UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 2 Informatique
INFO0303 – 2021/2022
***Resp. :* C. Rabat**

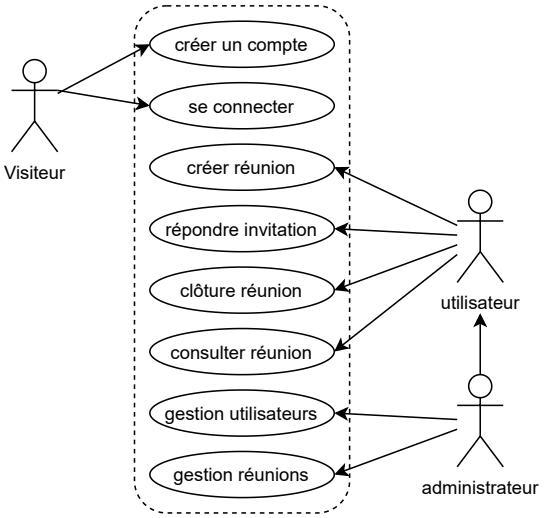
Devoir surveillé terminal

Durée 1h30
(Corrections)

Nous souhaitons développer une application Web en PHP pour organiser des réunions entre plusieurs personnes. Un utilisateur crée une réunion et y associe une liste de créneaux possibles, ainsi qu'une liste d'invités (adresses email uniquement). Chaque invité peut répondre aux demandes qui le concernent en sélectionnant ses créneaux disponibles. L'utilisateur peut ensuite clôturer la réunion en précisant le créneau sélectionné, ce qui empêche les invités par la suite de modifier leurs choix. Un administrateur à la possibilité de gérer intégralement le site (les utilisateurs, les réunions, etc.). C'est lui qui confirme la création des utilisateurs.

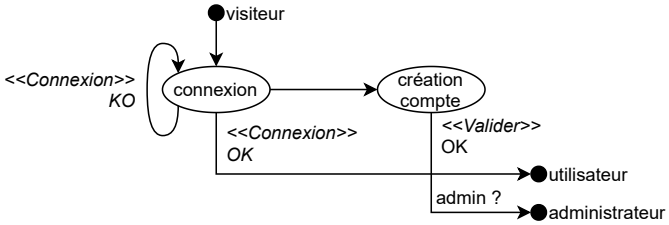
Exercice 1 (Modélisation - *6 pts*)
1°) Identifiez les acteurs du site et donnez les cas d'utilisation de l'application. *(1 point)*

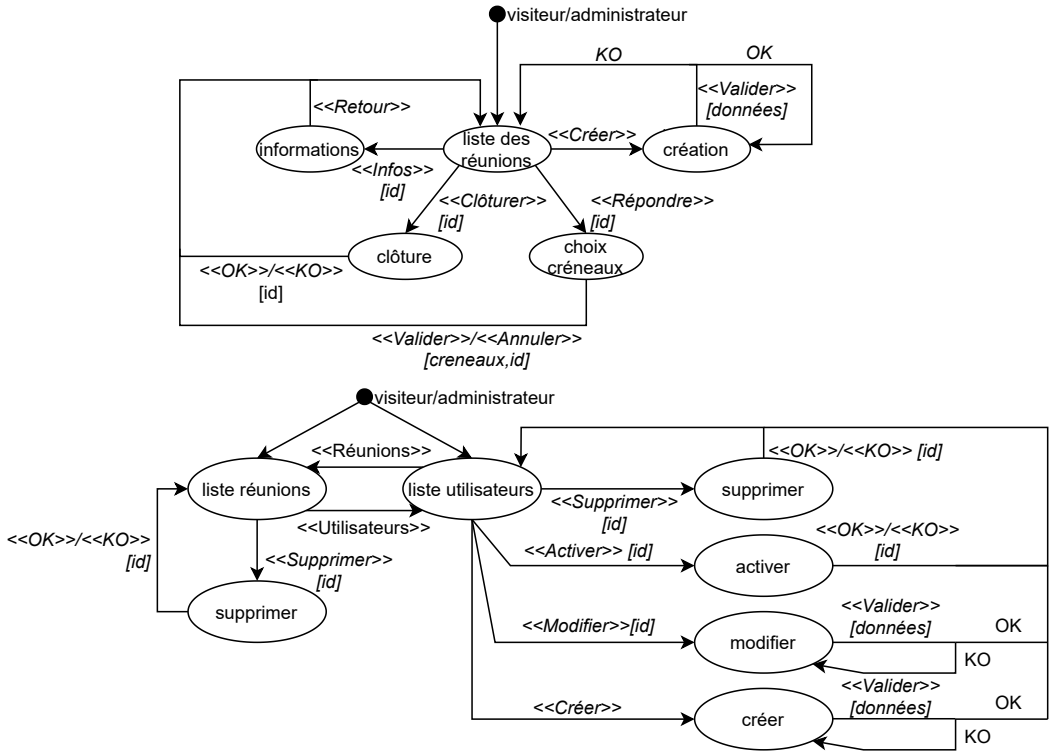
Solution : *Il y a 3 acteurs : le visiteur, l'utilisateur (un visiteur enregistré) et l'administrateur.*



