2025-2026

# CONCEPTION SITE WEB


NADAL NGAKI MUPATI

UNIVERSITE D'EVRY PARIS-SACLAY

2025-2026

# Document de conception d'un site web

Répondre au mieux aux questions et items ci-dessous. Le support de cours n°2 (Web design) peut aider à comprendre certaines questions.

## Analyse des besoins

- Attentes, contexte d'utilisation, objectifs du site :
  Mon site est un portfolio qui a pour objectif de présenter mon profil d'ingénieur de manière professionnel et structuré. Il sera utilisé pour les recruteurs, des entreprises et des responsables de stages ou d'alternance.
  Le site permet de mettre en valeur mes compétences, projets, expériences et motivations. Il vise à améliorer ma visibilité et à faciliter l'accès à mon profil……
  ……Existant (web ou autres moyens) :
  Il existe déjà des plateformes professionnelles comme LinkedIn ou GitHub permettant de présenter un profil et des projets.
  Cependant, ces outils restent limités en personnalisation et en organisation du contenu.
  Le site portfolio personnel permet de centraliser toutes les informations professionnelles.
  Il offre une présentation plus claire, personnalisée et adaptée à mon profil d'ingénieur……

### Définir les utilisateurs potentiels du site

- Tranche d'âge, profession, connaissances de l'internaute cible :
  Les utilisateurs ciblés sont principalement des recruteurs, ingénieurs, responsables RH et encadrants de stages, âgés de 25 à 55 ans.
  Ils ont de bonnes connaissances du web et consultent régulièrement des profils professionnels en ligne.
  Ils recherchent des informations claires, rapides et structurées……
- Matériel (plateforme, navigateur, bande passante) adéquat :
  Le site sera consulté sur ordinateur, tablette ou smartphone, via des navigateurs courants (Chrome, Firefox, Edge).
  Une connexion internet standard est suffisante, le site étant léger et optimisé……

En rouge, ci-dessous, il s'agit de questions à poser à des utilisateurs « tests » : il faut donc faire tester votre site à une ou plusieurs personnes différentes et à recueillir leurs impressions.

- L'objectif du site est-il clair ?
  Oui, le site présente clairement mon profil d'ingénieur, mes compétences et mes projets……
- L'audience du site peut-elle clairement s'identifier ?
  Oui, les recruteurs et responsables de stages peuvent facilement se reconnaître comme public cible……
- Le site est-il utile et pertinent pour ce public ?
  Oui, il fournit des informations essentielles pour évaluer un candidat et ses réalisations……

- Le site est-il intéressant ?
  Oui, le design et la présentation des projets rendent le site attrayant et professionnel.....
- Le site permet-il aux visiteurs de réaliser toutes les tâches qu'ils veulent accomplir ?
  Oui, les visiteurs peuvent consulter mon CV, mes projets et mes coordonnées facilement.
  .....
- Les visiteurs peuvent-ils accomplir facilement ces tâches ?
  Oui, la navigation est simple et intuitive, les informations sont bien organisées
  .....
- Le contenu et l'organisation des informations sont-ils cohérents avec l'objectif du site ?
  Oui, tout le contenu est structuré pour valoriser mes compétences et projets
- L'information importante est-elle facile à trouver ?
  Oui, le CV, les projets et les coordonnées sont accessibles rapidement depuis la page d'accueil.
  .....
  .....
- Toutes les informations sont-elles claires, faciles à comprendre et à lire ?
  Oui, les textes sont concis, les titres clairs et le formatage lisible.
  .....
- Le visiteur sait-t-il toujours où il est et comment faire pour aller où il veut ?
  Oui, le menu et les liens de navigation indiquent clairement la position sur le site.
  .....
  .....
- Le graphisme est-il agréable ?
  Oui, le design est moderne, sobre et professionnel, adapté à un portfolio.
  .....
  .....
- Les pages se chargent-elles suffisamment vite ?
  Oui, le site est léger et optimisé pour un chargement rapide sur tous les appareils.
  .....
  .....

