HEI API | Évaluation FastAPI & Postman - 23 juillet 2025

Pré-requis :

- Avoir un serveur uvicorn avec les modules de FastAPI fonctionnel
- Avoir une base de code sur FastAPI déjà présent sur un repository GitHub publique
- Avoir un Postman installé sur votre machine
- Remplir le formulaire suivant : https://forms.gle/xPEaCNErmhQc28vu9

NB :

- AUCUN DOCUMENT AUTORISÉ - Examen individuel
- Google, IA (ChatGPT, etc …) ou tout autre support ou source d'informations autre que les **codes** issus des derniers TD NON AUTORISÉ : seuls, les codes sont autorisés et **non** les énoncés
- Toute **tentative de fraude** entraînera immédiatement une note éliminatoire de zéro à la matière *(et potentiellement un redoublement, cf règlement)*
- Seuls, les projets qui sont push sur un repository GitHub **publique** vont être considérés et notés. Les projets n'ayant pas été push sur GitHub obtiendront directement une note éliminatoire de zéro.
- **15 minutes de retard** après le début des épreuves acceptées uniquement, au-delà de quoi, vous n'allez plus être autorisé à intégrer la salle.

Durée de l'épreuve : 1h30 > Dernier commit sur GitHub ne doit pas dépasser donc l'heure de début de l'épreuve + 1h30. Par exemple, si l'épreuve commence à 8h15, alors le dernier commit doit être 9h45. Malus pour des commits après la durée de l'épreuve.

Sujet pour K1-K2-K3 :

Q1 : Créer une route GET /hello, qui ne prend rien en paramètres et qui retourne un fichier HTML contenant le message "Hello" dans le corps. (3 points)

Q2 : Créer une route GET /welcome, qui prend le paramètre de requête "name" de type chaîne de caractère, et qui retourne le message "Welcome <name>" comme corps de la réponse avec une code de status 200 OK, où <name> est la valeur du paramètre fourni. Le corps de la réponse peut être soit en texte brute, soit en JSON avec l'attribut de votre choix, mais pas les deux. (3 points)

Q3 : Créer une route POST /players, qui prend dans le corps de la requête une liste d'objet JSON qui a les attributs suivants :

- Number, de type nombre entier

- Name, de type chaîne de caractères

Le comportement attendu par cette requête est de mémoriser en mémoire (vive), la liste des objets qui ont été fournis à travers le corps de la requête, et la réponse attendue est de retourner la liste contenant les objets students mémorisés en mémoire vive, c'est à dire ceux qui viennent d'être ajoutés et ceux qui ont déjà existés avant l'ajout. Le code de statut attendu possède la signification CREATED et pas le code de statut générique dont la signification est OK (3 points)

Q4 : Créer une route GET /players qui ne prend rien en paramètre, et qui retourne le contenu de la liste d'objets players actuellement stockés en mémoire avec un code de status 200 OK. (2 points)

Q5 : Créer une requête idempotente à travers une nouvelle route PUT /players, en utilisant l'attribut "Number" comme identifiant unique. Autrement dit, si le Number fournie dans le corps existe déjà, alors effectuer une modification si les valeurs ont été modifiées, sinon effectuer un ajout. (4 points)

Q6 : Pour chaque route qui a été définie plus tôt, vous allez maintenant uploader un capture d'écran pour illustrer que vous maîtrisez Postman comme client HTTP. Pour cela :

- Créez un nouveau dossier intitulé "postman" dans votre projet
- Renommer chaque fichier de la capture respectivement selon la route que vous avez créé précédemment. Par exemple, si votre route affichée sur Postman est GET /hello, cela correspond à la Q1, donc vous allez renommer le fichier Q1.

  Pour les 5 questions, vous devez donc avoir 5 images de captures d'écran intitulées Q1 jusqu'à Q5, si vous avez tout terminé.

Chaque requête sur Postman vaut 1 point, et le point est obtenu si et seulement si la route (fonctionnalité) marche correctement. Autrement dit, soit vous avez 1, soit vous avez 0.

BONUS 5 points (soit 5 points, soit 0) :
Dupliquez la route GET /students et renommer la route dupliquée en GET /players-authorized. Vérifiez que chaque requête reçue par le serveur pour cette route dupliquée contient l'en tête "Authorization" avec comme valeur "bon courage".

- Si aucune en-tête "Authorization" n'est trouvée, alors le serveur doit retourner une réponse indiquant que le client **n'est pas autorisé** à accéder à la ressource demandée et veillez à bien utiliser le code de statut correspondant.

- Si l'entête "Authorization" est bien trouvée, mais que sa valeur n'est pas la chaîne de caractère "bon courage", alors le serveur doit retourner une réponse indiquant que le client n'a pas les permissions nécessaires pour accéder à la ressource demandée, et veillez à bien utiliser le code de statut correspondant.

- Si l'en tête “Authorization” est bien trouvée avec la bonne valeur “bon courage”, alors le comportement reste le même que celui de la route initiale GET /players, c’est à dire le contenu de la liste d’objets players actuellement stockés en mémoire avec un code de status 200 OK.

En cas de besoin, voici une liste des méthodes natives des listes Python.

Méthode	Description
`append(x)`	Ajoute un élément à la fin
`extend(iterable)`	Ajoute plusieurs éléments
`insert(i, x)`	Insère un élément à une position donnée
`remove(x)`	Supprime la **1re occurrence** de `x`
`pop(i)`	Supprime et retourne l’élément à l’indice `i` (ou le dernier si omis)
`clear()`	Supprime tous les éléments
`index(x)`	Donne l’indice de la 1re occurrence de `x`
`count(x)`	Compte le nombre d’occurrences de `x`
`sort()`	Trie la liste (sur place)
`reverse()`	Inverse la liste (sur place)
`copy()`	Fait une copie superficielle de la liste