

Modalités

- Durée : **mini projet du 24/05/2024 à 16h45 au 02/06/2024 à 23h59** en dehors des créneaux des cours et projets, aucun retard ne sera accepté
- Technologies :
 - BackEnd : API Jersey basée sur **Tomcat 10 / JRE 17**. Stockage MySql
 - FrontEnd : Projet **Angular 17.3**
- **Recherches sur internet autorisées (citer vos sources pour respecter l'intégrité académique et éviter le plagiat.)**
- **Comme la recherche sur internet, s'entraider n'est pas interdit, à partir du moment où chacun maîtrise le code qu'il rendu, en répondant aux questions techniques posées lors de l'entretien.**

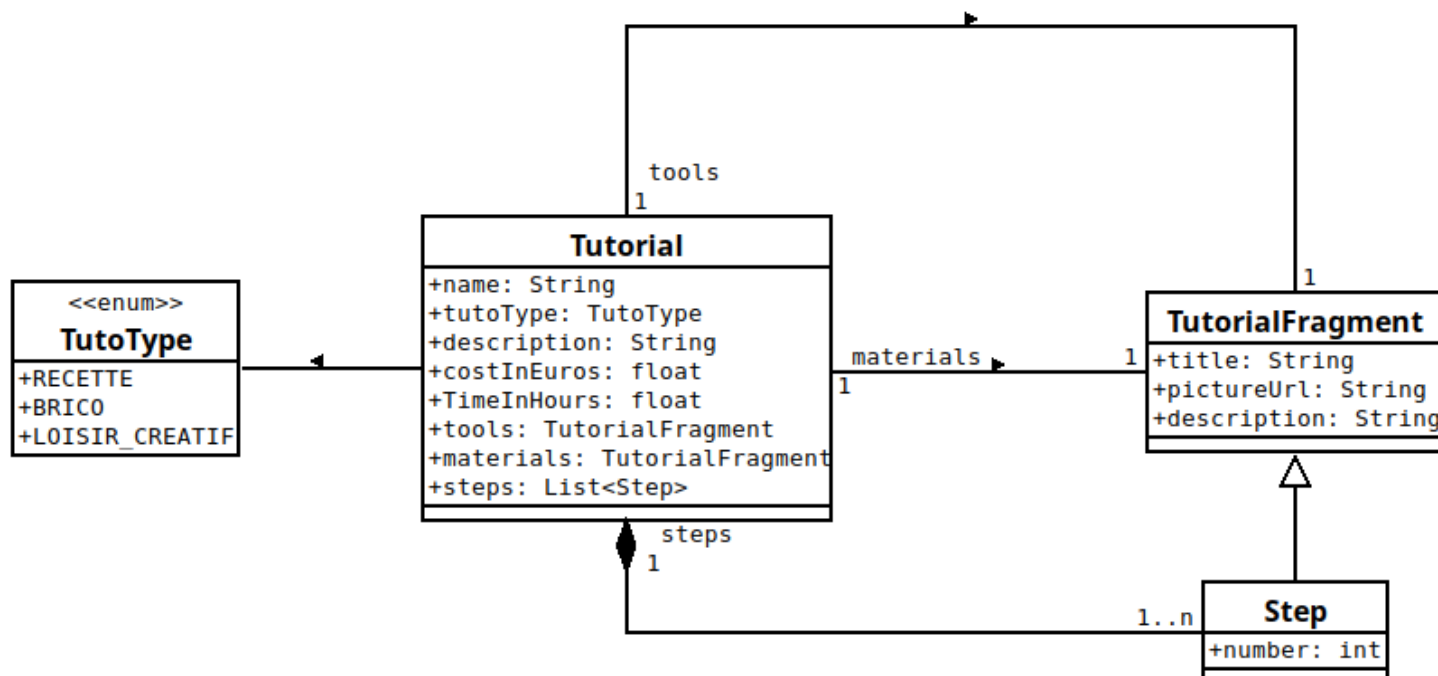
Contexte

Vous devez réaliser une application n-tiers permettant de réaliser des tutoriels de tout type sous la forme texte/images (les tutoriels vidéo ne seront pas gérés). Que ce soit des recettes de cuisine, des bricolages ou du loisir créatif.

