

SOMMAIRE

Introduction	2
Liste des compétences abordées	3
Résumé du projet	5
Cahier des charges	7
• Besoins	
• Objectifs	
• Cibles (personas)	
• User Stories	
• Minimum Viable Product (MVP)	
• Fonctionnalités détaillées	
• Wireframes	
• Charte graphique	
• Maquettes	
Spécifications techniques	25
Base de données	26
• Modèle Conceptuel de Données (MCD)	
• Modèle Logique de Données (MLD)	
• Dictionnaire des données	
• Routes (Endpoints)	
Exemples de conception	31
Tests	38
Conclusion	41
Remerciements	42

INTRODUCTION

Avant d'intégrer la formation de **Développeur Web et Web Mobile**, j'avais déjà un attrait marqué pour la programmation et le monde du numérique. Curieux de nature, j'ai commencé à apprendre par moi-même les bases du développement front-end, notamment en HTML, CSS et un peu de JavaScript. Cet apprentissage autodidacte m'a permis de réaliser mes premiers projets statiques, essentiellement des sites vitrines simples, et d'expérimenter avec des outils comme **WordPress**, qui m'ont donné une première approche des CMS et de la création de sites web.

Cette première expérience m'a confirmé mon intérêt particulier pour la conception d'interfaces et l'importance de l'expérience utilisateur (**UI/UX**). J'ai découvert que ce qui me plaisait le plus n'était pas seulement de coder, mais aussi de réfléchir à la manière dont les utilisateurs allaient interagir avec une application, et de rendre cette expérience la plus fluide et intuitive possible.

C'est dans ce cadre que j'ai intégré la formation de **développeur web et web mobile** à l'AFPA, afin d'acquérir des bases solides et professionnaliser ma pratique.

Dans le cadre de la formation, j'ai eu l'opportunité de travailler sur un projet plus ambitieux : **PlanIt**. Ce projet n'est pas seulement un exercice pratique, mais une véritable **synthèse de mon parcours**, réunissant les compétences acquises en front-end, en back-end et en conception. Ce projet m'a permis de mettre en pratique mes acquis, de confirmer mon attrait pour l'UI/UX et de franchir une étape importante dans mon apprentissage du développement web.

Au-delà de l'apprentissage technique, cette expérience est pour moi une véritable opportunité d'élargir mon champ de compétences et de donner naissance à un projet ambitieux, qui, je l'espère, continuera à évoluer après la formation.

LISTE DES COMPÉTENCES ABORDÉES

Modélisation et méthode Merise

Avant de commencer le développement, j'ai utilisé la méthode **Merise**¹ pour analyser et modéliser les besoins fonctionnels du projet. Cette approche m'a permis de structurer la réflexion autour des données et de clarifier les relations entre les entités principales : l'**Utilisateur** et la **Tâche**. La conception a suivi les étapes classiques, du **MCD (Modèle Conceptuel de Données)** au **MLD (Modèle Logique de Données)**, afin d'assurer une cohérence entre la vision métier et l'implémentation technique. Grâce à cette démarche, les contraintes comme l'unicité des adresses e-mail, la relation 1-N entre un utilisateur et ses tâches, ou encore l'ordre des tâches dans la semaine ont pu être identifiées et intégrées dès la conception. Cette phase de modélisation garantit que le projet repose sur une base solide et bien structurée.

Développement back-end

La partie back-end du projet repose sur un serveur **Node.js** construit uniquement avec les modules natifs, ce qui permet une compréhension approfondie des fondements du web. Le serveur expose une **API REST sécurisée** permettant la gestion de l'authentification et des tâches. Les utilisateurs peuvent s'inscrire, se connecter et maintenir une session grâce à des **tokens JWT** signés avec HMAC-SHA256 et comportant une date d'expiration. Les mots de passe sont protégés par un **hachage PBKDF2 avec sel**, garantissant un haut niveau de sécurité. Côté persistance, les données sont enregistrées dans un fichier JSON, représentant une base de données simplifiée et adaptée au contexte pédagogique. Le serveur implémente également un ensemble de bonnes pratiques : validation des entrées, gestion des erreurs, contrôle d'accès strict et protection contre les attaques temporelles via des comparaisons sécurisées.

