

Projet Web - Rapport

Joséphine Barthel - Ruitong Xu - Florent Amiable - Ibtissam Liedri

Mai 2018

Contents

1	Étude de l'existant	3
2	Définition des besoins de l'utilisateur	3
3	Solutions techniques	4
4	Plan du site	5
5	Base de données	6
6	Architecture logicielle	10
7	Intégration du Google Calendar	11
8	Notifications par email	11
9	Prolongements	11

1 Étude de l'existant

Guilde

Guilde est l'association de jeu de rôle de l'ENSIIE. Les membres de cette association se réunissent régulièrement en groupes d'en général 4 à 6 personnes pour pratiquer le jeu de rôle.

Ces réunions durent quelques heures et peuvent être organisées de façon occasionnelles ou récurrentes (par exemple hebdomadaire). Dans la suite de ce document, on fera référence à ces réunions comme des "tables" ou des "tables de jeu de rôles".

Organisation d'une table

Pour organiser une table, un membre de Guilde commence par envoyer un mail à la mailing list de l'association en indiquant quel type de jeu il souhaite organiser et quels horaires sont disponibles. Les joueurs qui sont intéressés répondent indiquant leurs disponibilités. Une fois qu'il a rassemblé suffisamment de joueurs, le créateur de la table (qu'on appellera "maître du jeu" par la suite) cherche un horaire auquel tout les joueurs sont disponibles. Pour cela, il utilise en général le site doodle.com. Un fois qu'un horaire commun a été trouvé, un évènement est créé sur le Google Calendar de l'association.

Ce processus prend en général quelques jours et implique l'utilisation de trois services différents. Le but de cette application est de simplifier le processus en regroupant tous ces services.

Trouver un maître du jeu

Parfois, il arrive qu'un membre de Guilde souhaite jouer à un jeu spécifique mais ne le connaisse pas suffisamment pour jouer le rôle de maître du jeu. Il doit alors chercher, parmi les membres de l'association, quelqu'un capable de proposer ce jeu.

Échange de documents

Enfin, les joueurs ont parfois besoin de certains documents pour participer aux tables (comme par exemple des fiches de personnage, des cartes...). Ces documents sont en général échangés par mail, ou via le wiki de l'association.

2 Définition des besoins de l'utilisateur

L'utilisateur de cette application est un membre de l'association de jeu de rôle Guilde. A travers un sondage auprès des membres de l'association, nous avons établi les besoins utilisateurs comme suit :

- Proposer une table aux autres utilisateurs en indiquant quel jeu et quels horaires sont proposés
- Être informé des tables qui sont proposées
- Indiquer son intérêt pour une table qui a été proposée
- Ajouter les tables qui le concerne à son agenda électronique (type Google Agenda)
- Connaître quels jeux peuvent faire jouer les autres utilisateurs
- Mettre des fichiers à disposition des autres utilisateurs

On réfléchit maintenant aux solutions techniques qu'on va proposer pour répondre à ces besoins, tout en respectant les contraintes externes.

3 Solutions techniques

Répondre aux besoins utilisateurs

Pour répondre au premier besoin on choisit d'utiliser un formulaire HTML pour permettre au maître du jeu de renseigner les informations de sa table.

On aura besoin d'une base de données pour stocker la liste de toutes les tables qui ont été proposées. On utilisera PostgreSQL comme SGBD et pgAdmin pour simplifier la gestion de la base.

Pour informer tous les utilisateurs des tables qui sont proposées, on veut continuer à utiliser la mailing list existante pour ne pas disrupter les habitudes des membres de l'association. On décide donc que le site enverra des mails automatisés à la mailing list.

Pour permettre aux utilisateurs d'indiquer leur intérêt pour une table, on choisit d'utiliser un système de commentaire comme sur un blog. Ceci à l'avantage de permettre aux utilisateurs de demander des précisions ou de préciser leurs disponibilités.

Google Calendar

Google Calendar propose une API qui permet entre autre d'insérer des événements dans un Calendar existant via une requête HTTP. On souhaite que notre site puisse faire des requêtes de ce type pour insérer des événements dans le Calendrier de l'association. Encore une fois, il s'agit de construire sur l'existant pour ne pas disrupter les habitudes des utilisateurs.

