

Cégep de Sainte-Foy – Hiver 2024
Programmation Web Dynamique – 420-W21-SF

Travail Pratique 2

22 avril 2024

Préparé par
Benjamin Lemelin

1 Résumé

Créer un site web dynamique en PHP permettant la rédaction et la consultation de pense-bêtes. Le site web devra utiliser une base de données pour conserver les données de l'utilisateur.

2 Conditions de réalisation

Valeur de la note finale	Contexte	Nombre de remises	Durée
25%	En équipe	2	3 semaines

3 Mise en contexte

Noto est une plateforme en ligne conçue pour faciliter la gestion et l'organisation de notes personnelles. À la manière de *Google Keep*, les notes comprennent un titre, un contenu et la possibilité de choisir la couleur de fond.



Les utilisateurs sont invités à créer un compte sur la plateforme afin de bénéficier pleinement de ses fonctionnalités. *Noto* offre ainsi un espace virtuel où il est possible de stocker et organiser des idées, des listes, des rappels et autres informations importantes, accessibles depuis n'importe quel appareil.



4 Tâches à réaliser

4.1. Création du projet

Copiez le projet fourni avec cet énoncé dans un sous-dossier nommé **tp2** à la racine de votre serveur. Il contient plusieurs fichiers dont des images et la configuration pour *Visual Studio Code*. Vous trouverez aussi avec cet énoncé un document de design en PDF contenant, entre autres, les maquettes ainsi que les couleurs utilisées. Ce document est assez large : ne soyez pas surpris de devoir l'agrandir.

Prudence

De préférence, n'utilisez pas le lecteur PDF de votre navigateur. Installez plutôt [Adobe Acrobat Reader](#), qui est plus adapté pour ce genre de document.

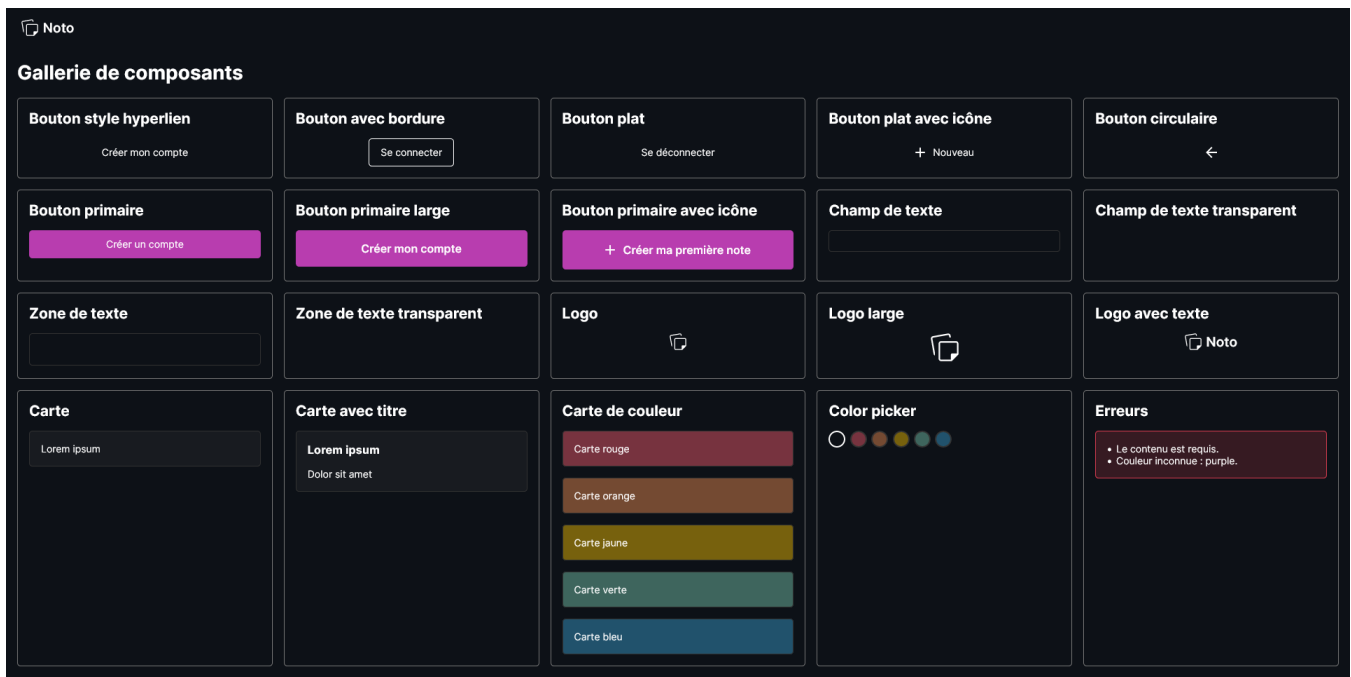
Une fois ouvert, agrandissez l'image avec la molette de la souris. Maintenez la touche **Espace** pour faire apparaître la main  et cliquez pour déplacer l'image. Notez que vous pouvez aussi utiliser la barre à outil sur le côté gauche pour activer l'outil main .

