|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | **Informaticien/-ne CFC**  Travail pratique individuel 2025 (TPI) |  | |
|  | |

Modèle de rapport v1.4

Candidat No 151464

Nom du candidat : Max Langenegger

Elevate – Projet TPI

### Sommaire

Elevate – Projet TPI 0

1 Résumé du rapport du TPI 2

1 Les grandes lignes du projet 3

1.1 Analyse de la situation initiale 3

1.2 Analyse de l’état désiré 3

1.3 Cahier des charges / exigences du système 3

1.4 Organisation du projet 3

2 Analyse préliminaire 4

1.1 Objectifs du système 4

2.1 Analyse de risque 4

1.2 Sécurité de l’information et protection des données 4

2 Concept 5

2.1 Exigences du système 5

2.2 Architecture du système 5

2.3 Plan d’intégration des systèmes 5

2.4 Concept d‘implémentation 5

2.5 Concept de formation 5

2.6 Concept de tests 5

2.7 Moyens nécessaires 5

3 Réalisation 6

3.1 Spécifications détaillées 6

3.2 Design du système 6

3.3 Configuration xyz 6

3 Test 7

3.1 Procédure de test 7

3.2 Protocol de test 7

3.3 Signature du protocole de test 7

3.4 PWA mode hors-ligne 7

3.5 Vérification connexion réseau PWA 9

3.6 Vérification connexion réseau PWA 10

4 Tests Techno 11

4.1 Test PWA 11

3.1 PWA mode hors-ligne 13

5 Conclusion 14

5.1 Améliorations possibles 14

5.2 Auto-évaluation 14

4 Bibliographie: liste des sources et références 15

5 Glossaire 16

6 Signatures 17

7 Annexes 18

# Résumé du rapport du TPI

Le Résumé du rapport du TPI est une présentation conceptuelle du travail effectué et du résultat attendu d’au maximum une page A4 qui permettent au lecteur une rapide compréhension du rapport de travail. . Il contient trois paragraphes : Situation de départ, mise en œuvre, résultats.

Pour plus de détails, veuillez vous référez au Manuel ICT - partie B : Documentation / rapport du TPI - Question 14 - page 81.

Dans de nombreuses courses comme la Millets Cup, qui se déroule sur les hauteurs de Lessoc, la gestion des résultats repose encore sur un système manuel. Chaque poste (sommet, slalom, touch, cible) est géré par un commissaire, qui enregistre les performances des concurrents sur papier avant de les transmettre à l’organisation. Ce processus est long et sujet aux erreurs : mauvaise retranscription des résultats, oublis, retards dans la mise à jour du classement, et difficulté à rectifier les erreurs.

Mon projet Elevate vise à moderniser et automatiser cette gestion des résultats. Grâce à une saisie numérique en temps réel, chaque commissaire pourra enregistrer directement les performances des concurrents via une interface simple et intuitive, réduisant ainsi les risques d’erreurs et les délais de traitement. Les résultats permettent d’afficher un classement en temps réel accessible à tous. De plus, un mode offline garantit la continuité de la saisie même en l’absence de connexion, avec une synchronisation une fois le réseau rétabli.

Elevate est un nouveau projet que je dois construire de A à Z, brique par brique pour finaliser mon TIP. Le seul fichier actuel lié à ce TPI est le cahier des charges que j’ai reçu.

# Les grandes lignes du projet

## Cahier des charges / exigences du système

Ce cahier des charges décrit de manière précise et mesurable les objectifs à atteindre pour la réalisation de l’application **Elevate**, destinée à la gestion des compétitions de marche et vol. Le système doit permettre la saisie, la consultation et la synchronisation des performances des concurrents par les commissaires, via une application mobile Progressive Web App (PWA) sécurisée, rapide et intuitive.

**A. Objectifs généraux**

* Développer une **application mobile de type PWA** fonctionnant sur tous les smartphones modernes, sans installation via un store.
* Offrir une **expérience utilisateur fluide**, même dans des zones avec une mauvaise couverture réseau.
* Garantir la **sécurité des données**, notamment par le hachage des mots de passe et la protection contre les injections SQL.
* Permettre une **synchronisation automatique ou manuelle des données** saisies hors ligne dès que la connexion est rétablie.
* Mettre à disposition un **classement consultable en temps réel**, sans authentification.

**B. Exigences fonctionnelles**

|  |  |  |
| --- | --- | --- |
| ID | Exigence fonctionnelle | Mesure de succès |
| EF1 | Affichage en temps réel du classement général | Classement trié automatiquement par points décroissants |
| EF2 | Détail complet des performances d’un concurrent | Affichage des types de postes, horaires, points |
| EF3 | Authentification des commissaires | Connexion via e-mail + mot de passe ou code PIN |
| EF4 | Mémorisation du login précédent pour simplifier la reconnexion | Champ prérempli après une première connexion |
| EF5 | Affichage des postes affectés à chaque commissaire | Liste filtrée automatiquement selon l’identifiant utilisateur |
| EF6 | Saisie d’un résultat avec formulaire interactif | Enregistrement réussi avec réinitialisation du formulaire |
| EF7 | Enregistrement de malus avec remarque obligatoire | Malus ajouté au concurrent, consultable dans ses résultats |
| EF8 | Fonctionnement offline avec synchronisation des résultats en attente | Données stockées localement et envoyées une fois connecter |
| EF9 | Indicateur graphique en cas de résultats non synchronisés | Icône ou message visible jusqu’à synchronisation complète |
| EF10 | Affichage d’un message d’erreur si le dossard est invalide | Notification claire sans plantage du système |

**C. Exigences techniques**

|  |  |  |
| --- | --- | --- |
| ID | Exigence technique | Mesure de succès |
| ET1 | Application développée en HTML, CSS, JS (client) et PHP/MySQL (serveur) | Fonctionnalités réparties selon architecture client/serveur |
| ET2 | Backend développé en PHP sous forme d’API REST sécurisée | Appels AJAX fonctionnels avec retours JSON |
| ET3 | Mot de passe des commissaires haché avec password\_hash() | Aucun mot de passe en clair dans la base |
| ET4 | Protection contre les injections SQL via requêtes préparées PDO | Aucune concaténation de chaînes dans les requêtes SQL |
| ET5 | Fichiers manifest.json et sw.js présents pour la gestion PWA | Fonctionnement hors ligne et possibilité d’installation |
| ET6 | Base de données MySQL avec structure claire (résultats, utilisateurs, postes, malus, etc.) | Schéma relationnel complet respecté |
| ET7 | Affichage responsive adapté aux mobiles | Navigation fluide sur petit écran, boutons accessibles |

**D. Exclusions du projet**

Conformément au mandat, certaines fonctionnalités sont **hors périmètre** :

* La gestion (création, modification, suppression) des **participants**.
* La gestion des **comptes utilisateurs** des commissaires.
* La gestion dynamique des **postes et affectations**.  
  Ces données seront saisies **manuellement en base de données avant la course**.