## Architecture de communication

- Arborescence des pages de mon site :
  - Nombre : 5 .....
  - Structure (possibilité de mettre un schéma) :
    Présentation (accueil) → compétences → Education → Projets → Contact
    La structure est simple et linéaire, accessible depuis un menu de navigation fixe.....
  - Objectifs de communication :
    Présenter clairement mon profil d'ingénieur.
    Mettre en valeur mes compétences et projets.
    Faciliter le contact avec les recruteurs et entreprises.....
  - Pour chaque page, définir le contenu :
    - Questions (besoins) :
    - Réponses (services) :
    - Objectifs
    - Cibles

**Présentation (ou accueil)**

Besoins : comprendre rapidement qui je suis.

Réponses : présentation synthétique et orientation vers les sections clés.

Objectif : capter l'attention.

Cibles : recruteurs et entreprises.

**Compétences**

Besoins : identifier mes compétences techniques.

Réponses : langages, outils, domaines maîtrisés.

Objectif : démontrer mon expertise.

Cibles : recruteurs techniques.

**Education**

Besoins : connaître mon parcours académique.

Réponses : diplômes, formations, certifications, établissements et spécialisations.

Objectif : valoriser ma formation.

Cibles : recruteurs et responsables de stages.

**Projets**

Besoins : évaluer mon expérience concrète.

Réponses : projets réalisés, descriptions, technologies utilisées.

Objectif : prouver mes compétences.

Cibles : entreprises et encadrants.

**Contact**

Besoins : me joindre facilement.

Réponses : email, CV, liens professionnels.

Objectif : faciliter le recrutement.

Cibles : recruteurs et entreprises

- Protocole de navigation :
  Menu principal fixe et clair sur toutes les pages.
  Navigation fluide et intuitive entre les différentes sections……

## Technologie

- Utilisation des technologies web suivantes (mettre en évidence une réponse) :
  - HTML : oui / non (version : HTML 4 / HTML 5 / XHTML / ne sait pas)
    Oui (version HTML 5)
  - CSS : oui / non (version : CSS 2 / CSS 3 / ne sait pas)
    Oui (version CSS 3)
  - Javascript : oui / non
    Oui, Javascript
  - PHP : oui / non
    Non, pour PHP. Etant donné que j'héberge mon site sur GitHub pages, ce dernier ne prends pas en charge les fichier PHP pour l'envoi des données du formulaire, c'est ainsi que j'ai opté pour la plateforme Formspree afin de gérer l'envoi des données du formulaire à mon adresse mail. J'opterai pour un formulaire avec du PHP lorsque j'hébergerai mon site sur un domaine compétent comme .fr, .com, …
  - Base de données et requête(s) SQL : oui / non (type de bdd : mysql / sqlite / access / autre)
    Bases de données/SQL : Non
  - Autre :
    GitHub (hébergement du code)
    Voici le lien :
    **https://nadal243.github.io/portfolio-de-nadal/**
- Respect des normes :
  - Quel *doctype* est utilisé ? <!DOCTYPE html>……
  - Les pages html, ont-elles été testées par le validateur html W3C ? oui / non
    - Si oui, combien d'erreurs ? …0…….  d'avertissements ? …5……
  - Le css a-t-il été testé par le validateur css W3C ? oui / non
    - Si oui, combien d'erreurs ? …0…….  d'avertissements ? …3……
- Expliquer, pour chaque usage de javascript ou de script php, ce que le script permet :

