Projet Web Backend Hetic

contact@ls-a.fr
https://www.ls-a.fr

Table des matières

Contexte

Dans le cadre de votre entreprise, il vous a été demandé de réaliser un système de billetterie en ligne. Pour ce faire, vous décidez de réaliser cet application en PHP.

Cette application disposera majoritairement d'APIs au travers d'un système d'authentification. Les personnes connectées pourront créer des événements et inscrire des visiteurs à ces événements. Une fois ces personnes inscrites, un lien pourra être généré pour télécharger leur billet.

Fonctionnalités

Votre application possèdera les fonctionnalités suivantes :

- Système d'authentification
  - Inscription
  - Connexion
  - Déconnexion
  - Validation du jeton
- Système de billetterie
  - Créer un nouvel événement
  - Modifier un événement
  - Annuler un événement
  - Inscrire un nouveau visiteur à l'événement
  - Annuler l'inscription d'un visiteur à l'événement
  - Générer un billet incluant un qrCode
  - Créer un lien pour télécharger ce billet
  - Créer un lien pour valider le billet (système anti-fraude)

Système d'authentification

> **Attention**
>
> Ce système doit être distinct et complètement indépendant du système de billetterie (y compris base de données). Par exemple, si ce système est lancé avec la commande `php -S localhost:8000`, le système de billetterie pourrait être lancé avec la commande `php -S localhost:8080`.

Le système d'authentification est une API avec trois méthodes détaillées ci-après. Il est associé à une base de données contenant deux tables :

- utilisateurs
- jetons

Le but de ce système d'authentification est de générer des chaînes de caractères **uniques et non prédictibles** à chaque fois qu'une personne se connecte avec succès à l'application. On appelera par la suite cette chaîne de caractères un "jeton".
Ce jeton doit être stocké dans la base de données tant que la personne est connectée.

Le système de billetterie interrogera le système d'authentification avec une demande simple : "Est-ce que ce jeton que je t'envoie correspond bien à un utilisateur connecté ?". Dans l'affirmative, le système de billetterie pourra effectuer l'action demandée. Dans le cas contraire, ce dernier renverra une erreur.

Inscription

Cette méthode est disponible uniquement pour faciliter le projet et vous permettre de créer des comptes utilisateur. Dans une version ultérieure du projet, votre entreprise empêchera l'inscription.

> **Notes**
>
> - Méthode : POST
> - URL : /register
> - Type de contenu : JSON
>
> Exemple de données envoyées :
>
> ```
> { "identifiant": "John", "motdepasse": "TheJohn10" }
> ```

Retours possibles :

- **En cas de mauvaises syntaxe du JSON de la requête : 400 - Bad Request**

```
{ "statut": "Erreur", "message": "JSON incorrect" }
```

- **En cas de succès : 200 - OK**

```
{ "statut": "Succès", "message": "" }
```

Connexion

Notes

- Méthode : POST
- URL : /login
- Type de contenu : JSON

Exemple de données envoyées :

```
{ "identifiant": "John", "motdepasse": "TheJohn10" }
```

Retours possibles :

- **En cas de mauvaises syntaxe du JSON de la requête : 400 - Bad Request**

```
{ "statut": "Erreur", "message": "JSON incorrect" }
```

- **En cas de mauvais identifiants : 401 - Unauthorized**

```
{ "statut": "Erreur", "message": "Identifiants incorrects" }
```

- **En cas de succès : 200 - OK**

```
{ "statut": "Succès", "message": "nQnn6173KJ76CN..." }
```

La propriété `message`, en cas de succès, renvoie un "jeton".
Ce jeton doit être également stocké côté serveur pour confirmer que oui, le système d'authentification a bien associé cette chaîne de caractères à l'utilisateur "John".

Déconnexion

Notes

- Méthode : GET
- URL : /logout
- Type de contenu : JSON

Exemple de données envoyées :

```
{ "jeton": "nQnn6173KJ76CN..." }
```

Retours possibles :

- **En cas de mauvaises syntaxe du JSON de la requête : 400 - Bad Request**

```
{ "statut": "Erreur", "message": "JSON incorrect" }
```

- **En cas de mauvais jeton : 401 - Unauthorized**

```
{ "statut": "Erreur", "message": "Jeton inconnu" }
```

- **En cas de succès : 200 - OK**

```
{ "statut": "Succès", "message": "" }
```

Cette action supprime côté serveur le jeton généré préalablement. Cette étape est importante pour l'action détaillée ci-après.