## Organisation du projet

Pour l’organisation de mon projet, j’ai choisi d’utiliser un planning en mode cascade qui se trouvera dans un fichier Excel disponible dans les annexes. Une feuille *Journal de bord* sera aussi présente dans ce classeur Excel qui servira à lister les heures passés avec la description des tâches effectuées.

Plusieurs sauvegardes du projet sont faîtes quotidiennement, une se trouve en local dans les fichiers de mon téléphone, une version est disponible sur mon [github](https://github.com/langeneggerm/TPI_Elevate) et une autre version se trouve sur une clé USB personnel. Ces trois supports permettent d’avoir plusieurs issues de secours si mon environnement de travail qui est OneDrive lâche.

Citer la méthode de gestion de projet utilisée (Méthode en phases, agile, scrum, …)

Etablir la liste des participants au projet avec leur rôle respectif.

Décrire comment sont gérées les sauvegardes de la documentation et du code (fréquence, supports utilisés, utilisation de logiciel de gestion de version, …)

# Analyse

## Objectifs du système

### Analyse de l’état actuel

Actuellement, la gestion des compétitions de marche et vol repose sur un système manuel et papier, comme c’est le cas lors d’événements tels que la Millets Cup. Chaque commissaire de course est chargé d’enregistrer les résultats des concurrents à la main sur des fiches ou formulaires. Ces documents sont ensuite transmis physiquement à l’organisation centrale pour traitement.

Ce mode de fonctionnement présente plusieurs **limitations majeures** :

* **Risque d’erreurs humaines** : les erreurs de saisie, les oublis ou les mauvaises retranscriptions sont fréquents et difficiles à corriger.
* **Temps de traitement long** : le processus de collecte, transmission et encodage des résultats est lent, ce qui retarde la mise à jour du classement.
* **Aucune visualisation en temps réel** : les participants, spectateurs ou autres commissaires n'ont pas accès à un classement à jour pendant la course.
* **Traçabilité limitée** : en cas de litige ou d’erreur, il est difficile de savoir qui a saisi quoi et à quel moment.
* **Dépendance au support papier** : les documents peuvent être perdus, abîmés ou mal interprétés, surtout dans un environnement extérieur et parfois difficile d’accès.

Ces limitations montrent que le système actuel n’est **ni fiable**, **ni efficace**, et qu’il ne répond plus aux besoins d’une compétition moderne. Il nuit à la réactivité des organisateurs et à l’expérience globale des participants.  
C’est sur la base de ce constat que le projet **Elevate** a été imaginé, avec comme but de **numériser** et **automatiser** l’ensemble du processus, tout en tenant compte des contraintes spécifiques liées au terrain (problèmes de réseau, conditions météo, simplicité d’utilisation).

### Analyse de l’état désiré

L’application désirée à la fin de mon mandat sera une application PWA avec plusieurs vues pour les différents besoins utilisable avec connexion ou sans connexion au réseau.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Dans un premier temps, quand nous accèderont au site, nous verront le classement actuel basé sur les informations présentes dans la base de données.

On pourra y voir notamment les diverses informations concernant les concurrents des courses comme leurs prénoms et noms de famille, leurs numéros de dossard, leurs nombre de points ainsi que lieu d’habitation.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.Quand on va cliquer sur un concurrent on va pouvoir avoir accès à ses l’entièreté des résultats que le concurrent à effectuer tout au long des courses

### Objectifs

Voici la liste détaillée des objectifs mesurables à atteindre dans le cadre du projet **Elevate**, tels que définis dans le cahier des charges :

1. **Objectifs fonctionnels**
2. **Afficher le classement en temps réel**
   * Classement trié par total de points décroissants
   * Affichage de la photo, dossard, nom, prénom, code postal, localité et score
   * Mise à jour dynamique sans rechargement de la page
3. **Afficher les résultats détaillés d’un concurrent**
   * Liste de tous les résultats enregistrés (type de poste, nom, date, heure, points)
4. **Afficher les malus d’un concurrent**
   * Remarque visible et liée à la bonne personne avec heure d’enregistrement
5. **Authentifier les commissaires de course**
   * Connexion par email + code PIN
   * Mémoire du login précédent pour simplifier les connexions suivantes
   * Système de session sécurisé
6. **Afficher les postes assignés à chaque commissaire**
   * Filtrage automatique selon l’utilisateur connecté
   * Visualisation simple des postes disponibles
7. **Permettre la saisie d’un résultat par un commissaire**
   * Formulaire de saisie avec dossard, heure, remarque
   * Saisie rapide et efficace, avec formulaire réinitialisé après enregistrement
   * Message d’erreur si le dossard est invalide
8. **Permettre la saisie d’un malus**
   * Champ de remarque obligatoire
   * Enregistrement lié au bon concurrent et au bon poste
9. **Gérer les saisies hors connexion (mode offline)**
   * Stockage local des résultats et malus si pas de réseau
   * Synchronisation automatique ou manuelle dès que le réseau est disponible
   * Indicateur graphique des résultats en attente
10. **Sécuriser les échanges de données**
    * Utilisation de requêtes préparées (protection contre les injections SQL)
    * Hachage des mots de passe avec password\_hash et vérification via password\_verify
11. **Objectifs techniques**
12. **Développer une PWA (Progressive Web App)**
    * Compatible avec tous les smartphones
    * Fonctionne sans installation
    * Inclut manifest.json et service worker (sw.js)
13. **Créer une API backend sécurisée en PHP**
    * Interface REST sans interface graphique
    * Fonctionne avec une base de données MySQL
    * Hébergement sur serveur local (WAMP) puis sur AlpHosting
14. **Garantir l’accessibilité et la rapidité de l’interface**
    * Interface simple et épurée
    * Grands boutons, navigation rapide, sans scrolling inutile
    * Chargements asynchrones pour optimiser la fluidité

## Choix de variante

Le premier choix que j’ai dû faire concerne le système de connexion que les commissaires vont utiliser pour se connecter à leurs comptes pour qu’ils puissent entrer des résultats dans la base de données.

