# Rapport SAE23

Seddiq Farsi | Selim Haddad | Léo Coquerel Le 01/05/2025

[Rapport SAE23 0](#_Toc199702528)

[Introduction 2](#_Toc199702529)

[Partie 1 : Solution de Gestion des Absences 3](#_Toc199702530)

[Objectifs et Fonctionnalités 3](#_Toc199702531)

[Technologies Utilisées et Mise en Œuvre 3](#_Toc199702532)

[Interfaces Clés 4](#_Toc199702533)

[Partie 2 : Slide Puzzle 5](#_Toc199702534)

[Objectifs et Fonctionnalités 5](#_Toc199702535)

[Technologies Utilisées et Mise en Œuvre 5](#_Toc199702536)

[Aperçu du Jeu 6](#_Toc199702537)

[Partie 3 : Le Défi de Maître Yoda 7](#_Toc199702538)

[Objectifs et Fonctionnalités 7](#_Toc199702539)

[Technologies Utilisées et Mise en Œuvre 7](#_Toc199702540)

[Résultat 7](#_Toc199702541)

[Partie 4 : TP JavaScript : Gestion d’Évènements et Interactions Utilisateur 8](#_Toc199702542)

[Objectifs et Fonctionnalités 8](#_Toc199702543)

[Technologies Utilisées et Mise en Œuvre 8](#_Toc199702544)

[Exemples d'Interfaces 9](#_Toc199702545)

[Conclusion 10](#_Toc199702546)

## Introduction

Le présent rapport détaille la réalisation de plusieurs mini-projets web dynamiques dans le cadre de la SAE23, menés par notre équipe. L'objectif principal était de concevoir et de développer des solutions interactives et fonctionnelles, en mettant en œuvre les connaissances acquises en HTML, CSS, JavaScript et PHP, ainsi qu'en gestion de bases de données (MySQLi).

Notre approche a consisté à travailler de manière collaborative, en utilisant des outils comme GitHub pour la gestion des versions et le partage du code, assurant ainsi une synchronisation efficace malgré des horaires de travail parfois différents. Chaque projet a été abordé avec une progression en difficulté, permettant d'explorer divers aspects de la programmation web dynamique et de renforcer notre compréhension des interactions utilisateur.

Nous avons veillé à séparer au mieux les différentes parties du code source pour une meilleure maintenabilité et lisibilité. Ce rapport fournit une vue d'ensemble de chaque projet, incluant les concepts techniques abordés et les fonctionnalités implémentées.

## Partie 1 : Solution de Gestion des Absences

Cette première partie du projet a consisté à développer un système complet de gestion des absences pour les étudiants, incluant des fonctionnalités de connexion, une feuille d'appel interactive et un mode de consultation des absences.

### Objectifs et Fonctionnalités

* **Authentification des Enseignants :** Un système de login sécurisé permettant aux enseignants d'accéder à la feuille d'appel.
* **Feuille d'Appel :** Une interface interactive pour les enseignants afin de marquer la présence, l'absence justifiée (ABJ) ou l'absence injustifiée (ABI) des étudiants pour un module et une heure de cours donnés, avec la prise en compte des cours de 3h.
* **Mode Consultation :** Une fonctionnalité pour consulter les absences par étudiant (nom, prénom, numéro d'étudiant) sur une période définie, offrant la possibilité de modifier les statuts d'absence.

