront donc en concordance avec l'univers des *Mimichants*. Nous avons donc choisi d'utiliser en majorité deux teintes de marron, et trois teintes de vert. Pour rendre le tout plus lumineux nous avons ajouté des pointes de jaune orangé pour les fleurs et du bleu pour représenter le ciel.

Afin de toujours garder les mêmes codes de couleurs, nous les avons faits figurer dans le fichier css.

#204F08==>vert foncé
#9ED61D ==>vert
#BCF59F==>vert clair
#301D16 ==>marron foncé
#8F6555 ==>marron clair
#FED52E ==>orangé



- L'orangé est utilisé pour les textes du corps du site. Par exemple la page de connexion, les dernières actions pendant l'absence du joueur et la gestion du compte sont en orange sur fond marron clair.

- Par contre, pour les pages du terrier, nous avons décidé d'utiliser le vert pour bien marquer que l'on accède ici à une partie « jeu » du site.

- De même les éléments de la carte sont composés en majorité de vert (menu de navigation, le bandeau et les onglets), pour la même raison.

- Les menus de gauche sont eux en vert et les liens sont représentés par des feuilles. Par contre les liens pour les services sont différents pour encore une fois marquer l'accès au « jeu »

Bandeau de gauche, icônes services, icônes menu principal

c) La structure

La taille est celle basique de tout site, c'est-à-dire 780 pixels de large, ceci afin de pouvoir répondre aux exigences des utilisateurs qui ont encore des écrans de 800 pixels. Le

site est constitué d'une entête (179 pixels de hauteur) avec le titre du jeu et quelques *Mimichants*, d'un menu à gauche (210 pixels de large), un corps et enfin un pied (40 pixels de hauteur).

Nous avons dû à plusieurs reprises réduire le menu de gauche car nous l'avions prévu trop grand et la carte ne logeait pas dans le corps du site. Pour la même raison, nous avons décidé d'enlever le bandeau de fond et le menu de gauche lorsqu'on accède à la carte de jeu et de les remplacer par un menu de navigation à gauche et par un fond de couleur marron foncé. Ces changements sont intervenus tard dans la construction du site et longtemps après que la charte graphique ait été fixée. Cela nous a obligé à modifier beaucoup de critères de construction et nous avons perdu du temps à cause de cela.

d) Les *Mimichants*

L'élaboration des *Mimichants* a été une des premières choses qui a été réalisée. Nous avions déjà dans l'idée de faire un jeu avec un visuel mignon et charmant comparé aux règles du jeu qui sont barbares.

Nous avons décidé de nous limiter à huit *Mimichants* différents plus un personnage utilisé pour les quêtes. Ils sont réalisés sous Illustrator pour avoir un trait net et ensuite pouvoir gérer facilement la transparence. On les détoure ensuite sous Photoshop.

Les règles du jeu nous imposent de pouvoir choisir la couleur du corps du *Mimichant* et de pouvoir changer les éléments du corps comme on le veut. Ainsi on doit pouvoir se retrouver avec un *Mimichant* avec un corps de renard, des oreilles de lapin et des pattes de grenouille et une queue de singe.

Les Mimichants

Dans un premier temps nous avons fait pour chaque *Mimichant* et pour chaque partie du corps 3 couleurs différentes. Les couleurs étaient donc limitées et cela ne correspondait pas à la demande des règles du jeu.

Par la suite nous avons adapté le code réalisé pour qu'il réponde aux exigences des autres groupes. Le deuxième code permet donc de sélectionner la couleur du pelage du *Mimichant* et les parties du corps.

Lors de la création d'un compte jeu, l'utilisateur doit créer le premier *Mimichant* qu'il incarnera. Un *Mimichant* se décompose en quatre parties : la queue, le corps, la tête et les oreilles.

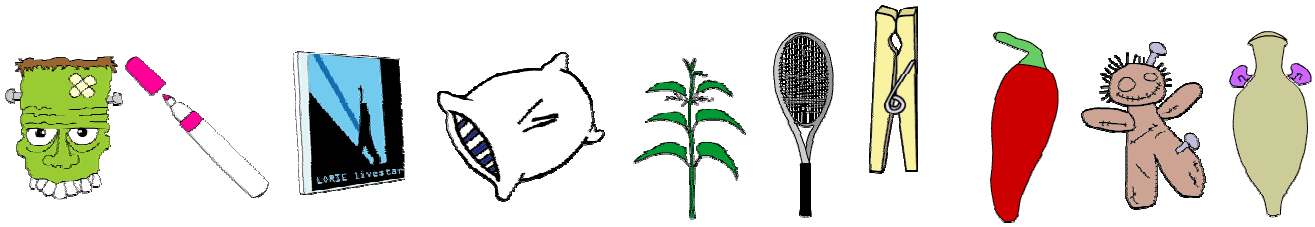
Il possède également d'une couleur de pelage permettant de le différencier de ces congénères. Etant donné que nous utilisions de la technologie HTTPRequest, il nous fallait trouver une méthode permettant
de visualiser l'aspect du *Mimichant* sans jamais réactualiser la page web.

Création du Mimichant

Nous avons donc décidé de ne pas afficher le *Mimichant* final avec la libGD, mais plutôt la superposition d'images transparentes représentant chaque partie du corps et d'un fond de couleur. Chaque partie, ainsi que la couleur (que l'utilisateur peut choisir grâce à la palette couleur), peuvent être modifiées avec l'aide d'un script javascript. Au final, l'affichage du *Mimichant* peut paraître quelquefois peu esthétique mais est toute fois la méthode la plus optimisée et est en concordance avec la politique définie par le groupe pour ce projet.

e) Les armes

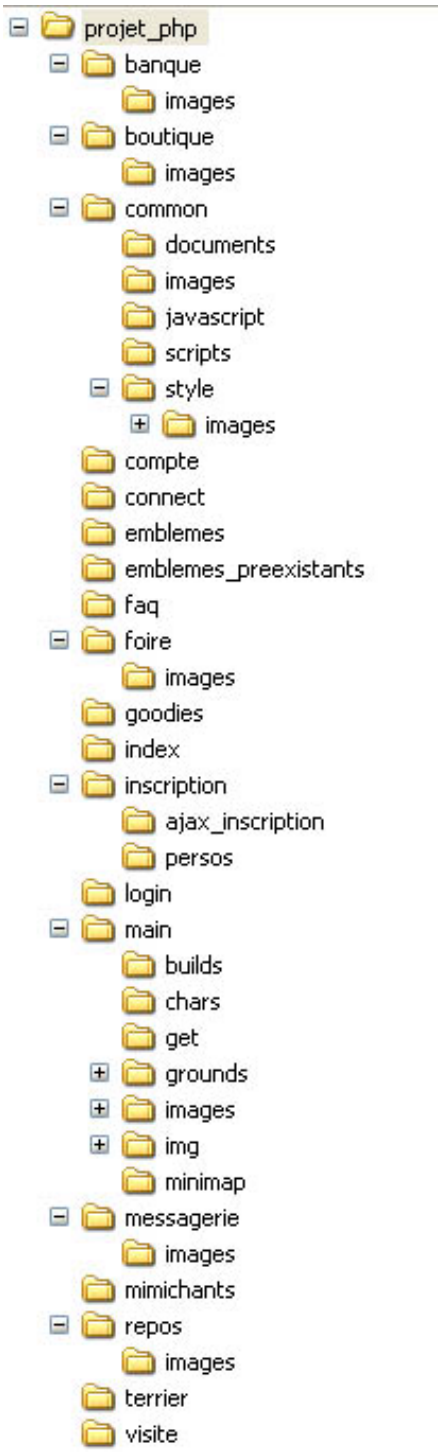
Les armes sont elles aussi dans un univers mignon et déluré pour respecter notre idée de départ. Elles sont faites sous Illustrator pour pouvoir plus facilement les utiliser par la suite. Leurs caractéristiques (dégât, bonus…) seront communes pour chacun des groupes. Seules les images des armes seront différentes. Nous avons fait 31 armes, un œuf et 6 potions.



Nos armes

2. L'organisation des fichiers

a) Architecture des fichiers (arborescence)



L'architecture des fichiers est assez classique. Un dossier pour chaque partie du site et en général dans chaque dossier figure un dossier image.

- **Banque :** Gestion du service banque
- **Boutique :** Gestion du service boutique
- **Foire :** Gestion du service foire
- **Repos :** Gestion du service repos

Ces 4 dossiers utilisent des scripts présents dans le dossier /common

- **Compte :** Gestion du compte
- **Connect :** Pour se connecter au site.
- **Emblèmes** et **emblèmes_préexistants** :
  Contient les emblèmes pour la guilde.

- **Faq :** Page faq
- **Goodies :** Page goodies
- **Visite :** Visite interactive du site
- **Inscription :** Gestion de l'inscription du joueur au site

Ces 3 pages sont accessibles uniquement depuis la page **Index**

- **Index :** Première page du site (page d'accueil)
- **Login :** Lors de la connexion
- **Main :** Page principale du jeu avec la carte, la mini carte et les onglets.