Le dilemme était de choisir entre un mot de passe ou un code PIN. J’ai choisi le mode de connexion avec le code PIN, qui sera plus utile à entrer lors de la connexion pour chaque personne sur le terrain grâce à sa rapidité d’exécution.

## Sécurité de l’information et protection des données

Les données présentes dans la base de données seront protégées de trois façons distinctes :

### Injections SQL

Pour ce projet, j’ai fait attention à la sécurité de ma base de données, notamment pour éviter les injections SQL. C’est un type d’attaque où quelqu’un essaie d’envoyer du code malveillant dans une requête SQL, par exemple via un formulaire ou une URL, pour accéder ou modifier des données sans autorisation.

Pour me protéger contre ça, j’utilise des requêtes préparées avec PDO. Ça permet de séparer le code SQL des données envoyées par l’utilisateur. Du coup, même si quelqu’un essaie d’injecter du code SQL dans un champ, ça ne fonctionnera pas.

Voici un exemple avec une des fonctions que j’ai utilisées :

            $queryPrepared = *$this*->pdo->prepare($query);

            $queryPrepared->execute($params);

Grâce à ça, les paramètres sont automatiquement sécurisés, et il n’y a pas de concaténation de texte dans les requêtes, ce qui évite les failles.

J’ai aussi mis des blocs try/catch pour gérer les erreurs proprement, sans afficher des messages trop techniques à l’utilisateur. En cas de problème, je peux aussi faire un rollback pour annuler une transaction, ce qui aide à garder la base de données dans un état cohérent.

Attaques XSS

Dans mon projet, j’ai aussi pris en compte les attaques XSS (Cross-Site Scripting). C’est un type d’attaque où un utilisateur malveillant essaie d’injecter du code JavaScript ou HTML dans une page web, souvent via un formulaire ou une URL. Le but, c’est que ce code soit exécuté dans le navigateur des autres utilisateurs, ce qui peut par exemple voler des informations ou modifier le contenu de la page.

Pour éviter ça, j’ai créé une fonction sanitizeInput qui nettoie toutes les données envoyées par les utilisateurs avant de les afficher. Elle fait deux choses importantes :

1. Elle enlève les espaces inutiles au début et à la fin du texte avec trim.
2. Elle transforme tous les caractères spéciaux (comme <, >, " ou ') en entités HTML avec htmlspecialchars.

Du coup, même si quelqu’un essaie d’envoyer un script comme ***<script>alert('XSS') </script>***, il sera transformé en texte inoffensif :  
***&lt;script&gt; alert(&#039;XSS&#039;)&lt;/script&gt;***

Voici la fonction utilisée :

function sanitizeInput($input)

{

    return htmlspecialchars(trim($input), ENT\_QUOTES, 'UTF-8');

}

Hachage de mot de passe et gestion de session

Pour sécuriser l’authentification dans mon projet, j’ai mis en place deux éléments essentiels : le **hachage des mots de passe** et la **gestion des sessions utilisateur**.

Quand un utilisateur se connecte, je ne compare jamais directement le mot de passe en clair. À la place, tous les mots de passe sont hachés lors de l’enregistrement (grâce à password\_hash) et je les vérifie ensuite avec password\_verify.  
Ça permet de ne jamais stocker de mots de passe lisible dans la base de données. Même si quelqu’un accède à la base, il ne pourra pas voir les vrais mots de passe.

Dans ma fonction de connexion (login), je fais par exemple :

        if (password\_verify($password, $user['PIN'])) {

            //connexion réussi

        }

Une fois le mot de passe vérifié, j’utilise les **sessions PHP** pour garder l’utilisateur connecté de manière sécurisée. Je stocke par exemple son email et son ID dans $\_SESSION, ce qui me permet de l’identifier sur les autres pages sans avoir à redemander ses infos à chaque fois.

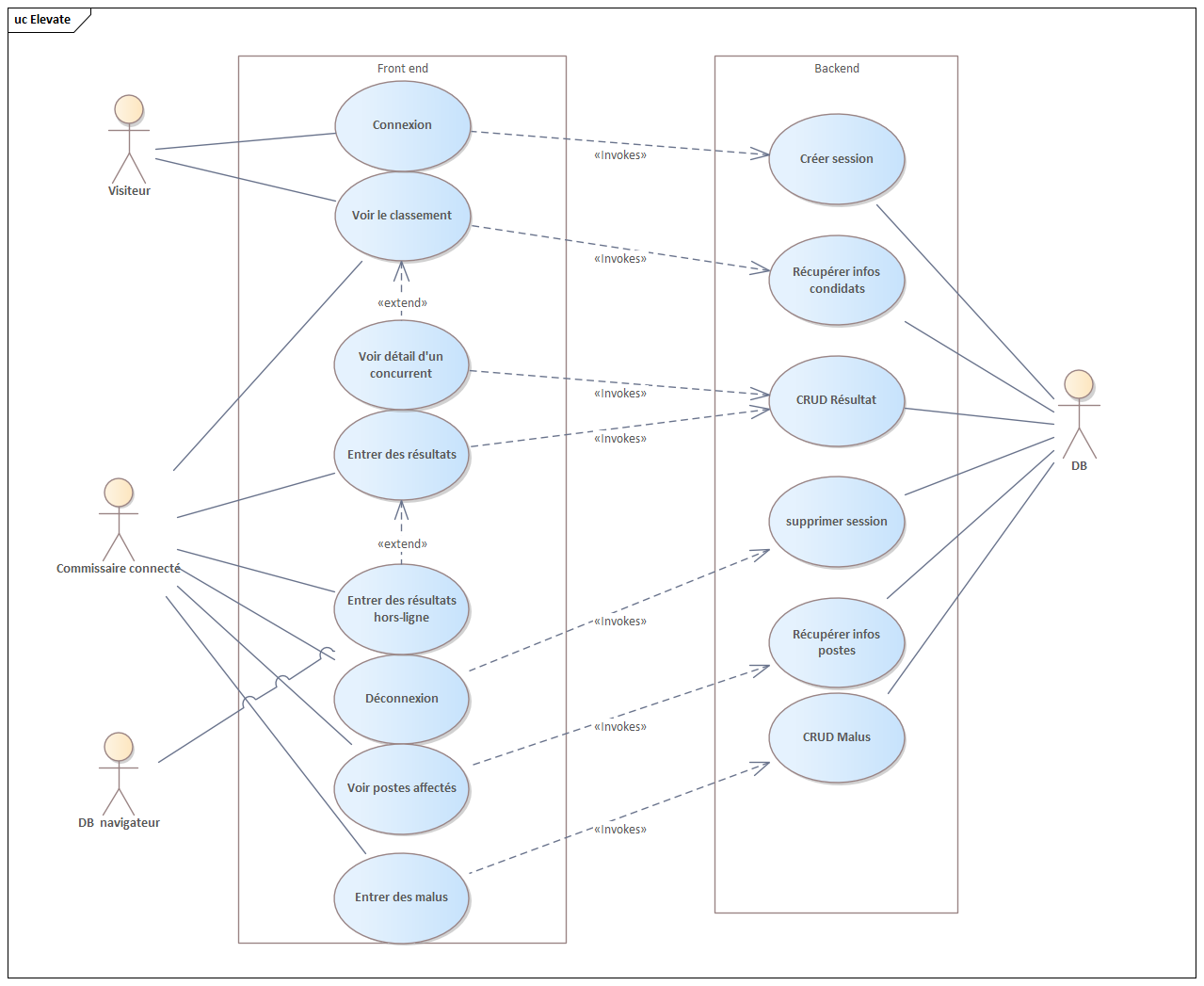
J’ai aussi créé un fichier sessionController.php qui centralise la gestion des sessions (comme la création, la vérification de connexion, et la déconnexion). Ça me permet de garder un code plus propre et plus facile à maintenir.