Ces tutoriels auront les caractéristiques suivantes :

- Un tutoriel **doit** avoir un nom, un type (à choisir entre « recette de cuisine », « Bricolage » et « Loisir créatif »), un descriptif, une liste d'outils ou ustensiles, une liste de matériel ou ingrédients ainsi qu'une photo. Il **doit** aussi contenir une ou plusieurs étapes permettant de réaliser le tutoriel. Il **peut** avoir un temps de réalisation (en heures) et un coût estimé (en euros), mais ce n'est pas obligatoire.
- Une liste d'outils ou ustensiles **doit** être composée d'une image ainsi que d'un texte précisant cette liste. Si plusieurs outils sont nécessaires, ils seront représentés sur le même cliché et détaillés dans le texte.
- Une liste de matériel ou ingrédients **doit** être composée d'une image ainsi que d'un texte précisant cette liste. Si plusieurs ingrédients sont nécessaires, ils seront représentés sur le même cliché et détaillés dans le texte.
- Une étape est aussi composée d'une image ainsi que d'un texte. Elle comporte aussi un titre ainsi qu'un numéro séquentiel permettant de connaître sa place dans le tutoriel (ex : étape n°1 : découpe, étape n°2 : collage, étape n°3 : peinture, ...).

Voici **une proposition** d'UML correspondant au cahier des charges ci-dessus :



Sécurisation

Afin de simplifier le développement, **aucune sécurisation ne sera demandée** par défaut, même si, dans une version définitive, la rédaction, la modification ou la suppression d'un tutoriel devrait être réservée à des profils « rédacteur » (un rédacteur ne pouvant modifier ou supprimer que ses propres tutoriels), eux-même gérés par des profils « administrateur ».

Du fait de cette limitation, **aucune connexion ne sera demandée**, tout utilisateur pourra visualiser, créer, modifier ou supprimer tout tutoriel.

Interface utilisateur

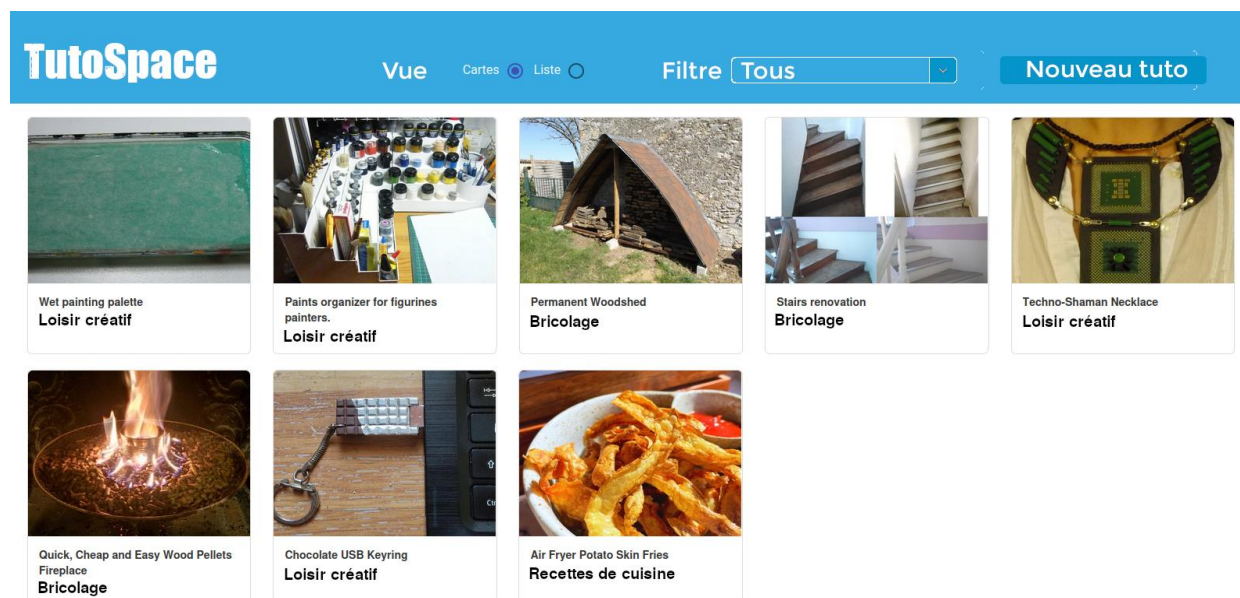
Écran principal

Description

L'écran principal de l'application comportera :

- Le nom de l'application (que vous pourrez choisir).
- Un moyen (via une list-box ou des radio-boutons) de changer le type d'affichage des tutoriels présentés :
 - soit via des « cartes » présentant une miniature moyenne de la photo du tutoriel avec, en dessous, le titre et le type de celui-ci (voir ci-dessous)
 - soit sous la forme d'une ligne comportant 4 colonnes (à vous de choisir l'ordre) :
 - une colonne affichant le type ;
 - une colonne affichant une petite miniature de la photo ;
 - une colonne affichant le titre ;
 - une colonne affichant tout (ou seulement le début si c'est trop long) de la description.
- Un filtre permettant de sélectionner tous les tutoriels ou seulement un type de tutoriel déterminé.
- Un bouton permettant de créer un nouveau tutoriel.
- À vous de déterminer quel moyen graphique (bouton, clic, ...) permettra de lancer l'édition et la suppression des tutoriels présents dans la liste.

