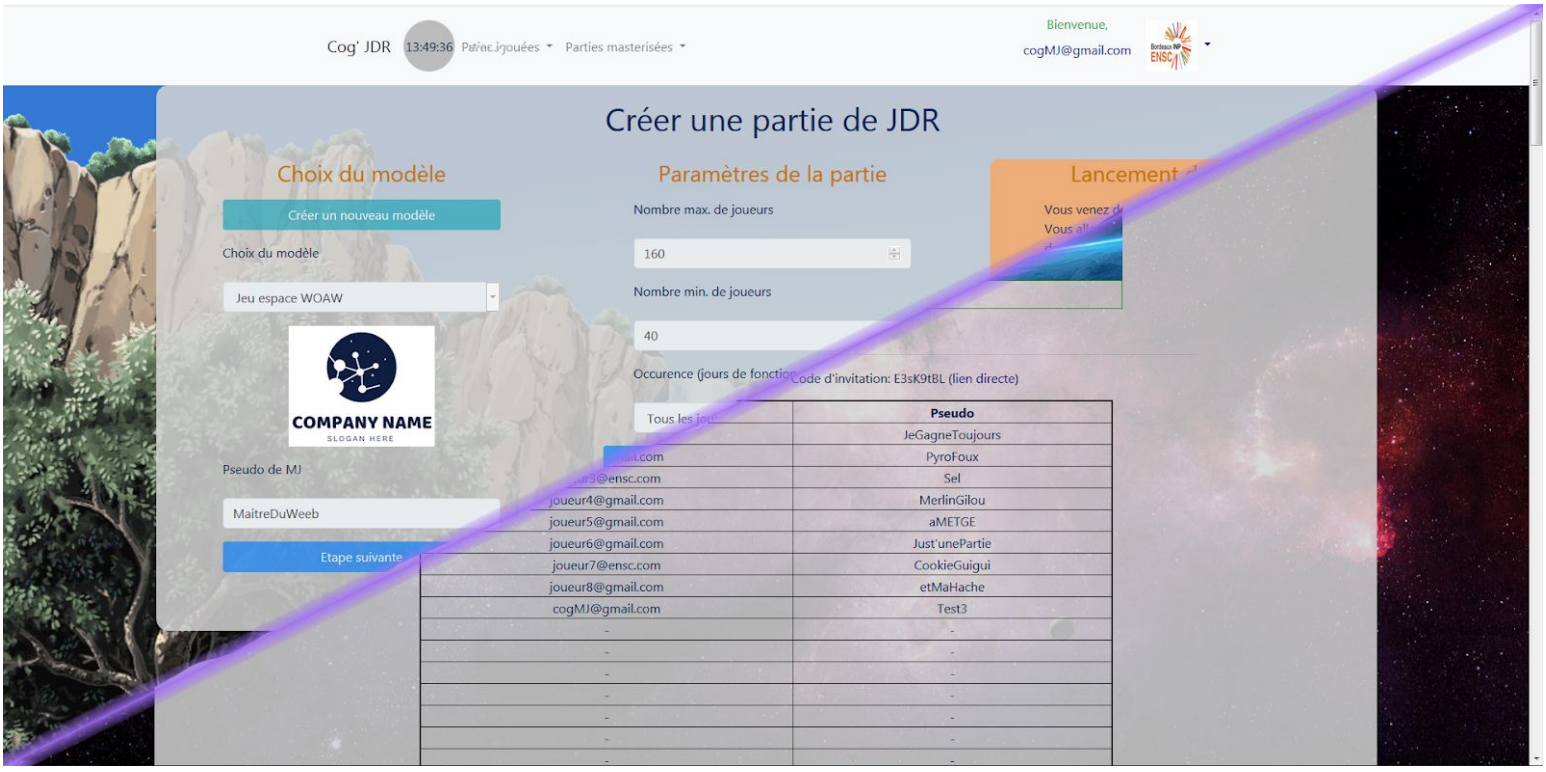
Cog'JDR

Rapport de projet Web



Auteurs

Hugo FOURNIER et Célestin GRENIER

Professeurs référents

CLERMONT Edwige, PESQUET Baptiste et THEVIN Lauren

Adresse du site hébergé :

https://cogjdr.000webhostapp.com/

Adresse du répertoire GitHub :

https://github.com/HugzFnr/ENSC-S6-Web-CogJDR

Introduction

Qu'est-ce que Cog'JDR ?

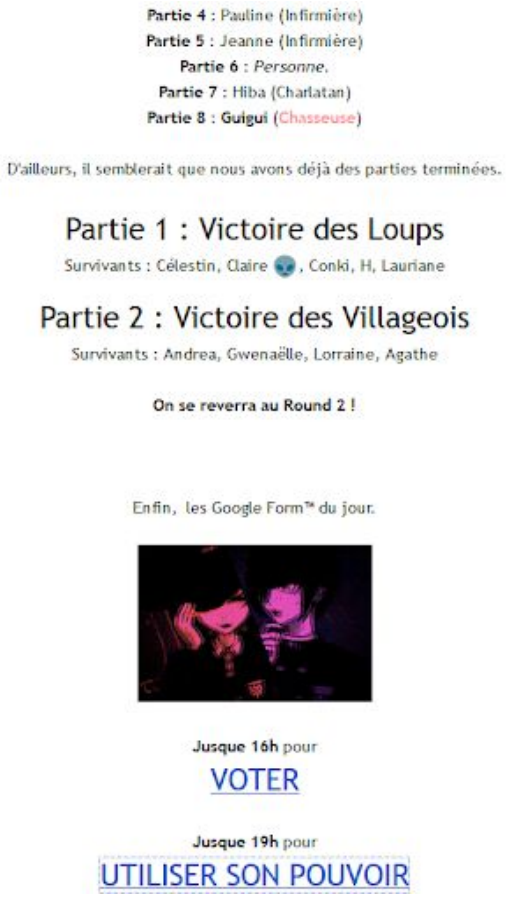
Contexte et problématique

Depuis au moins 2 années maintenant, les élèves de l'ENSC ont beaucoup apprécié des Jeux de Rôles (JDR) par boîte mail. Dans des parties regroupant plus des centaines de joueurs, ils ont comploté, se sont trahis, ont éliminé leurs adversaires, ont perdus, ont gagné, tous ensembles dans des équipes et des alliances. Ces jeux ont participé, notamment pour les 1A, à une meilleure cohésion au sein des élèves de l'école (et à une boîte mail trop remplie).

Concrètement, cela se passe grâce à des mails, des espaces de discussion comme Messenger et des Google Forms. Et cela repose sur les épaules (et les tableurs) d'un Maître du Jeu (MJ) généreux qui sacrifie une très grande partie de sa semaine pour pouvoir gérer la partie aux règles complexes et toujours novatrices (sans compter le temps pris pour concevoir le jeu auparavant). Enfin, les informations sont mal regroupées et cela crée souvent des confusions et des erreurs aussi bien côté joueur que côté MJ.