Grâce à cette approche, je protège à la fois les mots de passe des utilisateurs et leur session de navigation. C’est une étape importante pour éviter les vols de données ou les accès non autorisés.

                $\_SESSION['email'] = $email;

                $\_SESSION['id'] = $user['id'];

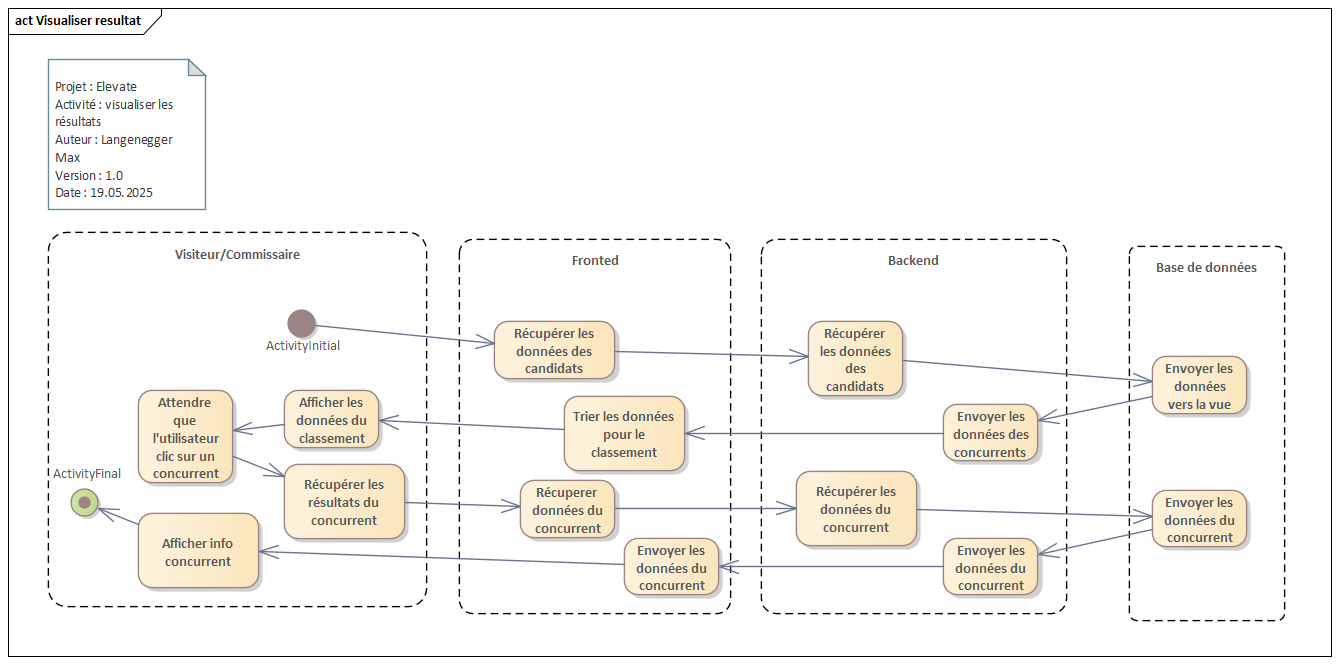
## Diagramme des cas d’utilisation

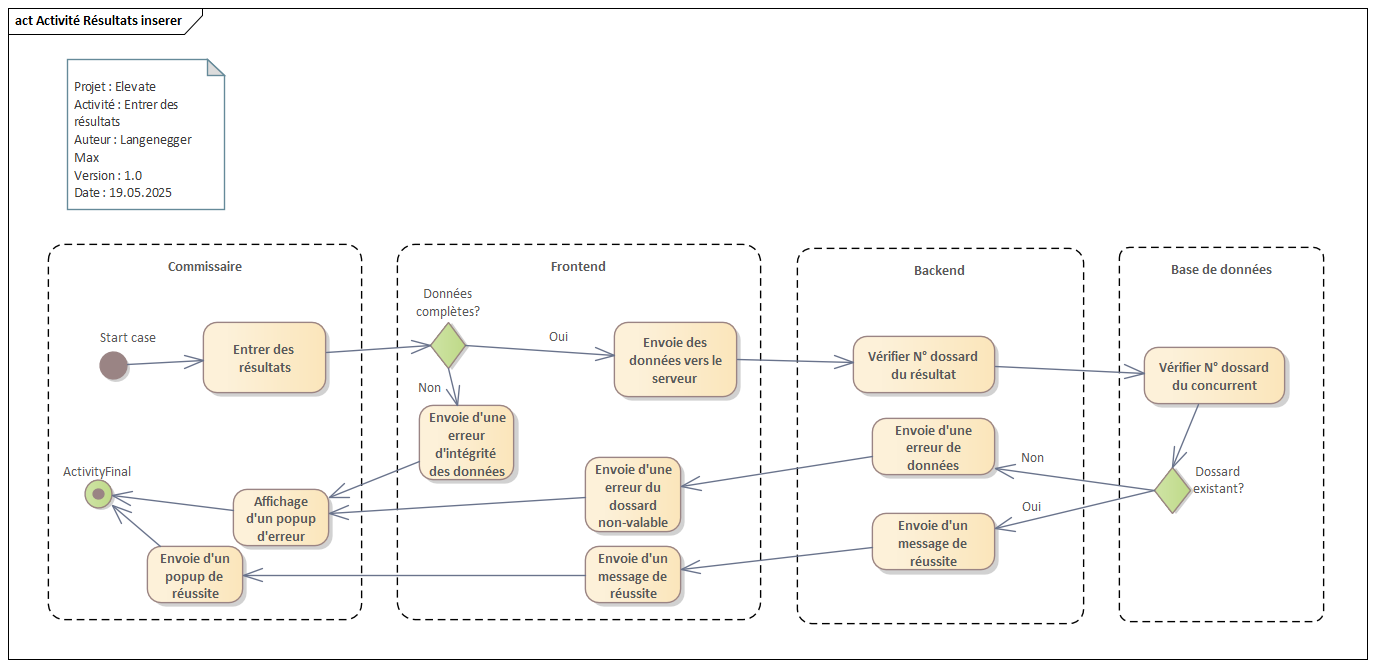
Les différents acteurs de mon application, côté utilisateur, seront soit des visiteurs ou soit des commissaires qui se sont connecter au préalable. Côté serveur, celui-ci englobe la base de données est les fichiers PHP qui constitueront le serveur en entier.