Fonction de recherche

On aura besoin d'une fonction de recherche sur les utilisateurs. En conséquence, il faut que les utilisateurs soient capable de s'identifier pour renseigner leurs

informations personnelles - on va ajouter à notre base de données des informations sur les utilisateurs. On aura aussi besoin d'un système d'authentification sécurisé.

Mise en ligne de fichiers

Enfin, on veut que notre site dispose d'un système de mise en ligne de fichiers. Ces fichiers seront liés au compte utilisateur, et pourront être liés à une table spécifique si l'utilisateur le souhaite. On utilisera l'input de type "file" qui est de base dans php et on stockera les documents sur le serveur qui héberge tout le site.

Environnement de développement

Notre environnement de développement utilisera le stack qui a été proposé par les responsables du projet : Linux - Apache - PostgreSQL - PHP. Ces différents éléments seront inclus dans des containers Docker. On ajoute dans un container supplémentaire pgAdmin4 pour simplifier la gestion de la base de donnée. On a choisit d'utiliser Bootstrap pour le CSS, aucun des membres du groupe n'ayant la compétence ou l'intérêt pour faire un CSS maison.

Style de code

Il a été décidé de coder le projet en anglais, ce qui présente beaucoup d'avantages comme par exemple éviter les ambiguïtés (on utilise le mot "game" pour faire référence à une table) et permettre d'avoir de l'aide sur des sites anglophones comme Stack Overflow (ce qui n'a finalement pas été nécessaire mais aurait pu). De plus, c'est un bon entraînement pour le milieu professionnel qui nous l'imposera certainement.

4 Plan du site

Le sondage montrait que l'accès rapide au calendrier était la priorité pour nos utilisateurs, nous avons donc choisi de l'afficher directement sur l'accueil.

La page Liste des tables contient la liste des toutes les tables : un administrateur peut les supprimer si nécessaire.

La page Créer une table contient un formulaire qui permet de renseigner les informations sur une table : nom de la table, système de jeu qui va être utilisé, durée en nombre de séances, horaires proposés, description, fichiers associés. Le formulaire est vérifié en JavaScript. L'ajout d'horaire et de fichiers se fait également en JavaScript puisque chaque ajout altère immédiatement la page pour indiquer ce qui va être envoyé à la base de données.

La page Voir les infos d'une table affiche les informations d'une table déjà créée. N'importe quel utilisateur peut télécharger les fichiers ou laisser un commentaire. Le maître du jeu peut ajouter un utilisateur qui a laissé un commen-

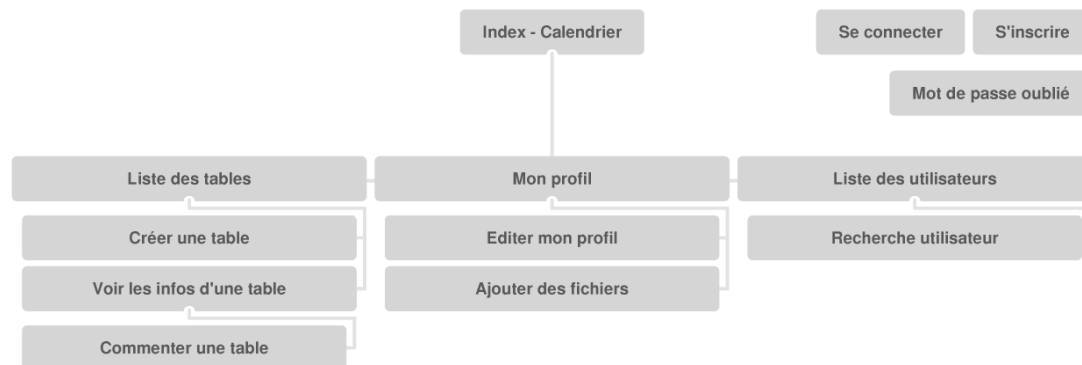


Figure 1: Plan du site

taire et qui n'est pas déjà joueur ou maître du jeu sur la table. Un administrateur peut supprimer un commentaire.