- **Messagerie :** Gestion de la messagerie du joueur
- ***Mimichant :*** Dossier contenant les *Mimichants* crées par les joueurs.

b) L'organisation des fichiers

L'organisation des fichiers a été standardisée pour une meilleure organisation :

- Les dossiers représentent tous une page ou un groupe de pages du site, excepté le dossier *common*. La page principale de chaque dossier contient le même nom que le dossier, à savoir si le dossier s'appelle *machin*, la page sera accessible à *machin/machin.php*.

- Le dossier *common* est le dossier qui contient tout ce qui est commun à plusieurs parties du site :
  - A la racine on trouve les templates des en-têtes et des pieds de page.
  - Dans le dossier *style*, on trouve le style de la page css.
  - Le dossier *scripts* contient les scripts PHP
  - Le dossier *javascript* contient les scripts Javascript
  - Le dossier *images* contient des images (non appelées par le css)
  - Le dossier *documents* contient des documents divers

  Chaque page utilise donc les templates communs pour ses en-têtes et ses pieds de page, les scripts communs, le style unique pour tout le site, etc … Tout ceci évite la redondance et permet une meilleure maintenance du site.

3. Choix du développement

a) Ajax

Le parti pris d'orienter notre implémentation du jeu sur les technologies AJAX est la réponse à notre désir de fournir à l'utilisateur une interface la plus ergonomique et réactive possible. AJAX était pour nous le choix de l'instantané, de l'application Web, en un mot, du confort de jeu. Il est ainsi employé pour les services tel l'inscription, la messagerie, etc. autant que pour le jeu en lui-même, soit la carte de jeu avec les modules attenants affichant les informations sur la partie et gérant les différents éléments de gameplay comme la copulation, le combat, la capture de terrain et bien d'autres.

b) Les templates

La toute première version du site ne contenait pas de templates, en raison de sa simplicité. Mais la complexité grandissante a rapidement requis d'utiliser des templates pour séparer le code PHP (programmation) du code HTML (interface).

Nous avons utilisé un système pour faire des templates, qui une fois fait et compris par les différents développeurs du site, permettait une grande simplicité et réutilisabilité du site : nous avons créé dans un fichier appelé templatize.inc.php deux fonctions PHP permettant de générer l'affichage avec un seul appel de fonction, prenant en paramètre les quatre fichiers templates qui vont servir à l'affichage, et un élément optionnel contenant des variables templates additionnelles. Ces variables additionnelles s'ajoutent aux variables templates communes au site (les liens entre les fichiers, par exemple) définies dans ces fonctions.

<u>Respect des standards :</u>

Pour un respect des standards du W3C (et parce que c'était explicitement marqué dans l'énoncé) les pages sont XHTML/CSS strict.

Problèmes et limitations :

La petite barre de connexion ou de menu en haut de la page bugge sous Internet Explorer (en même temps, c'est normal, c'est Internet Explorer…). Il ne s'agit cependant que d'un problème d'affichage, la connexion marche quand même.

IV. Les parties du site

1. La carte

Orientation AJAX :

Afin de découvrir les possibilités du WEB 2.0 et d'offrir la réactivité essentielle à un véritable confort de jeu, nous avons tiré parti des technologies AJAX.

L'ensemble des fonctionnalités propres à l'interface du jeu (carte globale et carte de jeu) sont réalisées d'abord en Javascript, qui, au moyen de l'objet « XmlHttpRequest », demande des pages au serveur. Notre architecture fait correspondre à chaque fonctionnalité une fonction Javascript et un script PHP. La page de jeu n'est ainsi jamais rechargée et les requêtes à la base de données sont minimales.

Ci-dessous, une liste exhaustive des correspondances fichiers/fonctions utilisés par la carte :

Fonctions Javascript (getEverything.js)	***Pages de script PHP (/main/get)***	***Description***
getCharPos(id) **getCharPosBack()**	getCharPos.php	Récupère les coordonnées du *Mimichant* d'identifiant id
moveChar(id, x_dir, y_dir) **moveCharBack()**	moveChar.php	Déplace un *Mimichant* (id) et actualise sa positon dans la base
getCaseInfos(x,y) **getCaseInfosBack()**	getGroundInfos.php getGroundChars.php	A la sélection d'une case : - positionne le marqueur de sélection - requête des informations détaillées (terrain, alliance, propriétaire, liste des *Mimichants* présents)
getGroundCharInfos() **getGroundCharInfosBack( )**	getGroundCharInfos.php	Infos détaillées sur le *Mimichant* ennemi sélectionné dans la liste d'un terrain
getGrounds(coord,x,y) **getGroundsBack()**	getGrounds.php	Requête demandant les terrains autour de (x, y)
getFlags() **getFlagsBack()**	getFlags.php	Requête demandant les flags des terrains de la zone affichée
getChars() **getCharsBack()**	getChars.php	Requête demandant les personnages présents dans la zone affichée

Ce découpage des pages de script PHP a permis un développement modulaire et indépendant de l'implémentation des communications avec le serveur. Ainsi, nos premiers tests pouvaient être réalisés sans nécessiter de base de données, simplement en simulant les réponses du serveur avec des scripts PHP ou des fichiers XML.

Dans la pratique, le système gagne autant en performance qu'en fonctionnalité : par exemple, les emblèmes de guilde des terrains peuvent être récupérés à tout moment, indépendamment des terrains eux-mêmes. Lorsqu'ils ne sont pas affichés, cela économise plusieurs jointures dans les requêtes à la base de données ; tandis qu'à tout moment, ils peuvent être récupérés, sans recharger les terrains. De plus, une fois les emblèmes « calculés », Javascript se contente de les rendre visibles ou non à la volée.

La structure de la carte est entièrement réalisée avec des positionnements absolus et l'utilisation de l'indice de profondeur « z-index », pour organiser l'indépendance des différentes couches (« layer ») du jeu.

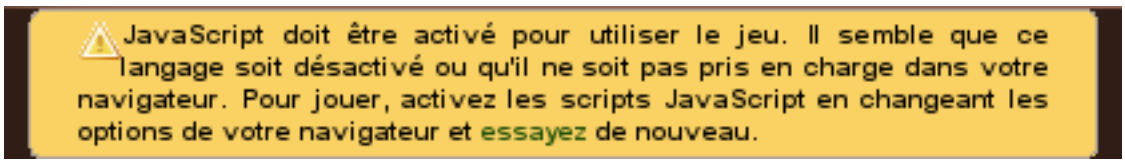
Toujours dans un souci d'optimisation, réduction des redondances CSS, minimisation de la quantité de code XHTML et compromis poids/calcul ont été des préoccupations permanentes pour alléger les requêtes à la base de données et les ressources associées.

La carte et la bandeau

Plus loin, la réactivité du système permet même d'afficher un bandeau de notification « temps-réel » sur les événements de la partie : très « tendance », cette fonctionnalité se rapprochant de l'ergonomie Gmail peut être appelée à tout moment par n'importe quelle fonction Javascript pour informer l'utilisateur.

Ce choix pour la technologie AJAX a néanmoins impliqué une concession importante : le système nécessite que le navigateur du client dispose d'un interpréteur Javascript, et que ce dernier ne soit pas désactivé. Dans l'idéal, il serait possible de développer pour chaque fonctionnalité Javascript son alternative PHP pour supporter tout environnement client. La charge supplémentaire de travail ne se justifiant pas dans notre optique de confort de jeu, nous avons écarté cet aspect.

Dans le cas où l'interpréteur Javascript est indisponible, un message d'avertissement est présenté à l'utilisateur (balise <noscript>) :



Fonctionnalités :

Combinés en quelques instants, les deux fonctionnalités communiquent pour une ergonomie à toute épreuve : un clic sur la mini carte (auto générée) permet directement de se déplacer sur la carte détaillée. Le système profite de la pertinence des technologies AJAX pour une réactivité inégalée. Plus loin encore...

La carte détaillée présente les fonctionnalités suivantes :

Carte et informations:

- affichage des bâtiments mines et terriers
- affichage des *Mimichants* de l'utilisateur sur la carte et d'icônes de présence pour les autres
- affichage / masquage des emblèmes de guilde de toutes les cases de la carte à la volée (pour une appréciation stratégique des conquêtes territoriales)
- affichage d'un bloc d'informations à la sélection d'une case : propriétaire, guilde - avec emblème – et liste des *Mimichants* présents
- affichage d'informations sur les *Mimichants* sélectionnés dans la liste d'un terrain

Navigation dans la carte :

- possibilité de forcer le rafraîchissement de la carte
- flèches de déplacement contrôlant, au choix, la vue détaillée ou le *Mimichant* sélectionné
- petite carte globale cliquable pour déplacer rapidement la vue détaillée, avec cible indiquant la position de la vue courante
- possibilité de recentrer la vue sur le *Mimichant* sélectionné
- recentrage automatique de la vue sur le *Mimichant* sélectionné ou lorsqu'il atteint une certaine marge au bord de la zone affichée (pour diminuer le nombre d'actualisation de la vue détaillée)

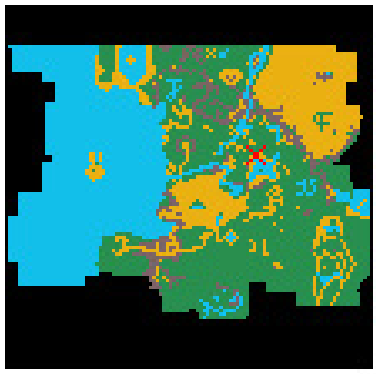
Intégration :

Templates et initialisation

Sans revenir sur le système de templates, on précisera que la carte détaillée étant entièrement gérée en Javascript – jusqu'à l'initialisation pour une meilleure évolutivité – il s'agissait de faire passer certaines variables depuis le PHP, comme l'identifiant et le login de l'utilisateur : l'appel au script « initMap(id, login) » est donc construit avec des variables de templates, pour assurer le lien PHP($_SESSION) et Javascript.

