Rapport du projet Web : My New LIIfE

Téodora Hovi, Enola Ruffine, Alexis Bourget, Maël Martin

2018/2019

Table des matières

1 Introduction

1.1 Concept

My New LIIfE est un projet de visual novel portant sur la vie à l'ENSIIE. Il vise principalement les élèves de l'ENSIIE et vise à apporter une vision parodique et humoristique de l'école. Il s'agit d'une plateforme web permettant de suivre une histoire visuelle et textuelle dont le déroulement sera modifié par les choix de l'utilisateur. Le joueur peut, lors du déroulement de l'histoire, avoir des interactions positives ou négatives envers des associations et éventuellement les rejoindre ou les quitter.

Chaque histoire possède plusieurs fins. Les choix et les actions du joueur influencent la fin qu'il obtient. Effectuer certaines actions permet de débloquer des achievements (ou haut-faits). Les histoires terminées, les statistiques obtenues lors de ces histoires et les achievements débloqués sont affichés sur le profil de l'utilisateur.

Un des intérêts de la plateforme est la modularité apportée au niveau des histoires : le système est totalement général et de nouveaux visuels, textes et scénarios peuvent être ajoutés facilement.

1.2 Objectifs

Dans le cadre de cette démo, les objectifs fixés sont les suivants :

— Avoir une interface de connexion permettant aux joueurs de s'inscrire afin d'accéder aux histoires.

— Avoir un profil où l'utilisateur peut consulter et modifier ses informations personnelles, ainsi que consulter ses parties précédentes et ses haut-faits.

— Avoir un système d'histoires fonctionnel, avec une boucle de jeu complète, c'est-à-dire :

- Détecter si l'utilisateur a une histoire en cours
- Lui proposer d'en créer une si ce n'est pas le cas
- Reprendre sa partie sinon
- Pouvoir choisir entre plusieurs options et changer les statistiques de l'utilisateur en conséquence
- Détecter lorsqu'un achievement est validé et l'afficher à l'écran
- Lorsque l'histoire est finie, afficher une popup donnant les stats de la run puis ajouter l'histoire aux histoires finies. L'utilisateur peut alors créer une nouvelle histoire.

— Tendre vers une forte modularité et flexibilité : a terme, il devrait être possible d'ajouter des histoires depuis le compte administrateur.

2 Page d'accueil

Problématique La page d'accueil du site est le premier challenge rencontré lors de la réalisation de ce projet. Sa visée était en effet double : pour les utilisateurs non inscrits, elle doit servir de présentation du jeu et inciter les visiteurs à s'inscrire. Pour un visiteur régulier, elle agit d'abord comme un portail de connexion puis comme un récapitulatif de son expérience de jeu.

Pour des raisons de confort de l'utilisateur, mais aussi de sécurité et de concision, nous souhaitions conserver une unique page pour remplir ces deux rôles. La page doit donc être générée différemment si l'utilisateur est connecté ou non.

Solution `print_functions.php` regroupe un ensemble de fonctions permettant d'afficher les différents éléments des pages du site web. Certaines, comme `printHeader()`, on simplement visée à éviter les redondances dans le code et à afficher de manière plus cohérente les éléments récurrents. Par exemple, si on souhaite modifier le header du site, il suffit de modifier la fonction correspondante.

D'autres fonctions permettent d'afficher la version "connecté" ou "visiteur" du site selon l'état de la variable de session `$_SESSION['name']`. Celle-ci n'est définie que si l'utilisateur s'est identifié avec succès à l'aide du formulaire de connexion. La fonction `printMain()` effectue donc une vérification sur cette variable puis appelle, accordément au résultat du test, soit `printMainLogOut()` soit `printMainLogIn()`.

3 Connexion, système de sessions

Problématique L'authentification et la gestion de compte étaient deux éléments imposés au projet. Pour les implémenter, nous nous sommes appuyés sur la mécanique de session présente de base introduite par PHP.

