2025 - TI 2 – Livre d’or

Objectifs

Création d’un formulaire responsive avec validation de formulaire permettant l’alimentation d’un livre d’or.

Compétences évaluées

Partie 1 - Test d’intégration, le 28 avril 2025

git : fork, clone, sauvegardes (commit), travail remis via un pull request sur GitHub

Wamp : création d’un hôte virtuel vers le dossier public du test

PhpMyAdmin : importation d’une DB contenant une table depuis un fichier .sql

Mysql (MariaDB) : exécutions de requêtes SELECT et INSERT en PDO

structure PHP : respect du Modèle Vue Contrôleur (**MVC**) en PHP

HTML / CSS : formulaire responsive

JS : Conditions à remplir pour valider l’envoi du formulaire

Partie 2 - Présentation, le 5 mai 2025

FTP : Le travail est mis en ligne sur le serveur du stagiaire, pendant la semaine de correction (entre le 29/04 et le 04/05).

SQL : Le travail est mis en ligne sur un serveur de base de données, pendant la semaine de correction (entre le 29/04 et le 04/05).

Présentation Publique : Présenter son travail de manière cohérente et pouvoir répondre aux questions concernant le code produit

Marche à suivre

- Créez un fork du dépôt TI2-2025 sur github.com puis clonez-le sur votre machine
- Créez un hôte virtuel dans WAMP sur le dossier …/TI2-2025/public/ nommé TI2-2025
- Créez une base de données dans phpMyAdmin en important le fichier ti2web2025.sql (en MariaDB ou MySQL)
- Enregistrez config.php.ini en config.php en vérifiant que cela correspond aux paramètres de votre base de données !

Consignes

Vous pouvez vous aider des fichiers vus au cours pour réaliser ce TI.
Vous pouvez utiliser des moteurs de recherche, mais pas d'**IA** !
Vous ne pouvez pas vous aider les uns les autres pendant ce premier jour de TI.

Il s'agit surtout d'une auto-évaluation de vos compétences !

Évitez de copier/coller des bouts de code **sans les comprendre**.
Pour prouver que vous avez compris, vous devriez être capable de les expliquer avec des '**commentaires personnels**', en **PHP** comme en **Javascript**.

Les IA ne doivent pas être utilisées pour ce TI, même en autocomplétion.

Vous pouvez par contre vous aider d'internet pour des bouts de code précis.

Faites un `commit` à chaque étape validée

Il doit y avoir un minimum de 10 `commits` pour ce TI, utilisez des noms qui représentent chaque étape de votre travail. **N'envoyez pas votre travail sur Github avant 16h30**

Tout commit après le 28 avril 2025 à 16h30 ne sera pas pris en compte lors de la première correction !

Pull Request

Faites un **`Pull Request` uniquement à la fin de votre TI**, **pas avant 16h30** !
Vérifiez que vous avez bien suivi les consignes avant de le faire.

Si vous avez fini et que vous êtes satisfait de votre travail, faites les points Bonus (vous pouvez les faire sur une autre branche, et envoyer une autre branche nommée bonus).

Si tout est bon, faites un `Pull Request` sur le dépôt original.

N'envoyez pas votre pull request avant `16h30` le jour du test

Occupez-vous ailleurs en attendant, n'aidez pas les autres stagiaires.
Des formateurs pourront vous aider si vous avez des questions.