Les visiteurs pourront consulter le classement actuel avec diverses informations comme le nom et prénom des concurrents, leurs numéros de dossard, leurs nombres de points ainsi que leur lieu d’habitation. Quand un utilisateur va cliquer sur un concurrent, il pourra alors avoir accès à des informations plus détaillé sur le concurrent et ces résultats. Il y aura une liste avec l’intégralité des résultats du concurrent choisi ainsi que d’autre informations sur celui-ci.

Pour qu’un visiteur passe d’un simple utilisateur à un commissaire, il devra se connecter à l’application à l’aide d’un code PIN pour vérifier son identité. Il aura alors accès à d’autre fonctionnalité de l’application comme voir les postes auquel il a été affecté ou encore entrer des résultats des différents concurrents. Le commissaire pourra toujours consulter le classement actuel. Si l’envie le prend, il peut sans autre se déconnecter pour redevenir un visiteur.

## Diagramme d’activité

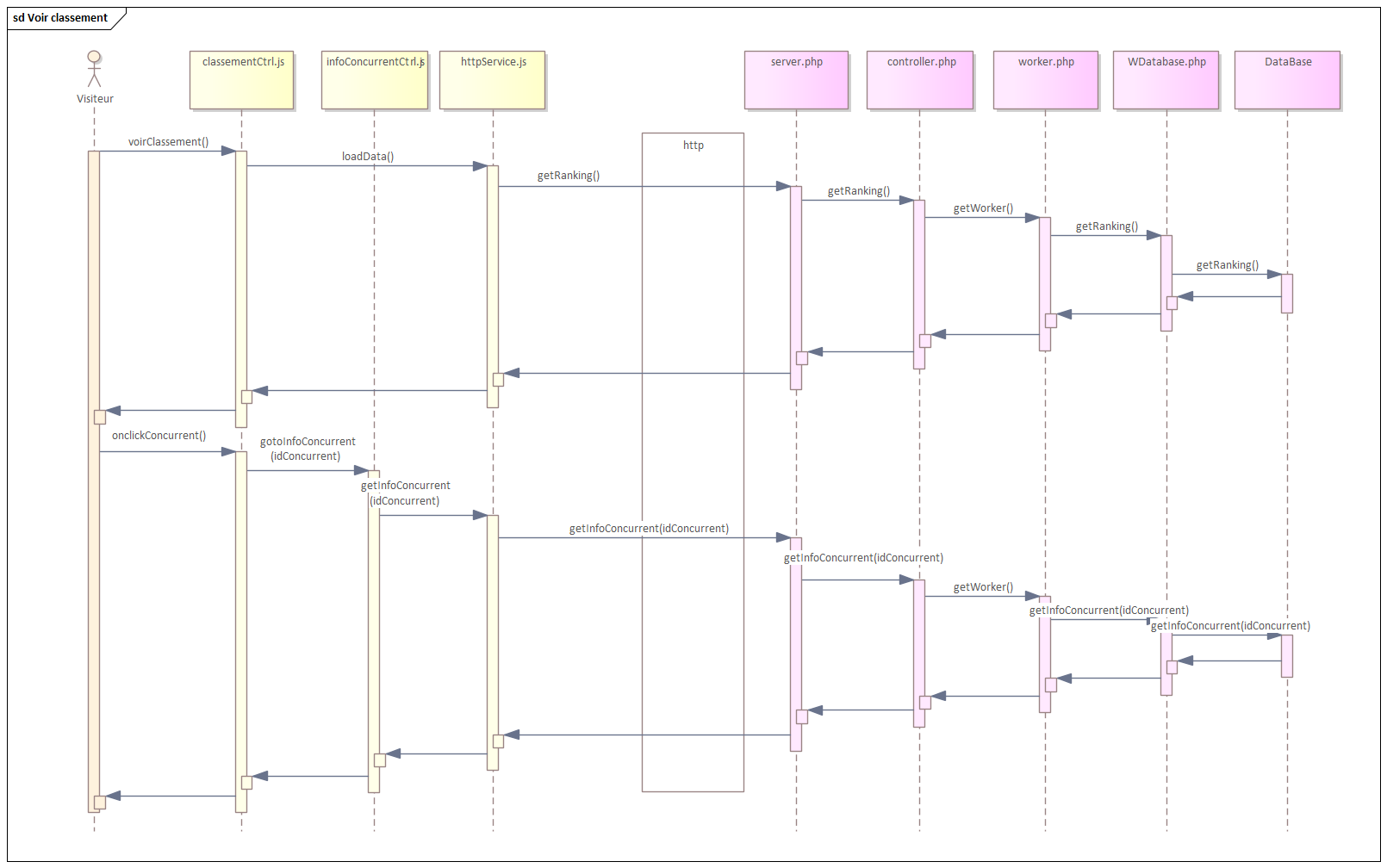
Cette activité se fait lors du chargement de la page principale dédié au classement, récupérer et afficher les résultats des différents concurrents pour établir le classement qui sera visible sur l’application. Dans un premier temps lors du chargement de la page, le visiteur va app

Cette activité

# Conception

## Diagramme de classe

## Diagramme de séquence

Cette séquence représente le cheminement de l’action pour récupérer le classement depuis le frontend qui commence par classementCtrl.js

## Exigences du système

## Architecture du système

Pour ce projet, j’ai organisé mon application en deux grandes parties : le client et le serveur. L’idée était de bien séparer ce qui est visible par l’utilisateur (le front-end) de ce qui gère les données et la logique métier (le back-end).