La page liste des utilisateurs contient la liste des tous les utilisateurs. N'importe quel utilisateur peut y lancer une recherche sur le nom d'utilisateur, le pseudo utilisateur, les tables proposées, les systèmes qu'il peut proposer (sous le nom "système meujeuté").

La page du profil affiche un profil utilisateur. Si l'utilisateur connecté est le propriétaire de ce profil, il peut l'éditer et lui ajouter des fichiers. N'importe quel utilisateur peut voir les informations et les fichiers de l'utilisateur. Un administrateur peut supprimer les fichiers.

Les pages Se connecter, S'inscrire et Mot de Passe oublié sont accessibles à tout utilisateur non connecté.

5 Base de données

Stockage des horaires

Un des challenges les plus intéressants du projet a été de stocker les informations relatives aux horaires des tables. Rappelons qu'une table peut être organisée de façon occasionnelle ("Le lundi 21 mai de 16h à 20h") ou récurrente ("Tous les lundis, de 16h à 20h").

De plus, un maître du jeu doit pouvoir proposer plusieurs horaires différents, en combinant les deux types si nécessaire ("Tout les mardis, de 20h à minuit ET le jeudi 8 mai, de 12h à 20h").

Le terme "Oneshot" désigne dans tout le code un horaire occasionnel. Le terme "Reccurent" désigne un horaire récurrent (par exemple hebdomadaire ou bihebdomadaire). Le terme Schedule désigne n'importe lequel de ces deux horaires.