Structure de la DB ‘ti2web2025’

```sql
DROP TABLE IF EXISTS `guestbook`;
CREATE TABLE IF NOT EXISTS `guestbook` (
                    `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
                    `firstname` varchar(100) NOT NULL,
                    `lastname` varchar(100) NOT NULL,
                    `usermail` varchar(200) NOT NULL,
                    `phone` varchar(20) NOT NULL,
                    `postcode` varchar(4) NOT NULL,
                    `message` varchar(500) NOT NULL,
                    `datemessage` datetime NOT NULL DEFAULT
current_timestamp(),
                                            PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

Importez la BDD dans PHPMyAdmin : `data/ti2web2025.sql` en MariaDB ou MySQL

4 pages PHP :

Ne changez pas les noms de fichiers et fonctions déjà présentes dans les fichiers de consignes.

config.php :

contient les constantes de connexion à la Database “ti2web2025”, mais également deux constantes pour le bonus (PAGINATION_NB et PAGINATION_GET).
Ce fichier ne sera pas présent sur github car il se trouve dans le .gitignore

/public/index.php :

Contrôleur frontal, toutes les requêtes passent par lui.
Il charge les dépendances (config.php, le modèle `model/guestbookModel.php`)
La connexion PDO est ouverte et fermée dans ce fichier, cette connexion doit afficher les erreurs et mettre les valeurs récupérées par défaut en tableaux associatifs.
Il gère les conditions et charge les fonctions du modèle. Il récupère les entrées utilisateurs et les gère.
Il appelle les fonctions des modèles pour l'insertion et l'affichage de données de la DB
Bonus : il gère la pagination et la variable `Get` liée.
Il charge la vue `view/guestbookView.php`, où on affichera les données nécessaires.

/model/guestbookModel.php :

Contient les fonctions de **chargement** de `getAllGuestbook()`
et d’**insertion** `addGuestbook()`.

Vous devez garder le typage et les noms des paramètres, ainsi que le type de retour.
Vos requêtes doivent utiliser PDO et Exception en cas d'erreurs SQL.
En cas d'erreur SQL un `die()` arrête le script et nous affiche le message d'erreur.
Les requêtes préparées sont obligatoires en cas d'entrée utilisateur.

```php
function getAllGuestbook(PDO $db): array
{
    // try catch
        // requête SQL
        // traitement des données
        // bonne pratique
        // si la requête a réussi,
        // renvoyer le tableau de(s) message(s)
        return [];

     // erreur de requête SQL
       // créer un die avec l'erreur

}


/*
Le nombre de caractères doit être vérifié pour ne pas dépasser le chiffre à côté du champs ainsi que:
- pas de tags,
- encodé en html,
- pas d'espaces avant et après
- Pas de champs vides.
 */
function addGuestbook(PDO $db,
                      string $firstname, // <= 100
                      string $lastname, // <= 100
                      string $usermail, // <= 200 et doit être un mail valide
                      string $phone, // <= 20 et ne doit contenir que des numériques (0 à 9 autant de x que souhaité)
                      string $postcode, == 4 et ne doit contenir que des numériques (0 à 9 autant de x que souhaité)
                      string $message // <= 500
): bool
{
    // traitement des données backend (SÉCURITÉ)

    // si pas de données complètes ou ne correspondant pas à nos attentes, on renvoie false
    return false;
    // requête préparée obligatoire
```

```
-
-     // try catch
-         // si l'insertion a réussi
-         // on renvoie true
-     // sinon, on fait un die de l'erreur
-
-  }
-
-
```

Bonus : il contient également getNbTotalGuestbook(), getGuestbookPagination() et pagination()

Ces fonctions doivent être sécurisées et utiliser PDO pour fonctionner.

/view/guestbookView.php :

Affiche la page HTML, en lien avec les js, css et images du dossier public

Permet l'affichage du formulaire

Permet l'affichage de tous les posts (messages précédents) du livre d'or

Affiche le nombre de message(s) dans la DB suivant le modèle suivant

- 0 message : "pas encore de message"
- 1 message : "Il y a 1 message"
- + d'un message : "Il y a X message**s**"

Affiche les erreurs / messages si nécessaire.

Bonus :

- Affichage de la pagination fonctionnelle (elle est déjà présente dans ´/model/guestbookModel.php´)
- La date du message est formatée au format français :
  "Le (27/04/2025 à 10h29)"
- Le retour automatique à la ligne est activé dans les messages affichés.

Structure des fichiers à utiliser

