Rapport de SAE
Année 2024/2025 – Semestre 4
Lilou DUFAU
Parcours IAMSI (D)

GUREKULTURA 

RAPPORT DE SAE

Application web de **gestion évènementielle** sur le **Pays Basque**

Groupe n° 7 :

CAMPISTRON Julian, DUFAU Lilou,

ETCHECOPAR Maxime, LAGÜE Théo,

MARTIN Edgar, MORANCE Kyllian

Établissement d'études : IUT de Bayonne et du Pays Basque

Enseignant référent : **Christophe MARQUESUZAÀ**

Table des matières

Table des figures

1. CONTEXTUALISATION

1.1. L'étudiante

...

1.2. LA SAE

Dans le cadre de mon BUT Informatique, j'ai pu participer à une SAE en groupe afin de développer mes compétences techniques.

1.2.1. Contexte du projet

Ce projet a été réalisé dans le cadre de la **SAE** (Situation d'Apprentissage Evalué) de mon BUT Informatique afin de pouvoir mettre en application les éléments vus en cours.

1.2.2. Objectifs du projet

GureKultura est une plateforme web dédiée aux étudiants du **Pays Basque** pour gérer et découvrir des **événements** locaux. Les utilisateurs connectés peuvent proposer des événements, soumis à validation par un modérateur. Une fois approuvés, ils apparaissent sur la page d'accueil avec des options de filtrage par catégorie. Les utilisateurs peuvent s'inscrire et commenter les événements. La plateforme propose aussi des actualités locales, publiées par des modérateurs, avec une section commentaires pour échanger et rencontrer d'autres passionnés de la culture basque.

1.2.3. Fonctionnalités effectuées

...

1.3. L'environnement de travail

...

1.3.1. L'environnement matériel

...

1.3.2. L'environnement logiciel

...

2. PRODUCTION

Durant la SAE j'ai pu produire différentes fonctionnalités comme la proposition d'évènements, leur modification etc .

2.1. Proposition d'évènements

Dans le cadre du projet, j'ai conçu et développé la fonctionnalité permettant aux utilisateurs de proposer des événements. Pour cela, j'ai mis en place plusieurs composants interconnectés, garantissant une gestion fluide et efficace de cette fonctionnalité.

1. Formulaire de proposition d'évènement

J'ai créé le formulaire de proposition d'événement dans un fichier **Twig** (*propEv.html.twig*). Ce formulaire permet à l'utilisateur de saisir toutes les informations nécessaires à la création d'un événement, notamment :Titre de l'événement, Description, Dates et heures de début et de fin, Lieu, Photo, Nombre de places, Catégorie, etc.

J'ai également intégré un système d'envoi des données via une requête **POST** vers le contrôleur que j'ai développé.

2. Développement du contrôleur **controller_propEv.class.php**

- J'ai mis en place le contrôleur *controller_propEv.class.php*, qui assure toute la logique métier liée à la proposition d'événement. Il fonctionne selon les étapes suivantes :
- **Validation de l'utilisateur connecté** : J'ai implémenté un contrôle de session pour s'assurer que seuls les utilisateurs connectés puissent

soumettre un événement. En cas de non-authentification, l'utilisateur est redirigé vers la page de connexion.

- **Récupération et validation des données du formulaire** : Toutes les données envoyées sont récupérées et soumises à des vérifications pour garantir leur validité et leur sécurité.
- **Création d'un objet Evenement** : Une fois les données validées, j'ai développé un mécanisme pour instancier un objet Evenement avec ces informations.
- **Insertion dans la base de données** : L'objet Evenement est ensuite transmis au DAO (*evenementDao*), que j'ai mis en place pour assurer une insertion sécurisée en base de données.

```php
<?php
$evenement = new Evenement(
    null,
    $donnees['titre'],
    $donnees['autorisationName'] ?? null,
    $donnees['description'],
    $donnees['email'],
    $donnees['tel'],
    $donnees['nomRep'],
    $donnees['prenomRep'],
    $donnees['debutDate'],
    $donnees['finDate'],
    $donnees['debutHeure'],
    $donnees['finHeure'],
    $donnees['lieu'],
    $donnees['photoName'] ?? null,
    false,
    $donnees['userId'],
    $donnees['cateId'],
    $donnees['nomCategorie'] ?? null,
    $donnees['nbPlaces'] ?? null
);
$managerEvenement->insert($evenement);
```

Figure 1

3. Modèle **Evenement.class.php**

- J'ai conçu la classe *Evenement*, qui représente un événement dans l'application. Cette classe contient :
- **Les propriétés de l'événement** (titre, description, dates, lieu, etc.).
- **Les getters et setters** pour accéder et modifier ces propriétés.
- **Un constructeur** permettant d'initialiser un objet *Evenement* avec les données fournies.

```php
<?php
private ?string $titre;
private ?string $description;
private ?string $lieu;
private ?int $nbPlaces;
```