Dans l'objectif de permettre aux élèves de jouer plus souvent à pleins de différents JDRs de ce type, de créer eux-même leurs JDR et de les faire jouer facilement, nous avons identifié une problématique : quel outil concevoir pour faciliter la gestion de JDR type boîte mail, côté joueur et surtout côté MJ ?

Mail de conclusion du Trône de Cognitique, JDR masterisé en Octobre 2018 par Adrien Metge



Partie 4 : Pauline (Infirmière)
Partie 5 : Jeanne (Infirmière)
Partie 6 : *Personne.*
Partie 7 : Hiba (Charlatan)
Partie 8 : **Guigui** (Chasseuse)

D'ailleurs, il semblerait que nous avons déjà des parties terminées.

Partie 1 : Victoire des Loups

Survivants : Célestin, Claire , Conki, H, Lauriane

Partie 2 : Victoire des Villageois

Survivants : Andrea, Gwenaëlle, Lorraine, Agathe

On se reverra au Round 2 !

Enfin, les Google Form™ du jour.

Jusque 16h pour
VOTER

Jusque 19h pour
UTILISER SON POUVOIR

Un des mails récapitulatifs du Ultimate Loup Garou, JDR masterisé en Janvier 2019 par Younès Rabii

Solution

Dans le cadre du projet Web S6 et en reprenant une partie des travaux du projet IHS 2018-2019 “MeuJeu” par Younès RABII, Guillaume CREUSOT et Louis HACHE, nous avons conçu et développé Cog’JDR (du moins une première version). C’est un site web plateforme de gestion de JDR type boîte mail qui vise à répondre à la problématique précédemment présentée.
Grâce des retours d’expérience de joueurs (dont, en trop grande partie, les nôtres), de MJs et aux maquettes du projet MeuJeu, nous avons spécifié les fonctionnalités de ce produit dans un Cahier des charges V1 (inclus dans l’archive rendue).

logo du projet MeuJeu

Adaptation au CDC demandé

Une majorité des fonctionnalités demandées dans le CDC initial du projet Web s'accordaient très bien avec les besoins que nous avons analysé. Ainsi, il inclut notamment les fonctionnalités suivantes qui sont similaires à celles demandées :

- inscription d'un utilisateur
- lancement d'une partie de jeu
- possibilités d'agir sur le cours de la partie côté joueur
- plus de possibilités côté MJ
- accès aux informations importantes de la partie côté joueur
- accès à toutes les informations côté MJ, dont l'état des joueurs et leurs actions
- espace de discussion en direct entre joueur ou entre MJ et joueurs
- gestion automatique d'actions selon l'état de la partie
- intégration de nouveaux modèles de JDR
- création et lancement de nouvelles parties
- gestion du temps en direct pour afficher l'heure et activer/désactiver des actions

Gestion de Projet

Organisation

Pour lancer le projet avec une bonne organisation, nous en avons d'abord identifié les deux principales phases : la conception, puis le développement. Afin d'être le plus efficace possible, notre fonctionnement a été différent dans chaque phase. Nous n'avons pas réalisé de planning prévisionnel car nous n'arrivions pas à estimer au lancement du projet la durée de chaque phase ni même à identifier toutes les tâches à réaliser car nos fonctionnalités n'étaient pas encore spécifiées. Cependant, nous avons identifié les activités principales et dans quelles ordres elles devaient être réalisées.

Tout au long du projet, nous avons donc privilégié des discussions régulières notamment grâce à des séances binômes mais aussi via Messenger et SMS. Notre Drive nous a permis regrouper et versioner nos documents d'organisation et de production écrite, dont nos premiers comptes-rendus d'avancement. A partir du maquettage de l'écriture des scripts BDD, ces comptes-rendus étaient rédigés dans les descriptions de commits de notre GitHub.

logos Messenger, Google Drive, GitHub

Phase de conception

Pour la phase de conception, nous avons travaillé exclusivement en séance présentielle en binôme pour s'assurer d'une compréhension globale du projet et permettre une meilleure efficacité dans la phase de développement à suivre.

Analyse des besoins

Nous avons commencé par l'analyse des besoins, grâce aux retours joueurs et MJs et au diaporama de soutenance du projet MeuJeu.

Spécifications et CDC

Ceci nous a permis de rédiger notre CDC pour le faire valider et spécifier les fonctionnalités de notre projet.

Conception de la base de données (MCD)

Une fois le CDC validé, nous avons pu concevoir notre complexe BDD par la réalisation d'un gargantuesque MCD en prenant en compte toutes nos fonctionnalités.

Maquettage // Implémentation de la BDD

Nous avons pu ensuite commencé à paralléliser des tâches.
Ainsi Célestin a commencé l'implémentation de la BDD par la traduction en schéma relationnel puis l'écriture des scripts SQL.
Pendant ce temps, Hugo a travaillé sur le maquettage des pages webs via le logiciel Axure.

Architecture du site // Arborescence des fichiers

Enfin, Hugo a conçu l'architecture du site en travaillant la navigation entre les pages sur les maquettes.
En parallèle, Célestin a conçu l'arborescence des fichiers dans le répertoire de notre code source.

Phase de développement

Une fois la phase de conception terminée a pu commencer la phase de développement.
Nous nous sommes alors spécialisés en fonction des tâches de conception réalisées précédemment : Célestin sur le back-end pour gérer, traiter et accéder aux données et Hugo sur le front-end pour l'interface graphique (GUI) et l'entrée de données par l'utilisateur.

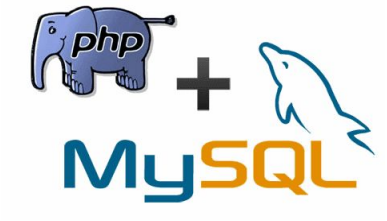
Front-end

logos HTML et CSS, Bootstrap, Jquery, JS

La partie front-end a été développée avec les langages et frameworks suivants : HTML 5 et CSS 3 (Bootstrap 4.3.1), JavaScript 1.8.5 (Jquery 3.3.1).
Nous avons priorisé les fonctionnalités les plus importantes pour l'expérience utilisateur sur notre site et avons développé dans l'ordre :

- la barre de navigation
- la page de création de modèle
- la page de création de partie, très inspirée de la précédente
- consulter et rejoindre une partie (joueur)
- effectuer une action (MJ et joueur)
- consulter et lancer une partie (MJ)

Back-end



logo php + MySQL

Le back-end a été développé avec les langages Php 5.6.35 et MySQL 8.0 (MariaDB).
Dans un souci de respect des consignes, nous avons d'abord réalisé les fonctionnalitées demandées dans le CDC initial :

- Espace de discussion
- Gestion de comptes
- Lancement de parties de JDR

Nous nous sommes ensuite attelés à la partie plus spécifique de notre CDC, réalisées par ordre de complexité croissante :