Le positionnement de la carte détaillée (« layers » en absolu) était protégé par des conteneurs en positionnement relatif pour les maintenir dans le flux de la page et s'est fait sans difficultés.

Carte globale



Le premier élément associé à la carte de jeu est la carte globale de navigation, qui doit mettre à jour la vue dans la carte principale. Développée dès le départ en vue de cette intégration, il fut relativement aisé de faire communiquer les deux systèmes, via Javascript.

Un fonction Javascript propre à la carte globale reçoit l'événement d'un clic et convertit les coordonnées de la souris en coordonnées dans la carte de jeu pour demander le rafraîchissement de cette dernière.

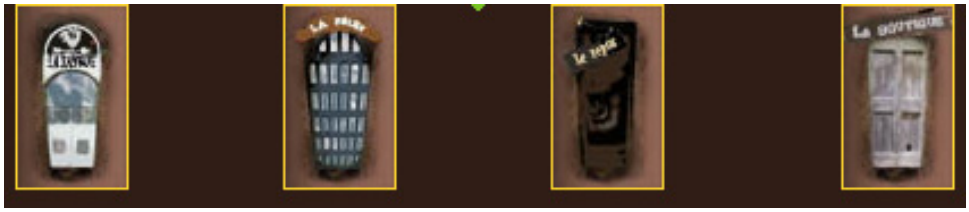
Des champs cachés, <intput type= « hidden » />, assurent la récupération en Javascript des informations calculées lors de la génération automatique de la carte mise au point par Clément Dony, en utilisant la bibliothèque GD.

Fonctionnalités de jeu

Fidèle à l'orientation AJAX, chaque fonctionnalité du jeu (banque, capture de terrain, copulation, attaque, etc.) est mise en correspondance avec le traitement serveur (PHP / +PL/SQL) par une fonction Javascript, qui aura l'avantage de pouvoir directement lui indiquer le « contexte » de l'appel, soient les variables telles que le *Mimichant* sélectionné ou encore le terrain courant, etc.

2. Les services

Les services ne sont accessibles que lorsque le *Mimichant* courant se trouve dans un terrier.



Portes d'accès aux services

a) La banque

Principe

Chaque joueur possède une réserve de chocopépites et chacun de ses *Mimichants* possède également sa propre réserve de chocopépites. La banque est un outil permettant au joueur de gérer l'ensemble de ses chocopépites et de les répartir entre ses *Mimichants,* à condition d'être dans un terrier.

Ainsi, la banque permet au *Mimichant* courant de déposer des chocopépites dans la réserve de son utilisateur ou d'en retirer.

Utilisation

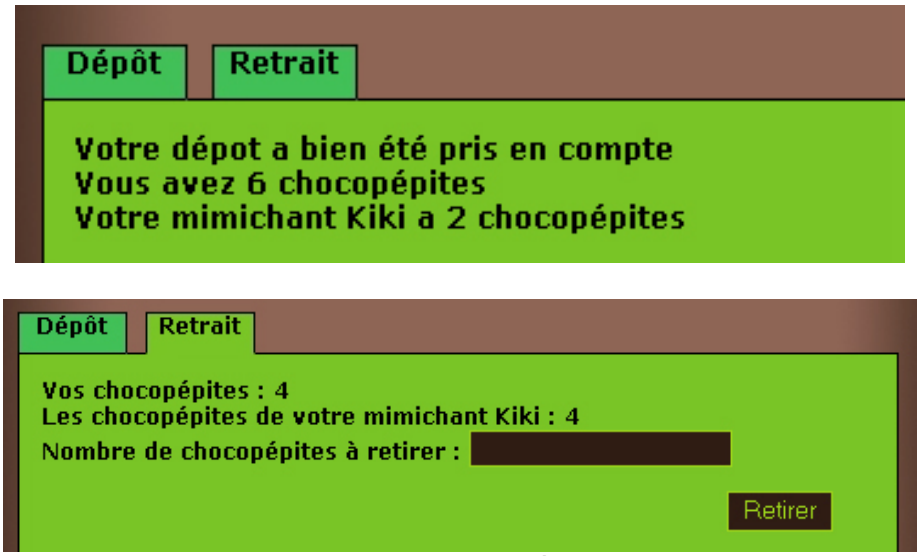
Lorsque le *Mimichant* courant se trouve dans un terrier, une porte « Banque » permet au joueur d'accéder à la banque.

La banque propose alors deux possibilités : déposer ou retirer des chocopépites.

Déposer des chocopépîtes

L'interface renseigne le joueur sur son nombre de chocopépites, le nombre de chocopépites de son *Mimichant* courant et rappelle le nom du *Mimichant* courant. Elle permet de saisir la somme à déposer, c'est-à-dire la somme à prélever chez le *Mimichant* et

à déposer chez le joueur. La procédure *depot* effectue ensuite le transfert : elle retourne « true » si celui-ci s'est bien passé, et « false » si le *Mimichant* n'avait pas assez de chocopépites. En fonction de la valeur renvoyée, on informe le joueur sur la réussite du transfert.



Retirer des chocopépites

L'interface donne les mêmes informations que précédemment et permet la saisie de la somme à retirer, c'est-à-dire la somme à prélever chez le joueur et à déposer chez le *Mimichant*. La procédure *retrait* effectue ensuite les transfert en vérifiant comme précédemment que le joueur possède suffisamment de chocopépites. Selon la valeur renvoyée, on informe le joueur sur la réussite du transfert.

Dans les deux cas, l'interface vérifie que la somme saisie par l'utilisateur est correcte, c'est-à-dire que le champ ne est pas vide ou que la somme ne contient bien que des chiffres.

b) La boutique

Principe

La boutique permet de vendre ou d'acheter de l'équipement pour le *Mimichant* courant.

Utilisation

Lorsque le *Mimichant* courant se trouve dans un terrier, une porte « Boutique » permet au joueur d'accéder à la boutique. La boutique propose alors deux possibilités : acheter ou vendre des objets.

Acheter des objets

L'interface liste l'ensemble des objets achetables et le nombre de chocopépites du joueur. En survolant un objet, une bulle d'information présente les caractéristiques de l'objet. En cliquant sur un objet, le joueur sélectionne cet objet. L'appui sur le bouton « Acheter » fait appel à la procédure *buy_obj* qui débite le compte du joueur de la somme correspondant à l'achat et équipe le *Mimichant* courant de ce nouvel objet.

Vendre des objets

L'interface liste l'ensemble des objets équipés par le *Mimichant* et le nombre de chocopépites du joueur. En survolant un objet, une bulle d'information présente les caractéristiques de l'objet. En cliquant sur un objet, le joueur sélectionne cet objet. L'appui sur le bouton « Vendre » fait appel à la procédure *sell_obj* qui crédite le compte du joueur de la somme correspondant à la vente et déséquipe le *Mimichant* courant de cet objet.

c) La foire

Principe

La foire permet de vendre ou d'acheter des *Mimichants*, hormis le *Mimichant* courant.

Utilisation

Lorsque le *Mimichant* courant se trouve dans un terrier, une porte « Foire » permet au joueur d'accéder à la foire.

La foire propose alors trois possibilités : acheter ou vendre des *Mimichants*, modifier la vente d'un *Mimichant*.

Acheter des Mimichants

L'interface liste l'ensemble des *Mimichants* achetables dans cette foire et le nombre de chocopépites du joueur. En survolant un *Mimichant*, une bulle d'information présente les caractéristiques de ce *Mimichant*. En cliquant sur un *Mimichant*, le joueur le sélectionne. L'appui sur le bouton « Acheter » fait appel à la procédure *buy_Mimichant* qui débite le compte du joueur de la somme correspondant à l'achat, change le propriétaire du *Mimichant* et crédite le compte de l'ancien propriétaire à condition que le joueur et son *Mimichant* courant remplissent les conditions nécessaires de l'achat. Un message informe le joueur sur la réussite de la vente.

Vente des Mimichants

L'interface liste l'ensemble des *Mimichants* du joueur hormis le *Mimichant* courant et affiche le nombre de chocopépites du joueur. En survolant un *Mimichant*, une bulle d'information présente les caractéristiques de ce *Mimichant*. En cliquant sur un *Mimichant*, le joueur le sélectionne. Il peut alors saisir le prix de vente qu'il désir (la saisie est vérifiée afin que la valeur soit bien un entier positif). L'appui sur le bouton « Mettre en vente » fait appel à la procédure *sell_Mimichant* qui vérifie que le joueur et son *Mimichant* courant remplissent

bien toutes les conditions nécessaires à la mise en vente d'un *Mimichant*. Un message informe le joueur sur la réussite de la mise en vente.