2°) Donnez les diagrammes de navigation de l'application. *(3 points)*

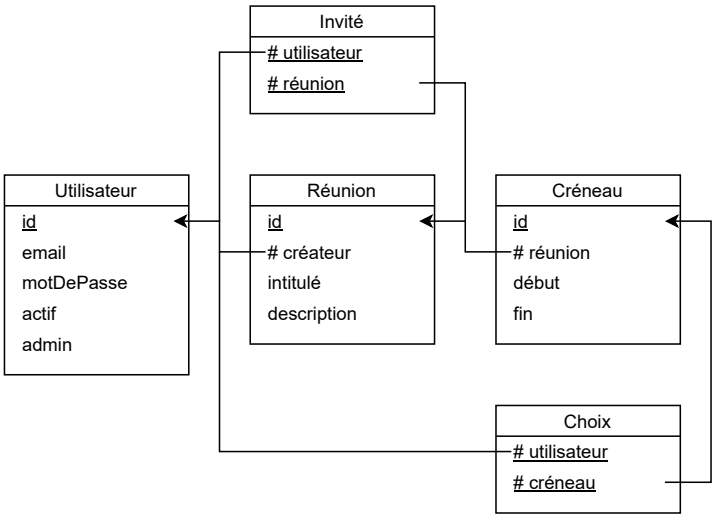
Solution :





3°) Quelles seraient les données nécessaires dans la base de données ? Proposez un MLD et expliquez votre choix. *(2 points)*

Solution : *Les utilisateurs sont caractérisés par leur adresse email et leur mot de passe, ainsi qu'un état actif (par défaut à faux tant que l'administrateur ne le change pas). On peut également ajouter un booléen pour savoir si l'utilisateur est administrateur. Une réunion doit être caractérisée par un intitulé, une description (un texte d'explication) et par l'utilisateur qui l'a créée. Un créneau possible pour une réunion correspond un début et une fin et doit être associé à une réunion. Les invités à une réunion sont caractérisés par les identifiants de l'utilisateur et de la réunion. De même, les choix des créneaux sont caractérisés par les identifiants du créneau et de l'utilisateur.*



Exercice 2 (Programmation PHP - *6,5 pts*)
Dans un premier temps, nous faisons le choix de développer l'application en PHP sans *framework*.

1°) Proposez une classe `Reunion` permettant de représenter une réunion avec les méthodes classiques (constructeur, getters, conversion en chaîne de caractères). *(1,5 point)*

Solution :

```php
class Reunion {
  private int $id;
  private int $createur;
  private string $intitule;
  private string $description;

  public function __construct(int $id, int $createur, string $intitule, string $description)
      {
    $this->id = $id;
    $this->createur = $createur;
    $this->intitule = $intitule;
    $this->description = $description;
  }

  public function getId() : int { return $this->id; }
  public function getCreateur() : int { return $this->createur; }
  public function getIntitule() : string { return $this->intitule; }
  public function getDescription() : string { return $this->description; }

  public function __toString() : string {
    return $this->intitule;
  }
}
```

2°) Proposez un formulaire permettant à l'utilisateur de saisir les données d'une réunion (pour le moment, sans les créneaux). *(1 point)*