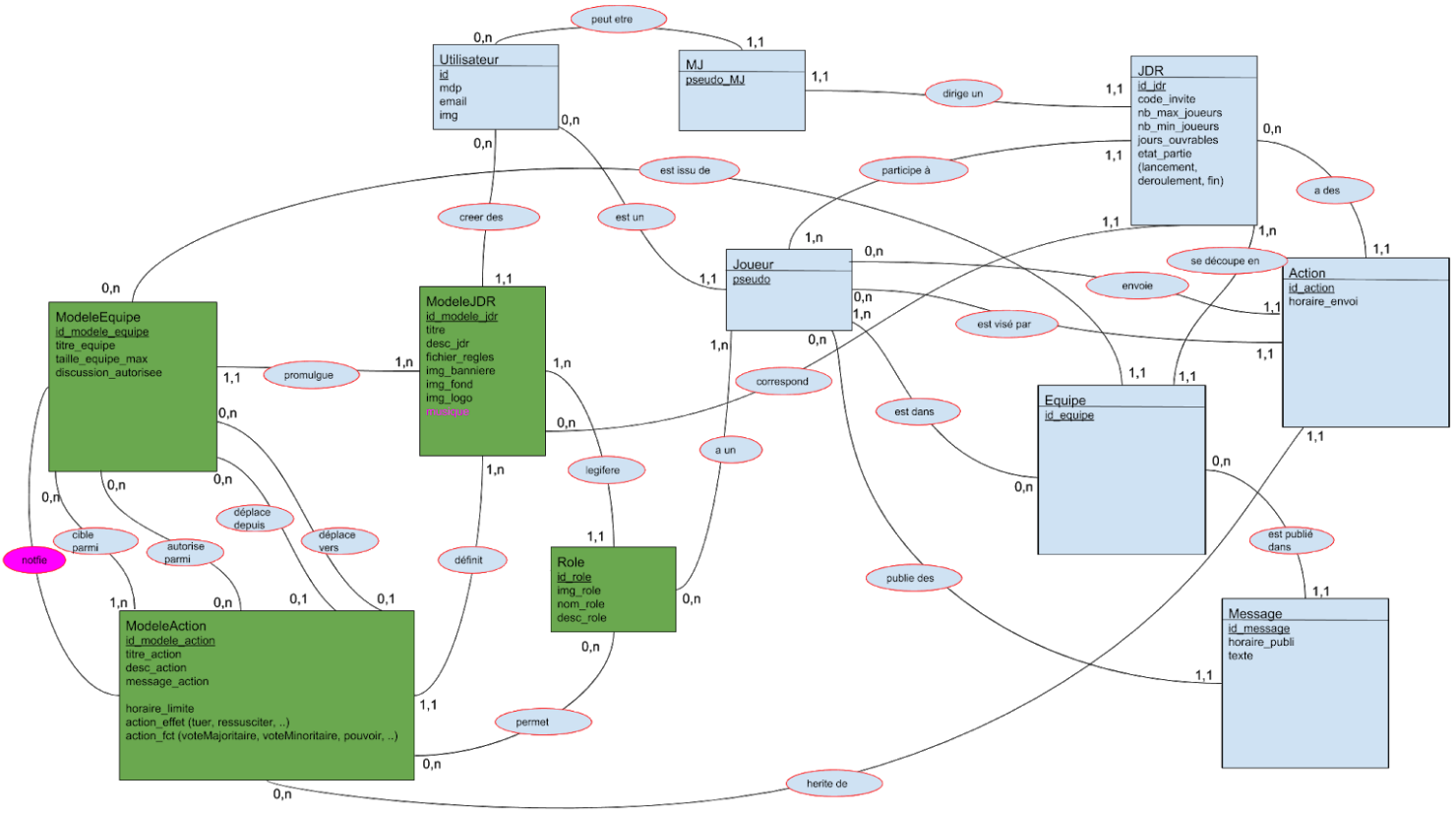
- actions
  - formulaire à remplir par un joueur,
  - automatisations telles que déplacer le joueur d'une équipe à une autre (e.g.: Vivants → Morts);
- création :
  - de parties de JDR
  - de modèles de JDR.

Modélisation

MCD et schéma relationnel

Modèle Conceptuel des Données

La base de données présentée ci-dessous à été conçue avec l'idée de faciliter la création de JDRs en employant des modèles "statiques" qui sont réutilisables (tables en vert).



Les éléments en violet / rose seront ajoutés dans les prochaines versions.

Tables de modèles

Pour que deux JDR issus d'un même modèle puissent se dérouler en parallèle, il suffit qu'ils aient le même `id_modele_jdr` pour avoir accès aux même `ModeleJDR`, `ModeleAction`, `ModeleEquipe` et `Role`.
Une autre stratégie viable aurait été de manipuler deux bases de données : une pour les modèles et une autre pour les données de parties en cours.

Utilisateur joueur

La différenciation entre utilisateur, MJ et joueur a été faite avec l'hypothèse suivante : tout utilisateur peut avoir envie de créer son JDR, et d'utiliser des pseudos différents selon le thème du jeu. Egalement, avoir une table "Joueur" est indispensable pour la gestion des messages et des actions puisque nous avons fait une seconde hypothèse : les joueurs voudront pouvoir jouer sur plusieurs parties différentes en même temps.

Discussion et équipes

Pour limiter le nombre de tables et factoriser certains traitements, la table équipe représente aussi bien une instance d'un modèle d'équipe au sein d'un partie que des espaces de discussions.
Tous les JDR on également par défauts les équipes "Tous", "Vivants", et "Morts". Dans l'équipe "Tous", la discussion n'est pas autorisé pour que les morts ne racontent pas d'histoires.
Le modèle d'un équipe à deux joueurs (un message privé, ou MP) correspond à l'`id_modele_equipe = 0` et n'est reliée à aucun modèle de JDR puis qu'il peut être utilisé par chaque JDR.
Chaque modèle d'équipe précise si la discussion entre joueurs y est autorisé (le MJ peut toujours poster des messages). La notion de message est donc intrinsèquement liée à la notion d'équipe.

Schéma relationnel

Le schéma relationnel ainsi obtenu est le suivant :

```
Utilisateur (id, mdp, email, img)

ModeleJDR (
      id_modele_jdr, #id_createur, titre,
      desc_jdr, fichier_regles, img_banniere, img_fond, img_logo
)
ModeleEquipe (
      id_modele_equipe, #id_modele_jrd, titre_equipe,
      taille_equipe_max, discussion_autorisee
)
Role (
      id_role, #id_modele_jrd,
      img_role, nom_role, desc_role
```

```
)

ModeleAction (
      id_modele_action, #id_modele_jrd,
      titre_action, desc_action, message_action,
      horaire_activ,
      action_effet_id_modele_equipe_depart,
      action_effet_id_modele_equipe_arrive,
      action_fct
)
→ Cible (id_modele_equipe_cible, id_modele_action)
→ Autorise (id_modele_equipe_autorise, id_modele_action)
→ Permet (id_role, id_modele_action)

MJ (id_mj, #id_utilisateur, #id_jrd_dirige, pseudo_mj)
Joueur (id_joueur, #id_utilisateur, #id_jrd_participe, pseudo)
→ EstUn (#id_joueur, #id_role)

JDR (
      id_jdr, #id_modele_jrd,
      code_invite,
      nb_max_joueurs, nb_min_joueurs,
      jours_ouvrables,
      etat_partie
)
Action (
      id_action, #id_modele_action,
      #id_jdr,
      #id_joueur_cible, #id_joueur_effecteur,
      horaire_envoi
)

Equipe (id_equipe, #id_modele_equipe, #id_jdr)
→ EstDans (#id_joueur, #id_equipe)

Message (id_message, #id_joueur, #id_equipe, horaire_publi, texte)
```

Dictionnaire des données

Nous n'avons pas réalisé de dictionnaire des données car nous n'en avons pas perçu utilité, n'étant qu'un groupe de 2 et ayant fait la conception de la BDD ensemble. Egalement, nous ignorions le concept et avons manqué de temps.