Modification d'une vente

L'interface liste l'ensemble des *Mimichants* mis en vente du joueur et affiche le nombre de chocopépites du joueur. En survolant un *Mimichant*, une bulle d'information présente les caractéristiques de ce *Mimichant*. En cliquant sur un *Mimichant*, le joueur le sélectionne. Il peut alors retirer le *Mimichant* de la vente ou diminuer son prix de vente. Un message informe le joueur sur la réussite du retrait de la vente du *Mimichant* sélectionné ou la diminution du prix de vente.

3. La messagerie et le carnet d'adresses

La messagerie est la partie du site qui va permettre une communication interne entre les joueurs.

Nous avons essayé de concevoir la messagerie en respectant les outils qui existent déjà dans les messageries comme Hotmail, Yahoo !,…

Pour la réception de messages :

Le joueur dispose d'une interface où les messages qu'il a reçus sont rangés dans l'ordre d'arrivée. Il sait si ses messages ont été lus ou non. Il a la possibilité de les « marquer comme non lu » ou même encore de les supprimer.

Pour l'envoi de messages :

Pour envoyer un message, l'interface est simple, avec comme champs, le nom de la personne à qui on envoie le message, l'objet et le message. La date et l'heure sont récupérées dynamiquement par PHP.

Pour répondre aux attentes du sujet, l'utilisateur peut envoyer un même message à plusieurs utilisateurs.

Le carnet d'adresses :

Lorsque l'utilisateur veut envoyer un message, il a la possibilité d'enregistrer les noms d'utilisateurs de ces camarades dans un « carnet d'adresses ». Ainsi, lorsqu'il veut envoyer un message, il lui suffit de cliquer sur le nom de la personne dans le carnet

d'adresses et le nom sera dynamiquement ajouter dans le champ « A :». Si le joueur clique sur plusieurs noms, ils seront ajoutés à la suite et séparés par des virgules.

L'intérêt d'une messagerie :

La messagerie joue un rôle important dans le jeu, il permet aux utilisateurs de communiquer, de faire des demandes tels que pour la copulation ou des échanges entre les joueurs. C'est pourquoi la messagerie est ludique pour permettre à tous les utilisateurs de pouvoir y accéder.

Système de messagerie

4. L'interface d'échange

Les règles de l'échange

L'interface d'échange est évoquée très rapidement dans les règles du jeu, mais les modalités d'un échange n'y sont pas précisées. De plus, elle a complètement été mise de côté lors de la conception de la base de données.

La décision a donc été prise (assez tardivement) d'implémenter une version simple de cette interface.

Les règles fixées sont les suivantes :

- Un *Mimichant* peut proposer son équipement à l'échange contre un objet qu'il désire.
- Tous les *Mimichants* qui possèdent un tel objet voient l'annonce dans l'interface d'échange.
- Un *Mimichant* intéressé peut alors accepter l'offre. L'échange est réalisé et la proposition effacée de la base.
- Le *Mimichant* qui a déposé la proposition peut la retirer à tout moment (tant que personne n'y a répondu).

L'implémentation

A l'heure où nous écrivons, seule la partie « base de données » a été réalisée pour l'interface d'échange.

Une table a dû être rajoutée au schéma initial de la base, et des procédures stockées ont été écrites. Ces procédures permettent de déposer une proposition, de la retirer, de récupérer toutes les propositions déposées par le *Mimichant*, de récupérer toutes celles potentiellement intéressantes pour le *Mimichant*, et d'accepter une offre.

L'interface elle-même sera constituée de 3 onglets : l'un affichant les propositions des autres *Mimichants* avec possibilité d'y répondre, un second affichant les propositions du *Mimichant* courant avec possibilité de les retirer, et un dernier permettant de déposer une proposition.

Les améliorations possibles

Cette interface est volontairement très simple, car nous l'avons implémentée tard dans le projet et avons donc pensé qu'il était préférable qu'elle soit basique et opérationnelle plutôt que non viable par manque de temps.

Cependant, nous avons réfléchi aux améliorations possibles :

- Ne permettre le troc qu'aux *Mimichants* qui se trouvent sur des cases voisines : il faudrait pour cela modifier la procédure qui renvoie les annonces « intéressantes » pour vérifier la distance entre les *Mimichants* avant de renvoyer le résultat.
- Enlever des points d'actions aux *Mimichants* qui réalisent l'échange
- Permettre d'échanger plusieurs objets à la fois (exemple : une grosse arme contre deux petites…) : cela nécessiterait une table de plus dans la base car la relation serait alors « n à n » au lieu de « 1 à 1 »
- Permettre de proposer plusieurs échanges pour un même équipement : « je veux échanger mon objet contre un objet X OU un objet Y ». Il faudrait un id supplémentaire qui serait identique pour les propositions relatives au même équipement, et supprimer toutes les propositions lorsqu'une est acceptée.
- Gérer les objets « utilisés » : permettre de refuser l'échange avec un objet déjà utilisé, ou remise à 0 de l'utilisation…

5. Gestion du compte utilisateur

Utilisation d'Ajax pour l'inscription :

Pour une meilleure performance et un meilleur *look and feel* de l'inscription, celle–ci a été réalisé au moyen de l'objet XmlHttpRequest de Javascript. Le principe est que lorsqu'on remplit son login et son mot de passe, et qu'on clique sur continuer, cela lance une fonction Javascript, qui lance elle-même une page PHP. Dans le script PHP correspondant on teste si l'utilisateur existe déjà, et si le login et le mot de passe sont corrects. La page générée en XML sera interprétée en retour par une autre fonction Javascript qui détecte les erreurs de login et mot de passe et va faire des opérations sur le code HTML de la page initiale en conséquence, laissant apparaître un message d'erreur ou bien la suite de l'inscription.

Ajax est utilisé deux fois dans l'inscription, qui est d'ailleurs séparé en deux pages. Pourquoi avoir fait deux pages, alors qu'on aurait pu faire tout dans une seule page, avec Ajax ? Parce que cela évite pour l'utilisateur d'avoir à effectuer un scroll vertical sur la page.

Limitations :

Sur le serveur de la fac, l'utilisation de l'objet XmlHttpRequest de Javascript provoque des problèmes sur les explorateurs de type Mozilla. On peut complètement corriger le problème en lisant la FAQ du site qui explique comment configurer correctement le navigateur.

Ajax pose aussi des problèmes sous Internet Explorer mais notre objectif premier était le fonctionnement sous Mozilla.

Gestion du compte :

La gestion du compte est plus simple que l'inscription, dans la mesure où il n'y a plus lieu de s'occuper des *Mimichants*, mais seulement de l'utilisateur.

Code commun à l'inscription et à la gestion du compte :

L'inscription et la gestion du compte utilisent des fonctions PHP communes, en particulier le code de chargement d'un emblème de guilde.

Ce code redimensionne les emblèmes de guilde (uploadés avec $_FILE) grâce à la bibliothèque GD.

6. Affichage Informations utilisateur et *Mimichant*

Une partie du site est dédiée à l'affichage d'informations sur l'utilisateur et sur ses Mimichants.

Principe

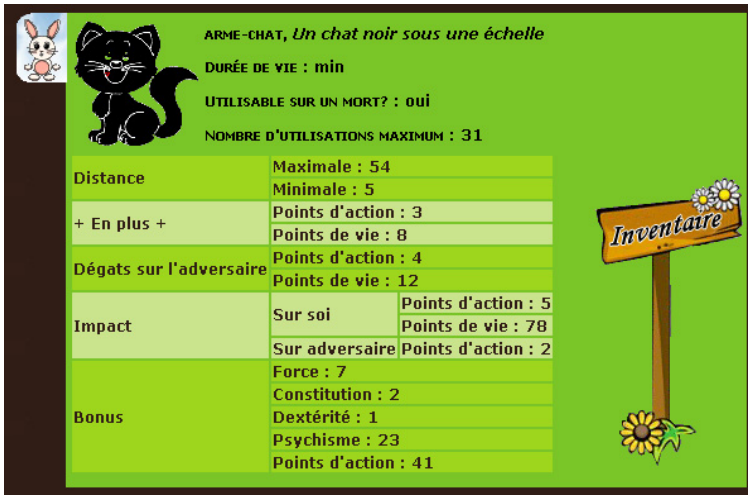
A la connexion, les informations concernant l'utilisateur sont affichées sur la page principale et le bandeau à gauche. Est également affiché l'historique depuis la dernière connexion : pour chaque *Mimichant*, un récapitulatif du nombre de ses points d'action, chocopépites…, pour le joueur, le nombre de fois où ses *Mimichants* se sont fait attaquer et le nombre de demandes de copulation.

