Épreuve finale

Services Web- Hiver 2024

Intentions pédagogiques

Ce travail permettra à l'élève de démontrer sa capacité à développer un service d'échange de données REST et d'effectuer son intégration dans une application de test.

Il est en lien avec la compétence *00SV : Effectuer le développement de services d'échange de données*.

Mise en situation

L'entreprise pour laquelle vous travaillez veut développer une application de liste de tâches et vous veux sur le projet. Votre rôle dans le développement sera de créer le service web selon les spécifications du gestionnaire de projet. La conception de l'interface sera faite par une autre équipe mais on vous demande de développer une page d'ajout d'utilisateur pour les tests à l'interne. Comme ce service sera utilisé par deux autres équipes de développeur vous devrez rédiger une documentation exhaustive du service Web.

Responsabilités des élèves et de l'enseignant

- Compléter le travail individuellement à partir des consignes suivantes (élèves)
- Autoévaluer le travail à l'aide de la grille de vérification (élèves)
- Participer à la rencontre formative (élèves et enseignant)
- Répondre aux questions de élèves (enseignant)

Plagiat et utilisation des outils de générations de code

- L'usage des outils de génération de code est interdit dans ce travail.
- Vous devez être en mesure de comprendre et d'expliquer le fonctionnement de l'entièreté de votre code.
- Tout code emprunté d'une source externe doit être cité conformément à ce qui est décrit dans cette page : https://techinfo.profinfo.ca/code-emprunte/
- Tout manquement pourra être considéré comme un plagiat.

Consignes

Le service web qu'on vous demande de développer permettra de faire une gestion d'une liste de tâche. Chaque tâche comporte un titre, une date de début et d'échéance et un statut (*terminée* ou *en cours*). Optionnellement on peut aussi y ajouter une description plus détaillée, une date d'échéance et une liste de sous-tâches. Une sous-tâche comporte uniquement un titre et un statut (*terminée* ou *en cours*). On devra fournir une clé api associé à un usager pour accéder au service et les tâches seront regroupées par usager. Donc on ne pourra consulter que les tâches de la clé qui est fournis dans la requête.

1) Préparation de la base de données

Vous retrouverez en annexe A un diagramme entité-relation représentant la base de données à créer. Créez un script SQL qui prendra en charge la création de la base de données, des tables et l'insertion des données initiales au besoin.

2) Le service Web

Voici les fonctionnalités à inclure dans le service:

- Il doit être développé en JavaScript avec la librairie Express
- Le code doit être structuré en routes, contrôleurs et modèles.
- Chaque route doit comporter une gestion des erreurs et retourner le code HTTP adéquat (Succès, ressource non trouvée, erreur serveur, erreur d'autorisation)
- Les données retournées par les requêtes doivent toujours être au format JSON.
- La structure des réponses est laissée à votre discrétion.
- Toutes les erreurs de code 500 doivent être inscrite dans un fichier de journal des erreurs. (Vous pouvez utiliser la librairie *Morgan* pour vous aider)
- Toutes les routes doivent être documentées avec openAPI et une route doit permettre d'accéder à la documentation.

Routes à créer

Voici la liste des routes à créer. Vous devez utiliser la méthode HTTP et le format de paramètres le plus adéquat pour chaque situation. Vous pouvez utiliser les noms que vous souhaitez pour les routes mais vous devez respecter les concepts décrits dans les notes de cours à la section *Les bonnes pratiques dans la conception d'un API*.

L'accès à toutes les routes doit être protégées par une clé api sauf pour les routes d'ajout d'un utilisateur, de récupération ou de génération d'une nouvelle clé et la documentation.

- ❑ **Afficher la liste de toutes les tâches de l'usager.**
  Par défaut seulement les tâches incomplètes seront affichées mais vous devez permettre l'option de les afficher toutes.

- ❑ **Afficher le détail d'une tâche**
  Son titre, sa description, son statut, sa date de début et d'échéance et la liste de ses sous-tâches ainsi qu'un indiquant détaillant si la sous-tâche est terminée ou non.

- ❑ **Ajouter, modifier, modifier le statut et supprimer une tâche**
  Le rôle de la route « modifier le statut » est uniquement de changer le statut de la tâche.

- ❑ **Ajouter, modifier, modifier le statut et supprimer une sous-tâche**
  Pour la gestion des sous-tâches vous avez deux options :
  - Refaire une série de routes comme pour la gestion des tâches
  - À chaque changement d'une sous-tâche (ajout, modification, changement de statut, suppression) vous utilisez la route de modification de la tâche et vous retournez toutes les informations de la tâche et de ses sous-tâches.