### Technologies Utilisées et Mise en Œuvre

* **Frontend (HTML/CSS) :**
  + Les fichiers index.php, appel.php, et consultation.html définissent la structure des pages de connexion, de la feuille d'appel et du formulaire de consultation, respectivement.
  + Le fichier style.css est utilisé pour styliser l'ensemble des interfaces, incluant les formulaires, les tableaux et les éléments de navigation, afin d'assurer une présentation cohérente et intuitive.
* **Backend (PHP) :**
  + index.php gère l'authentification des enseignants en vérifiant les identifiants soumis ($\_POST['user'], $\_POST['mdp']) par rapport à la base de données. En cas de succès, l'enseignant est redirigé vers appel.php avec son idEnseignant.
  + appel.php permet l'enregistrement des absences. Il récupère la liste des étudiants depuis la base de données et génère dynamiquement des champs radio pour chaque étudiant. Il gère également l'heure du cours et l'option "cours de 3h" pour insérer une ou deux entrées d'absence dans la base de données via des requêtes INSERT.
  + consultation.php traite les requêtes de consultation. Il filtre les absences en fonction des critères de recherche (nom, prénom, ID étudiant, période) soumis via consultation.html et affiche les résultats dans un tableau. Il permet également de modifier les absences via des listes déroulantes.
  + update.php est un script PHP qui reçoit les données de modification d'absence via une requête POST et met à jour les enregistrements correspondants dans la base de données via des requêtes UPDATE.
* **Base de Données (MySQLi) :**
  + Le fichier SAE23.sql définit la structure de la base de données SAE23, incluant les tables Absences, Enseignant, et Etudiant avec leurs relations et clés primaires.
  + Les scripts PHP utilisent l'extension mysqli pour établir la connexion à la base de données (new mysqli('localhost','root',"",'SAE23')) et exécuter les requêtes SQL ($bdd->query($sql)).

### Interfaces Clés

**Écran de Connexion (Login)**

L'interface de connexion permet aux enseignants de s'authentifier.

**Feuille d'Appel**

Après connexion, l'enseignant accède à la feuille d'appel où il peut sélectionner un module et marquer les absences.

**Mode Consultation**

Le mode consultation offre un aperçu des absences et permet des modifications.

## Partie 2 : Slide Puzzle

Cette partie du projet a consisté à développer un jeu de puzzle coulissant interactif en utilisant les technologies web fondamentales.

### Objectifs et Fonctionnalités

* **Jeu de Puzzle 4x4 :** Implémentation d'un puzzle classique où une image est découpée en 15 pièces mélangées, avec un espace vide pour permettre le mouvement des pièces.
* **Interaction Utilisateur :** Les joueurs peuvent déplacer les pièces adjacentes à l'espace vide par un simple clic, dans le but de reconstituer l'image originale.
* **Compteur de Coups :** Affichage en temps réel du nombre de mouvements effectués par le joueur, ajoutant un élément de défi.
* **Image Personnalisée :** Utilisation d'une image de l'IUT Rouen (image.jpg) pour le puzzle, rendant le jeu plus pertinent pour le contexte du projet.

### Technologies Utilisées et Mise en Œuvre

* **HTML (Puzzle.html) :**
  + La structure du jeu est définie par un tableau (<table>) de 4x4 cellules.
  + Chaque cellule contient un bouton (<button>) qui représente une pièce du puzzle.
  + L'attribut onclick sur chaque bouton est utilisé pour appeler la fonction JavaScript move() avec l'index de la pièce.
* **CSS (Puzzle.css) :**
  + Le fichier Puzzle.css est responsable de la stylisation de l'ensemble du jeu.
  + Il définit la taille fixe des boutons (100x100px) et utilise les propriétés background-image, background-size, et background-position pour afficher la partie correspondante de l'image image.jpg sur chaque bouton.
  + La classe .empty est appliquée au bouton représentant l'espace vide, le rendant transparent et visuellement distinct.
  + Des transitions CSS sont utilisées pour un effet visuel fluide lors du survol des boutons.
* **JavaScript (Intégré dans Puzzle.html) :**
  + Un tableau v est utilisé pour stocker l'ordre actuel des pièces (numéros 1 à 15 et une chaîne vide pour l'espace).
  + La fonction u() (update) parcourt le tableau v et met à jour le style de chaque bouton (backgroundImage, backgroundPosition) pour refléter l'état actuel des pièces. Elle met également à jour le compteur de coups (c).
  + La fonction move(i) est le cœur de la logique du jeu :
    - Elle prend l'index du bouton cliqué.
    - Elle vérifie les indices des pièces adjacentes (haut, bas, gauche, droite).
    - Si une pièce adjacente est l'espace vide (v[j]===""), elle échange la position de la pièce cliquée avec l'espace vide ([v[i],v[j]]=[v[j],v[i]]).
    - Le compteur de coups (c) est incrémenté, et la fonction u() est appelée pour rafraîchir l'affichage.
  + Au chargement, le tableau v est mélangé aléatoirement (v.sort(()=>Math.random()-.5)) pour initialiser le puzzle.