Nous avons également placé sous la carte des onglets dont le nombre équivaut au nombre de *Mimichants* de l'utilisateur. Dans chaque onglet, représenté par l'image du *Mimichant* en question, sont affichées toutes les informations utiles au jeu (nom, sexe, chocopépites, points de vie, points d'action…). L'utilisateur a également la possibilité de voir tous les objets appartenant à l'équipement ou à l'inventaire du *Mimichant*.

Utilisation

Lorsque l'utilisateur sélectionne un des *Mimichants* dans les onglets, il est repéré et sélectionné sur la carte. De même, lorsqu'un *Mimichant* est sélectionné dans la carte, l'onglet affriché est celui qui lui correspond.

Onglet informations sur le Mimichant



Inventaire du Mimichant

Informations sur l'arme sélectionnée

7. Détail des procédures utilisées

a) Copulation

La procédure de copulation est appelée dès qu'un *Mimichant* choisit de copuler avec un autre *Mimichant*. Il y a alors des paramètres à vérifier.

On vérifie tout d'abord, grâce à la procédure checkpa, que les deux *Mimichants* ont assez de points d'action. Le mâle doit avoir au moins 12 points d'action, et la femelle, elle, doit posséder tous ses points d'action (24, soit un par heure).

Ensuite, nous faisons appel à une autre procédure, isheteronotpregnant, qui vérifie que les deux Mimichants n'appartiennent pas au même utilisateur, qu'ils sont de sexes opposés, et que la femelle n'est pas déjà enceinte (on vérifie dans la table eggs que le Mimichant femelle n'apparaît pas déjà en tant que mère). Si ces conditions ne sont pas remplies, un message est envoyé au(x) utilisateurs(s) concerné(s).

Si ces conditions sont remplies, la copulation peut avoir lieu : on retire tous ses points d'action à la femelle, la moitié au mâle, et l'œuf est créé dans la table egg. La procédure copulation prend pour paramètres : l'identifiant de la femelle, du mâle, la date courante et un identifiant à 1 pour l'objet car c'est un œuf.

L'éclosion

24 heures après la copulation, l'œuf doit éclore. Nous avons créé une procédure, checkeggandenoughspace, qui se charge de vérifier que, dans la table eggs, la date de copulation est passée de 24h, qui ajoute à l'équipement de la femelle l'objet œuf et qui supprime aussitôt l'œuf de la table eggs. On utilise les timestamp pour comparer les dates.

La naissance du Mimichant

24 heures après (soit 48h après la copulation), nous vérifions, grâce à cette même procédure, que la femelle a assez d'espace vital pour ses *Mimichants* avant de créer un nouveau Mimichant et de supprimer l'œuf de son équipement. En cas d'espace insuffisant, un message lui est envoyé. Cette procédure est appelée tout le temps et recherche dans la table equipements tous les objets ayant un obj_id à 1.

Le père reçoit une compensation à la fin de la copulation, par le biais de l'augmentation de son score de psychisme, dextérité, constitution et de force physique.

La réussite finale de cette procédure est signalée par l'envoi à la mère d'un message signalant l'heureux évènement.

Récapitulatif des procédures propres à la copulation

Nom	***Arguments***	***Opérations effectuées***	***Retour***
checkpa	*Integer, boolean* id du *Mimichant*, sexe du *Mimichant*	Vérifie si le *Mimichant* a assez de P.A. pour copuler, en fonction de son sexe.	*boolean* **true** si l'opération s'est bien passée et **false** sinon
isheteronotpregnant	*integer, integer* id du *Mimichant* 1, id du *Mimichant* 2	Vérifie si le couple est bien hétérosexuel, si les deux *Mimichants* appartiennent à des utilisateurs différents et si la femelle n'est pas déjà enceinte.	*boolean* **true** si vérifications validées et **false** sinon
copulation	*integer, integer* id premier *Mimichant*, id du second	Appel à chekpa ; Appel à isheteronotpregnant ; Réactualisation PA Insertion d'un œuf dans la table eggs	*Boolean* **true** si l'opération s'est bien passée et **false** sinon
checkeggandcheckenoughspace	aucun	24h après la copulation, insère l'objet de type 1 dans la table equipements, enlève l'œuf de la table eggs. 48h après copulation, vérifie si le futur Mimichant aura assez d'espace vital. Enlève l'objet de type 1 précédemment créé de la table equipements si vérification validée.	*boolean* **true** si l'opération s'est bien passée et **false** sinon
create_Mimichant_by_repro	*integer, character varying, text, integer, integer, integer, integer, integer, integer* user_id,char_name,char_infos,charIdMum,charIdDad,char_strength,char_const,char_dext,char_psy	Crée un *Mimichant* au graphisme aléatoire, avec pour parents les deux *Mimichants* ayant copulé	*Boolean* **true** si l'opération s'est bien passée et **false** sinon

b) Banque

Les procédures de la banque ont été relativement simples à développer. En effet, le fonctionnement de ce service était clair, et les actions réalisées par les procédures ne nécessitaient que quelques requêtes SQL simples (SELECT et UPDATE). Elles constituaient donc un moyen idéal d'apprendre vite à coder une procédure stockée.

Quatre procédures ont été codées pour la banque. Deux d'entre elles réalisent une action (c'est-à-dire mettent à jour des tables), les deux autres servent à récupérer des informations pour afficher dans l'interface.

Nom	***Arguments***	***Opérations effectuées***	***Retour***
depot	*integer,integer* id du *Mimichant*, montant à déposer à la banque	Après avoir vérifié que le montant à déposer est inférieur à la somme d'argent portée par le *Mimichant*, le transfert d'argent est effectué, et la taxe de 5% est prélevée au passage.	*boolean* **true** si l'opération s'est bien passée et **false** sinon
retrait	*integer,integer* id du *Mimichant*, montant à retirer de la banque	Après avoir vérifié que le montant à retirer est inférieur au solde du compte, le transfert d'argent est effectué.	*boolean* **true** si l'opération s'est bien passée et **false** sinon
solde_character	*integer* id du *Mimichant*	La somme d'argent portée par le *Mimichant* est lue dans la base de données.	*integer* argent porté par le Mimichant
solde_user	*integer* id du *Mimichant*	Le solde du compte en banque de l'utilisateur propriétaire du *Mimichant* est lu dans la base de données.	*integer* solde du compte en banque de l'utilisateur

c) Foire aux *Mimichants*

La difficulté de ces procédures résidait dans la compréhension des règles et de la base de données. La description de ce service dans les règles laisse certains détails dans l'obscurité, notamment ce qu'il advient de l'équipement non vendu, ou de l'argent du *Mimichant*. De plus certains détails manquaient dans la base de données, ou étaient très peu explicites.

Des décisions ont donc été prises :

- L'équipement non vendu est perdu pour tout le monde (donc supprimé des tables correspondantes).
- L'argent porté par le *Mimichant* est perdu.
- Un *Mimichant* ne peut pas se vendre lui-même, il ne peut vendre que les autres *Mimichants* du même utilisateur. Les points d'actions lui sont enlevés, mais la taxe est payée par le compte en banque de l'utilisateur.
- De même, lorsque qu'un *Mimichant* achète un autre *Mimichant*, les points d'actions lui sont enlevés, mais c'est l'utilisateur qui paye.

Quatre procédures ont été développées, dont une servant uniquement à récupérer des informations.

Nom	*Arguments*	*Opérations effectuées*	*Retour*
sell_Mimichant	*integer, integer, integer, integer, boolean* id du *Mimichant* à vendre, id du *Mimichant* qui vend, id de terrain du terrier, prix souhaité, vente de l'équipement ou non	La fonction vérifie que le *Mimichant* courant ne se vend pas lui-même, qu'il a assez de points d'action et d'argent, puis le *Mimichant* est mis en vente, la taxe et les PA sont prélevés et l'équipement est supprimé si nécessaire.	*boolean* **true** si l'opération s'est bien passée et **false** sinon
buy_Mimichant	*integer,integer* id du *Mimichant* à acheter, id du *Mimichant* qui achète	La fonction vérifie que l'acheteur a assez de points d'action, d'argent et d'espace vital, puis effectue le transfert d'argent, change le propriétaire du *Mimichant* et le retire de la vente.	*boolean* **true** si l'opération s'est bien passée et **false** sinon
lower_price	*integer, integer, integer* id du *Mimichant* en vente, nouveau prix, id du *Mimichant* qui est à la foire	La fonction vérifie que le nouveau prix est bien inférieur à l'ancien, que le *Mimichant* a assez de points d'action, puis change le prix en prélevant les points d'action.	*boolean* **true** si l'opération s'est bien passée et **false** sinon
mimi_envente	*integer, integer* id du *Mimichant* qui est dans le terrier, id de terrain du terrier	Les *Mimichants* du propriétaire du *Mimichant* courant qui sont à vendre dans cette foire sont lus dans la base de données.	*setof characters* les *Mimichants* de l'utilisateur à vendre dans cette foire

d) Annonce

Pour les annonces, nous avons créé deux procédures qui ont été assez simples à développer. Les procédures des annonces servent seulement à récupérer les annonces enregistrées ou à en enregistrer. Les requêtes étaient donc assez ludiques, nous avons utiliser des select et insert.