Vous aurez à écrire vous-même le code CSS de l'application. Si vous le préférez, vous pouvez également utiliser [Tailwind](#), mais ce n'est pas obligatoire. Dans tous les cas, il vous faudra tout de même utiliser le fichier **app.css**, inclus avec le projet, car il modifie la police de caractère.

4.2. Galerie de composants (Facultatif)

Il est conseillé de recréer les différents composants du site avant de créer les pages. Pour faciliter le travail, vous trouverez dans le projet de départ un fichier nommé **gallery.php** listant la majorité des composants. Il ne vous restera qu'à créer les styles CSS dans **app.css** (ou avec *Tailwind*).

Voici à quoi cela devrait ressembler. N'oubliez pas de consulter le document de design pour les détails.



4.3. Page d'accueil

Créez la page d'accueil (**index.php**) en vous fiant aux maquettes. Les boutons (des hyperliens) mènent à leurs pages respectives (lisez le libellé). Le logo et le nom de l'application (en haut à gauche) mène à la page d'accueil (c'est la même chose pour les autres pages). Sur la version mobile, le texte au centre de la page est légèrement plus petit, mais la disposition ne change pas (consultez le document de design).

Cette page change de vue si l'utilisateur est connecté. Consultez la section 4.6 pour les détails.

4.4. Page de création de compte

Créez la page de création de compte (**sign-up.php**) en vous fiant aux maquettes. Conservez les comptes dans une base de données (voir le script **CreateDB.sql**). N'oubliez pas de chiffrer le mot de passe.

i Information

Vous devez faire des [Data Access Object](#) pour chaque table. Cela dit, implémentez uniquement les opérations nécessaires; vous n'avez pas à faire toutes les méthodes.

En cas d'erreur de la part de l'utilisateur, affichez les messages en haut du formulaire. Voici les validations :

Champ	Détails
Courriel <i>Texte</i>	<ul style="list-style-type: none">• Obligatoire.• Non vide (pas seulement des espaces).• Doit être une adresse de courriel valide.
Mot de passe <i>Password</i>	<ul style="list-style-type: none">• Obligatoire.• Non vide (l'utilisateur peut mettre des espaces).
Confirmation <i>Password</i>	<ul style="list-style-type: none">• Obligatoire.• Doit être égal au mot de passe.

Information

La récupération et la validation des formulaires doit se faire avec des [Data Transfer Object](#). Il vous faudra un DTO pour chaque formulaire (que ce soit un formulaire **GET** ou un formulaire **POST**).

4.5. Page de connexion

Créez la page de connexion (**sign-in.php**) en vous fiant aux maquettes. Lorsque l'utilisateur est connecté, assurez-vous de le conserver dans la session pour plus tard. En cas d'erreur de la part de l'utilisateur, affichez les messages en haut du formulaire. Voici les validations :

Champ	Détails
Courriel <i>Texte</i>	<ul style="list-style-type: none">• Obligatoire.• Non vide (pas seulement des espaces).• Doit être une adresse de courriel valide.
Mot de passe <i>Password</i>	<ul style="list-style-type: none">• Obligatoire.• Non vide (l'utilisateur peut mettre des espaces).

4.6. Page de consultation des notes

Créez la page de consultation des notes (**index.php**) en vous fiant aux maquettes. Il s'agit d'une vue différente de la page décrite à la section 4.3. Les boutons (de simples hyperliens) mènent à leurs pages respectives (lisez le libellé). Le bouton « Se déconnecter » déconnecte l'utilisateur (détruisez la session).

Si l'utilisateur n'a aucune note d'associée à son compte, le site doit afficher un message invitant l'utilisateur à se créer une note. Sinon, affichez une grille avec les notes de l'utilisateur. Pour tester cette partie, créez manuellement quelques notes dans la base de données.