```
.gitignore
README.md
config.php
config.php.ini
data/
	data/ti2web2025.sql
	data/fig1_vue_smartphone.png
	data/fig2_vue_tablette.png
	data/fig3_vue_desktop.png
	data/Grille d'évaluation TI2 2025.pdf
	data/TI2 2025.pdf
public/
	public/index.php
	public/js/
		public/js/validation.js
	public/css/
		public/css/style.css
	public/img/
		public/img/email.png
		public/img/favicon.png
		public/img/sign-up-amico.png

view/
	view/guestbookView.php

model/
	model/guestbookModel.php
```

Interface visuelle

Le livre d'or est affiché dans la vue **../view/guestbookView.php** via le contrôleur frontal **../public/index.php**

Dans la partie supérieure, on retrouve un formulaire qui permet au visiteur d'ajouter un nouveau post, avec son prénom, son nom, son e-mail, son numéro de téléphone, son code postal et son message.

Dans la partie inférieure, on retrouve la liste des messages postés précédemment par d'autres visiteurs.

Cette vue est donc constituée de plusieurs éléments :

- un titre principal ("Livre d'or")
- une image d'illustration
- un formulaire
- un titre secondaire ("Les messages précédents")
- Le nombre de message(s) dans la table (voir PHP)
- *Bonus pagination (voir PHP)*
- l'ensemble des messages déjà sauvegardés dans la base de données
- *Bonus pagination (voir PHP)*

Le formulaire est composé de :

- plusieurs **champs texte** (tous obligatoires) pour le prénom, le nom, le code postal et le numéro de téléphone portable du visiteur
- un **champ e-mail** pour l'adresse e-mail du visiteur
- une **zone de texte** pour le message du visiteur (**limité à 300 caractères**)
- un message sous cette zone de texte indique le **nombre de caractères introduits**
- un **bouton d'envoi**