La partie client contient tout ce qui s’affiche dans le navigateur :

* views/ : Ce dossier contient les différentes pages HTML de l’application, comme la page de connexion (login.html), la page du classement, ou encore les pages de saisie pour les commissaires.
* js/controller/ : Tous les fichiers JavaScript qui contrôlent les différentes pages (par exemple loginCtrl.js gère la logique de la page de connexion).
* js/worker/ : Ici, on retrouve par exemple httpService.js qui gère les requêtes vers le serveur.
* sw.js et manifest.json : Ces fichiers permettent d’ajouter des fonctionnalités PWA (comme l’installation sur le bureau ou le cache offline).

La partie server s’occupe de toute la logique côté back-end :

* controllers/ : On y retrouve le contrôleur principal et le sessionController.php qui gère les connexions, la création et la destruction des sessions utilisateur.
* workers/ : C’est là que sont gérés tous les traitements plus techniques.
  + database/ : Contient la classe de connexion à la base de données (WDatabaseConnection.php) et les fonctions de requêtes (WDatabase.php).
  + worker.php : Fait le lien entre les contrôleurs et la base de données.
* exceptions/ : Pour gérer proprement les erreurs (ex : DatabaseException).
* config.php : Contient les informations de configuration (comme les identifiants de la base).
* server.php : Le point d’entrée de l’API, qui route les requêtes vers le bon contrôleur.

Cette organisation me permet d’avoir un code clair, bien structuré et facilement maintenable. Elle facilite aussi la réutilisation des fonctions, la sécurisation du code, et la gestion des différentes responsabilités dans le projet.

## Plan d’intégration des systèmes

## Concept d‘implémentation

## Concept de formation

## Concept de tests

|  |  |
| --- | --- |
| Fonctionnalité testée | Résultat attendu |
| Afficher le classement | Le classement s'affiche en ordre décroissant de points avec les infos des concurrents. |
| Afficher les résultats et les malus d’un concurrent | Les résultats et les malus liés au dossard sélectionné s’affichent correctement. |
| Rafraîchir le classement | Le classement se met à jour avec les dernières données sans recharger toute la page. |
| Se rendre à la page de login | L’utilisateur est redirigé vers l’écran de connexion. |
| Se connecter en tant que commissaire | L’utilisateur est authentifié et redirigé vers la sélection de postes. |
| Afficher les postes selon le commissaire connecté | Seuls les postes affectés au commissaire connecté sont visibles. |
| Entrer un résultat pour un concurrent | Le résultat est enregistré, et une confirmation s’affiche. Le formulaire est réinitialisé. |
| Entrer un malus pour un concurrent | Le malus est enregistré avec la remarque et l’heure, puis visible dans les détails. |

## Moyens nécessaires

Conseil: D’autres chapitres peuvent s’ajouter à cette partie en fonction du projet.

# Réalisation

## Spécifications détaillées

## Design du système

# Test

## Procédure de test

## Protocol de test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Objet testé | Description du test | Attente | Résultat | Visa |
| 1.0 |  |  |  |  |  |

## Signature du protocole de test

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
|  |  |  |

## PWA mode hors-ligne

### Procédure de test

*index.html*

### Protocol de test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Objet testé | Description du test | Attente | Résultat | Visa |
| 1.0 |  |  |  |  |  |

### Signature du protocole de test

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
|  |  |  |
|  |  |  |

## Vérification connexion réseau PWA

### Procédure de test

Tests technologiques pour l'enregistrement d'une image dans une base de données MySQL et la récupération d'une image depuis une base de données afin de l'afficher dans une page HTML.

### Protocol de test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Objet testé | Description du test | Attente | Résultat | Visa |
| 1.0 |  |  |  |  |  |

Protocole de test détaillé avec Visa.

### Signature du protocole de test

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
|  |  |  |

## Vérification connexion réseau PWA

### Procédure de test

### Protocol de test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | Objet testé | Description du test | Attente | Résultat | Visa |
| 1.0 |  |  |  |  |  |

Protocole de test détaillé avec Visa.

### Signature du protocole de test

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
|  |  |  |

# Tests Techno

## Test PWA

Tests technologiques pour la mise en place d'une application PWA de base. Ce test permettra de créer un site avec un manifest permettant l’installation du site en tant qu’application PWA basique sans aucune autre fonctionnalité.  
  
J’ai commencé par créer un simple fichier html qui possède un lien vers un manifest ainsi qu’un texte *Hello World PWA*.

<html>

<head>

    <link *rel*="manifest" *href*="manifest.json" />

</head>

<body>

Hello World PWA

</body>

</html>

J’ai aussi créé le manifest permettant l’installation de l’application avec différents paramètres tels que le nom de l’application, l’url de l’application et les icones qui seront affiché sur l’application

{

    "name": "Test PWA",

    "short\_name": "PWA",

    "description": "Ce site est un site de test pour une PWA",

    "start\_url": "/testTechno/PWASimple",

    "display": "standalone",

    "background\_color": "#ffffff",

    "theme\_color": "#a70d0d",

    "icons": [

        {

            "src": "bob\_192x192.png",

            "sizes": "192x192",

            "type": "image/png"

        },

        {

            "src": "bob\_512x512.png",

            "sizes": "512x512",

            "type": "image/png"

        }

    ]

}

Lors de l‘accès au [site](https://elevate.emf-infopro-tpi.ch/testTechno/PWASimple/), celui-ci nous demande bien si l’on veut installer le site en tant qu’application.

Une image contenant texte, capture d’écran, Police, logo

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, Système d’exploitation, Page web

Le contenu généré par l’IA peut être incorrect.

L’application est fonctionnelle et téléchargeable sur navigateur ET aussi téléphone mobile.

Une image contenant texte, capture d’écran, dessin humoristique, graphisme

Le contenu généré par l’IA peut être incorrect.

## PWA mode hors-ligne

Tests technologiques pour faire fonctionner une application PWA en mode hors ligne (cache des fichiers). Il faut ajouter un serviceWorker qui va gérer les fichiers que le navigateur va mettre en cache. Cela permet de rendre l’application disponible même hors réseau.

Pour permettre l’ajout et l’utilisation d’un serviceWorker, j’ai d’abord créé un lien entre le fichier html et le fichier sw.js. J’ai ajouté un bout de JavaScript dans le html qui va vérifier que le navigateur supporte les serviceWorker et enregistre sw.js si le navigateur est correct.