Illustration 1: Exemple de page d'accueil



Actions utilisateur


- Quand l'utilisateur demande à changer de type de vue (cartes ou liste), cela changera le mode d'affichage la liste principale.
- Si l'utilisateur sélectionne un filtre, la liste principale devra refléter ce filtrage.
- Quand l'utilisateur clique sur le bouton « Nouveau », la liste sera remplacée par l'écran de détail tutoriel en mode création (donc avec ses champs vides).
- Quand l'utilisateur choisit d'éditer l'un des tutoriels présentés par la liste (quel que soit le moyen que vous aurez choisi), celle-ci sera remplacée par l'écran de détail tutoriel en mode édition (donc avec ses champs pré-remplis avec les données du tutoriel édité)
- Quand l'utilisateur choisit de supprimer l'un des tutoriels présentés par la liste (quel que soit le moyen que vous aurez choisi), celui-ci sera supprimé après une ultime confirmation de l'utilisateur. La liste devra alors répercuter cette suppression.

Écran de détail tutoriel

Description

Il vous est laissé toute liberté pour afficher le détail du tutoriel. Gardez cependant à l'esprit les règles métier imposées dans le paragraphe « Contexte » en début de ce document. Notamment le fait qu'un tutoriel doit avoir une liste d'outils, une liste de matériel et au moins une étape (mais peut en contenir plusieurs).

TutoSpace Abri à bûches Bricolage



Normal B I U G H1 H2 H3 H4 " </> = 🔗 Ix HTML

Ce tutoriel explique comment fabriquer un abris à bûches à partir d'anciennes douelles de gros tonneau et de divers matériaux.

Cliquez pour ajouter une image

Normal B I U G H1 H2 H3 H4 " </> = 🔗 Ix HTML

Cliquez pour ajouter une image

Normal B I U G H1 H2 H3 H4 " </> = 🔗 Ix HTML

Ajouter étape Valider

Illustration 2: Exemple de détail tutoriel

Actions utilisateur

Quelles que soient les options choisies, l'utilisateur doit pouvoir

- charger une image à chaque fois que c'est nécessaire.
- rentrer du texte (avec un éditeur WYSIWYG si possible).
- ajouter une étape ou en supprimer une existante. Si une étape autre que la dernière est supprimée, la numérotation des étapes doit être recalculée.

Enfin l'utilisateur ne peut valider tant que toutes les règles ne sont pas respectées.

API

Tous les stockages de votre application doivent passer par l'API, **il ne peut y avoir de stockage de données (sauf temporaire) dans votre front Angular.**

Logging

Chaque entrée dans l'un des services de votre API doit faire l'objet d'une trace de niveau INFO dans un fichier de logging appelé « TutoAPI_xx.log » où xx est un numéro séquentiel. Un nouveau fichier (avec un nouveau numéro) doit être créé **si et seulement si** la taille du précédent dépasse les 100Ko.

Toutes les sorties d'un service doivent aussi faire l'objet d'une trace au même endroit avec un niveau dépendant du statut HTTP de la sortie :

- INFO si on sort du service avec un statut OK (HTTP 200) ou CREATED (HTTP 201) ;
- WARNING si c'est un statut de type erreur externe (HTTP 4xx) ;
- ERROR si c'est un statut de type erreur interne du serveur (HTTP 500).

Tous les appels à la méthode « printStackTrace() » de la classe « Exception », **quel que soit l'endroit dans le code où elle est appelée**, doivent être remplacés par des traces en Log de cette stack trace. Si ces Exceptions interviennent ailleurs que dans un service, elles devront être tracées dans un fichier unique (pas de RollingFile) nommé « TutoErrors.log ».

Par défaut, le projet sera configuré pour tracer les logs de niveau INFO ou supérieur.

Tests Unitaires

Chacun de vos services doit être testé unitairement en utilisant une base de données volatile en RAM type hsqldb. **Les tests doivent être pertinents** et vérifier les règles définies dans le paragraphe « Contexte » en début de ce document. La couverture de test sur les classes de services de l'API devra être **au minimum de 85 %**.