Maquettes

L'intégralité des maquettes (en .rp ou exportées en .html), réalisées sous Axure (et dont l'interactivité est en grande partie fonctionnelle) sont disponibles dans l'archive déposée ou sur notre GitHub.
Nous ne présenterons ici que les deux principales, dont une majorité des autres pages sont des versions simplifiées.

Création

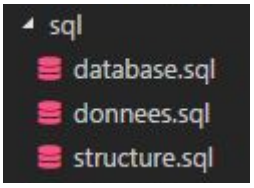
Cette première maquette, volontairement très similaire à celle de la création de partie, a représenté un des défis majeurs de l'Interface Utilisateur (IU) et de l'Expérience Utilisateur (EU) de ce site : la création d'un modèle de JDR. C'est un processus long et complexe que nous souhaitons faciliter pour qu'un maximum de personnes puisse créer.
Ainsi, nous avons minimisé la surcharge informationnelle en faisant apparaître les menus progressivement, quand ils sont nécessaires. Le fonctionnement par étape est indispensable car les premiers paramètres (par exemple le nombre d'équipes) déterminent combien de fois devront être remplis les panneaux suivants (par exemple celui pour créer une équipe).
Enfin, ce maquettage a été réalisé en se référant très souvent au CDC et au MCD pour vérifier l'implémentation de toutes les fonctionnalités et la correspondance avec la BDD.

Ecrans de jeu

Les écrans de jeu incluent la consultation de partie et le lancement de partie, que cela soit pour le joueur, ou pour le MJ qui a plus d'informations et d'interactions possibles. Tous ces écrans incluent l'espace de discussion qui est propre à chaque partie.
Le défi ici était de regrouper au maximum les informations et les actions possibles sur la partie, sans surcharger l'utilisateur et en gardant des espaces pour les images et gifs prévus dans le modèle de jeu.

Base de données implantée

Scripts SQL



Les scripts SQL sont disponibles en annexe de ce rapport. Lors du développement, nous avons utilisé 3 fichiers, avec chacun un rôle précis :

- `database.sql` : création, si elle n'existe pas, la base de données et accord de tous les droits d'accès à l' 'admin' ;
- `structure.sql` : drop toutes les tables, ajout des tables (en UTF-8, moteur innodb pour l'utilisation de clés étrangères) ;
- `donnees.sql` : contient des données d'exemple longtemps utilisées pour automatiser les tests d'intégration.

Le fichier `structure.sql` correspond à la traduction directe du modèle relationnel et, de ce fait, à toute la structure de la base de donnée de Cog'JDR. On y trouve donc toute les contraintes telles que les clés étrangères ou les `NOT NULL`.
La raison pour laquelle ce fichier drop toutes les table est également pour automatiser et rendre plus rapide les testes d'intégration : si on avait besoin de réinitialiser les données ou de modifier les contraintes entre tables, il suffisait d'exécuter ce fichier sur phpMyAdmin.
Enfin, les données récupérées de l'utilisateur passent toujours par des fonctions de sécurité.

Technologies

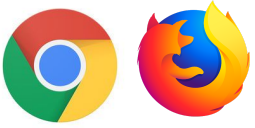
Pour le développement en local du site, nous avons utilisé les logiciels et versions installés sur les ordinateurs de l'école, à savoir :

- Xampp (5.6.35 - Control Panel v3.2.2),
- Apache/2.4.33 (Win32),
- OpenSSL/1.0.2n PHP/5.6.35 (x86),
- phpMyAdmin (4.8.0).
- Visual Studio Code

logos Xampp, MariaDB, Visual Studio Code

Le site à été testé sur les navigateurs suivants et sur leur mode mobile :

- Chrome 73.0.3683.86 (Build officiel) (64 bits)
- Firefox 60.6.1esr (64 bits)
- Internet Explorer 11.0.9x (faible compatibilité, non recommandé)

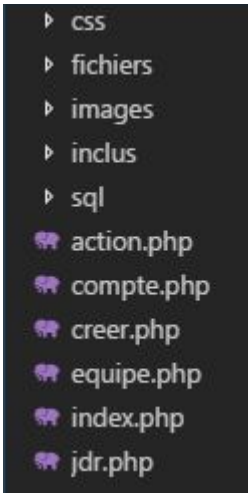


logos Chrome et Firefox

Pour gérer, stocker et tenir à jour notre code, nous avons utilisé GitHub via GitHub Desktop.

Site web

Architecture



Architecture des fichiers

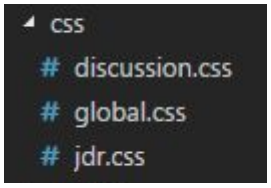
L'architecture des fichiers à été conçue pour que l'utilisateur n'ait accès qu'à quelques fichiers dont le nom évoque le rôle.

Par exemple la page `jdr.php` est utilisée pour afficher l'état d'un JDR :

- à un utilisateur quelconque avant de rejoindre,
- au MJ avant le lancement,
- à un joueur ou au MJ en cours de partie.