    <script *type*="text/javascript" *src*="sw.js"></script>

    <script>

        if ('serviceWorker' in navigator) {

      window.addEventListener('load', () => {

        navigator.serviceWorker.register('/testTechno/PWAHorsLigne/sw.js')

          .then(*registration* => {

            console.log('Service Worker enregistré avec succès:', *registration*);

          })

          .catch(*error* => {

            console.log('Erreur lors de l’enregistrement du Service Worker:', *error*);

          });

      });

    }

    </script>

*sw.js*

const CACHE\_NAME = 'v1';

const FILES\_TO\_CACHE = [

  '/favicon.ico',

  '/testTechno/PWAHorsLigne/',

  '/testTechno/PWAHorsLigne/index.html',

  '/testTechno/PWAHorsLigne/sw.js',

  '/testTechno/PWAHorsLigne/logo/bob\_192x192.png',

  '/testTechno/PWAHorsLigne/logo/bob\_512x512.png',

  '/testTechno/PWAHorsLigne/parapente.jpg',

  '/testTechno/PWAHorsLigne/manifest.json'

];

self.addEventListener('install', *event* => {

*event*.waitUntil(

    caches.open(CACHE\_NAME)

      .then(*cache* => {

        console.log('Mise en cache des fichiers');

        return *cache*.addAll(FILES\_TO\_CACHE);

      })

  );

});

self.addEventListener('activate', *event* => {

*event*.waitUntil(

    caches.keys().then(*keys* =>

*Promise*.all(

*keys*.filter(*key* => *key* !== CACHE\_NAME)

            .map(*key* => caches.delete(*key*))

      )

    )

  );

});

self.addEventListener('fetch', *event* => {

*event*.respondWith(

    caches.match(*event*.request)

      .then(*response* => *response* || fetch(*event*.request))

  );

});

Maintenant l’application est disponible et visible même lorsque la connexion est interrompue.

Dans un premier temps sur navigateur, la page affiche que nous sommes hors connexion.

Une image contenant dessin humoristique, smiley, sourire, émoticône

Le contenu généré par l’IA peut être incorrect.

Puis la page et les fichiers présent dans le cache s’affiche correctement et tous les fichiers sont bien présents dans le cache du navigateur :

Une image contenant capture d’écran, texte, Logiciel multimédia, Logiciel de graphisme

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, nombre, Police, ligne

Le contenu généré par l’IA peut être incorrect.

## PWA statuts réseau

Tests technologiques pour établir si une connexion réseau est active ou non dans une application PWA. Analyser les moyens possibles pour savoir si une connexion est perdue ou rétablie.

Pour effectuer ce test j’ai ajouté un fichier *main.js* qui permet de vérifier l’état actuel du réseau pour ensuite afficher un message différent en fonction de celui-ci

$(document).ready(function () {

    document.getElementById('status').innerHTML = navigator.onLine ? 'online' : 'offline';

    var target = document.getElementById('target');

});

function handleStateChange() {

    var timeBadge = **new** *Date*().toTimeString().split(' ')[0];

    var newState = document.createElement('p');

    var state = navigator.onLine ? 'online' : 'offline';

    newState.innerHTML = '' + timeBadge + ' Connection state changed to ' + state + '.';

    console.log(newState.innerHTML);

    target.appendChild(newState);

}

window.addEventListener('online', handleStateChange);

window.addEventListener('offline', handleStateChange);

Une image contenant texte, capture d’écran, oiseau

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran

Le contenu généré par l’IA peut être incorrect.

## Envoi et réception d’image

Tests technologiques pour l'enregistrement d'une image dans une base de données MySQL et la récupération d'une image depuis une base de données afin de l'afficher dans une page HTML.

Pour effectuer ce test j’ai utilisé deux fichiers php, un qui va s’occuper d’envoyer les images et d’afficher du HTML et un autre qui va récupérer les images dans la base de données.

*Upload\_base64.php*

<?php

$host = 'localhost';

$dbname = 'db\_tt';

$username = 'root';

$password = '';

try {

    $pdo = new *PDO*("mysql:host=$host;dbname=$dbname", $username, $password);

} catch (*PDOException* $e) {

    die("Erreur de connexion : " . $e->getMessage());

}

if (isset($\_FILES['image'])) {

    $imageData = file\_get\_contents($\_FILES['image']['tmp\_name']);

    $base64Image = base64\_encode($imageData);

    $stmt = $pdo->prepare("INSERT INTO t\_img (img) VALUES (?)");

    $stmt->bindParam(1, $base64Image);

    $stmt->execute();

    echo "Image encodée en base64 enregistrée avec succès dans `t\_img` !";

}

?>

<!DOCTYPE *html*>

<html>

<head>

    <meta *charset*="UTF-8">

    <title>Envoyer une image (base64)</title>

</head>

<body>

    <h1>Uploader une image en base64</h1>

    <form *action*="upload\_base64.php" *method*="post" *enctype*="multipart/form-data">

        <input *type*="file" *name*="image" *required*>

        <input *type*="submit" *value*="Envoyer">

    </form>

    <p><a *href*="view\_base64.php">Voir les images enregistrées</a></p>

</body>

</html>

Le fichier upload va d’abord créer une connexion à la base de données et ensuite vérifier si il y a bien une image a envoyer pour ensuite l’encoder en base64 et l’ajouter à la base de données.  
Les deux fichiers php vont envoyer et recevoir les images sur la base de données stockée en local grâce à WAMP

*View\_base64.php*

<?php

$host = 'localhost';

$dbname = 'db\_tt';

$username = 'root';

$password = '';

try {

    $pdo = new *PDO*("mysql:host=$host;dbname=$dbname", $username, $password);

} catch (*PDOException* $e) {

    die("Erreur de connexion : " . $e->getMessage());

}

$stmt = $pdo->query("SELECT img FROM t\_img");

$images = $stmt->fetchAll(*PDO*::FETCH\_ASSOC);

?>

<!DOCTYPE *html*>

<html>

<head>

    <meta *charset*="UTF-8">

    <title>Images enregistrées (base64)</title>

</head>

<body>

    <h1>Images dans la base (base64)</h1>

    <?php foreach ($images as $img): ?>

        <div *style*="margin-bottom:20px;">

            <img *src*="data:image/jpeg;base64,<?= htmlspecialchars($img['img']) ?>" *style*="max-width:300px;">

        </div>

    <?php endforeach; ?>

</body>

</html>

View va s’occuper de récupérer les images et de les afficher sur la page actuelle.