- ❑ **Ajouter un utilisateur**
  La route doit prendre en paramètre un nom, un prénom, une adresse courriel et un mot de passe, créer l'usager dans le système et lui retourner une clé api.

- ❑ **Récupérer sa clé api ou en redemander une nouvelle**
  En fournissant une adresse courriel et le bon mot de passe le système va retourner la clé api de l'utilisateur. Un paramètre optionnel permet de générer une nouvelle clé et de la retourner à l'utilisateur.

3) Interface de gestion des utilisateurs

- Vous devez créer une page HTML avec un script JavaScript qui fournit une interface permettant de :
  - Créer un nouvel utilisateur
  - Récupérer ou régénérer une nouvelle clé api.
- Utilisez la fonction Fetch en JavaScript pour interagir avec les deux routes de gestion des utilisateurs de votre api.
- Vous trouverez à l'annexe B un « wireframe » de la page à créer. Vous devez vous le reproduire le plus fidèlement possible.

4) Test de l'API avec Postman

- Créez une collection Postman avec une requête fonctionnelle pour chacune des routes de votre service Web.
- La collection doit se nommer **sw_épreuvefinale_da**.

5) Mise en ligne du service Web

- Votre **service Web** ainsi que la **base de données** associées doivent être mise en ligne sur le site de Render (https://render.com/)
- Vous pouvez utiliser votre propre hébergeur mais vous devez le validez avec votre enseignant.

6) Documentation du service web

- Toutes les routes de l'api doivent être documentées au format OpenAPI.
- Vous devez créer une page de présentation de la documentation avec le module Swagger-UI.
- Une route de votre api doit permettre d'afficher la page de documentation. Cette route est accessible à tous et n'est pas protégée par la clé api.

Grille d'évaluation

CRITÈRES	INDICATEURS	ÉCHEC (0%)	ACCEPTABLE (60%)	BIEN (80%)	EXCELLENT (100%)
Création juste d'une base de données à partir d'un modèle.	• Créer correctement la base de données. • Insérer les données initiales selon les besoins. • Respecter le modèle de données.	Le script de création de la base de données reproduit **insuffisamment** le modèle fournit. Les ajouts au modèle sont **rarement** pertinents.	Le script de création de la base de données reproduit **faiblement** le modèle fournit. Les ajouts au modèle sont **parfois** pertinents.	Le script de création de la base de données reproduit **partiellement** le modèle fournit. Les ajouts au modèle sont **souvent** pertinents.	Le script de création de la base de données reproduit **fidèlement** le modèle fournit. Les ajouts au modèle sont **toujours** pertinents.
	Résultat /5	**0-2**	**3**	**4**	**5**
Programmer la logique applicative du service.	• Programmer et intégrer des mécanismes d'authentification.	**Aucune** route de gestion de l'utilisateur sont programmées correctement. **Peu** de routes de gestion de la liste de tâches sont protégées par une clé d'api.	**Certaines** routes de gestion de l'utilisateur sont programmées correctement. **Certaines** routes de gestion de la liste de tâches sont protégées par une clé d'api.	La **majorité** des routes de gestion de l'utilisateur sont programmées correctement. La **majorité** des routes de gestion de la liste de tâches sont protégées par une clé d'api.	**Toutes** les routes de gestion de l'utilisateur sont programmées correctement. **Toutes** les routes de gestion de la liste de tâches sont protégées par une clé d'api.
	Résultat / 15	**0-8**	**9-10**	**11-13**	**14-15**
	• Programmer la réception des données d'entrées.	**Peu** de routes de gestions de la liste de tâches sont programmées correctement. Les paramètres sont **rarement** envoyés dans la requête http de manière approprié. Les paramètres sont **rarement** récupérés efficacement par le service Web.	**Certaines** routes de gestions de la liste de tâches sont programmées correctement. Les paramètres sont **parfois** envoyés dans la requête http de manière approprié. Les paramètres sont **parfois** récupérés efficacement par le service Web.	La **majorité** des routes de gestions de la liste de tâches sont programmées correctement. Les paramètres sont **souvent** envoyés dans la requête http de manière approprié. Les paramètres sont **souvent** récupérés efficacement par le service Web.	**Toutes** les routes de gestions de la liste de tâches sont programmées correctement. Les paramètres sont **toujours** envoyés dans la requête http de manière approprié. Les paramètres sont **toujours** récupérés efficacement par le service Web.
	Résultat / 15	**0-8**	**9-10**	**11-13**	**14-15**
	• Manipulation correcte des données de la base de données	**Peu** de requêtes à la base de données sont construites selon un choix **approprié** de commandes et de paramètres. Les résultats des requêtes sont **rarement** récupérés et utilisés de manière adéquate.	**Certaines** requêtes à la base de données sont construites selon un choix **approprié** de commandes et de paramètres. Les résultats des requêtes sont **parfois** récupérés et utilisés de manière adéquate.	La **majorité** des requêtes à la base de données sont construites selon un choix **approprié** de commandes et de paramètres. Les résultats des requêtes sont **souvent** récupérés et utilisés de manière adéquate.	**Toutes** les requêtes à la base de données sont construites selon un choix **approprié** de commandes et de paramètres. Les résultats des requêtes sont **toujours** récupérés et utilisés de manière adéquate.
	Résultat / 15	**0-8**	**9-10**	**11-13**	**14-15**