Il convient d'abord de se demander si on souhaite conserver cet héritage dans

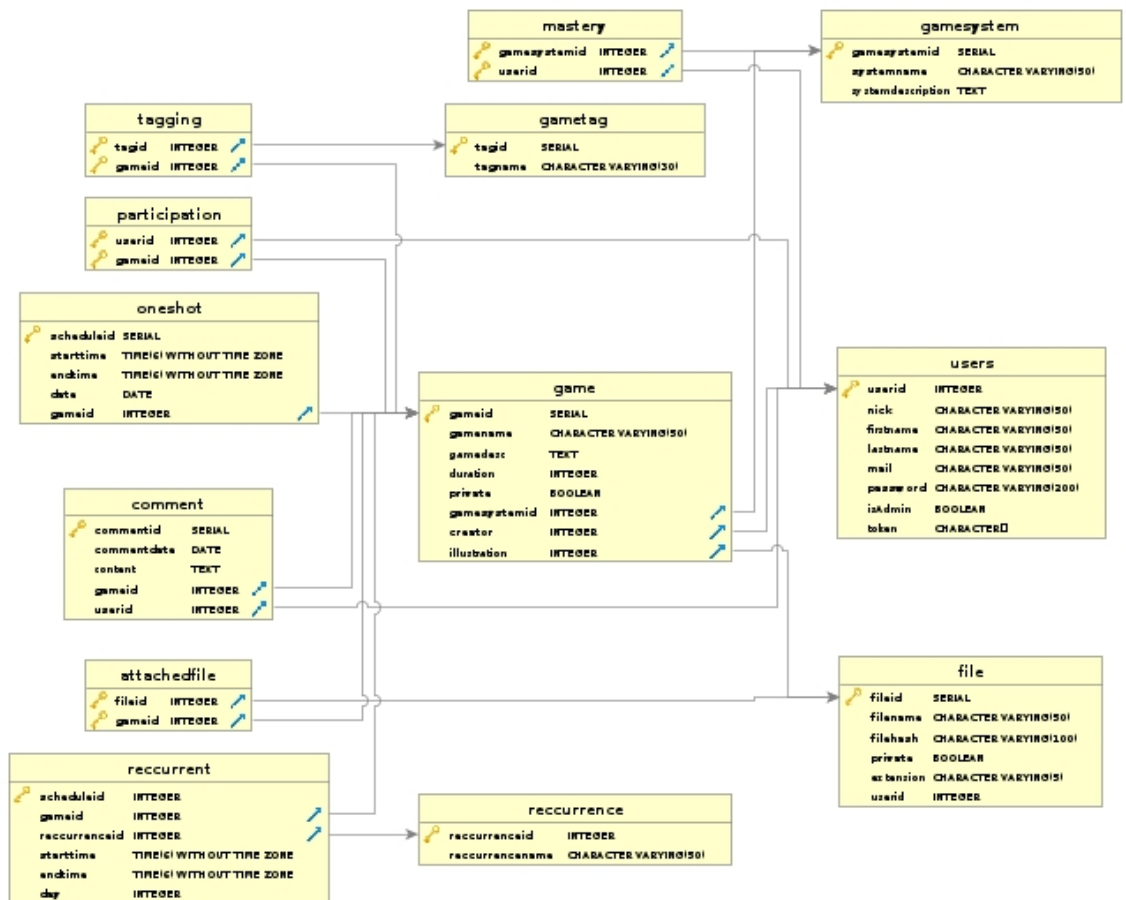


Figure 2: Le MCD de notre base de données

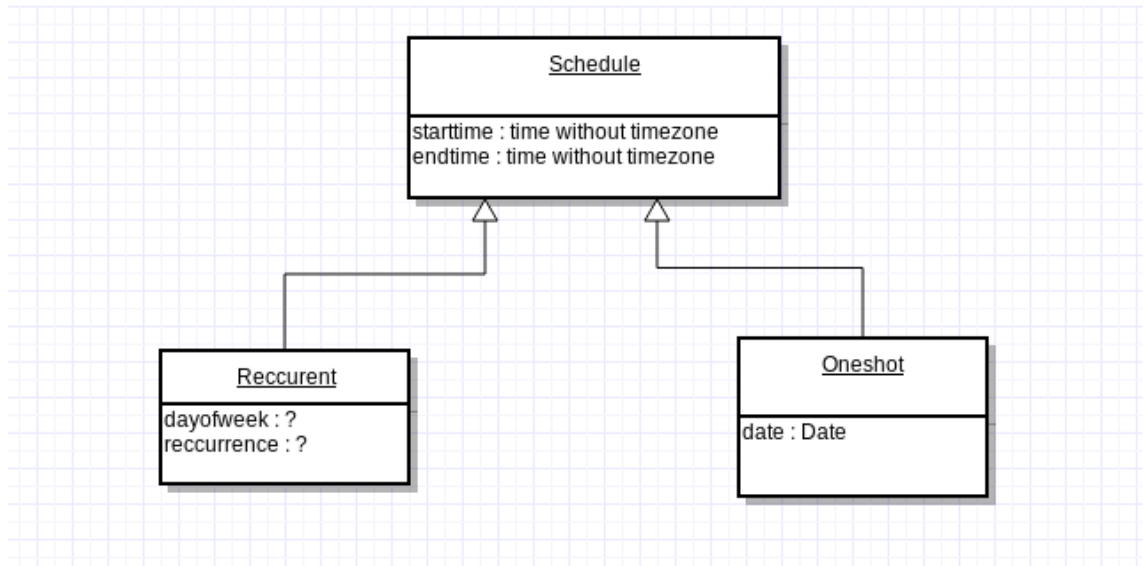


Figure 3: La structure de données des horaires - initial

la base de données : en effet, oneshot et recurrent sont mutuellement exclusifs : il n'existe pas d'horaire à la fois récurrent et occasionnel. Cependant, il existe des attributs communs : starttime et endtime (heure de début et heure de fin". On a finalement choisi de créer deux tables séparée dans la base de données (sans héritage) pour ne pas multiplier les jointures.

Dans le code PHP, on a une seule classe appelée "Schedule" qui gère à la fois l'insertion des oneshots et des recurrent avec ses méthodes statiques insert_recurrent() et insert_oneshot().