Une image contenant texte, capture d’écran, Police, ligne

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, dessin humoristique, art

Le contenu généré par l’IA peut être incorrect.

## Requêtes en attentes et base de données locale

Tests technologiques sur les moyens d'enregistrer des données (les résultats en attente) en local et de gérer ces données : cache, base de données locale, fichier, autre.

*index.html*

<!DOCTYPE *html*>

<html *lang*="fr">

<head>

  <meta *charset*="UTF-8">

  <title>PWA Images</title>

  <link *rel*="manifest" *href*="manifest.json" />

  <script *type*="text/javascript" *src*="sw.js"></script>

  <script *src*="js/db.js"></script>

  <script *src*="js/main.js"></script>

  <script>

    if ('serviceWorker' in navigator) {

      window.addEventListener('load', () => {

        navigator.serviceWorker.register('/testTechno/PWARequeteDonneesCache/sw.js')

          .then(*registration* => {

            console.log('Service Worker enregistré avec succès:', *registration*);

          })

          .catch(*error* => {

            console.log('Erreur lors de l’enregistrement du Service Worker:', *error*);

          });

      });

    }

  </script>

</head>

<body>

  <h1>PWA - Gestion des images</h1>

  <div *id*="etatConnexion" *style*="padding: 10px; font-weight: bold;"></div>

  <input *type*="file" *id*="imageFile" *accept*="image/\*">

  <button *id*="uploadBtn">Envoyer l’image</button>

  <button *id*="loadBtn">Charger les images</button>

  <div *id*="resultat"></div>

</body>

</html>

Le fichier html contient les divers liens vers les fichiers JavaScript ainsi qu’un input, deux boutons et un div qui va accueillir le résultat des fichiers php.

*main.js*

  document.addEventListener('DOMContentLoaded', () => {

async function envoyerImage() {

  const fileInput = document.getElementById('imageFile');

  const file = fileInput.files[0];

  if (!file) return;

  const formData = **new** *FormData*();

  formData.append('image', file);

  if (navigator.onLine) {

    try {

      const res = await fetch('upload\_base64.php', {

        method: 'POST',

        body: formData

      });

      if (res.ok) {

        alert(await res.text());

      } else {

        console.error('Réponse non OK, stockage local.');

        await saveImageOffline(file);

      }

    } catch (err) {

      console.error('Erreur réseau, stockage local.');

      await saveImageOffline(file);

    }

  } else {

    console.log('Hors ligne, image stockée localement.');

    alert("Les images seront envoyé lorsque vous serez connecté à internet");

    await saveImageOffline(file);

  }

}

  async function chargerImages() {

    if (navigator.onLine) {

      console.log("chargement en ligne")

      fetch('view\_base64.php')

        .then(*res* => *res*.json())

        .then(*data* => {

          const container = document.getElementById('resultat');

          container.innerHTML = '';

*data*.forEach(*img* => {

            const image = document.createElement('img');

            image.src = `data:image/jpeg;base64,${*img*}`;

            image.style.width = '200px';

            image.style.margin = '10px';

            container.appendChild(image);

          });

        });

    } else {

      try {

        console.log("chargement hors ligne")

        const images = await getAllImagesOffline();

        const container = document.getElementById('resultat');

        container.innerHTML = '';

        images.forEach(*img* => {

          const image = document.createElement('img');

          const reader = **new** *FileReader*();

          reader.onloadend = function() {

            image.src = reader.result;

            image.style.width = '200px';

            image.style.margin = '10px';

            container.appendChild(image);

          };

          reader.readAsDataURL(*img*);

        });

      } catch (err) {

        console.log('Erreur lors du chargement des images locales:', err);

      }

    }

  }

  // Boutons JS

  document.getElementById('uploadBtn').addEventListener('click', envoyerImage);

  document.getElementById('loadBtn').addEventListener('click', chargerImages);

  function mettreAJourEtatConnexion() {

const etat = document.getElementById('etatConnexion');

if (navigator.onLine) {

  etat.textContent = '🟢 Connecté à Internet';

  etat.style.color = 'green';

} else {

  etat.textContent = '🔴 Hors ligne';

  etat.style.color = 'red';

}

}

mettreAJourEtatConnexion();

window.addEventListener('online', () => {

mettreAJourEtatConnexion();

console.log('Connexion retrouvée. Envoi des images locales.');

syncImages();

});

window.addEventListener('offline', () => {

mettreAJourEtatConnexion();

console.log('Perte de connexion');

});

});

Ce fichier va principalement vérifier l’état du réseau et sélectionner les différentes méthodes à utiliser en fonction de l’état de celui-ci. Ce que l’on veut avec ce test c’est stocker les requêtes que l’on fait lorsque l’on est sans connexion pour ensuite les envoyer quand on récupère

# Conclusion

Conclusion sur le projet en général. Les objectifs sont-ils atteints ? Si non, pourquoi ?

## Améliorations possibles

Y a-t-il des améliorations possibles (nouvelles fonctionnalités, amélioration et/ou modification du design, modification du support de données…) ?

## Auto-évaluation

Quels sont mes sentiments vis-à-vis du travail effectué. Enoncé des facteurs qui expliquent le succès ou l’échec de la réalisation des objectifs du projet.

# Bibliographie: liste des sources et références

Conseil: Cette partie peut contenir une liste des sources et éléments littéraires utilisés.

# Glossaire trier par ordre alphabetique

|  |  |
| --- | --- |
| Terme | Signification |
|  |  |
|  |  |
|  |  |

# Signatures

Je soussigné déclare que les informations contenues dans ce rapport de travail pratique individuel rendu ce jour le XX.XX.20XX dans le cadre de la procédure de qualification de mon CFC d’informaticien/-ne, ne sont pas plagiées. Toutes les informations de sources extérieures ainsi que les informations fournies par des tiers durant le déroulement du travail sont consignées.

L’apprenti/-e doit signer la documentation avant la remise afin de témoigner de la justesse des informations qui y figurent.

|  |  |  |
| --- | --- | --- |
| Date | Nom | Signature |
|  |  |  |

# Annexes

Code imprimé, protocole de séances, etc.: spécifier complètement et annexer à la documentation.

Listings de scripts et programmes. Le travail fourni par la candidate ou le candidat doit être entièrement documenté. Omettre le code généré automatiquement s’il n’est pas nécessaire à la compréhension.

Dans le cas où des manuels ont été créés, vous pouvez les ajouter ici en tant qu’annexe.