Solution Le déroulement de l'opération de connexion est le suivant :

1. L'utilisateur 'submit' le formulaire de connexion
2. Ses informations sont redirigées vers une page `login.php` par la méthode POST.
3. La fonction `checkUser()` est appelée. Cette fonction récupère d'un côté les données de l'utilisateur et de l'autre un éventuel mot de passe stocké en base de données correspondant au pseudo fourni par l'utilisateur. Une comparaison couronnée de succès initialise la variable de session `$_SESSION['name']` avec la valeur `$_POST['pseudo']`, un échec renvoie un message d'erreur.

Selon le retour de `checkUser()`, l'utilisateur est soit redirigé vers l'accueil où il est désormais connecté, soit reste sur la page où s'affiche un message d'erreur. Toute requête de l'utilisateur connecté se fait par la suite via cette variable qui n'est réinitialisée que par déconnexion ou si les cookies sont vidés.

Le système de déconnexion est extrêmement similaire. La fonction `logOut()` située dans `print_function.php` appelle `session_destroy()` avant de rediriger l'utilisateur vers l'accueil.

4 Page de jeu

Problématique Dans le cadre d'un projet se présentant comme un "visual novel en ligne", la partie jeu du site s'annonçait comme la plus important mais également la plus problématique. Le temps moyen passé sur chaque étape étant très faible, de l'ordre de la seconde, et de nombreuses images se succédant, recharger la page à chaque choix aurait profondément nui à l'UX et aurait fait perdre beaucoup de son intérêt au sujet.

4.1 Structure d'une histoire

Afin de justifier les choix ayant été fait afin de répondre à la problématique suivante, il est nécessaire de se pencher sur le modèle adopté pour la gestion des histoires. Un court lexique est proposé ci-dessous afin d'en faciliter la compréhension.

Node Un node désigne une étape d'une histoire. Une histoire est constituée d'une succession de nodes dont le contenu est principalement : un texte, qui peut être une voix narrative ou un dialogue, une image de background et une image de foreground.

Choix Les choix permettent de naviguer entre les nodes. Un choix relie deux nodes de manière unique. Ici, choix ne signifie pas forcément dilemme : certains nodes, notamment dans les passages narratifs, ne proposent qu'un seul choix.

Fin Une fin est un node qui ne possède pas de choix. Un node sans choix ne peut correspondre qu'à une unique fin.

Prérequis Un prérequis est une condition nécessaire à l'affichage d'un choix. Il s'agit d'une condition sur le node d'arrivée du choix. Il peut porter sur l'appartenance du joueur à une association dans l'histoire en cours, sur son affinité avec les différentes associations ou sur les variables d'alcoolémie, de fantomisation et de sérieux.

4.2 jQuery et AJAX

Les informations concernant les histoires n'étant pas en clair dans le front du site, l'utilisation d'AJAX est apparue comme nécessaire. L'intégralité des actualisations de la base de données, des statistiques de l'utilisateur, de l'histoire en cours se fait donc en temps réel afin de fournir une expérience confortable au joueur tout en évitant un décalage entre les informations en temps réel de l'utilisateur et celles stockées en base de données.

Etant donnée que cette technologie n'avait été qu'évoquée en cours, sa prise en main a constitué un enjeu majeur dans l'avancée du projet.

Les scripts js contenant les appels jQuery sont au maximum situés dans la balise `<head>` des pages concernées. Ces derniers sont réduits au strict minimum favorisent l'utilisaion des fonctions situées dans `game_functions.js` qui effectuent le gros des appels jQuery. Ces appels constituent principalement de `$.post` et de `$.getJSON`, qui exécutent les scripts PHP regroupés dans le dossier `php_script` puis rechargent de manière appropriée le contenu de la page.