Ceci était une erreur car crée du code pas très "orienté objet", comme ce qui suit.

```

// contr leur
$onshots = $game->one_shot_schedules();
$reccurrents = $game->reccurrent_schedules();

// vue
foreach($onshots as $oneshot)
{
    echo " <li class='list-group-item'> Le ".
        date("d/m/Y", strtotime($oneshot->date))."
        de ".$oneshot->starttime."      ".$oneshot->endtime." </li>";
}
foreach ($reccurrents as $reccurrent)
{
    $reccurrence=Reccurrence::id_to_reccurrence($reccurrent->reccurrence
  
```



```

        echo "<li class='list-group-item'> Le ".$int_to_dayofweek($reccurrent
            de ".$reccurrent->starttime."      ".$reccurrent->endtime.">
    }

```

La solution aurait été d'avoir deux classes Oneshot et Reccurrent hériter de Schedule et implémenter toutes les deux leur version d'une méthode display(). Notre code ci-dessus devient :

```

$schedules = $game->schedules;

// ...
foreach($schedules as $schedule)
{
    echo "<li class='list-group-item'>".
        $schedule->display()."
    </li>";
}

```

Stockage des heures

Enfin, se pose la question du type de variable. On choisit d'utiliser le type "time without timezone" de pgsql pour stocker les heures : toutes les activités de l'association ayant lieu dans une zone géographique minuscule, on préfère s'épargner la complexité de gérer les fuseaux horaires. Le jour de la semaine à d'abord été une enum, mais on s'est rendu compte que le PHP et le JS suivaient la convention suivante : 0 = dimanche, 1 = lundi... On a donc choisi de stocker le jour de la semaine comme un entier inférieur à 7 et d'utiliser une méthode int_to_dayofweek(int) en PHP pour faire la conversion.

Stockage des récurrences

Finalement la récurrence. Comment représenter qu'un évènement arriver une fois par semaine, une fois par mois, une fois toutes les trois semaines ? On a pris la décision de créer une table spéciale, recurrence, qui contient une id et une chaîne de caractère qui indique la fréquence ("toutes les semaines", "toutes les deux semaines")...

Avec du recul, on aurait pu simplement stocker un nombre de semaine entre chaque évènement et créer une autre méthode de conversion en PHP.

Sécurité

On a bien veillé à ne pas stocker les mots de passe dans la base de donnée en texte normal : à la place, on a hash chaque mot de passe au moment de l'inscription en utilisant la fonction PHP `password_hash`. Lors de l'authentification, le hash du mot de passe entré est comparé contre

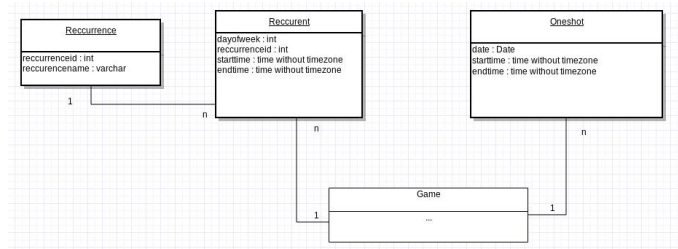


Figure 4: La structure de données des horaires - version fianle

6 Architecture logicielle

MVC

On choisit d'utiliser l'architecture Modèle-Vue-Contrôleur pour notre site. Ceci signifie que la partie du code en charge d'afficher les informations est totalement séparée de celle en charge de faire les requêtes aux serveurs et de faire fonctionner la logique de l'application. Ces deux parties sont mises en relation par le contrôleur, qui appelle les fonctions de l'un et de l'autre pour faire le lien entre les deux.

Cette architecture a les avantages suivants

- le site est simple à maintenir : un changement esthétique ne peut pas avoir d'impact sur le fonctionnement et inversement
- l'équipe peut collaborer sur le code car la personne qui s'occupe de la mise en forme ne touche pas aux fichiers qui concernent le fonctionnel
- une nouvelle personne peut intégrer le projet facilement : le code est séparé en fichiers d'une taille réduite qui ont une seule fonctionnalité.

Arborescence des fichiers

- public/ : contient les pages du site web (les contrôleurs).
 - * actions/ : pages de validations des formulaires
 - * js/ : fichiers JavaScript.
 - * css/ : feuilles de style.
 - * logos/ : images et logos.
- src/app/ : contient la logique de l'application. Ces fichiers ne sont pas accessibles directement par les utilisateurs
 - * helpers.php : contient l'autoloader, la fonction de connexion à la base de données, et des fonctions utilitaires générales.
 - * models/ : contient les modèles sous forme de classe PHP.