Cette vue doit **adapter l'affichage** en fonction de la largeur d'écran (**voir les figures dans les pages suivantes**, comme exemples) :

- vue "smartphone"
- vue "tablette"
- vue "PC desktop"

Commun pour les trois vues

- Le choix de l'image d'illustration, des polices de caractères, des couleurs, les bordures et des ombrages est **laissé libre**.
- Idéalement, les champs du formulaire seront alignés pour une **belle présentation**.
- Lors de l'envoi du formulaire, un **message d'avertissement** apparaît sous le titre du formulaire, pour signaler si le message est enregistré ou non dans la base de données
- La mise en page doit être réalisée avec du **CSS natif** (Flexbox et/ou Grid), **mais pas avec Bootstrap !**

Vue “Smartphone” : écrans d’une largeur inférieure à 480 pixels

- le **titre principal** est **centré**
- une **image** d’illustration est **centrée** sous le titre principal
- un formulaire est centré sous l’image et les **labels sont au-dessus des champs**
- le titre secondaire est sous le formulaire
- Le nombre de message(s) dans la table (voir PHP)
- *Bonus pagination (voir PHP)*
- l’ensemble des messages sont sous le titre secondaire
- *Bonus pagination (voir PHP)*

(voir **figure 1** dans les pages suivantes)

Vue “Tablette” : écrans d’une largeur supérieure à 480 pixels et inférieure à 960 pixels

- le **titre principal** est **centré**
- une **image** d’illustration est **centrée** sous le titre principal
- un formulaire est centré sous l’image et les **labels sont à côté des champs**
- le titre secondaire est sous le formulaire
- Le nombre de message(s) dans la table (voir PHP)
- *Bonus pagination (voir PHP)*
- l’ensemble des messages sont sous le titre secondaire
- *Bonus pagination (voir PHP)*

(voir **figure 2** dans les pages suivantes)