Défi (Facultatif)

Vous pouvez utiliser une grille de 3 colonnes en mode *Desktop*, ou créer une grille qui ajuste automatiquement le nombre de colonnes en fonction de la taille de l'écran. Dans ce cas, assumez que la taille maximale d'une note est de **400px**.

4.7. Page de création de note

Créez la page de création (**new.php**) de note en vous fiant aux maquettes. Les champs du formulaire sont transparents, de sortes à être intégré au design. Assurez-vous de ne pas oublier le bouton de l'en-tête (comprendre hyperlien) pour retourner à la page précédente.

En cas d'erreur de la part de l'utilisateur, affichez les messages **en bas** du formulaire. Voici les validations :


Champ	Détails
Titre <i>Texte</i>	<ul style="list-style-type: none">• Facultatif.
Contenu <i>Texte long</i>	<ul style="list-style-type: none">• Obligatoire.• Non vide (pas seulement les espaces).
Couleur <i>Boutons radio</i>	<ul style="list-style-type: none">• Obligatoire.• Doit être une des valeurs possibles :<ul style="list-style-type: none">○ clear○ red○ orange○ yellow○ green○ blue

Défi (Facultatif)

En JavaScript, modifiez l'arrière-plan du formulaire pour correspondre à la couleur choisie.

4.8. Page de modification de note

Créez la page de création (**edit.php**) de note en vous fiant aux maquettes. Il s'agit du même formulaire qu'à la page précédente, à la différence qu'il sert à modifier une note existante.

Pour modifier une note, l'utilisateur doit cliquer sur le bouton  de la note à modifier. Ce bouton (un simple hyperlien) n'est visible que si la note est survolée. Il mène à la page **edit.php**, incluant un paramètre dans l'URL pour indiquer l'identifiant de la note à modifier. Par exemple, pour la note 3 :


```
<a href="edit.php?id=3">
```

Vous trouverez cette valeur dans la variable **\$_GET** (tout comme les formulaires).

Information

Remarquez qu'il a très peu de différences visuelles entre cette page et la précédente. En fait, seul le bouton d'envoi du formulaire change. Pourquoi pas réutiliser la même vue ?

4.9. Suppression d'une note

Retournez dans la page de consultation de note (`index.php`). Ajoutez un bouton  sur les notes, permettant de la supprimer. Ce bouton (un simple hyperlien) n'est visible que si la note est survolée. Pour gérer cette opération, le plus simple est de créer une page `delete.php` recevant en paramètre l'identifiant de la note à supprimer. Par exemple, pour supprimer la note 3 :

```
<a href="delete.php?id=3">
```

4.10. Tests unitaires

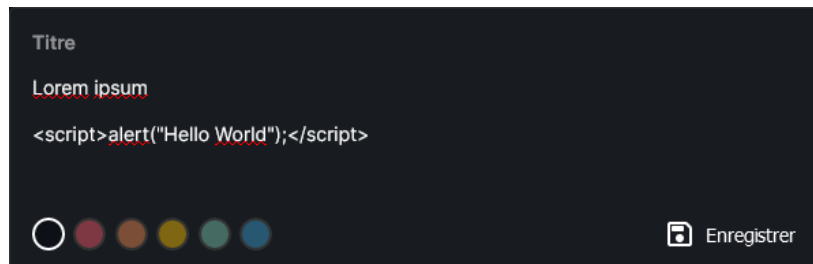
Il vous est demandé de tester unitairement vos *DTO*. Veuillez vérifier que les validations sont fonctionnelles. Il s'agit de la seule chose à tester pour ce travail.

Prudence

Les tests unitaires sont importants dans la mesure **où ils sont utiles**. Par exemple, il est inutile de tester unitairement des *getters*, car le potentiel à l'erreur est pratiquement inexistant. À l'inverse, le potentiel à l'erreur de vos validations est beaucoup plus élevé.

4.11. Prévention du XSS (*Cross Site Scripting*)

Dans votre site, essayez de créer une note ayant le contenu suivant :



À votre retour sur la page de consultation des notes, une boîte de dialogue devrait apparaître. Le script que vous avez tapé dans la note s'est exécuté.



Ce que vous avez devant vous est un type d'attaque informatique très répandue du nom de *Cross Site Scripting* (XSS). Elle consiste à injecter du code *JavaScript* dans une formulaire dans le but d'exécuter ce code sur l'ordinateur d'autres utilisateurs du site. Ce genre d'attaque exploite le fait que de nombreux sites n'effectuent pas [l'assainissement](#) de leurs données.