CRITÈRES	INDICATEURS	ÉCHEC (0%)	ACCEPTABLE (60%)	BIEN (80%)	EXCELLENT (100%)
	• Programmer l'envoi des données de sortie	Les données retournées par les requêtes sont **rarement** bien formatées au format JSON Le code de statut HTTP de la réponse est **rarement** adéquat.	Les données retournées par les requêtes sont **parfois** bien formatées au format JSON Le code de statut HTTP de la réponse est **parfois** adéquat.	Les données retournées par les requêtes sont **souvent** bien formatées au format JSON Le code de statut HTTP de la réponse est **souvent** adéquat.	Les données retournées par les requêtes sont **toujours** bien formatées au format JSON Le code de statut HTTP de la réponse est **toujours** adéquat.
	Résultat / 15	**0-8**	**9-10**	**11-13**	**14-15**
Programmer une application de test utilisant le service	• Utiliser de façon appropriée le service Web • Convertir les données fournies par le service Web en données exploitable par l'application de test.	Une page HTML permet de tester **faiblement** les routes de gestions de l'utilisateur. La page HTML respecte **faiblement** le wireframe. Une collection Postman permet de tester **faiblement** les routes du service Web.	Une page HTML permet de tester **certaines** routes de gestions de l'utilisateur. La page HTML respecte **sommairement** le wireframe. Une collection Postman permet de tester **certaines** routes du service Web.	Une page HTML permet de tester **la majorité** des routes de gestions de l'utilisateur. Une collection Postman permet de tester **la majorité** des routes du service Web.	Une page HTML permet de tester **la majorité** des routes de gestions de l'utilisateur. La page HTML respecte **parfaitement** le wireframe. Une collection Postman permet de tester **l'intégralité** des routes du service Web.
	Résultat / 20	**0-11**	**12-13**	**14-17**	**18-20**
Configurer et migrer le service Web chez un hébergeur Web	• Appliquer correctement de la procédure de migration du service sur le serveur externe. • Appliquer de manière rigoureuse les mesures de sécurité	Le service Web **n'est pas en ligne** et chez un hébergeur. La base de données **n'est pas en ligne** chez un hébergeur. Les variables d'environnement utilisées dans le code **ne sont pas** configurées adéquatement chez l'hébergeur.	Le service Web **est en ligne** et chez un hébergeur. La base de données est **en ligne** chez un hébergeur. Les variables d'environnement utilisées dans le code sont **partiellement** configurées adéquatement chez l'hébergeur.		Le service Web **est en ligne** et chez un hébergeur et fonctionnel. La base de données est **en ligne** chez un hébergeur et accessible à distance. Les variables d'environnement utilisées dans le code sont **toutes** configurées avec adéquatement chez l'hébergeur.
	Résultat / 5	**0-2**	**3**	**4**	**5**
Rédiger la documentation	• Rédiger une documentation claire et exhaustive du service Web.	**Peu** de routes sont correctement documentées. Les routes sont **rarement** documentées selon la norme **OpenAPI**.	**Certaines** routes sont correctement documentées. Les routes sont **parfois** documentées selon la norme **OpenAPI**.	Une **majorité** de routes sont correctement documentées. Les routes sont **pour la plupart** documentées selon la norme **OpenAPI**.	**Toutes** les routes sont correctement documentées. Les routes sont **toutes** documentées selon la norme **OpenAPI**.
	Résultat / 10	**0-5**	**6**	**7-8**	**9-10**
Total / 100		Qualité de langue (10% maximum, -0,25 par faute) =			

CRITÈRES	INDICATEURS	ÉCHEC (0%)	ACCEPTABLE (60%)	BIEN (80%)	EXCELLENT (100%)
Note finale / 100		Commentaires :			

Modalités d'évaluation

Évaluation formative

- Une rencontre formative de rétroaction sera planifiée avec l'enseignant dans la semaine du 28 avril 2025. Cette rencontre est optionnelle.
- Une liste de vérification sera fournie pour vous permettre de vous autoévaluer.