Vue “PC desktop” : écrans d’une largeur supérieure à 960 pixels

- le **titre principal** est **centré**
- une **image** d’illustration est **à côté du formulaire** et ils sont sous le titre principal
- dans le formulaire, les **labels sont à côté des champs**
- le titre secondaire est sous le formulaire
- Le nombre de message(s) dans la table (voir PHP)
- *Bonus pagination (voir PHP)*
- l’ensemble des messages sont sous le titre secondaire
- *Bonus pagination (voir PHP)*

(voir **figure 3** dans les pages suivantes)

Figures (exemples)

Figure 1 : vue “smartphone”

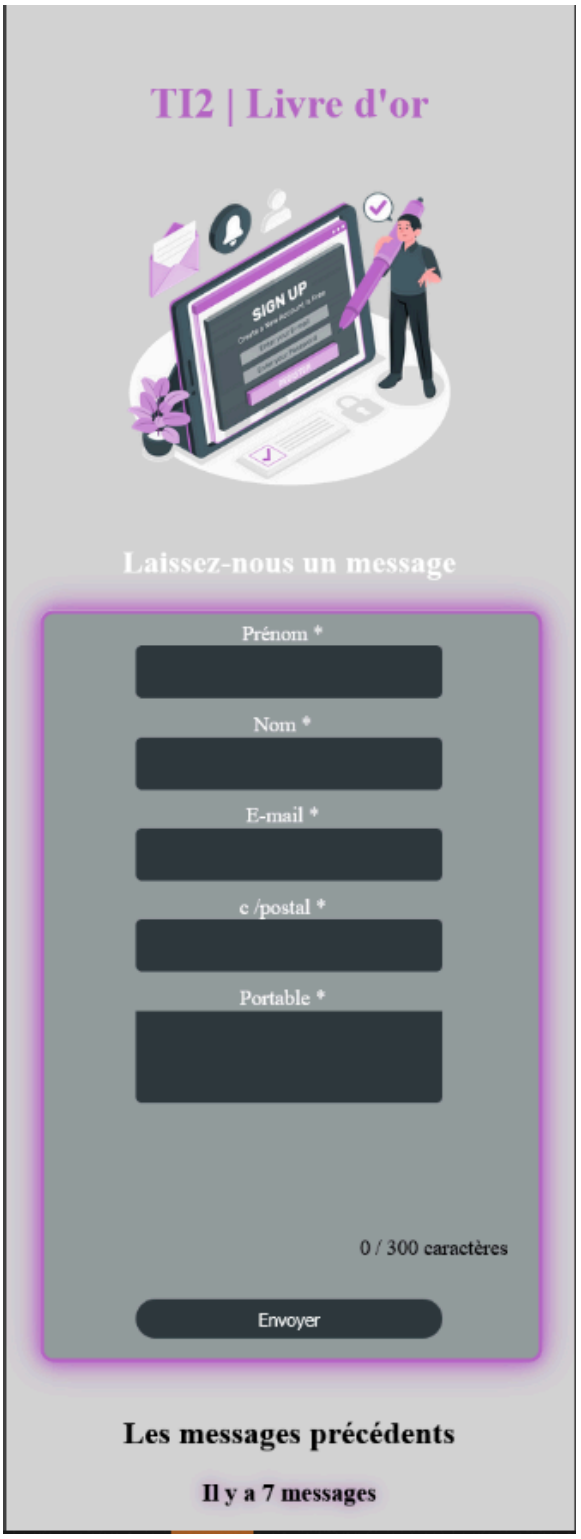

Figure 2 : Vue “Tablette”

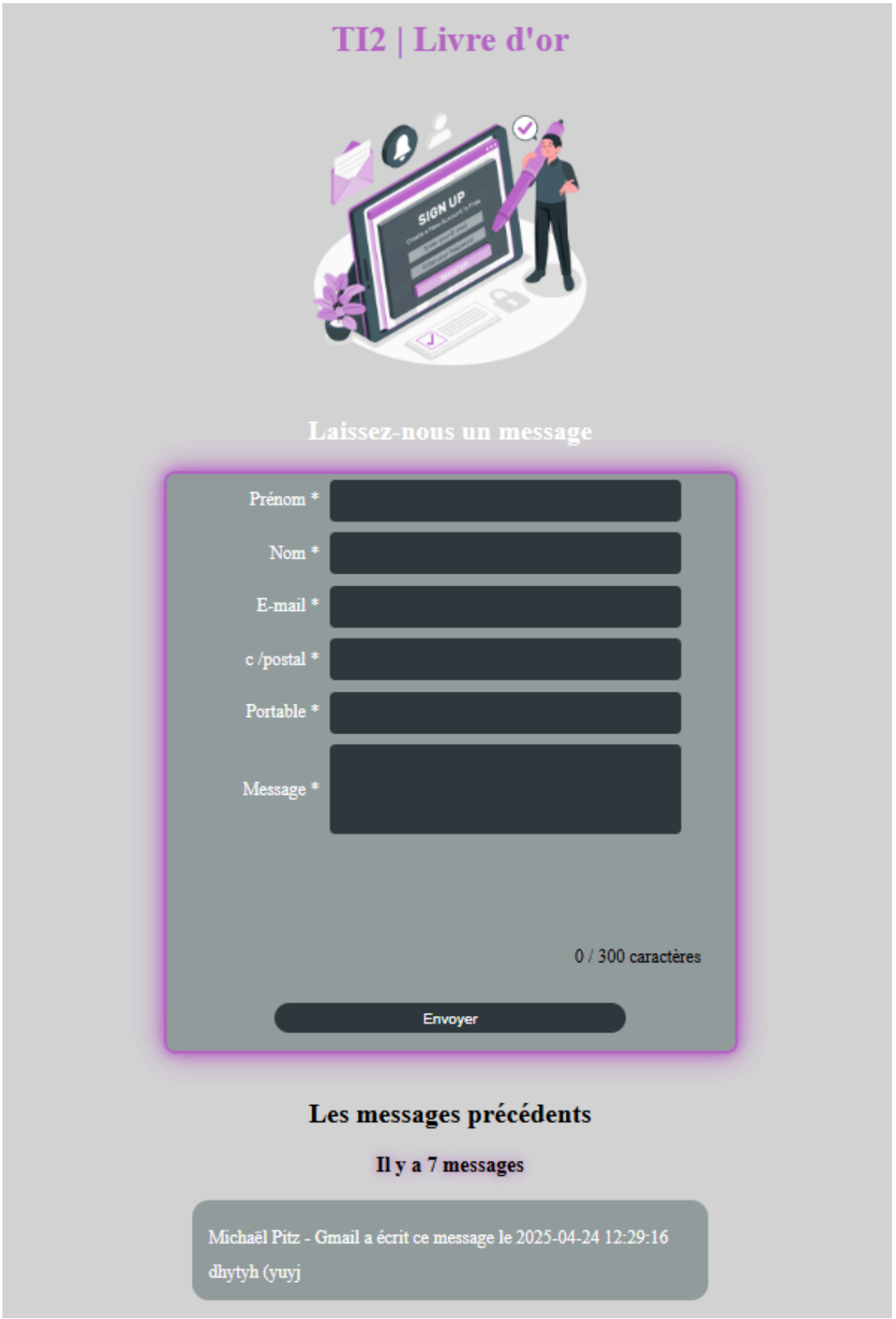

Figure 3 : vue “PC Desktop”

Javascript

Consignes :

1. Tous les champs du formulaire sont obligatoires
2. Sur le formulaire les **champs à vérifier en javascript sont** :

    - ☐ Le champ pour l'adresse **email**
    - ☐ Le champ pour le **code postal Belge**
    - ☐ Le champ pour le **numéro de téléphone Belge**
    - ☐ Le champ pour le **message**

3. À la soumission du formulaire :

    - Utilisez **JavaScript** pour valider les 3 champs.
    - Si un champ est invalide, **affichez un message d'erreur clair en rouge sous le titre**.
    - Si tous les champs sont valides, **affichez un message de succès en vert**.

4. Utilisez des **expressions régulières (regex)** pour faire les vérifications suivantes :

    - **Email** : doit avoir un format standard (ex : `prenom.nom@mail.com`)
    - **Code postal** : doit contenir **exactement 4 chiffres**
    - **Numéro de téléphone** : doit **commencer par 04** et contenir **10 chiffres au total** (ex : `0498150882`)

Conseil :

Vous pouvez utiliser la méthode JavaScript `regex.test(valeur)` pour tester vos regex.