- * views/ : contient les vues sous forme de fichiers principalement HTML.
 - layouts/ : contient les têtes et les pieds de pages pour les différents types d'utilisateurs (connectés ou non).
 - * class/ : contient des classes statiques qui regroupent certaines méthodes utiles pour un domaine (authentification, envoi de mail...)
- documents/ : contient les documents mis en ligne par les utilisateurs
 - data/ : contient le fichier .sql avec la base de données

7 Intégration du Google Calendar

Il est facile d'intégrer le Google Calendar dans le site au moyen d'un élément HTML de type iframe : le code est fourni directement sur le site Google Calendar. Malheureusement, nous n'avons pas eu le temps d'intégrer les requêtes d'insertion : cette fonctionnalité est donc remise à la partie "Prolongement".

8 Notifications par email

En PHP, la fonction mail() prend en paramètre l'expéditeur, le sujet et le corps du message. Elle appelle ensuite le programme sendmail (si on est sous Linux comme c'est le cas avec notre environnement de développement) qui envoie le mail au destinataire.

Malheureusement, notre serveur sous Docker à l'intérieur d'une machine virtuelle, ce qui complexifie le processus : la machine virtuelle ne peut directement envoyer des mails par Internet.

Pour cela, on a d'abord essayé d'ajouter un container qui contient un serveur SMTP. Ceci c'est avéré compliqué car Linux n'utilise pas SMTP de base : il aurait été nécessaire d'installer ssmtp sur la machine virtuelle ce qui s'est avéré trop compliqué.

Nous avons alors tenter d'installer le MTA (mail transfert agent) Postfix sur la machine hôte et de lui faire écouter les ports par lequel la machine virtuelle envoie les mails : ceci a également échoué.

Devant la perte de temps que représentait ce problème nous avons décider de le remettre aux prolongements du projet. En conséquence, les fonctionnalités qui nécessitent des mails automatisés (réinitialisation du mot de passe, signalement des nouvelles tables...) ne marchent pas encore.

9 Prolongements

A ce jour le projet n'est pas encore tout à fait livrable. Les éléments suivants doivent absolument être ajoutés pour répondre aux besoins principaux des utilisateurs :

- Faire fonctionner l'envoi de mail
- Permettre les insertions dans le Google Calendar
- Utilisation du LDAP Arise à la place du système d'authentification annuel afin de permettre aux utilisateurs de se connecter avec leur compte Arise (association de réseau de l'école).

Ce dernier élément a été remis à plus tard à cause de l'exigence d'un système d'authentification "maison" mais nous estimons qu'il s'agit d'un élément essentiel pour permettre aux membres de l'association d'utiliser notre site avec un minimum d'effort.

L'envoi de mails et de requêtes automatiques à Google Calendar sont absolument nécessaires car si les utilisateurs doivent passer par les même étapes qu'avant en plus du site, ils n'utiliseront pas le site.

Parmi les éléments secondaires qui pourraient être ajoutés, on compte :

- La possibilité de retirer des horaires à une table en cours de création. Dans le formulaire de création de table actuel, si un utilisateur se trompe lors de l'ajout d'un horaire il est obligé de recharger la page et de recommencer. Ceci n'est pas souhaitable.
- La possibilité d'archiver une table une fois qu'elle est lancée. Avec le site actuel, les propositions de tables vont s'accumuler jusqu'à ce que la liste devienne illisible. Une solution serait de permettre au maître du jeu d'archiver sa table une fois qu'il a trouvé des joueurs. Les tables archivées ne sont pas affichées par défaut dans la liste des tables.
- une fonction de recherche dans la liste des tables. Les tables auraient des tags qui renseignent sur leurs thématiques, leur style, leur univers (par exemple : "d&d", "warhammer", "narratif", "exploration"). Il est possible de faire une recherche dans les tables par tag.
- une illustration pour une table. Au lieu d'un simple lien, chaque table dans la liste des tables est représenté par une vignette avec son nom, son système et son illustration.

Vous pouvez remarquer que certaines de ces fonctionnalités ont déjà du support dans la base de données : c'est parce qu'elles ont été prévues au départ et on été abandonnées (pour ce rendu du moins).