Évaluation finale (pondération 50%)

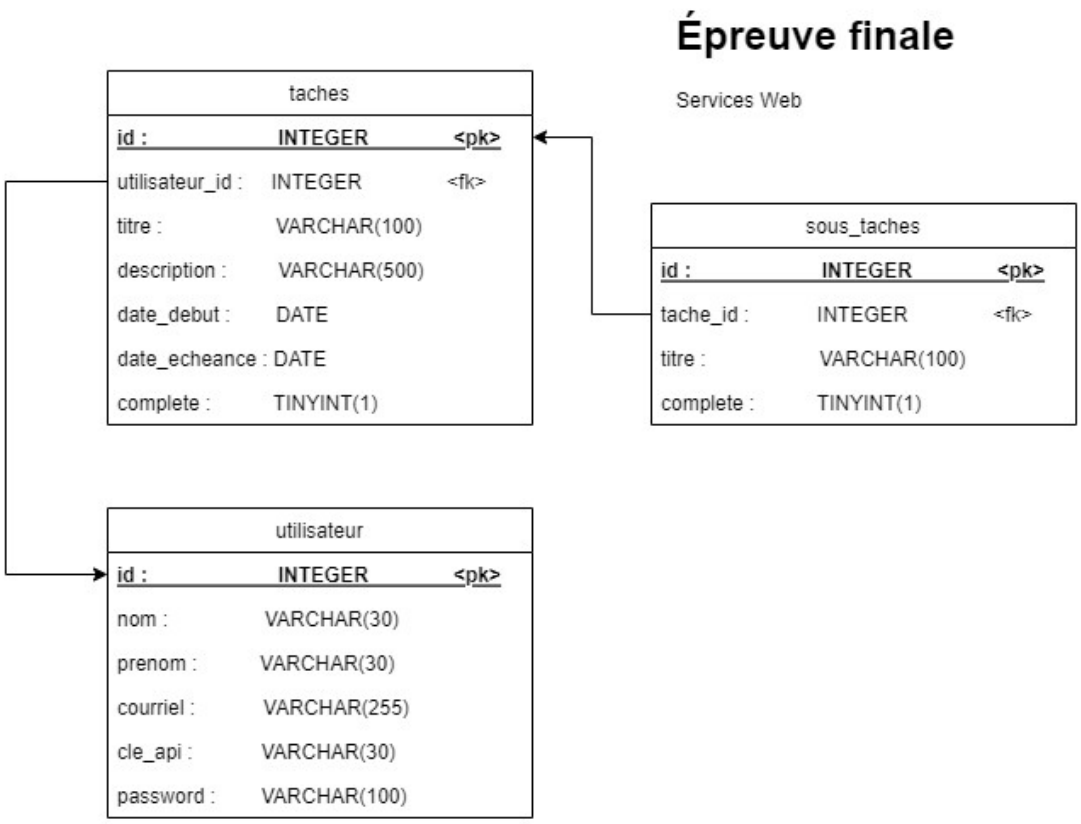
La date limite pour la remise du travail est le 16 mai 2025 à 17h00.

La qualité de la langue sera évaluée dans la documentation de votre service web. Une pénalité de 0,25% par faute de français sera retranchée de votre note.

Le projet devra être remis dans deux dépôts Github qui prendront la forme suivante :

- Un dépôt GitHub contenant votre service Web. (Il peut être le même que celui utilisé pour la mise en ligne sur *Render*)
- Un deuxième dépôt GitHub contenant les répertoires et fichiers suivants :
    - Un répertoire **Application JS** contenant la page Web de gestion de l'usager
    - Un fichier **Migration_bd.sql** contenant le script de migration *SQL* de la base de données.
    - Le fichier **Liste de vérification.docx**.
    - Le fichier d'exportation au format JSON de votre collection Postman.

Annexe A- Diagramme entité-relation



Annexe B- Wireframe de la page de gestion des utilisateurs