Nom	*Arguments*	*Opérations effectuées*	*Retour*
setannounce	Identifiant du *Mimichant*, id du terrain, message	La procédure récupère la date courante, vérifie si l'utilisateur appartient à la guilde qui détient le terrier, si oui l'utilisateur ne paiera pas son dépôt d'annonce, dans le cas contraire, on retire 5 Chocopépites à l'utilisateur. Après ça, l'annonce est enregistrée dans la base.	*boolean* **true** si l'opération s'est bien passée et **false** sinon
readannounce	*integer*	Récupère toutes les annonces enregistrés sur l'id de terrain passé en paramètre.	*boolean* **true** si l'opération s'est bien passée et **false** sinon

e) Création du *Mimichant*

Il s'agit de réaliser une procédure pour créer un *Mimichant*. Pour des raisons pratiques (arguments différents) nous avons fait deux procédures différentes, l'une appelée lors de la création du tout premier *Mimichant* et une seconde lors de l'éclosion d'un œuf. En effet lors de l'éclosion d'un œuf, l'utilisateur ne choisit pas lui-même les caractéristiques physiques de son futur *Mimichant*. Dans notre jeu les lois sur la génétique sont prises en compte. Il n'y a cependant pas de gène récessif et le futur enfant a autant de chance de ressembler à son père qu'a sa mère (1 chance sur deux pour chaque partie du corps et la couleur).

Premier Mimichant (procédure creation_Mimichant_premier)

Cette procédure prend beaucoup d'arguments en paramètre car de nombreuses caractéristiques sont choisies par l'utilisateur (forme, couleur et caractéristiques du *Mimichant*). Nous avons décidé de fixer les PV, PA, PaMax et EquipMax aléatoirement pour rendre le jeu un peu plus corsé. Cette procédure permet aussi de remplir les infos sur la guilde de l'utilisateur.

Eclosion d'un œuf (procedure create_Mimichant_by_repro)

Cette procédure est appelée après celle sur la reproduction. Elle prend en paramètre (user_id, char_name, char_infos, charIdMum, charIdDad et les caractéristiques de force). Les autres caractéristiques du futur *Mimichant* sont soient décidées aléatoirement soient génétiquement. Lorsqu'un œuf éclos l'utilisateur est dirigé vers une page ou il peut choisir le nom, le descriptif et les caractéristiques de force de son *Mimichant*. La procédure est ensuite appelée.

8. L'administration du site

Pour gérer un tel jeu, une administration est nécessaire afin de centraliser les informations de tous les joueurs et d'y avoir un accès rapide. On peut ainsi ajouter, modifier ou supprimer des informations concernant les joueurs et/ou les *Mimichants*.

L'administration offre la possibilité d'ajouter des objets via une interface, ce qui permet au fil du temps d'offrir la possibilité aux utilisateurs de découvrir des nouveautés.

Au départ, nous souhaitions que notre interface d'administration soit la plus complète possible, c'est-à-dire qu'elle gère les utilisateurs, les *Mimichants*, les objets mais aussi les pnjs (les quêtes) et les décorations et qu'elle ait une messagerie qui permet d'envoyer un message à toutes les personnes inscrites sur le site.

Id	Nom	Mail	Nb CP	Nb mimichants	Voir mimis	Modifier	Supprimer
1	Steph	melybloo@hotmail.com	6000	2			
2	Lucie	lucioleO6@hotmail.com	5000	2			
3	Phrounz	phrounz@mail.fr	0	0			
4	julie	nine-bill@wanadoo.fr	0	4			
21	robert		0	0			
22	John	naloj@ludologi.com	1210	1			
24	mickey	moi@lui.com	0	0			
31	clem	clem@mail.fr	6	1			
49	Nin	nin@mail	0	0			
52	Sev	vngn	0	0			
57	Toto	toto	0	1			
74	Cath		2000	1			

Gestion des utilisateurs via l'administration du site

Pour le moment, l'administration permet de gérer les utilisateurs, *les Mimichants* et l'insertion d'objets.

V. Bilan

1. Les problèmes rencontrés

AJAX

Bien que découvrant la technologie AJAX, nous avons rapidement été confrontés à ses contraintes et limites. D'abord par rapport à la compatibilité des navigateurs : l'implémentation diffère finalement peu entre Internet Explorer et les navigateurs Gecko ; mais la gestion de la sécurité sous Mozilla, Firefox et Netscape posait au départ des restrictions imposant une modifications des paramètre du navigateur peu acceptable pour l'utilisateur lambda.

Le développement sous forme de petits modules s'est heurté aux limites de l'interpréteur Javascript au niveau des traitements asynchrones : sources de bug en raison de la persistance de variables locales empêchant les appels simultanés à certaines fonctions génériques.

Finalement, nous avons choisi à regret de cadrer l'utilisation de XmlHttpRequest sur des traitements synchrones, garantissant le suivi des variable de jeu, mais diminuant de beaucoup l'intérêt d'AJAX car les appels sont alors bloquants, bien que très rapides.

Javascript

La personne responsable du développement de la carte de jeu a finalement autant codé en Javascript qu'en PHP, sinon davantage. Pour un projet de cette envergure, le manque d'orientation objet de Javascript 1 (et/ou 1.5) se révèle relativement contraignant car on ne peut profiter de l'organisation d'un système de classes/objets et les passages de valeurs sont souvent réalisés par l'emploi de variables globales.

Les améliorations de Javascript 2.0 à venir (typage des variables, paramètres et retour de fonctions, types supplémentaires : int, double, decimal, Class, Type, classes, namespaces et types d'accès (public, private...), héritage, interfaces, packages, iterateurs, générateurs, redéfinition d'opérateurs comme en C++, etc.) se révèleront un véritable atout de ce point de vue.

L'ergonomie du site

L'identité graphique et l'ergonomie ont été pensées en amont. Néanmoins, au fil du travail sur certains modules et notamment la carte, certaines dimensions et formes ont dû être repensées, en vue d'offrir le meilleur confort de navigation.

L'enregistrement de la carte dans la base de données

Au cours des réunions de création des règles du jeu, la façon de gérer les informations liées à la carte n'a pas été définie. Nous avions donc créé un éditeur permettant de ranger de façon ordonnée dans une base de données une carte rectangulaire d'origine (0,0). Nous gérions notre carte en fonction de ces informations, en optimisant au mieux notre code.

Un autre groupe avait également implémenté un éditeur qui ne disposait pas les informations de façon non-ordonnée dans la base, afin de permettre de créer une carte d'origine indéfinie, de forme indéfinie et modifiable. Nous avons donc dû modifier notre code pour se calquer sur le modèle précédemment cité.

La création des *Mimichants*

Ce problème relève plus d'une communication externe, c'est-à-dire avec les deux autres groupes de la classe. En effet par manque de communication et d'explication, lors de la création du *Mimichant*, le choix de sa couleur de pelage n'a pas été envisagé de la même façon par les trois groupes. Nous imposions les couleurs sur chaque partie du corps, alors que les autres groupes définissaient une couleur globale pour le *Mimichant*. Après de longues discussions, nous avons adapté notre code sur le modèle des autres groupes.

Difficultés à comprendre les règles

Les règles ayant étaient définies par un groupe composé essentiellement de joueurs invétérés, il a été difficile au départ de les comprendre. De plus, certaines parties n'ont pas été comprises de la même façon par tous les groupes. Pour une bonne compréhension et en cas de litige, nous devions demander à Flavien, seule personne connaissant toutes les subtilités du jeu, car se trouvant à la fois dans l'équipe de création des règles, les ayant rédigé, et faisant également parti de l'équipe base de données.

La base de données

Au final, la base de données devait être commune aux trois groupes. En attendant d'avoir accès à la base de données de M. Bouvier, nous avions une base temporaire sur laquelle les 3 groupes travaillaient. Lorsque la base de M. Bouvier a été mise en place, certains ont travaillé dessus et d'autres ont continué sur la première. Du coup, certaines informations étaient dans l'une et d'autres dans l'autre. Nous aurions du rapidement tout transférer sur une seule et même base afin d'avoir tous la même version.

De plus, lors de la création de la base, toutes les possibilités n'ont pas été envisagés. Nous nous sommes vite rendu compte qu'il manquait des champs voir même des tables. Quand l'implémentation des procédures a commencé, chacun a rajouté un ou plusieurs champs dans les tables voire modifié des noms, ce qui fait qu'il n'existe plus de version .doc avec toutes les informations concernant la base de données du projet, et qui a généré des erreurs lors du passage de notre base à celle de M. Bouvier.