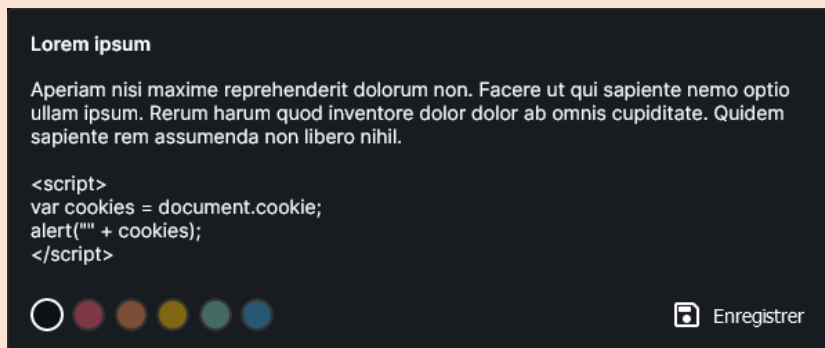
Fort heureusement, en PHP, il est possible d'envoyer toute chaîne de caractère potentiellement dangereuse à la fonction `htmlspecialchars`. Cette fonction s'occupe de substituer tout caractère pouvant être interprété comme du code HTML par des caractères de remplacement inoffensifs.

```
echo htmlspecialchars("<script>alert('Hello World');</script>");
```

Faites-en sortes d'assainir les données avant de les afficher dans vos pages. Il n'est pas nécessaire d'assainir les données avant de les placer dans la base de données : juste avant l'affichage.

Prudence

L'exemple précédent n'était pas très dangereux, mais le XSS n'en reste pas moins un vecteur d'attaque très sérieux. Voici un autre exemple permettant de voler des cookies. Bien que ce code ne fasse que les afficher dans un dialogue, il serait très facile de les envoyer à un serveur distant.



5 Contraintes

- Tout le texte doit être en français. Les écrans doivent **respecter les maquettes** montrées.
- Le code doit être en **anglais**. Les commentaires peuvent être en français si souhaité.
- Vous devez faire usage de [vues](#) pour **séparer la logique applicative** du visuel.
- Vous devez utiliser des **constantes** lorsque nécessaire.
- Vous devez **préremplir les champs** de vos formulaires avec les données précédemment envoyées.
- Vous devez faire des [Data Access Object](#) pour gérer les accès à la **base de données**.
- Vous devrez faire des [Data Transfer Object](#) pour **obtenir et valider** les données de formulaires.

6 Conseils

- Respectez les bonnes pratiques : découpez votre code en fonctions s'il se répète, appliquez les standards de nommage du langage et indentez correctement.
- Développez votre application directement sur la machine virtuelle en vous connectant en *SSH* avec *Visual Studio Code*. Cela vous donnera accès à une meilleure auto-complétion.
- Utilisez [git](#) pour partager le code au sein de votre équipe. Faites des *commit/push* régulièrement. Néanmoins, prenez quand même la peine de bien séparer le travail (par exemple, un *DAO* chaque).

7 Collaboration et plagiat

Ce travail ne peut être le produit d'une collaboration entre équipes ou de partage de code¹. Si de tels comportements sont observés, le professeur attribuera automatiquement la note de 0 %. Aussi, tout étudiant peut être convoqué à une rencontre afin de s'assurer de sa compréhension du travail remis.

Tout acte de plagiat, de tricherie et de fraude sera sanctionné. Constitue notamment un plagiat, une tricherie ou une fraude tout acte de copier, de fournir ou de recevoir volontairement de l'information lors d'un examen, de reproduire en tout ou en partie le travail d'une autre personne, qu'il s'agisse d'un document imprimé, multimédia ou électronique, sans y faire expressément référence, de remplacer un étudiant ou de se faire remplacer lors d'un examen, de remettre un travail réalisé par une autre personne, d'obtenir, posséder ou utiliser frauduleusement des questions ou réponses d'examen, d'utiliser du matériel, des applications, des sites Web ou des ressources non autorisés, de se faire aider par une autre personne lorsqu'il est demandé de réaliser l'évaluation seul, de falsifier les résultats de travaux ou d'examens.

Source : Politique d'évaluation des apprentissages du collège

8 Modalités de remise

8.1. Remise 1

Cette remise comprend les points 4.3 à 4.5. Il s'agit d'une remise partielle.