**Explication** :

Ce script sert à afficher un texte animé avec un effet « machine à écrire », tout en s'adaptant automatiquement quand la langue du site change. Pour éviter les bugs visuels, il commence par arrêter et supprimer toute animation déjà en cours. Ensuite, il récupère les nouvelles phrases traduites, soit depuis un dictionnaire de données, soit via une valeur de secours si nécessaire. Une fois les textes prêts, il relance Typed.js avec des vitesses bien réglées afin d'afficher les phrases en boucle de façon fluide, naturelle et agréable pour l'utilisateur.

```
// Instance globale pour Typed.js afin de pouvoir la recréer lors d'un changement de langue
let typedInstance = null;

// Initialise ou réinitialise l'animation Typed.js selon la langue
function initTyped(lang) {
```

```
    const typedElement = document.querySelector('.typed');
    if (!typedElement || typeof Typed === 'undefined') return;

    // Détruit l'instance précédente si elle existe
    if (typedInstance && typeof typedInstance.destroy === 'function') {
        typedInstance.destroy();
    }

    // Récupère la liste des textes à partir des traductions
    let raw = TRANSLATIONS[lang] && TRANSLATIONS[lang]['hero.typed'];
    if (!raw) raw = TRANSLATIONS['fr']['hero.typed'] || '';

    // Fallback éventuel sur l'attribut data-typed-items si rien n'est
défini
    if (!raw) {
        raw = typedElement.getAttribute('data-typed-items') || '';
    }

    let strings = raw.split(',').map(s => s.trim()).filter(Boolean);
    if (!strings.length) return;

    typedInstance = new Typed('.typed', {
        strings,
        loop: true,
        typeSpeed: 100,
        backSpeed: 50,
        backDelay: 2000
    });
}
```

**Explication** :

Cette fonction parcourt tous les éléments de la page marqués d'un attribut spécifique pour remplacer leur contenu par la traduction correspondant à la langue sélectionnée. Elle sécurise l'affichage et utilise le français comme roue de secours si une traduction manque, tout en synchronisant l'animation de texte Typed.js pour qu'elle reste cohérente. Enfin, elle enregistre ce choix dans la mémoire du navigateur afin que le site conserve automatiquement la même langue lors de la prochaine visite de l'utilisateur.

```
// Applique les traductions aux éléments possédant l'attribut data-i18n
function applyTranslations(lang) {
    if (!TRANSLATIONS[lang]) return;
    document.querySelectorAll('[data-i18n]').forEach(el => {
        const key = el.getAttribute('data-i18n');
        const text = TRANSLATIONS[lang][key] || TRANSLATIONS['fr'][key];
        if (text !== undefined) {
            // Utilise textContent pour éviter d'injecter du HTML
            el.textContent = text;
        }
```

```
    });

    // Réinitialise l'animation Typed.js avec la bonne langue (si présente sur la page)
    initTyped(lang);

    // Enregistre la langue choisie dans le localStorage
    try { localStorage.setItem('site_lang', lang); } catch (e) { /* ignore */ }
}
```

**Explication** :

Ce code configure les boutons de sélection de langue en leur assignant une action de traduction immédiate lors du clic de l'utilisateur. Au chargement du site, il identifie automatiquement la langue à utiliser en consultant la mémoire du navigateur ou les paramètres par défaut de la page. Cette automatisation garantit que l'interface s'affiche instantanément dans la bonne version tout en mémorisant les préférences de visite pour les prochaines connexions.

```
// Initialise les boutons de changement de langue
function initLanguageButtons() {
    const btnFr = document.getElementById('ln-fr');
    const btnEn = document.getElementById('ln-en');
    if (btnFr) btnFr.addEventListener('click', () => applyTranslations('fr'));
    if (btnEn) btnEn.addEventListener('click', () => applyTranslations('en'));
}

// Initialise la langue lors du chargement de la page
document.addEventListener('DOMContentLoaded', () => {
    initLanguageButtons();
    const saved = (function() { try { return localStorage.getItem('site_lang'); } catch (e) { return null; }})();
    const defaultLang = saved || document.documentElement.lang || 'fr';
    applyTranslations(defaultLang);
});
```