Vous pouvez même commencer par écrire les tests avant les services eux-mêmes afin de vous initier au TDD (Test Driven Development).

Aucun test particulier n'est demandé sur les autres classes du projet.

Persistence

Le stockage sera réalisé dans une base MySQL en utilisant le framework JPA 2.1 et son implémentation EclipseLink 2.5.2.

Les classes du modèle JPA seront gérées via un ou des DAO(s) accessible(s) par **un singleton** DaoFactory.

Livrables

Vous rendrez votre travail à la fin du temps imparti sur le devoir Teams.

Livrable Front

Le livrable de la partie front de votre projet devra contenir au moins un zip du dossier « src » (**UNIQUEMENT LE DOSSIER SRC**) de votre projet Angular. Ce ZIP devra IMPERATIVEMENT être nommé comme suit :

S4IL_FRONT_202406_<votreCampus>_<votreNom>_<votrePrenom>.zip

(en changeant, bien entendu, les valeurs <votreCampus>, <votreNom> et <votrePrenom> par les valeurs correspondantes).

Si vous avez ajouté des bibliothèques externes à votre projet (via un « ng add xxx ») ou si vous avez des éléments supplémentaires à fournir à votre correcteur front, vous pouvez ajouter à votre rendu un fichier d'explication ou de précisions qui sera nommé :

S4IL_FRONT_202406_<votreCampus>_<votreNom>_<votrePrenom>_README.txt

Livrable Back

Le livrable de la partie back de votre projet consistera en un export de votre projet Eclipse :

❑ Sélectionnez le projet « Click Droit→Export...→General→Archive File »

❑ Vérifiez que le projet (et uniquement) s4il_miSemestre est bien coché.

❑ Vous déposerez cette archive dans Teams. Ce ZIP devra IMPERATIVEMENT être nommé comme suit :

S4IL_BACK_202406_<votreCampus>_<votreNom>_<votrePrenom>.zip

(en changeant, bien entendu, les valeurs <votreCampus>, <votreNom> et <votrePrenom> par les valeurs correspondantes).

Si vous avez des éléments supplémentaires à fournir à votre correcteur back, vous pouvez ajouter à votre rendu un fichier d'explication ou de précisions qui sera nommé :

S4IL_BACK_202406_<votreCampus>_<votreNom>_<votrePrenom>_README.txt

Note sur les livrables

Un mini projet de ce type constitue une somme de travail supplémentaire conséquente, par rapport à un devoir sur un créneau ou même une journée, pour vos correcteurs. De ce fait, votre projet doit se lancer sans autre manipulation que :

- l'import de votre projet back dans Eclipse puis son lancement (modulo le nom du runtime Tomcat qui peut changer de votre poste à celui du correcteur).
- le remplacement du dossier src d'un projet Angular vierge par celui extrait de votre zip puis le lancement de ce projet via un « ng serve ».

Les seules exceptions à cette règle qui seront acceptées seront les précisions apportées via les fichiers README.txt fournis et le correcteur sera seul juge de la pertinence ou non des justificatifs fournis. En cas de doutes, n'hésitez pas à poser des questions en amont (et pas la veille du rendu à 0h42!).

Listes des compétences évaluées

Comme d'habitude, nous vous précisons les compétences évaluées. **Prenez bien le temps de les lire et, pour chacune, posez-vous la question « Mon rendu couvre-t-il bien cette compétence ? »**

Nous avons trop souvent vu des rendus qui en faisaient plus que ce qui était demandé mais faisaient pourtant, par oubli en général, l'impasse sur une ou plusieurs compétences, pourtant listées dans ce paragraphe. Aller au rattrapage pour ce genre d'étourderie, ça peut être assez frustrant (pour le correcteur aussi).

Compétences Front

- Interpolation et property bind
- Utiliser event binding
- Créer et utiliser model
- Créer et utiliser component
- Créer et utiliser service
- Mettre en place des routes
- Redirection automatique
- Créer formulaire
- Utiliser des observables
- Effectuer des requêtes HTTP
- Employer les bonnes pratiques

Compétences Back

Module « Chaîne de Build »

- Intégrer Maven
- Gérer dépendance/repository
- Organiser projet (packages)

Module « Tests automatisés »

- Créer tests JUnit pertinents
- Etiqueter tests (annotations)

Module « Architecture-3-tiers-back »

- Mettre en place les DAO
- Gérer persist./annot. JPA
- Créer des NamedQueries
- Réaliser une API Rest
- Tracer les erreurs via Logger
- Employer les bonnes pratiques