### Aperçu du Jeu

Le jeu se présente sous la forme d'une grille où chaque bouton représente une partie de l'image.

## Partie 3 : Le Défi de Maître Yoda

Cette section du projet a abordé un défi algorithmique inspiré du "Défi de Maître Yoda", visant à construire la plus longue chaîne de mots selon une règle spécifique.

### Objectifs et Fonctionnalités

* **Chaîne de Mots :** Déterminer la plus longue séquence de mots à partir d'une liste donnée, où chaque mot de la séquence doit commencer par les trois dernières lettres du mot précédent.
* **Algorithme de Recherche :** Implémentation d'une logique pour parcourir l'ensemble des mots et identifier la chaîne la plus longue possible en respectant la règle.

### Technologies Utilisées et Mise en Œuvre

* **PHP (tri.php) :**
  + Le script PHP contient une liste prédéfinie de mots ($mots).
  + Il construit un **graphe** ($graphe) où chaque mot est une clé, et sa valeur est un tableau des mots qui peuvent le suivre (ceux dont les trois premières lettres correspondent aux trois dernières lettres du mot actuel). Cette structure permet de modéliser les connexions entre les mots.
  + L'algorithme parcourt chaque mot de la liste comme un point de départ potentiel pour une chaîne.
  + Pour chaque départ, il construit une chaîne ($chemin) en sélectionnant de manière gloutonne le prochain mot valide non encore visité dans la chaîne actuelle.
  + Il maintient un tableau $visites pour éviter les boucles infinies et s'assurer que chaque mot n'est utilisé qu'une seule fois par chemin.
  + Après avoir exploré un chemin complet à partir d'un mot de départ, il compare la longueur de ce $chemin avec la $meilleure chaîne trouvée jusqu'à présent et la met à jour si la nouvelle est plus longue.
  + Finalement, le script utilise echo pour générer directement du HTML affichant la $meilleure chaîne trouvée, jointe par l'opérateur ➔.
* **HTML/CSS (tri.css) :**
  + Le fichier tri.php génère une page HTML simple qui affiche le titre du défi et le résultat de l'algorithme.
  + Le fichier tri.css est utilisé pour styliser cette page, définissant la police, les couleurs, les marges et l'apparence des titres et des paragraphes pour une lecture agréable du résultat.

### Résultat

Le script PHP génère et affiche la suite de mots la plus longue trouvée.

## Partie 4 : TP JavaScript : Gestion d’Évènements et Interactions Utilisateur

Cette partie a été un Travaux Pratiques approfondi sur la gestion des événements et les interactions utilisateur en JavaScript, divisée en plusieurs mini-applications interactives.

### Objectifs et Fonctionnalités

* **Bases de la Gestion d'Événements :** Maîtriser l'attachement d'événements via l'attribut HTML onclick, la propriété DOM Level 0 onproperty, et la méthode recommandée addEventListener().
* **Manipulation de l'Objet Event :** Utiliser les propriétés clés (event.target, event.clientX, event.clientY, event.key) et les méthodes (event.preventDefault(), event.stopPropagation()) de l'objet Event pour contrôler le comportement des interactions.
* **Interactions Avancées :** Prévenir le comportement par défaut des formulaires et des liens, et implémenter la délégation d'événements pour gérer les interactions sur des éléments ajoutés dynamiquement.
* **Applications Interactives :**
  + **Compteur :** Un compteur numérique simple avec des boutons d'incrémentation, décrémentation et réinitialisation.
  + **Formulaire de Validation :** Un formulaire avec une validation en temps réel de la longueur du nom de l'utilisateur.
  + **Liste de Tâches (ToDo List) :** Une application complète pour ajouter, marquer comme terminé et supprimer des tâches, avec persistance des données.
  + **Changement de Thème :** Une fonctionnalité pour basculer entre un thème clair et sombre de l'interface, avec sauvegarde du choix de l'utilisateur.
  + **Glisser-Déposer (Drag and Drop) :** Un élément interactif qui peut être déplacé librement par l'utilisateur dans une zone définie.