Mais permet aussi d'avoir des lien d'invitation simples tels que :

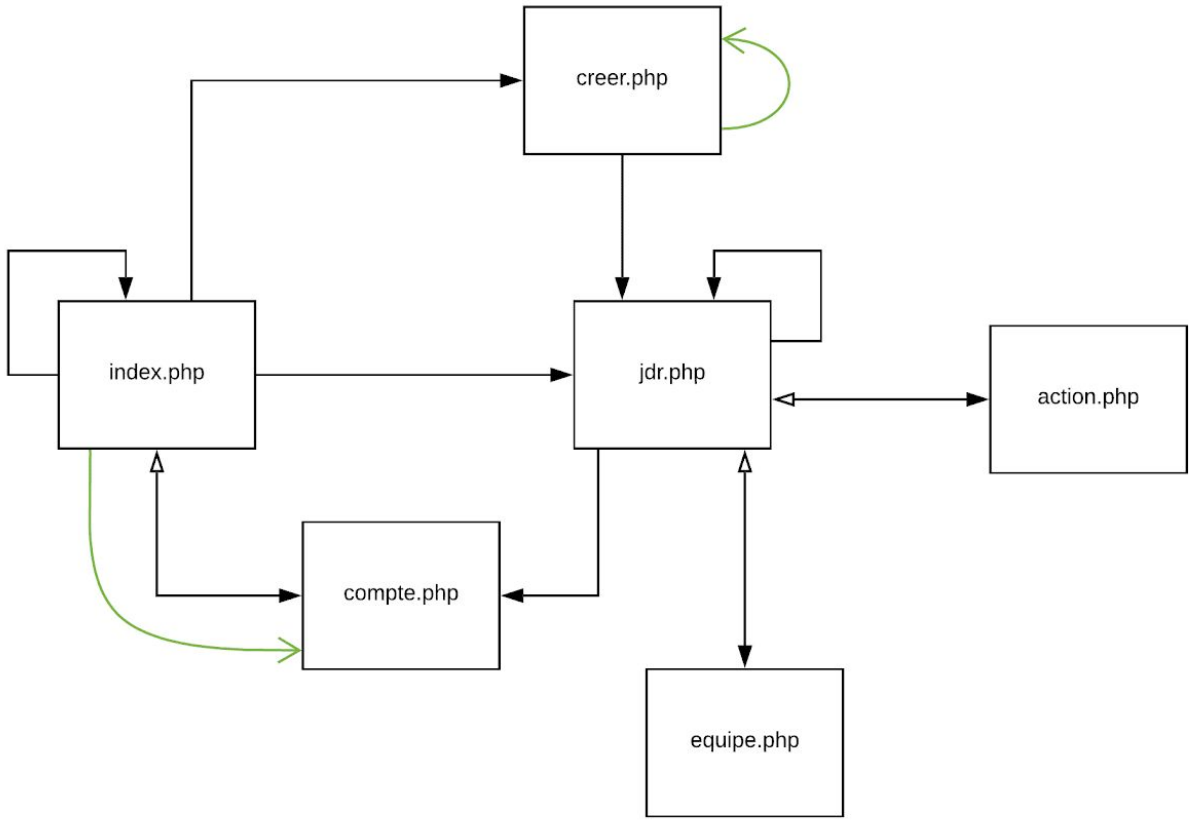
```
.../jdr.php?code=12345678
```



Les CSS sont rassemblés dans un dossiers et nommé selon les usage / fichiers dans lequel ils sont employés. Le fichier `global.css` est inclus dans toutes les pages du site.

Afin qu'une page intègre les fichiers CSS qui lui sont propres, on précise la valeur de la variable `$__css_necessaires` (comme étant une list) just avant d'inclure le `page_debut.php`.

Lien entre pages et redirections



Légende :

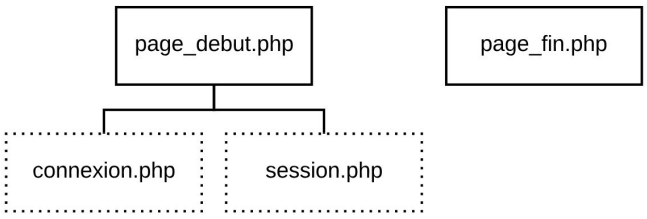
- les flèches vertes représentent un envoi de fichiers,
- les flèche noires représentent des lien de navigation :
  - manuels (l'utilisateur a cliqué sur un lien ou confirmer un formulaire),
  - automatiques ou forcé (l'utilisateur est redirigé automatiquement).

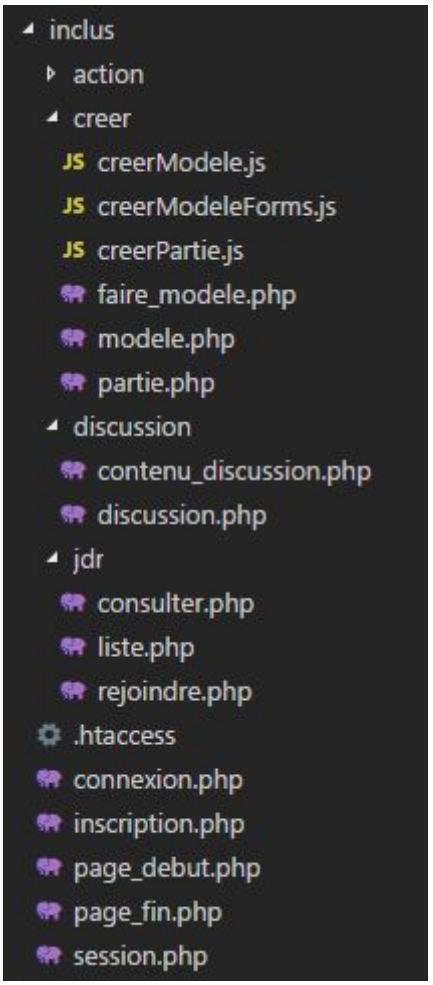
Bien sûr, l'utilisateur peut utiliser les liens de la barre de navigation rapide pour revenir vers `index.php`, `jdr.php` (pour afficher la liste des JDR ou accéder à un JDR auquel il participe), `compte.php` (pour afficher et éditer son compte) ou `creer.php` (pour créer un JDR) à tout moment.

En accord avec ce que nous avions planifié, sur la version réalisée, la page `jdr.php` se présente comme le centre de notre site d'un point de vue utilisateur, reliant toute les pages entre elles. Des page telles que action.php et `equipe.php` ne sont accessibles qu'après être passé par `jdr.php` puisqu'elles relèvent d'un JDR spécifique.

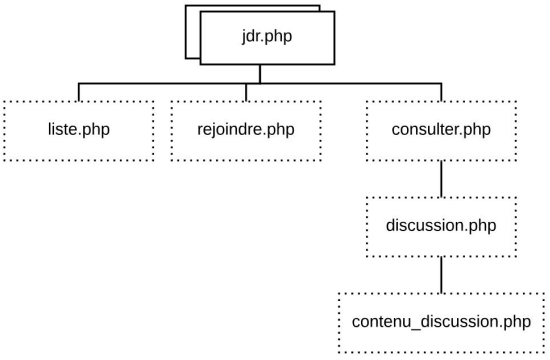
Programmes inclus

Comme seul 6 fichiers sont accédés par l'utilisateur, on utilise l'inclusion de fichiers selon l'action [intended].