Figure 2

4. DAO **evenement.dao.php**

J'ai développé le **DAO** *EvenementDao*, qui est responsable des interactions avec la base de données. Il contient une méthode *insert* permettant d'insérer un nouvel événement dans la table *gk_evenement*.

```php
<?php
public function insert(Evenement $evenement) {
    $sql = "INSERT INTO gk_evenement (titre, description, dateDebut, dateFin, lieu, nbPlaces, ...)
            VALUES (:titre, :description, :dateDebut, :dateFin, :lieu, :nbPlaces, ...)";
    $pdoStatement = $this->pdo->prepare($sql);
    $pdoStatement->execute([
        ':titre' => $evenement->getTitre(),
        ':description' => $evenement->getDescription(),
        ':dateDebut' => $evenement->getDateDebut(),
        ':dateFin' => $evenement->getDateFin(),
        ':lieu' => $evenement->getLieu(),
        ':nbPlaces' => $evenement->getNbPlaces(),
        ...
    ]);
}
```

Figure 3

5. Validation et gestion des erreurs

- Pour garantir la fiabilité du processus, j'ai mis en place un système de gestion des erreurs dans le contrôleur :
- **Gestion des exceptions** : Si une erreur survient lors de l'insertion des données, un message d'erreur clair est affiché à l'utilisateur.
- **Journalisation des erreurs** : J'ai intégré un mécanisme permettant d'enregistrer les erreurs dans les logs afin de faciliter le débogage et la maintenance.

```php
<?php
catch (Exception $e) {
    $messageErreur = $e->getMessage();
    error_log("Error inserting event: " . $messageErreur);
    return false;
}
```

Figure 4

6. Affichage des catégories

Pour améliorer l'expérience utilisateur, j'ai ajouté une fonctionnalité permettant de récupérer dynamiquement les catégories depuis la base de données via le **DAO** *CategorieDao*. Ces catégories sont ensuite affichées dans le formulaire sous forme de liste déroulante, garantissant une classification intuitive des événements.

7. Redirection et confirmation

Après une insertion réussie, j'ai mis en place une redirection automatique vers **la liste des événements de l'utilisateur**. Cette fonctionnalité permet à l'utilisateur de visualiser rapidement son événement et d'assurer un bon suivi de ses propositions.

2.2. Tests et validation

Pour les tests, nous testions nous même les fonctionnalités afin de savoir si le rendu était celui attendu ou non.

3. MÉTHODOLOGIE

3.4. Méthode de travail

...

3.2. Technologies utilisées

...

4. BILAN

4.1. Sommaire

...

4.2. Apport aux compétences

...

4.3. Conclusion

...

4.4. Glossaire

- **Twig** : Moteur de templating utilisé dans Symfony pour générer des vues dynamiques en PHP.
- **Formulaire (HTML/Twig)** : Interface permettant à un utilisateur de saisir des informations qui seront envoyées à un serveur.
- **Requête POST** : Méthode HTTP permettant d'envoyer des données à un serveur pour traitement.
- **Contrôleur (Controller)** : Fichier PHP qui gère la logique métier en recevant et traitant les requêtes utilisateur.

- **Validation utilisateur** : Processus permettant de vérifier si un utilisateur est authentifié avant d'accéder à certaines fonctionnalités.
- **Objet (Classe PHP)** : Structure de programmation définissant des propriétés et des méthodes pour représenter une entité (ex. : un événement).
- **Getter et Setter** : Méthodes permettant d'accéder et de modifier les propriétés d'un objet de manière sécurisée.
- **DAO (Data Access Object)** : Classe responsable de l'interaction avec la base de données pour récupérer, insérer, modifier ou supprimer des données.
- **Base de données** : Système permettant de stocker et organiser des informations de manière structurée.
- **Insertion de données** : Action d'enregistrer des informations saisies dans un formulaire dans une base de données.
- **Exception (Erreur)** : Problème survenant lors de l'exécution d'un programme, pouvant être intercepté et géré.
- **Logs** : Fichiers ou entrées enregistrant des événements ou erreurs pour faciliter le suivi et le débogage.
- **Catégories (BDD)** : Données récupérées depuis une table spécifique pour être affichées dynamiquement dans un formulaire.
- **Redirection** : Action de renvoyer un utilisateur vers une autre page après une action spécifique.
- **Tests fonctionnels** : Vérification de la conformité d'une fonctionnalité par rapport aux attentes en effectuant des essais manuels ou automatisés.