Remettez votre projet sur LÉA, dans la section travaux, à l'intérieur d'une archive *Zip*. **Incluez dans votre remise un fichier texte avec le nom et le matricule de tous les membres de l'équipe.** Une pénalité de 15 % est appliqué en cas de retard de moins d'une journée. Au-delà de ce délai, le travail est refusé et la note de 0 % est automatiquement attribuée.

Supprimez les fichiers et dossiers inutiles, c'est-à-dire :

- Le dossier **.vs**.
- Cet énoncé, le document de design et la grille de correction.

8.2. Remise 2

Cette remise comprend l'intégralité du projet. Les modalités sont les mêmes qu'à la première remise.

¹ Du code identique avec des variables renommées et/ou des espaces introduits est considéré comme du plagiat.

9 Évaluation

Fonctionnalités – Le Quoi – 50%
Page d'accueil : <ul style="list-style-type: none">• S'affiche si utilisateur non-connecté.
Page de création de compte : <ul style="list-style-type: none">• Courriel obligatoire, non vide et valide.• Mot de passe obligatoire.• Confirmation obligatoire et valide.• Compte créé si formulaire valide.
Page de connexion : <ul style="list-style-type: none">• Courriel obligatoire, non vide et valide.• Mot de passe obligatoire.• Connexion au compte dans la session s'il existe.
Page de consultation des notes : <ul style="list-style-type: none">• S'affiche si utilisateur est connecté.• Invite de création de note si vide.• Grille de notes de l'utilisateur (responsive).• Contrôles visibles seulement si note est survolée.• Suppression d'une note spécifique.• Déconnexion de son compte.
Page de création/modification de note : <ul style="list-style-type: none">• Titre, facultatif.• Contenu (texte long) obligatoire et non vide.• Couleur (bouton radio) obligatoire et valide.• Note créé/modifiée si formulaire valide.
Formulaires : <ul style="list-style-type: none">• Liste de messages d'erreur de validation.• Formulaires vides au lancement.• Données conservées au travers des changements de page.
Mise en page : <ul style="list-style-type: none">• Respect des maquettes.• Mise en page <i>Responsive</i> et <i>Mobile First</i>.

Compétences – Le Comment – 50%
<p>Préparer la base de données (00SU-3) (incluant, mais sans s'y limiter) :</p> <ul style="list-style-type: none"> • Rédaction de requêtes SQL (syntaxe générale). • Rédaction et utilisation de <i>Data Access Object</i>. • Utilisation de requêtes préparées. • Utilisation de transaction lors des requêtes en écriture.
<p>Programmer la logique applicative coté serveur (00SU-5) (incluant, mais sans s'y limiter) :</p> <ul style="list-style-type: none"> • Rédaction de code en PHP (syntaxe générale). • Classement des fichiers et des dossiers respecte les standards du cours. • Séparation de la logique application et du visuel. • Utilisation du bon verbe Http (GET, POST) pour l'envoi et la réception de formulaires. • Application du <i>Post/Redirect/Get</i> pour les formulaires. • Rédaction et utilisation de <i>Data Transfer Object</i>. • Validations des données. • Usage de sessions.
<p>Programmer la logique applicative coté client (00SU-6) (incluant, mais sans s'y limiter) :</p> <ul style="list-style-type: none"> • Rédaction de code en HTML (syntaxe générale). • Rédaction de code en CSS (syntaxe générale).
<p>Contrôler la qualité de l'application (00SU-7) (incluant, mais sans s'y limiter) :</p> <ul style="list-style-type: none"> • Rédaction de tests unitaires avec PHPUnit. • Tous les tests unitaires passent avec succès.

Pénalités – Rigueur
<p>Qualité générale du code (incluant, mais sans s'y limiter) :</p> <ul style="list-style-type: none"> • Propreté générale du code. • Mauvaises pratiques de programmation. • Formatage ou indentation déficiente. • Standard de nommage non respecté. • Cohérence de l'ensemble de l'œuvre. • Orthographe et grammaire (0.5% jusqu'à concurrence de 20 %).
<p>Remise du travail (incluant, mais sans s'y limiter) :</p> <ul style="list-style-type: none"> • Erreur ou avertissement à l'interprétation du code. • Non-respect des consignes de remise. • Fichier inutile ou temporaire remis avec le projet. • Travail remis en retard.

Important

La note de la catégorie « Compétences » dépend de la catégorie « Fonctionnalités ». Un étudiant ayant **25/50** pts sur les fonctionnalités (soit **50%**) ne pourra avoir plus de **25/50** pts sur les compétences (équivalent à **50%**).