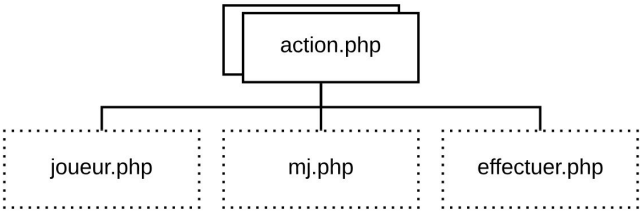




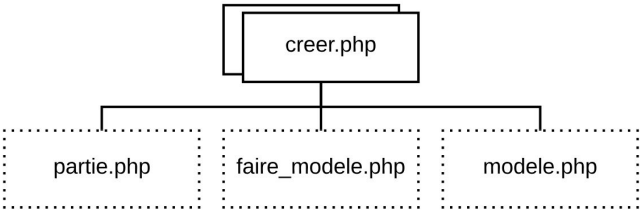
Les fichiers `page_debut.php` et `page_fin.php` sont inclus dans toutes les pages puisqu'ils contiennent respectivement le header et le footer.
Bien sûr, les fichiers `connexion.php` et `session.php` sont inclus via `require_once` pour éviter la duplication de fonction et les multiples connexion à la base de données.



Le fichier `jdr.php` à l'arborescence de dépendance la plus complexe notamment car consulter la partie donne accès à l'espace de discussion et demande d'afficher la liste des joueurs dans la partie.



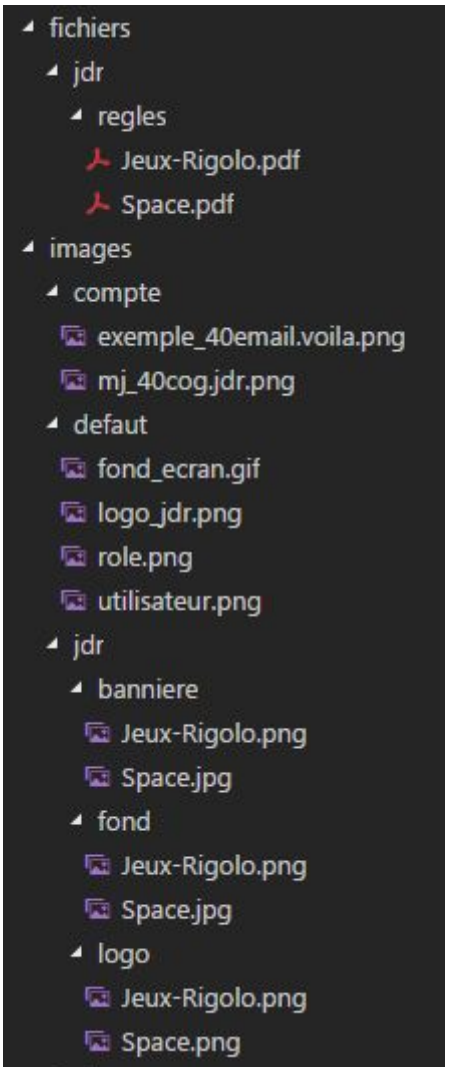
La plupart des pages présentent une interface joueur et une interface MJ. Pour la page d'action, cela à été séparé en deux fichiers car les deux interfaces diffèrent trop (d'une part un formulaire avec la liste des cible potentielles, et d'autre part un tableau associant cible et votant). Le fichier



Utilisé pour créer des modèles et des parties à partire de modèles. Le fichier `faire_modele.php` n'affiche pas de contenu : il s'occupe d'insérer le modèle selon le formulaire de `modele.php` dans la base de données.

Fichiers des utilisateurs

Les fichiers envoyés par les utilisateurs sont reçus et stockés en utilisant les fonctions appropriées de `connexion.php` : `send_image` et `send_file`.
Les images ont été regroupées selon leur utilisation (par exemple les images concernant les JDR sont réunies dans `./images/jdr/` et réparties selon leur usage en `banniere/`, `fond/` et `logo/`).



Ecrans finaux

Comme pour les maquettes, nous ne détaillerons ici que les deux écrans principaux.

Création

L'écran final de création est assez proche de la maquette conçue. Nous avons pu y ajouté des flash lumineux verts ou violets pour respectivement symboliser la validation d'une étape (étape suivante) ou l'annulation d'une étape (étape précédente). Cette version est agréable à utiliser mais peut devenir pénible pour un grand nombre d'équipes, de rôles, et surtout d'actions. Le passage à un écran de smartphone est tout à fait "responsive" et fait s'empiler horizontalement les panneaux.

Ecrans de jeu

L'écran de jeu de consultation de la partie du MJ est aussi proche de la maquette. Une vignette permettant de compter le nombre d'actions envoyées pour chaque action a été ajouté, ainsi qu'un tableau de bord interactif permettant de déplacer les joueurs dans une équipe. Enfin, la bannière et le fond s'adaptent au modèle de jeu crée par le MJ, permettant de développer l'imaginaire du JDR.

Plan de tests

Sans pouvoir parler de véritables plans de tests, nous avons pris soin durant notre développement et même notre conception de rester critique, faire des retours, et tester les récentes implémentations.

Intégrations partielles

A chaque intégration de nouvelles fonctionnalités et avant de commit des changements (enfin, presque à chaque fois), nous les testions en local avec les technologies précédemment citées pour s'assurer de leur bon fonctionnement (i.e : fonctionnement prévu et pas d'erreur) une fois intégrées au site. Nous réalisions également des tests sur les fonctionnalités potentiellement impactés par ces changements pour s'assurer qu'une nouvelle version n'était pas en fait un retour en arrière.

Parcours utilisateurs

Tout le long de la conception et du développement, nous avons pris le point de vue de potentiels utilisateurs pour savoir quels fonctionnalités tester en priorité et quels cas étaient le plus à prévoir. Egalement, il s'agissait de voir si elles étaient compréhensibles et utilisables.

Sécurité

Tous les champs saisis ou modifiés par l'utilisateur ont été testé face à des tentatives d'injection SQL et XSS avec succès. La page de création de modèles de JDR n'étant pas entièrement complété, elle n'a pas encore été testée en profondeur.

Conclusion

Difficultés et solutions

Au cours de ce projet, nous avons rencontré plusieurs difficultés face auxquelles nous avons essayé des solutions.

Notamment, la phase de conception a été très longue, nous avions donc prévu beaucoup d'heures pour le développement sur la fin du projet

Également, le téléversement des fichiers sur la page de création de modèle nous a coûté beaucoup de temps car l'envoi de données se faisait initialement par une requête Ajax Jquery. Sur la version présentée, les images ne fonctionnent donc pas mais nous avons passé beaucoup de temps à essayer des solutions à ce problème qui ne nécessitaient pas une refonte importante de la page et de ses scripts JS. Aucune n'a marché, mais nous avons commencé à développer une prochaine version où les données pourront s'envoyer par requête XHMLHttp qui transmet correctement les fichiers.

Écarts par rapport au CDC initial

Nous ne pouvons pas parler d'ajouts par rapport au CDC initial du projet Web ainsi nous comparerons à notre CDC V0 fourni au début du projet. Cette version 1 du site remplit toutes les exigences spécifiées mais présente plusieurs écarts de fonctionnalités.

