Etablissement : ISET-Charguia	**Département :** Technologies de l'Informatique
Unité : Framework Côté Client	**Année Universitaire :** 2025 - 2026
Classes : DSI 21 & DSI 22	**Enseignante :** Amel TRIKI

Mini Projet : Développement d'un site pour valoriser le patrimoine Tunisien

Le ministère du tourisme vous sollicite pour développer un site qui met en valeur le patrimoine Tunisien.

Le patrimoine peut être :

- Archéologique (Amphithéâtre d'El Jem, Dougga, Carthage, Bulla Regia...)
- Architectural & Historique (Médinas de Tunis, Sousse; Ksour du Sud; Mosquées; Synagogues...)
- Naturel (Parcs nationaux, Ichkeul, Chott el-Jérid, Oasis...)
- Immatériel (Musique, costumes, traditions, cuisine, savoir-faire artisanal comme la poterie, le tissage...)

C'est à vous de déterminer la **thématique exacte sur** laquelle portera votre site.

Vous devez donc développer une application web de type CRUD[1] permettant de valoriser le patrimoine choisi

L'application doit comporter 2 parties :

- Une partie back office destinée à l'administrateur du site
- Une partie front office destinée aux internautes qui pourraient consulter votre application (sans authentification).

Chaque élément du patrimoine (site, parc, musique, tenue, ….) est décrit par :

- Un identifiant (id) de type string (unique, non géré manuellement)
- Un nom (ou libellé ou intitulé, …) pour désigner le nom du patrimoine
- Une photo
- 4 champs obligatoires à définir en plus de type : string, booléen, Date et number

Exemple : adresse (string), ouvert(boolean), dateCréation (Date) et prixEntree (number)

- Un champ de type tableau d'**objets** (Exp : liste des commentaires, lieux, …)
- D'autres champs si vous le souhaitez (catégorie, description, ...)

La nomination des champs est **libre**.

[1] CRUD : Create, Read, Update & Delete : Ajouter, afficher, modifier et supprimer

L'application doit répondre à des exigences fonctionnelles et techniques.

Exigences fonctionnelles :

1. Partie back office

La partie back office est destinée à l'administrateur, elle doit permettre de :

1.1. Consulter, ajouter, modifier, supprimer un patrimoine (que vous avez défini)

1.2. Rechercher un patrimoine

1.3. S'authentifier à l'application

1.4. Modifier son mot de passe

1.5. Ajouter une fonctionnalité complémentaire de votre choix

2. Partie front office

La partie front office est destinée à l'internaute qui va visiter le site, elle doit permettre de :

2.1. Consulter la liste des patrimoines. L'affichage doit être partiel en ne ciblant que quelque champs (par exemple : nom, photo et catégorie).

2.2. Consulter le détail d'un patrimoine (tous les champs), après sélection dans la liste.

2.3. Rechercher un patrimoine selon **2 critères combinables** à définir. L'utilisateur peut combiner ou utiliser séparément les 2 critères. Par exemple, il est possible de rechercher un patrimoine dont le nom commence par "a" et dont le prix d'entrée est inférieur à un certain montant, ou simplement de rechercher par nom.

2.4. Consulter une information externe (ex. : photos, météo actuelle, etc.) issue d'une API gratuite de votre choix, à l'emplacement que vous jugerez pertinent.

2.5. Consulter une page d'informations statiques présentant le site et ses (vous)

2.6. Ajouter une fonctionnalité complémentaire de votre choix

Exigences techniques :

L'application doit définir :

- Définir une **interface typescript** pour représenter le patrimoine et d'autres interfaces si nécessaire
- Contenir un ou plusieurs **services** pour la gestion de la logique métier.
- Afficher la **liste des patrimoines dans un composant parent**, chaque patrimoine étant représenté par un **composant enfant**.
- Inclure un ou plusieurs **pipes** dont au moins un **pipe personnalisé**
- Utiliser le **binding de classe** ([class])
- Disposer d'un **menu de navigation** adapté à l'internaute et d'un **menu distinct pour l'administrateur**.

- Définir **des routes différentes** pour l'affichage de la liste et du détail d'un patrimoine (front office).
- Inclure **des routes imbriquées (routes enfants)**.
- Utiliser des **formulaires réactifs avec validations**.
- Utiliser un ou plusieurs **frameworks CSS** au choix : *Bootstrap*, *Angular Material*, *Materialize CSS*, *Tailwind CSS*, etc.

De même l'application doit :

- Pouvoir accéder à un **serveur JSON**, les données étant sauvegardées dans **un fichier json unique.**
- Être versionnée avec Git et déposée sur Github
- Etre déployée sur Surge.sh ou autre plateforme équivalentz

Travail à rendre:

Vous devez remettre :

1. Au plus tard le **dimanche 2 Novembre 2024, 23h55** : sur la plateforme iset2026.uvt.tn : Une première version de la fiche de projet à compléter
2. Au plus tard le **dimanche 23 Novembre 2024, 23h55** sur la plateforme iset2026.uvt.tn :

- Un dossier compressé qui comporte :
- Votre projet Angular (sans node_modules ni .angular, ni .vscode) qui porte le nom : **Projet_Nom1_Nom2** où Nom1 et Nom2 désignent les noms du binôme (Exemple : Projet_Makni_BenAli).
- Une fiche de projet à compléter par le binôme/ monôme
- Un commentaire indiquant l'URL vers votre dépôt sur Github

(Exemple : https://github.com/Makni/Projet_Makni_BenAli.git)

Bonus:

Vous pouvez être bonifié dans le cas où :

- Vous intégrez un backend de votre choix (Node JS, Spring Boot,…)
- Vous utilisez d'autres concepts avancés d'Angular : les signals, les animations,…
- Vous utilisez un framework CSS autre que Bootstrap destiné à Angular tel qu'Angular Material.

Système d'évaluation:

L'évaluation portera sur un ensemble de critères tels que :

- La réponse aux exigences fonctionnelles et techniques
- La qualité du code, tests effectués, …
- L'ergonomie de l'application
- L'utilisation de git et de Github
- La complexité et l'implémentation des 2 fonctionnalités ajoutées

Il est à noter que :

- **Le meilleur mini-projet par classe se verra octroyé la note de 20.00 /20.00**
- 2 codes identiques seront sanctionnés d'un **zéro**.
- Des codes copiés d'une IA ou du net seront lourdement pénalisés.
- Le projet peut être fait en binôme ou en monôme, c'est au choix, c'est la qualité du projet qui sera évaluée indépendamment du nombre d'étudiants qui y travaillent
- Le mini-projet fera l'objet d'une validation, tout étudiant qui ne saura pas expliquer son code ou qui n'assistera pas à la séance de validation aura une note de **zéro**

L'application doit être fonctionnelle/exécutable ne comportant pas d'erreurs de compilation, autrement, la note sera systématiquement au-dessous de 5.