Validation du jeton

Notes

- Méthode : POST
- URL : /verify
- Type de contenu : JSON

Exemple de données envoyées :

```
{ "jeton": "nQnn6173KJ76CN..." }
```

Retours possibles :

- **En cas de mauvaises syntaxe du JSON de la requête : 400 - Bad Request**

```
{ "statut": "Erreur", "message": "JSON incorrect" }
```

- **En cas de jeton incorrect : 401 - Unauthorized**

```
{ "statut": "Erreur", "message": "Jeton incorrect" }
```

- **En cas de succès : 200 - OK**

```
{
  "statut": "Succès",
  "message": "",
  "utilisateur": {
      "identifiant": "John"
  }
}
```

Cette action est appelée par le système de billetterie. Elle permet de renvoyer les informations utilisateur associées à un jeton tout en vérifiant si la personne est bien authentifiée.

Système de billetterie

Le système de billetterie disposera de différentes pages accessibles sans être identifié :

- Connexion
- Déconnexion
- Afficher un billet
- Valider un billet

Et différentes pages accessibles après être identifié (espace réservé à l'équipe de gestion des événements) :

- Créer/Modifier/Annuler un événement
- Ajouter/Annuler un visiteur à l'événement
- Visualiser les événements et leurs inscrits

Pour les explications suivantes, on considèrera qu'un billet possède au moins quatre propriétés :

- L'événement auquel il est relié
- Le visiteur auquel il est relié
- Un code **unique et non prédictible** "public"
- Un identifiant **unique et non prédictible** "privé"

Le code public sert à valider un billet et est constitué d'une trentaine de caractères alphanumériques. Toute personne peut le demander, en parallèle du nom du visiteur, pour valider l'inscription. Ce code public sera encodé dans un qrCode, pour valider en quelques secondes que le billet n'a pas été contrefait.

L'identifiant privé, quand à lui est constitué de 6 à 10 caractères contenant des lettres en majuscules ou des chiffres. Il est prévu pour être "simple" à recopier et n'est donné qu'au propriétaire du billet. En effet, ce code permet de générer et d'imprimer le billet sur le système de billetterie. Toute personne disposant de ce code pouvant générer le billet, il ne doit pas être communiqué.

Connexion

Affiche un formulaire de connexion. Le formulaire doit présenter deux champs, avec le couple identifiant/mot de passe. Une fois les données récupérées, le système de billetterie enverra **lui-même** une requête au système d'authentification.

> **Notes**
>
> En cas de succès, il faudra stocker le jeton dans la session de l'utilisateur (donc côté serveur). N'oubliez pas d'afficher de nouveau le formulaire, avec un message d'erreur, si les identifiants sont incorrects.

Déconnexion

Ce lien, quand il est cliqué, doit effectuer trois actions :

- Envoyer une requête au système d'authentification pour déconnecter l'utilisateur (page /logout)
- Supprimer la session de l'utilisateur
- Rediriger vers la page de connexion

Afficher un billet

Affiche un formulaire pour accéder au billet. Le formulaire doit contenir deux champs :

- Le nom du visiteur
- L'identifiant privé du billet

Une fois les données récupérées, si les informations sont valides, le billet doit s'afficher. Un billet contient à minima les informations suivantes :

- Nom du visiteur
- Nom de l'événement
- Date de l'événement
- Date de génération de ce billet
- qrCode permettant de valider le billet

Le qrCode sera par la suite scanné "dans la vraie vie" par les équipes en charge de l'événement. Ce qrCode contiendra un unique lien de la forme suivante :

```
http://localhost:8080/validate?nom=John&code=hE6yuK981AAb...
```

Cette action est détaillée plus bas, dans la section "Valider un billet".

> **Notes**
>
> Pour ce projet, il n'est pas nécessaire de générer le billet au format PDF. Une simple page HTML contenant ces informations est suffisante : si le visiteur veut imprimer son billet, il pourra tout simplement utiliser la fonction "Imprimer" de son navigateur favori.

Valider un billet

Affiche un formulaire pour valider un billet. Le formulaire doit contenir deux champs :

- Le nom du visiteur
- Le code public du billet

Ce formulaire doit envoyer les données en GET, sous la forme suivante :

```
http://localhost:8080/validate?nom=John&code=hE6yuK981AAb...
```

Il est important que les données soient envoyées en GET, car ce lien doit être encodé dans le qrCode.
Cette action retournera :

- En cas de succès :
  - Une simple page web avec un fond vert
  - Un code HTTP 200
- En cas d'erreur :
  - Une simple page web avec un fond rouge
  - Un code HTTP 401

Ainsi, avec un simple lien, il sera possible de vérifier si le billet existe ou non. C'est une des mesures importantes pour lutter contre la contrefaçon.