### Technologies Utilisées et Mise en Œuvre

* **HTML :**
  + Les fichiers HTML fournissent la structure de base pour chaque mini-application (boutons, paragraphes, champs de texte, listes, div déplaçable).
  + Des id sont utilisés pour identifier de manière unique les éléments à manipuler avec JavaScript.
* **CSS :**
  + Les fichiers CSS définissent le style visuel de toutes les interfaces.
  + Ils gèrent la mise en page (flexbox), l'apparence des boutons et des champs de texte, et les styles spécifiques pour les zones interactives (ex: #zoneSurvol, #zoneDessin).
  + TP\_Selim.css utilise des variables CSS (:root) pour faciliter le changement de thème (clair/sombre) et applique des styles améliorés pour le compteur et les messages.
  + Des classes comme .champ-erreur sont utilisées pour indiquer visuellement les erreurs de validation.
* **JavaScript :**
  + **Gestion des Événements :**
    - JSLeo.js démontre les différentes méthodes d'attachement d'événements (onclick, addEventListener) et l'utilisation de l'objet Event pour les événements de souris (mouseover, mouseout, mousemove) et de clavier (keydown, input).
    - Il montre également event.preventDefault() pour les formulaires (monFormulaire) et les liens (monLien).
    - La délégation d'événements est illustrée sur la listeDynamique pour gérer les clics sur des <li> ajoutés dynamiquement.
  + **Applications Interactives (JSSelim.js) :**
    - **Compteur :** La variable c est manipulée par les écouteurs de clic sur les boutons moins, plus, reset. La fonction majC() met à jour le textContent de l'élément #compteur.
    - **Formulaire :** La fonction valNom() valide la longueur du champ nomIn. L'événement submit du formulaire utilise e.preventDefault() et affiche des messages de succès ou d'erreur.
    - **Tâches :** Les fonctions loadT() et saveT() interagissent avec localStorage (localStorage.getItem(), localStorage.setItem(), JSON.parse(), JSON.stringify()) pour la persistance des tâches. creerT() crée dynamiquement les éléments <li> avec leurs boutons d'action. La délégation d'événements est implicitement utilisée car les écouteurs sont attachés aux boutons créés dynamiquement.
    - **Thème :** La fonction applyTh() lit le thème depuis localStorage au chargement. Le bouton toggle-theme bascule la classe theme-sombre sur le <body> et met à jour localStorage.
    - **Glisser-Déposer :** Les événements mousedown, mousemove, et mouseup sont attachés à l'élément #element-deplacable. Le code calcule le déplacement de la souris (deltaX, deltaY) et met à jour les propriétés style.left et style.top de l'élément, en s'assurant qu'il reste dans les limites de son conteneur parent.

### Exemples d'Interfaces

**Compteur et Formulaire**

Le compteur et le formulaire illustrent les interactions de base.

**Liste de Tâches et Thème**

La liste de tâches et le sélecteur de thème montrent des interactions plus avancées et la persistance des données.

## Conclusion

Ce projet de mini-projets web dynamique a été une expérience d'apprentissage enrichissante. Nous avons pu consolider nos compétences en développement web frontend et backend, en abordant des aspects variés allant de la gestion de bases de données à l'interactivité avancée en JavaScript.

Les défis rencontrés, notamment les problèmes de mise en page réactive et l'adaptation au format mobile (non directement visible dans les captures, mais un aspect important du développement web), nous ont permis d'approfondir nos connaissances et de trouver des solutions robustes. La collaboration au sein de l'équipe a été essentielle pour surmonter ces obstacles, renforçant nos capacités à communiquer et à travailler efficacement ensemble.

Nous sommes fiers des réalisations de ces mini-projets, qui démontrent notre capacité à concevoir et à implémenter des applications web fonctionnelles et interactives, répondant aux exigences du cahier des charges.