**Explication** :

Ce script permet d'alterner entre un mode clair et un mode sombre tout en mémorisant le choix de l'utilisateur dans son navigateur. Il modifie dynamiquement l'apparence du site en ajoutant ou retirant une classe spécifique au document et met à jour l'icône du bouton de bascule pour refléter l'état actuel. Au démarrage, il applique automatiquement le thème enregistré ou se base sur les préférences système de l'ordinateur pour garantir un confort visuel immédiat dès l'ouverture de la page.

```
/* ========== GESTION DU THÈME (Clair/Sombre) ========== */
// Définit le thème en clair ou en sombre
function setTheme(theme) {
```

```
    try { localStorage.setItem('site_theme', theme); } catch (e) { /* ignore */ }
    if (theme === 'dark') {
        document.documentElement.classList.add('dark');
    } else {
        document.documentElement.classList.remove('dark');
    }
    // Met à jour l'icône du bouton de bascule
    const btn = document.getElementById('theme-toggle');
    if (btn) btn.textContent = theme === 'dark' ? '☼' : '☾';
}

// Bascule entre les thèmes clair et sombre
function toggleTheme() {
    const current = (function() { try { return localStorage.getItem('site_theme'); } catch (e) { return null; }})();
    const next = current === 'dark' ? 'light' : 'dark';
    setTheme(next);
}

// Initialise le thème au chargement (respecte les préférences système)
function initTheme() {
    const btn = document.getElementById('theme-toggle');
    if (btn) {
        btn.addEventListener('click', toggleTheme);
    }
    const saved = (function() { try { return localStorage.getItem('site_theme'); } catch (e) { return null; }})();
    const preferred = saved || (window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches ? 'dark' : 'light');
    setTheme(preferred);
}

document.addEventListener('DOMContentLoaded', initTheme);
```

**Explication** :

Ce bloc de code gère l'interactivité du formulaire de contact en affichant ou masquant dynamiquement le champ "Qui êtes-vous ?" selon que l'utilisateur se présente comme un recruteur ou non. Il intègre un outil de validation du format d'email et déclenche l'affichage d'un message de chargement visuel lors de la soumission des données vers le service Formspree. Enfin, le script automatise la gestion des champs obligatoires et initialise l'ensemble de ces comportements dès le chargement de la page pour garantir une expérience utilisateur fluide et sans erreurs.

```
/* ========== FORMULAIRE DE CONTACT ========== */
// Valide le format de l'email
function isValidEmail(email) {
    const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
```

```
    return regex.test(email);
}

// Gère l'affichage conditionnel du champ "Qui êtes-vous ?" en fonction du champ recruteur
function initRecruteurField() {
    const recruteurSelect = document.getElementById('recruteur');
    const quiEtesVousContainer = document.getElementById('qui-etes-vous-container');
    const quiEtesVousInput = document.getElementById('qui-etes-vous');

    if (recruteurSelect && quiEtesVousContainer && quiEtesVousInput) {
        recruteurSelect.addEventListener('change', function() {
            if (this.value === 'Non') {
                quiEtesVousContainer.style.display = 'block';
                quiEtesVousInput.setAttribute('required', 'required');
            } else {
                quiEtesVousContainer.style.display = 'none';
                quiEtesVousInput.removeAttribute('required');
                quiEtesVousInput.value = '';
            }
        });
    }
}

// Gère l'envoi du formulaire de contact avec Formspree (uniquement si contact-form existe)
document.addEventListener('DOMContentLoaded', function() {
    initRecruteurField();

    const form = document.getElementById('contact-form');
    if (form) {
        form.addEventListener('submit', function(e) {
            const reponseDiv = document.getElementById('form-reponse');

            // Si la validation native du navigateur passe, affiche un message de chargement
            // et laisse Formspree gérer l'envoi réel
            reponseDiv.innerHTML = `<div class="alert"><span class="loader"></span>Envoi en cours...</div>`;
        });
    }
});
```