Créer/Modifier/Annuler un événement

Cette section est réservée aux personnes authentifiées. Elle est constituée d'une ou plusieurs pages, avec des formulaires, permettant de :

- Créer un nouvel événement : Nom, Date, ...
- Modifier les données d'un événement
- Annuler un événement : dans ce cas, l'événement continue d'exister dans la base de données, mais il n'est plus possible d'éditer des billets ou d'y inscrire des visiteurs

Inscrire/Annuler l'inscription à un événement

Cette section est réservée aux personnes authentifiées. Elle est constituée d'une ou plusieurs pages, avec des formulaires, permettant de :

- Inscrire une personne à un événement existant : après saisie de son nom, l'identifiant public et le code privé du billet doivent être générés automatiquement.
- Annuler l'inscription d'une personne à un événement : supprime la personne de la liste des inscrits, supprime aussi l'identifiant public et le code privé du billet.

Visualiser les événements et leurs inscrits

Cette section est réservée aux personnes authentifiées. Elle est constituée d'une ou plusieurs pages, permettant de consulter la liste des événements avec leurs noms, leurs dates, et la liste des inscrits à chaque événement.

Bonus : mesure supplémentaire anti-contrefaçon

Ajoutez une action de "consommation" des billets. Cette adresse ne pourra être appelée qu'une fois par billet et permettra d'éviter que deux personnes aient le même billet. Cette action de "consommation" des billets sera réalisée au scan du qrCode par l'équipe de l'événement.

Etape 1 :

Ajoutez un troisième code généré au billet, le code de consommation. Comme celui public, il doit être **non prédictible et complexe**.

Etape 2 :

Créez une nouvelle action de consommation d'un billet, avec une adresse similaire à celle pour la validation d'un billet. Exemple :

```
http://localhost:8080/consume?nom=John&code_consommation=AA86hj5pM...
```

Cette action recevra :

- le nom du visiteur
- le code de consommation

Cette action sera un succès si le nom est bien associé au code et si le billet n'a pas déjà été consommé. Une fois la consommation réalisée, le billet sera considéré comme "consommé".

Cette action renverra :

- En cas de succès :
  - Une simple page web avec un fond vert
  - Un code HTTP 200
- En cas d'erreur :
  - Une simple page web avec un fond rouge
  - Un code HTTP 401

Etape 3 :

Dans la page de visualisation des inscrits à un événement, ajoutez un code couleur pour savoir quel billet a été consommé : cela permettra de tracer des statistiques sur la participation aux événements.
Ajoutez également deux boutons, permettant de considérer :

- Un billet non consommé comme "déjà consommé"
- Un billet consommé comme "non consommé"

Focus sur l'interface

Vous êtes libres d'utiliser Bootstrap, Tailwind, jQuery, ou autre bibliothèque frontend permettant de réaliser rapidement une interface agréable et responsive.

Attention, la logique doit rester côté backend et le rendu de la page doit être effectué côté serveur. Vous n'êtes pas autorisé(e) à utiliser des outils comme React, Angular ou VueJS.

Modalités

Les modalités de ce projet sont les suivantes :

- Ce projet devra être réalisé en groupe de 6 à 8 personnes
- Votre rendu devra être effectué dans une archive ZIP envoyée par mail, nous vous

conseillons donc une taille inférieure à 25Mo (voir livrables).

- Les seuls langages autorisés pour ce projet sont HTML, CSS et JavaScript côté front, ainsi que PHP et SQL côté back.
- Vous êtes autorisés à utiliser composer, tant que les seuls packages installés sont GuzzleHttp et/ou ceux vus en cours.
- Les frameworks PHP ne sont pas autorisés.

Le rendu du projet

Le projet doit être rendu avant le 30 avril 2023 à 23h59, à l'adresse renaud@ls-a.fr. Votre rendu devra être accompagné de la liste des membres de votre groupe. Ce rendu devra être réalisé sous la forme d'un ZIP en pièce jointe du mail. Attention donc à la taille de votre rendu. Les rendus par liens (WeTransfer, Git) ne seront pas pris en compte.

Bonus potentiels

L'ajout de fonctionnalités supplémentaires ou un travail particulièrement qualitatif sur un élément du projet peut donner droit à des points supplémentaires.

Malus potentiels

Des malus pourront être appliqués :

- En cas de copier/coller trop important d'une source, que ce soit internet ou un autre groupe
- Retard d'envoi du projet
- Hors sujet
- Faille de sécurité manifeste et vue en cours (requêtes SQL non préparées, mots de passe non hashés)