4.3 Création d'une histoire

La première étape est de gérer la création d'une nouvelle histoire. Lorsque la page de jeu est chargée, une requête à la DB vérifie si l'utilisateur possède une histoire en cours. Si c'est le cas, le node auquel il se trouve, ses statistiques et son avancée dans l'histoire sont restaurés et il peut reprendre sa partie.

Si l'utilisateur ne possède pas d'histoire en cours, la page charge alors les histoires disponibles (pour l'instant sans appel à la DB, il s'agit d'une fonctionnalité supplémentaire qui consitue une des pistes d'amélioration) et propose à l'utilisateur d'en lancer une.

Le système pour afficher l'une ou l'autre des configurations est similaire à celui utilisé sur la page d'accueil, et on pourra retrouver les fonctions correspondantes dans `print_functions.php`.

Cliquer sur le bouton "Jouer" déclenche l'insertion de la base de données d'une nouvelle histoire courante pour l'utilisateur. Les statistiques sont toutes initialisées à 50. L'argument passé à la fonction `initNewStory(int)` correspond au premier node de l'histoire.

4.4 Actualisation des visuels et des statistiques

L'actualisation de ce que voit l'utilisateur se fait via une fonction `actualizeFront()` qui est déclenchée dès que la page a fini de se charger, et à chaque fois que l'utilisateur effectue une action ayant un impact sur la base de données.

Le rôle de cette fonction est d'afficher les informations correspondant au node courant, c'est à dire l'étape de l'histoire sur laquelle se trouve l'utilisateur. Ces informations sont :

- Les images (foreground et background)
- Le texte
- Les statistiques actuelles de l'utilisateur
- Un éventuel achievement débloqué par l'utilisateur
- Les choix accessibles à l'utilisateur en fonction du node sur lequel il se trouve et de ses statistiques

La récupération des deux premiers item se fait via un script `getBg.php` qui contrairement à ce qu'indique son nom, renvoie un JSON contenant le background, le foreground et le text correspondant au node actuel. Ces trois élément sont ensuite placés dans les divisions correspondantes.

L'affichage de ces trois informations simultanément dans un même espace a d'ailleurs constitué une autre problématique importante puisque des notions un peu plus poussées de CSS telles que le z-index, le positionnement relatif et absolu de divisions et les attributs align-self de Flex-Box ont dû être mobilisées.

4.4.1 Affichage des choix

Le caractère théoriquement infini du nombre de choix possibles pour un même node, et les conditions particulières d'affichage de ces derniers ont posé problème un moment. La récupération des choix correspondant à un node est triviale : il suffit d'effectuer une jointure sur l'id du node courant et l'id de départ du choix.

Cependant, ces choix ne doivent être affichés que si l'utilisateur remplit les prérequis, ceux-ci ne portant pas sur les choix, mais sur les noeuds d'arrivés. Une fonction `checkRequirements($request, $user_stat)` a donc été implémentée, toujours dans `database_access.php`. Cette fonction est appelée par le script `getChoices.php` qui effectue une requête préparée successivement sur tous les choix possibles pour le node en cours.

La complexité de cette opération fait que `getChoices.php` contient des appels à la BDD alors que l'objectif était qu'elles soient toutes regroupées dans `database_access.php`.

4.4.2 Affichage des achievements

Un script permettant de vérifier si le node courant débloque un achievement inédit est également exécuté et affiche une pop-up donnant quelques informations. L'apparition et la disparition de la pop-up se fait via deux fonctions js trouvables dans `game_functions.js`.

4.5 Actualisation de la base de données

Là où `actualizeFront()` récupère les informations de la base de données pour les afficher, `actualizeBack()` fait l'inverse et met la base de données à jour en fonction des actions de l'utilisateur, la principale étant le clic sur un choix. Le clic est détecté via jQuery, qui envoie l'objet cliqué à `actualizeBack()` qui en extrait l'id. L'id de la div correspondant au choix cliqué transmet en effet l'information sur le node d'arrivée du choix.