**Explication** :

Ce bloc d’instructions configure plusieurs cartes de compétences pour qu'elles redirigent automatiquement l'utilisateur vers la page competence.html  dès qu'une d'entre elles est cliquée.

```
/* ========== RACCOURCIS VERS LES COMPÉTENCES ========== */
// Ajoute les clics sur les cartes de synthèse des compétences
['competence-item1','competence-item2','competence-item3'].forEach(id => {
    const el = document.getElementById(id);
    if (el) el.addEventListener('click', () => { window.location.href =
'competence.html'; });
});
```

**Explication** :

Ce script regroupe deux outils d'interaction pour la navigation et les documents. La première fonction affiche une fenêtre modale en changeant sa visibilité à l'écran grâce à son identifiant HTML. La seconde permet de fermer la fenêtre modale en cliquant sur une croix située en haut à droite.

```
/* ========== GESTION DES MODALES ========== */
// Ouvre une modale par son identifiant
function openModal(modalId) {
    document.getElementById(modalId).style.display = "block";
}
// Ferme une modale par son identifiant
function closeModal(modalId) {
    const modal = document.getElementById(modalId);
    if (modal) {
        modal.style.display = "none";
    }
}
```

**Explication** :

Ce script permet d'organiser et de trier dynamiquement les projets affichés sur la page projet.html grâce à un système de catégories. Lorsqu'un utilisateur clique sur un bouton de filtre, le code identifie la catégorie sélectionnée, masque les projets qui ne correspondent pas et met à jour en temps réel un compteur pour indiquer le nombre de projets visibles. Il gère également l'aspect visuel en mettant en évidence le bouton actif, garantissant ainsi une navigation fluide et une présentation claire du site dès l'ouverture de la page.

```
/* ========== FILTRAGE DES PROJETS ========== */
// Fonction de filtrage des projets pour la page projets.html
document.addEventListener('DOMContentLoaded', function() {
    // Récupère tous les boutons de filtre et les projets dans le DOM
    const filterButtons = document.querySelectorAll('.filter-btn');
    const projectItems = document.querySelectorAll('.project-item');
    const countElement = document.getElementById('count');

    /**
     * Filtre les projets par catégorie
     * @param {string} category - Catégorie à afficher ("all" montre tous
les projets)
     */
```

```
    function filterProjects(category) {
        let visibleCount = 0;
        // Affiche/masque les projets selon la catégorie
        projectItems.forEach(item => {
            if (category === 'all' || item.getAttribute('data-category')
=== category) {
                item.style.display = 'block';
                visibleCount++;
            } else {
                item.style.display = 'none';
            }
        });
        // Met à jour l'affichage du nombre de projets
        if (countElement) {
            countElement.textContent = visibleCount;
        }
    }

    // Ajoute les écouteurs de clic à tous les boutons de filtre
    filterButtons.forEach(button => {
        button.addEventListener('click', function() {
            // Retire la classe 'active' de tous les boutons
            filterButtons.forEach(btn => btn.classList.remove('active'));
            // Ajoute la classe 'active' au bouton cliqué
            this.classList.add('active');
            // Applique le filtrage
            const category = this.getAttribute('data-category');
            filterProjects(category);
        });
    });

    // Initialise le compteur de projets au chargement
    if (countElement) {
        countElement.textContent = projectItems.length;
    }
});
```

## Charte graphique

- Polices de caractères :
  Titres : Montserrat, sans-serif ……
  Textes : Open Sans, sans-serif……
- Couleurs :
  Couleur principale : #1E88E5 (bleu)
  Couleur texte : #222222 (gris très foncé)
  Couleur liens/accents : #E53935 (rouge léger) ……
- Fond de pages :
  Fond uni très clair (#F5F5F5) avec sections blanches (#FFFFFF) et légères ombres……

- Styles :
  Design épuré et moderne, beaucoup d’espaces blancs, boutons arrondis, icônes simples, effets de survol (hover) doux sur les liens et cartes de projets……
- Modèle de pages (dessiner des maquettes des pages) :