2. Les améliorations possibles

Même si nous avons au final un projet assez complet, il y a des points qui pourraient être améliorés :

- Les Personnages Non Joueurs ne sont pas complètement intégrés au jeu.
- L'administration n'étant pas complète, on pourrait y ajouter la possibilité d'avoir des statistiques sur les joueurs, leurs habitudes de jeu (les objets qu'ils utilisent le plus,…) afin de mieux optimiser le jeu par la suite.
- La navigation pourrait être optimisée de même que l'ergonomie générale du site car il est difficile de passer de la carte aux onglets des *Mimichants* situés en dessous. Nous utilisons pour cela deux scroll bars ce qui n'est pas pratique pour l'utilisateur. De plus, le changement de statut d'un objet, de « équipé » à « déséquipé » fait sortir des onglets, ce qui coupe la navigation.
- Nous pourrions rechercher automatiquement la résolution de l'écran de l'utilisateur, afin d'adapter notre site à ses dimensions.
- Il a été envisagé une « visite guidée » du site, accessible à partir de la page d'accueil, avec des imprim-écrans explicatifs.
- Il serait agréable visuellement pour l'utilisateur de gérer les raccords entre les textures de la carte.

- L’adaptation à Internet Explorer serait complémentaire de celle à Firefox, trop restrictive.

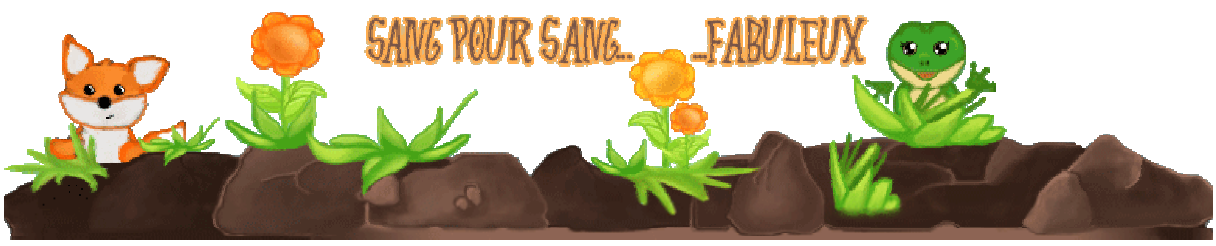
3. Résultats obtenus

Au final, nous avons un projet assez complet qui regroupe la plupart des parties demandées par le sujet. Nous avons essayé de respecter les règles du jeu au maximum sauf pour l’interface d’échange où il n’y avait pas d’informations dans les règles, et sommes assez satisfaits du résultat obtenu.
Tous les modules suivants ont été implémentés :

- Gestion de compte
- Gestion de la carte principale
- Gestion des deux cartes utilisant la libGD (minicarte et carte de jeu)
- Système de messagerie
- Interface d’échange
- Terrier des *Mimichants* : Les Services
- Gestion de l'inventaire du *Mimichant*
- Page état du personnage
- Gestion du BackOffice

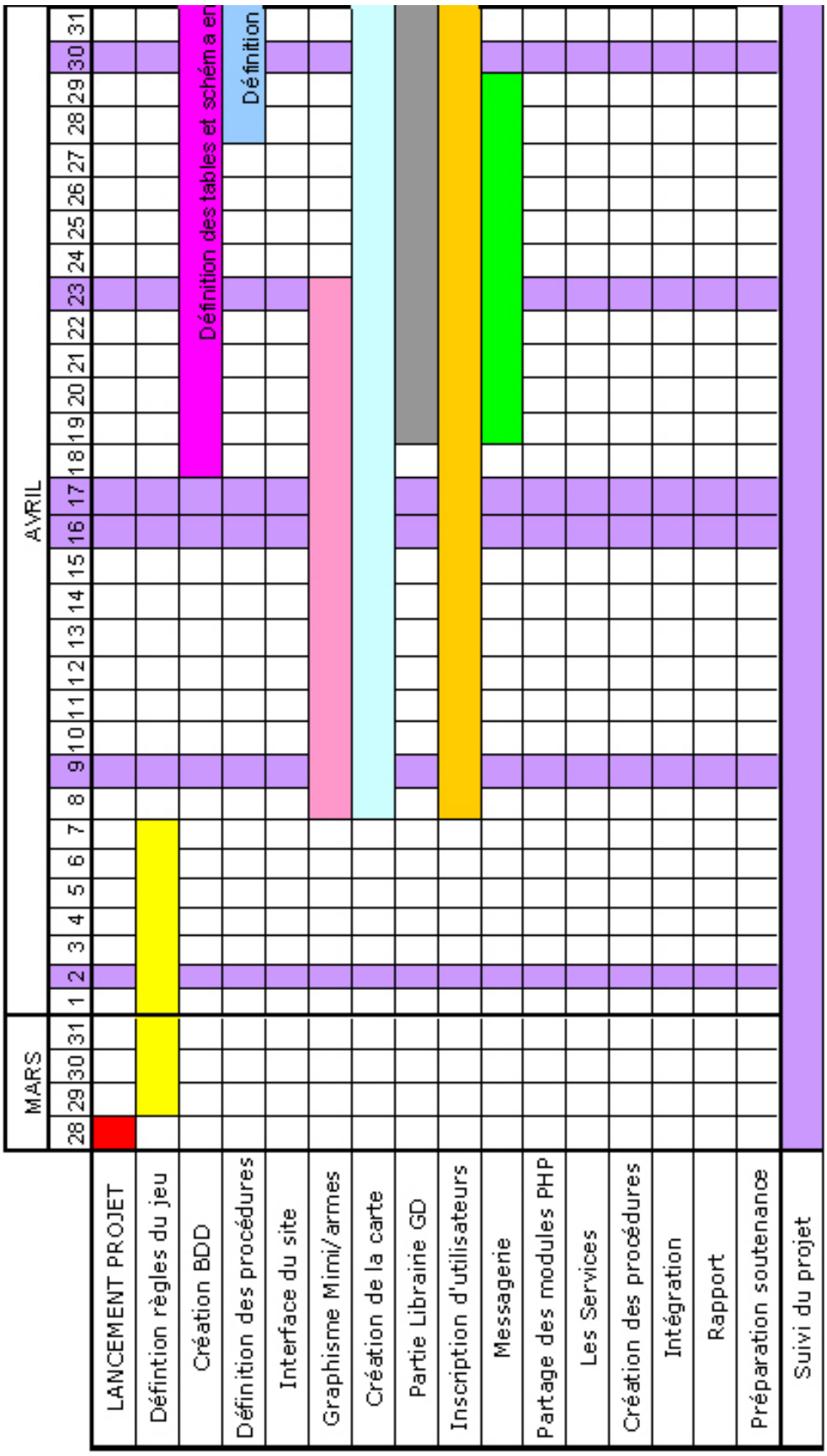
Conclusion

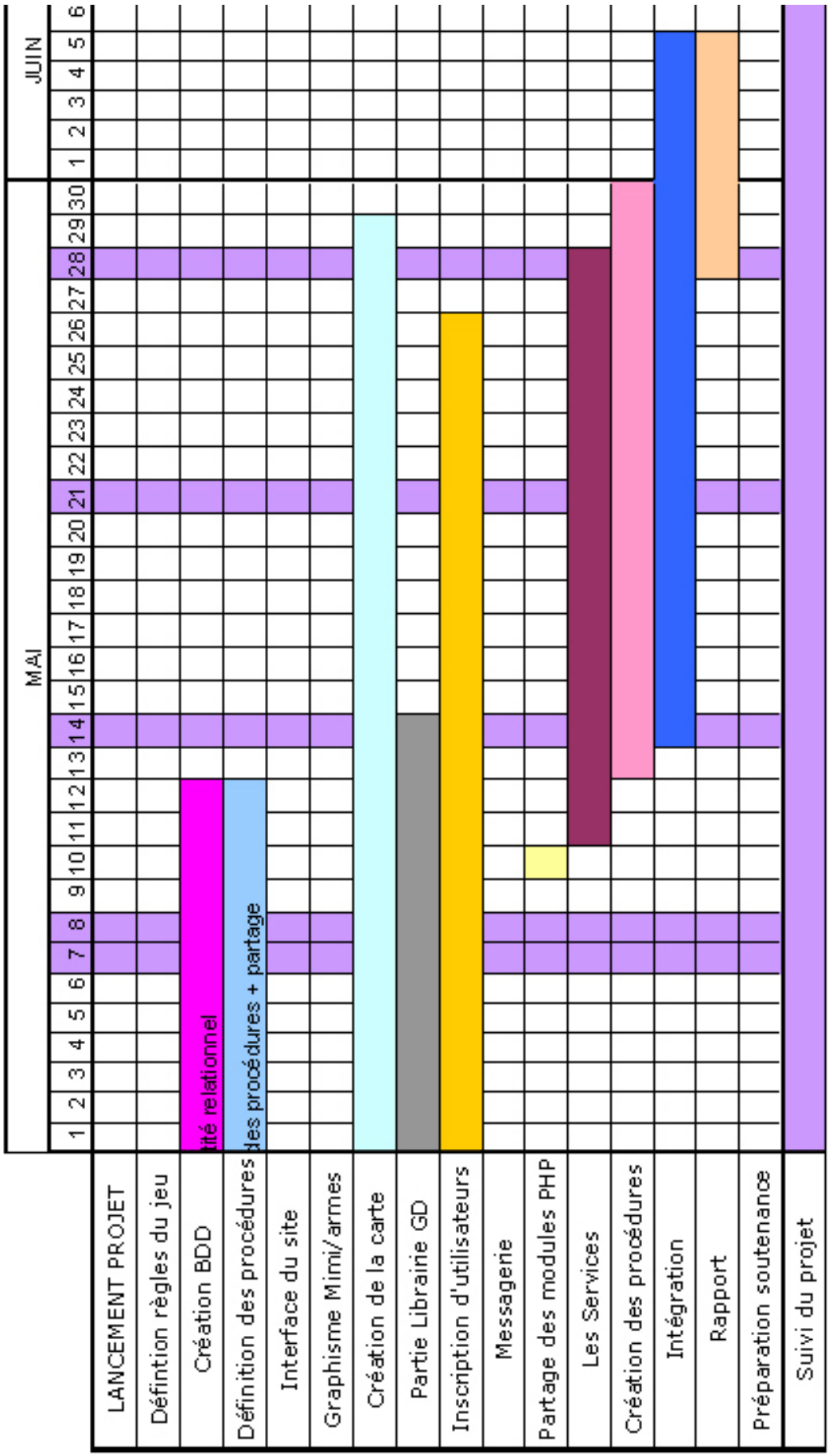
Ce travail de groupe a été intéressant car responsabilisant. De nombreuses décisions ont dû être prises, tant au sein du groupe qu'avec les autres groupes, ce qui fut une bonne expérience de gestion de projet et d'organisation. D'un point de vue du contenu, nous avons également beaucoup appris dans des domaines différents et avons notamment pu mettre en application et développer les connaissances acquises lors des cours de Php.