Fonctionnalités ajoutées

- les joueurs peuvent télécharger les règles en .pdf envoyées par le MJ lors de la création du modèle
- les horaires limites des actions sont affichées sur la page pour le MJ et le joueur
- une horloge affiche l'heure en direct
- le nombre de votes envoyés par les joueurs pour une action est visible par le MJ

Fonctionnalités non implémentées

- les images choisies par le MJ lors de la création du modèle ne sont pas envoyées à la BDD
- les pouvoirs des rôles ne fonctionnent pas
- les joueurs ne peuvent pas savoir qui est dans leur équipe si leurs camarades ne parlent pas
- les messages automatiques d'action ne sont ni configurables ni optionnels

Avenir du projet

Cog'JDR est un projet que nous allons continuer dans l'objectif de l'ouvrir aux élèves de l'école au cours de l'année prochaine. Ainsi, tout au long du projet, nous avons gardé cette perspective et essayé de maintenir un code le plus compréhensible possible. Egalement, nous avons commencé la rédaction d'un CDC V2, de rapports de bugs, de plans de tests et de plans de conception d'une seconde BDD pour gérer les statistiques d'utilisation.

Cette version 1 est la version Présentable.
Nous prévoyons des tests sur smartphone, des correctifs, et au minimum les fonctionnalités du CDC V1 pour arriver à la version 2 qui sera la version Testable.
Avec cette version 2, nous organiserons des tests utilisateur avec des MJs expérimentés, et débutants, et des parties avec des joueurs en comité restreint.
En exploitant ces tests et en développant une charte graphique plus attractive, nous arriverons à la version 3 qui sera ouverte aux élèves de l'école pour tester le site avec des parties à un grand nombre de joueurs, et pour s'amuser ♥.

Annexe

Database.sql

```
create database if not exists cogjdr character set utf8 collate
utf8_unicode_ci;
use cogjdr;

grant all privileges on cogjdr.* to 'cogjdr_user'@'localhost' identified
by 'admin';
```

Structure.sql

```
drop table if exists Message_;
drop table if exists EstDans;
drop table if exists Equipe;
drop table if exists Action_;
drop table if exists EstUn;
drop table if exists Joueur;
drop table if exists MJ;
drop table if exists JDR;
drop table if exists Permet;
drop table if exists Autorise;
drop table if exists Cible;
drop table if exists ModeleAction;
drop table if exists Role_;
drop table if exists ModeleEquipe;
drop table if exists ModeleJDR;
drop table if exists Utilisateur;


create table Utilisateur (
   id integer not null primary key auto_increment,
   mdp varchar(255) not null,
   email varchar(32) not null,
   img varchar(128) not null
) engine=innodb character set utf8 collate utf8_unicode_ci;
```

```
create table ModeleJDR (
   id_modele_jdr integer not null primary key auto_increment,
   id_createur integer,
   titre varchar(32) not null,
   desc_jdr varchar(1024) not null,
   fichier_regles varchar(128) not null,
   img_banniere varchar(128) not null,
   img_fond varchar(128) not null,
   img_logo varchar(128) not null,

   foreign key (id_createur) references Utilisateur (id)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table ModeleEquipe (
   id_modele_equipe integer not null primary key auto_increment,
   id_modele_jdr integer,
   titre_equipe varchar(32) not null,
   taille_equipe_max integer not null,
   discussion_autorisee boolean not null,

   foreign key (id_modele_jdr) references ModeleJDR (id_modele_jdr)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table Role_ (
   id_role integer not null primary key auto_increment,
   id_modele_jdr integer,
   img_role varchar(128) not null,
   nom_role varchar(128) not null,
   desc_role varchar(512) not null,

   foreign key (id_modele_jdr) references ModeleJDR (id_modele_jdr)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table ModeleAction (
   id_modele_action integer not null primary key auto_increment,
   id_modele_jdr integer,
   titre_action varchar(32) not null,
   desc_action varchar(512) not null,
   message_action varchar(1024) not null,
   horaire_activ time not null,
```

```
    action_effet_id_modele_equipe_depart integer,
    action_effet_id_modele_equipe_arrive integer,
    action_fct enum('voteMajoritaireTous', 'voteMinoritaireTous',
'voteMajoritairePremier', 'voteMinoritairePremier',
'voteMajoritaireNul', 'voteMinoritaireNul', 'pouvoir') not null,

    foreign key (action_effet_id_modele_equipe_depart) references
ModeleEquipe (id_modele_equipe),
    foreign key (action_effet_id_modele_equipe_arrive) references
ModeleEquipe (id_modele_equipe),
    foreign key (id_modele_jdr) references ModeleJDR (id_modele_jdr)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table Cible (
    id_modele_equipe_cible integer,
    id_modele_action integer,

    foreign key (id_modele_equipe_cible) references ModeleEquipe
(id_modele_equipe),
    foreign key (id_modele_action) references ModeleAction
(id_modele_action)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table Autorise (
    id_modele_equipe_autorise integer,
    id_modele_action integer,

    foreign key (id_modele_equipe_autorise) references ModeleEquipe
(id_modele_equipe),
    foreign key (id_modele_action) references ModeleAction
(id_modele_action)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table Permet (
    id_role integer,
    id_modele_action integer,

    foreign key (id_role) references Role_ (id_role),
    foreign key (id_modele_action) references ModeleAction
(id_modele_action)
```

```
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table JDR (
   id_jdr integer not null primary key auto_increment,
   id_modele_jdr integer,
   code_invite varchar(8),
   nb_max_joueurs integer not null,
   nb_min_joueurs integer not null,
   jours_ouvrables enum('5 jours', '6 jours', '7 jours') not null,
   etat_partie enum('lancement', 'deroulement', 'fin') not null,

   foreign key (id_modele_jdr) references ModeleJDR (id_modele_jdr)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table MJ (
   id_mj integer not null primary key auto_increment,
   id_utilisateur integer,
   id_jdr_dirige integer,
   pseudo_mj varchar(32) not null,

   foreign key (id_utilisateur) references Utilisateur (id),
   foreign key (id_jdr_dirige) references JDR (id_jdr)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table Joueur (
   id_joueur integer not null primary key auto_increment,
   id_utilisateur integer,
   id_jdr_participe integer,
   pseudo varchar(32) not null,

   foreign key (id_utilisateur) references Utilisateur (id),
   foreign key (id_jdr_participe) references JDR (id_jdr)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table EstUn (
   id_joueur integer,
   id_role integer,

   foreign key (id_joueur) references Joueur (id_joueur),
   foreign key (id_role) references Role_ (id_role),
```

```
    primary key (id_joueur, id_role)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table Action_ (
    id_action integer not null primary key auto_increment,
    id_modele_action integer,
    id_jdr integer,
    id_joueur_cible integer,
    id_joueur_effecteur integer,
    horaire_envoi timestamp not null default CURRENT_TIMESTAMP,

    foreign key (id_modele_action) references ModeleAction
(id_modele_action),
    foreign key (id_jdr) references JDR (id_jdr),
    foreign key (id_joueur_cible) references Joueur (id_joueur),
    foreign key (id_joueur_effecteur) references Joueur (id_joueur)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table Equipe (
    id_equipe integer not null primary key auto_increment,
    id_modele_equipe integer,
    id_jdr integer,

    foreign key (id_modele_equipe) references ModeleEquipe
(id_modele_equipe),
    foreign key (id_jdr) references JDR (id_jdr)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table EstDans (
    id_joueur integer,
    id_equipe integer,

    foreign key (id_joueur) references Joueur (id_joueur),
    foreign key (id_equipe) references Equipe (id_equipe),

    primary key (id_joueur, id_equipe)
) engine=innodb character set utf8 collate utf8_unicode_ci;

create table Message_ (
```

```
    id_message integer not null primary key auto_increment,
    id_joueur integer,
    id_equipe integer,
    horaire_publi timestamp not null default CURRENT_TIMESTAMP,
    texte varchar(1024) not null,

    foreign key (id_joueur) references Joueur (id_joueur),
    foreign key (id_equipe) references Equipe (id_equipe)
) engine=innodb character set utf8 collate utf8_unicode_ci;


insert into ModeleEquipe (`id_modele_equipe`, `id_modele_jdr`,
`titre_equipe`, `taille_equipe_max`, `discussion_autorisee`) values (0,
null, 'MP', 2, 1);
```