Solution :

```php
...
<form method="post" action="dest.php">
  <label for="intitule">Intitulé</label><input type="text" name="intitule"/>
  <label for="description">Description</label>
  <textarea name="intitule"></textarea>
  <button type="submit">Valider</button>
</form>
...
```

3°) Donnez le contenu du script PHP permettant de récupérer les données depuis le formulaire (en ajoutant les vérifications nécessaires) et de créer un objet `Reunion`. *(2 points)*

Solution :

```php
<?php
include("Reunion.php");

if(isset($_POST['intitule'])) $intitule = $_POST['intitule'];
if(isset($_POST['description'])) $description = $_POST['description'];
$reunion = new Reunion(-1, -1, $initutle, $description);
...
```

4°) Comment permettre à l'utilisateur d'ajouter autant de créneaux que nécessaire depuis le formulaire ? Comment les récupérer ensuite en PHP ? *(2 points)*

Solution : *Une solution est d'utiliser Javascript. En cliquant sur un bouton, on appelle une fonction Javascript qui crée à la volée deux champs texte (pour le début et la fin). Le nom (champ `name`) correspond à un tableau (avec des crochets). Pour récupérer les données en PHP, il suffit de regarder les tableaux et de parcourir leurs cases.*

Exercice 3 (*Laravel - 7,5 pts*)

Finalement, nous choisissons d'utiliser *Laravel* pour développer cette application.

1°) Quels éléments doivent être mis en place dans *Laravel* pour réaliser l'application ? Expliquez chacun d'entre eux. *(2 points)*

Solution : *Il faut :*

- *Les migrations : une pour chaque table ; permet de créer les tables*
- *Les contrôleurs : un contrôleur pour gérer chaque élément (réunion, invitations, créneaux) ; permettent de gérer les accès, de récupérer les données depuis la base et d'appeler les vues*
- *Les modèles : réunion, invitation, créneau ; permettent de manipuler les données de la base*
- *Les vues : une pour chaque page et un ou plusieurs templates (pour les données communes) ; mise en forme, affichage des données, formulaires*
- *Les requêtes : pour vérifier automatiquement les données saisies (ajout/modification)*
- *Les seeders/factories : uniquement pour créer l'administrateur (ou pour générer des données de test)*

On peut parler également de la configuration de l'application avec le fichier `.env` *(configuration du nom de l'application, de l'accès à la base de données).*

2°) Qu'est-ce qui caractérise une vue dans le modèle MVC ? *(1 point)*

Solution : *Le principe d'une vue est de mettre en forme et d'afficher des données passées par le contrôleur pour l'utilisateur. Elle contient principalement du HTML.*

3°) Quelles vues seraient nécessaires dans cette application ? *(1 point)*

Solution : *Il faudrait un template pour toutes les pages, voire un template spécifique pour la page de connexion. Chaque vue hérite ensuite de ces templates. Il faudrait une vue pour chaque page : liste des réunions, information d'une réunion, création, clôture, choix, etc.*

4°) Expliquez l'intérêt et le fonctionnement des migrations dans *Laravel*. *(2 points)*

Solution : *Chaque migration permet de modifier la structure de la base de données (ajout de tables, modification ou suppression de tables). Elles possèdent une méthode permettant également de faire des retours en arrière pour annuler les modifications. Dans une application en mode production, cela permet d'éviter de perdre des données (si une sauvegarde de la base de données est réalisée en parallèle). L'un des intérêts des migrations est de simplifier l'installation de l'application ou son transfert d'un serveur à un autre ou son installation. Par exemple, dans le cas du partage du code via GIT, la structure de la base de données n'est pas échangée contrairement aux migrations. Avec l'appel à* `php migrate`*, la structure est mise à jour.*

5°) Expliquez à quoi servent les *seeders* et les *factories*. Est-ce nécessaire pour une application en production ? *(1,5 point)*

Solution : *Un seeder permet de remplir le contenu d'une table en s'aidant d'une factory. Cette dernière permet de créer un élément, parfois en utilisant un faker pour générer des données aléatoires.*

Les seeders et les factories servent lorsque l'application est en mode développement. Pour le mode production, c'est utile pour pré-remplir certaines tables en spécifiant des données par défaut ou bien en créant des utilisateurs (comme l'administrateur de l'application).