Annexes

1. Retro planning




MAI
JUIN
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
1 2 3 4 5 6
LANCEMENT PROJET
Définition règles du jeu
Création BDD
tité relationnel
Définition des procédures
des procédures + partage
Interface du site
Graphisme Mimi/armes
Création de la carte
Partie Librairie GD
Inscription d'utilisateurs
Messagerie
Partage des modules PHP
Les Services
Création des procédures
Intégration
Rapport
Préparation soutenance
Suivi du projet

2. La base de données (liste non exhaustive)

• **users**	• **characters**
user_id *(int)* user_login *(string)* user_pass *(string)* user_mail *(string)* user_guild_name *(string)* user_guild_color *(string)* user_guild_emblem *(binary)* user_guild_infos *(string)* user_guild_master_id *(int)* user_date_crea *(date)* user_last_connect *(date)* user_money *(int)*	char_id *(int)* #user_id (propriétaire) *(int)* char_pos *(point)* char_name *(string)* char_infos *(string)* char_sex *(bool)* char_money *(int)* char_pv *(int)* char_pa (points d'actions) *(int)* char_papm (pa par minute) *(int)* char_equipmax (objs équip max) *(int)* char_skincolor *(string6)* char_head (graphisme) *(int)* char_ears (graphisme) *(int)* char_corps (graphisme) *(int)* char_tail (graphisme) *(int)* char_strength (force) *(int)* char_const (constitution) *(int)* char_dext (dexterité) *(int)* char_psy (psychisme) *(int)* #char_id_dad(int) #char_id_mum(in) **char_pvmax (int)**
• **grounds**	• **decorations**
ground_id *(int)* # ground_type_id *(int)* ground_pos *(point)* # user_id *(int)* ground_build *(int)* (0=rien,1=terrier,2=mine) # deco_id *(int)* #obj_id (int)	deco_id *(int)* deco_name *(string)* deco_infos *(string)* deco_obstacle *(bool)* deco_graph *(string)* (graphisme)
• **equipments**	• **characters_sold**
equip_id *(int)* char_id *(int)* obj_id *(int)* obj_date *(date)* **obj_equip(Boolean)** **equip_used(int)** **equip_duration(int)**	sold_id *(int)* char_id *(int)* sold_price *(int)* ground_id *(int)*
• **towns**	• **messages (cf. table send)**
# ground_id *(int)* town_name *(string)* town_infos *(string)* town_take_pa *(int)* town_gain *(int)*	message_id *(int)* # user_send_id *(int)* message_subject *(string)* message_text *(string)* message_read *(boolean)* message_date *(date)*
• **objects**	• **mines**

obj_id *(int)* obj_name *(string)* obj_infos *(string)* ~~obj_weight *(int)*~~ obj_duration (durée de vie en minutes) *(int)* obj_use_dead (utilisable sur mort) *(bool)* obj_use_skill_id *(int)* obj_use_dist_max *(int)* obj_use_dist_min *(int)* obj_use_pa_self *(int)* obj_use_pa_target *(int)* obj_use_pv_self *(int)* obj_use_pv_target *(int)* obj_use_limit (utilisations) *(int)* obj_impact_skill_id *(int)* obj_impact_pa_self *(int)* obj_impact_pa_target *(int)* obj_impact_pv_self *(int)* obj_impact_dam_min *(int)* obj_impact_dam_max *(int)* obj_equip *(bool)* (objet nécessitant d'être équipé pour être utilisé) obj_equip_pa *(int)* (nombre de PA dépensés pour équiper l'objet) obj_equip_defense *(int)* obj_equip_bonus_strength *(int)* obj_equip_bonus_const *(int)* obj_equip_bonus_dext *(int)* obj_equip_bonus_psy *(int)* obj_equip_bonus_pa *(int)* ground_id (int) obj_inshops (bool) (boutique ou pas)	# ground_id *(int)* mine_name *(string)* mine_infos *(string)* mine_capacity *(int)* mine_graph *(string)* (graphisme) mine_take_pa *(int)* mine_gain *(int)*
• **services**	• **ground_types**
service_id *(int)* service_name *(string)* service_infos *(string)* service_cp *(int)* service_pa *(int)*	type_id *(int)* type_name *(string)* type_infos *(string)* type_defense *(int)* type_move_pa (PA perdu) *(int)* type_take_pa (PA perdu) *(int)* type_gain *(int)*
• **skills**	• **towns_services**

skill_id *(int)*
skill_name *(string)*
skill_infos *(string)*

skill_up_dodge *(int)*
skill_up_impact_melee *(int)*
skill_up_impact_distance *(int)*
skill_up_training *(int)*
skill_up_hurt *(int)*
skill_up_psy *(int)*
skill_up_fight_melee *(int)*
skill_up_fight_distance *(int)*
skill_up_move *(int)*

town_service_id *(int)*
town_id *(int)*
service_id *(int)*

- **pnjs**

pnj_id *(int)*
pnj_type *(boolean)*
quest_id *(int)* (id de quete. 0:pas de quete)
pnj_dialon *(string)* (dial accueil)
pnj_dialshop *(string)* (dial boutique)
pnj_sell *(bool)*
pnj_buy *(int)* (val max achetée. <0:n'achete rien, =0:prend gratuit)
pnj_graph *(string)*
pnj_pos (*point*)

- **quests**

quest_id *(int)*
quest_dialon *(string)* (dial accueil)
quest_dialoff *(string)* (dial fin)
obj_id_get *(int)* (obj recherché)
obj_id_win *(int)* (obj offert)
quest_nbCP_win *(int)*
quest_nbobj_win *(int)*

- **characters_skills**

char_skill_id *(int)*
char_id *(int)*
skill_id *(int)*
skill_lvl *(int)*

- **grounds_visibles**

vis_id *(int)*
ground_id *(int)*
user_id *(int)*

- **events**

event_id *(int)*
user_id *(int)*
char_id *(int)*
event_name *(string)*
event_infos *(string)*
event_date *(date)*

- **send**

message_id
user_receive_id

eggs

id_egg
id_char_mum
id_char_dad
date_copulation **(timestamp)**
id_object

3. Liste des procédures réparties par groupe

Groupe	Procédure :
1	Déplacement dans la carte
2	Revenus journaliers
1	Capture de terrain
x	Utilisation objets (combats, soins, résurrection de quelqu'un, - PA)
1	Résurrection (aller dans la ville la plus proche)
1	Alliances : - Rentrer dans une alliance - Sortir d'une alliance - Virer quelqu'un de l'alliance
3	Messagerie
	Terrier :
1	- Salle de repos (- de PA, - de CP, + de PV)
3	- Banque (dépôt, retrait)
2	- Boutique : - achat : avec CP du Mimichant qui achète - rente : CP sur Mimichant
3	- Foire aux Mimichants - achat de mimi (- de PA, - de CP) - vente de mimi (- de PA, - de CP)
3	- Crieur public - poser une annonce (- de CP, - de PA) - lire une annonce (- de PA)
2	Mine : - minage (- de PA, + de CP, utilisation des compétences pour + ou - de CP) - épuisement (déplacer mine, fixer nouveau nombre de CP)
3	Copulation : - création oeuf sur femelle - envoi de message de demande - éclosion -> destruction oeuf, création Mimichant s'il y a de la surface vitale)
? à voir *(faite par le groupe 3)*	Transfert d'objet (- de PA)
2	Quêtes (validation de quêtes)
3	Création de Mimichant (procédure appelée 1 fois à la création du compte et à chaque fois qu'un oeuf éclos)
2	Objet sur terrain : - ramasser (- de PA) - déposer (- de PA)

4. Schéma relationnel