Donnees.sql

```
delete from Cible where 1;
delete from Autorise where 1;
delete from ModeleAction where 1;
delete from Message_ where 1;
delete from EstDans where 1;
delete from Equipe where 1;
delete from ModeleEquipe where id_modele_equipe != 0;
delete from Joueur where 1;
delete from MJ where 1;
delete from JDR where 1;
delete from ModeleJDR where 1;
delete from Utilisateur where 1;


insert into Utilisateur (`id`, `mdp`, `email`, `img`) values
    (null,
'$2y$10$hXBu1vDgRNw7cliHI2pZgOpNI5q4LlAshFU70Dg9KanzccApfiGy6',
'exemple@email.voila', 'exemple_40email.voila.png'),
    (null,
'$2y$10$lyClN0AX5HnrZAPFMyga2uF6CoIoKv2dq2Hnju3Q29n0PvKLg/5YW',
'mj@cog.jdr', 'mj_40cog.jdr.png'),
```

```
    (null,
'$2y$10$Bvs9nWFl89CtEFH18Iy9gOZ.QmpizdrYWFUtei5LvZWtmIfZVRwjy',
'coucou@coucou.coucou', 'coucou_40coucou.coucou.gif');

insert into ModeleJDR (`id_modele_jdr`, `id_createur`, `titre`,
`desc_jdr`, `fichier_regles`, `img_banniere`, `img_fond`, `img_logo`)
values
    (null, 2, 'Jeux Rigolo', 'Ceci est la description du Jeux Rigolo :).
Du coup je remplis cet espace avec un texte long et plein de fautes de
fran&ccedil;ais en tout genre. Dailleurs, ce jeu, tout aussi rigolo
qu&apos;il soit, ne se joue qu&apos;à 12 joueurs max... donc bon,
voil&agrave;.', 'Jeux-Rigolo.pdf', 'Jeux-Rigolo.png', 'Jeux-Rigolo.png',
'Jeux-Rigolo.png'),
    (null, 2, 'Space', 'Ceci est une description assez courte. Un super
JDR dans l&apos;espace, WOAW &num;paieTaDescription', 'Space.pdf',
'Space.jpg', 'Space.jpg', 'Space.png');

insert into JDR (`id_jdr`, `id_modele_jdr`, `code_invite`,
`nb_max_joueurs`, `nb_min_joueurs`, `jours_ouvrables`, `etat_partie`)
values
    (null, 1, "12da0fde", 12, 0, '7 jours', 'lancement');

insert into MJ (`id_mj`, `id_utilisateur`, `id_jdr_dirige`, `pseudo_mj`)
values
    (null, 2, 1, "xXMeuJeuXx");

insert into Joueur (`id_joueur`, `id_utilisateur`, `id_jdr_participe`,
`pseudo`) values
    (null, 1, 1, 'Joueur'),
    (null, 3, 1, 'Sel');

insert into ModeleEquipe (`id_modele_equipe`, `id_modele_jdr`,
`titre_equipe`, `taille_equipe_max`, `discussion_autorisee`) values
    (null, 1, 'Vivants', -1, 1),
    (null, 1, 'Sac', 11, 1),
    (null, 1, 'Morts', -1, 1),
    (null, 1, 'Tous', -1, 0);

insert into Equipe (`id_equipe`, `id_modele_equipe`, `id_jdr`) values
    (null, 1, 1),
```

```
    (null, 2, 1),
    (null, 3, 1),
    (null, 4, 1);

insert into EstDans (`id_joueur`, `id_equipe`) values
    (1, 1),
    (2, 1),
    (2, 2),
    (1, 4),
    (2, 4);

insert into Message_ (`id_message`, `id_joueur`, `id_equipe`,
`horaire_publi`, `texte`) values
    (null, 1, 1, NOW(), "Hey, salut !"),
    (null, 1, 1, NOW(), "Comment ca va ?"),
    (null, 1, 1, NOW(), "Héo.."),
    (null, 1, 1, NOW(), "Ya qqun ? Tu répond ?"),
    (null, 2, 2, NOW(), "c'est plustot calme ici..."),
    (null, 2, 1, NOW(), "Tutecalm."),
    (null, null, 2, NOW(), "oui lol"),
    (null, 2, 2, NOW(), "ha slt"),
    (null, null, 1, NOW(), "TAISEZ VOUS!");

insert into ModeleAction (`id_modele_action`, `id_modele_jdr`,
`titre_action`, `desc_action`, `message_action`, `horaire_activ`,
`action_effet_id_modele_equipe_depart`,
`action_effet_id_modele_equipe_arrive`, `action_fct`) values
    (null, 1, "Vote des villageois", "Les villageois votent pour la
personne qu'ils veulent br&ucirc;ler. Qui pensez-vous &ecirc;tre un loup
?", "$cible.pseudo; ($cible.email;) a &eacute;t&eacute; choisis &agrave;
la majorit&eacute;e ($vote.nb_majoritaire; sur $vote.nb_total;) pour
&ecirc;tre br&ucirc;l&eacute;(e) ce soir...", "23:24:25", 1, 3,
"voteMajoritaireTous"),
    (null, 1, "Modele d'action test no 2", "Titre de l'action",
"Description de l'action", "10:11:12", null, 2, "voteMajoritaire");

insert into Cible (`id_modele_equipe_cible`, `id_modele_action`) values
    (1, 1),
    (1, 2);
```

```
insert into Autorise (`id_modele_equipe_autorise`, `id_modele_action`)
values
    (1, 1),
    (2, 2);

insert into Role_ (`id_role`, `id_modele_jdr`, `img_role`, `nom_role`,
`desc_role`) values
    (null, 1, "bla", "d-Role", "Fait un blague qui transforme un joueur
en sac.");

insert into Permet (`id_role`, `id_modele_action`) values
    (1, 2);
```