L'id du choix devient donc l'id du nouveau node courant. Cette information est envoyée à un script PHP qui met à jour l'histoire courant du joueur avec le nouveau node. Une fonction de callback permet ensuite de modifier les statistiques de l'utilisateur selon les modificateurs spécifiés dans le nouveau node.

Enfin, afin de combler le nouveau décalage entre l'affichage et la base de données, `actualizeFront()` est appelée et révèle à l'utilisateur les nouveaux visuels et l'impact de son précédent choix.

4.6 Fin d'une histoire

La fin d'une histoire est détectée lorsque la liste des choix possibles à l'utilisateur est nulle. Alors le contenu de la div `choices` est remplacé par un bouton "Finir l'histoire". Ce bouton va afficher une popup qui révèle à l'utilisateur la fin obtenu et ses différentes statistiques, et parallèlement upload l'histoire en cours (désormais finie) dans la liste des histoires finies. Puis l'histoire actuelle du joueur est supprimée afin qu'il puisse en recommencer une.

Remarque Lors des tests du site on peut remarquer que l'upload des histoires dans la liste des histoires finies a été mis en commentaire. Il s'agit d'une mesure prise à cause d'un mauvais design de la base de données qui interdit à un utilisateur de finir deux fois une même histoire un même jour avec des statistiques identiques, ce qui était le cas durant les tests. Cette fonction est cependant tout à fait opérationelle.

5 Ajout de nouveaux scénarios et histoires

L'obejctif de notre site était d'être très adaptatif et de faciliter l'ajout de nouvelles histoires simplement en les insérant dans la base de données. Etant donné le caractère fastidieux des requêtes d'insertions devant être tapées, nous avons mis au point un parser en Python qui, sur la base de fichiers CSV contenant les informations nécessaires à l'histoire, génère les requêtes SQL nécessaires pour l'insertion des nodes, choices et ends correspondant. Son utilisation est détaillée dans le README et des fichiers d'exemple sont fournis.

6 Difficultées rencontrées

Si le rendu s'est bien fait sous Docker, un fichier "version_perso" a également été déposée. En effet, suite à des problèmes avec Docker, seul un membre du groupe a réussi à s'en munir. Afin que tout le monde puisse contribuer au projet, celui-ci a donc été principalement réalisé sur les sites perso et la DB MySQL fournis par Arise. La transition ne s'étant effectuée que tardivement, la version perso demeure à ce jour plus stable que la version Docker. Afin de fournir une meilleure expérience de test, nous avons jugé pertinent de rendre également cette version qui reflète mieux la qualité du travail que nous espérions fournir.

Comme spécifié dans le README, le lien du site perso est "hovi.iiens.net/MyNewLiife/php_accueil.php".

7 Conclusion

7.1 Pistes d'amélioration

Le projet MyNewLIIfE étant un projet ambitieux avec de nombreuses possibilités d'expansion, il était évident que nous ne parviendrions pas à toutes les intégrer durant ce projet. Cependant, nous comptons réellement poursuivre le développement dans le futur, en implémentant notamment la possibilité de voir les dernières fins des utilisateurs dans le Panthéon, et en offrant plus d'option de personnalisation de personnages. Evidemment, nous comptons également continuer à implémenter les histoires, mais celles-ci ne nécessitent déjà plus de travail de développement réel.

Un autre point qu'il sera nécessaire d'améliorer si MyNewLIIfE prend de l'ampleur est la sécurité du site. Ce dernier est actuellement faible aux injections MySQL et la transmission des informations sur les noeuds suivants via l'id le rend très sensible à la triche. De même les pages censées n'être accessibles que par connexion le sont directement par l'adresse. Même si cela n'influe pas sur la sécurité des données, il sera urgent de le corriger.

Mais malgré le fait que nous n'avons pas pu accomplir tout ce que nous avions prévus, nous sommes néanmoins fiers de ce que nous avons accompli et de l'expérience acquise dans des domaines peu abordés en cours, notamment AJAX et jQuery.