Développement front-end

Sur le plan du front-end, j'ai mis l'accent sur la création d'une interface claire, fluide et responsive. L'application utilise **HTML5, CSS3 et JavaScript moderne** sans framework externe afin de renforcer la maîtrise des bases que j'ai acquies pendant ma formation. L'interface propose une expérience utilisateur intuitive : un système d'onglets pour naviguer entre l'inscription et la connexion, un tableau de bord hebdomadaire affichant les tâches par jour, et des modales pour ajouter ou gérer ses activités. Le **drag & drop** permet de réorganiser facilement les tâches, tandis que le code JavaScript gère l'état de la session et la communication avec l'API. L'adaptabilité est assurée par des **media queries** et des unités modernes comme clamp et minmax, garantissant une expérience fluide sur ordinateurs, tablettes et smartphones. Enfin, des mesures de sécurité côté client, comme les validations de formulaires et le stockage local limité aux informations essentielles, viennent compléter l'approche sécurisée de l'ensemble du projet.

¹ La méthode Merise est une approche de conception de système d'information développée en France dans les années 1970, structurée autour de modèles conceptuels, organisationnels et techniques pour séparer les données des traitements et garantir une analyse rigoureuse avant l'implémentation.

Documentation et outils : Docker et GitHub

Le projet **PlanIt** ne se limite pas au développement fonctionnel : une attention particulière a été portée à la **documentation et aux outils de collaboration**. Le code est versionné avec **Git** et hébergé sur **GitHub**, ce qui permet de conserver un historique complet des évolutions, de collaborer efficacement et de garantir la traçabilité. La rédaction d'un **dossier technique** accompagne le projet afin de détailler l'architecture, les choix technologiques et les perspectives d'évolution. Par ailleurs, l'application est pensée pour être intégrée dans un environnement **Docker** afin de simplifier son déploiement et son exécution sur différents systèmes. La conteneurisation facilite la reproductibilité de l'environnement, réduit les problèmes liés aux dépendances et prépare le projet à un déploiement dans un cadre professionnel.

Tests et validation

Enfin, une partie des compétences abordées réside dans la **validation et les tests** de l'application. Bien que le projet repose sur des technologies simples, chaque fonctionnalité a été testée de manière manuelle afin de garantir sa robustesse et sa conformité au cahier des charges. Les tests se concentrent sur les scénarios essentiels : inscription, connexion, ajout de tâche, modification par glisser-déposer, suppression et gestion de session. Les erreurs courantes (mauvais identifiants, champ vide, accès non autorisé) ont également été vérifiées pour s'assurer que l'application réagit correctement et de manière sécurisée. Ces tests, bien que manuels, permettent d'acquérir une rigueur méthodologique et préparent le terrain à l'intégration future de tests automatisés (unitaires ou end-to-end).

RÉSUMÉ DU PROJET

Au départ **PlanIt** était sensé être une application de to-do list, mais j'ai rapidement vu les limites de ce genre d'application et j'ai voulu concevoir quelque chose qui sera utile autant dans la vie personnel que dans le monde professionnel : dans un monde où les activités professionnelles et personnelles se multiplient, il devient essentiel de disposer d'un outil clair et efficace pour organiser ses journées. Beaucoup d'applications existantes (comme Trello) proposent des solutions complexes, souvent surchargées de fonctionnalités, ce qui peut décourager les utilisateurs recherchant avant tout la simplicité. J'ai donc conçu **PlanIt** comme un outil minimaliste et intuitif, centré sur l'essentiel : permettre à chacun de planifier ses tâches de manière fluide, intuitive et sécurisée.

Ce projet s'adresse à un public varié : étudiants souhaitant organiser leur emploi du temps hebdomadaire, professionnels désireux de mieux gérer leurs priorités, ou encore toute personne cherchant à équilibrer ses activités quotidiennes. Accessible depuis un simple navigateur, il ne nécessite ni installation lourde, ni connaissances techniques avancées, ce qui le rend particulièrement inclusif et universel. **PlanIt** se positionne ainsi comme un outil de productivité accessible à tous, mettant en avant la simplicité et la sécurité des données personnelles.

Objectifs principaux

Les objectifs de **PlanIt** se base autour de trois axes complémentaires :

- **Simplicité d'utilisation** : offrir une interface ergonomique qui ne nécessite aucun apprentissage particulier. L'utilisateur doit pouvoir créer, déplacer et supprimer une tâche en quelques clics, sans se perdre dans des menus complexes.
- **Organisation visuelle et intuitive** : proposer une vision claire de la semaine sous forme de tableau, où les tâches sont réparties par jour et peuvent être réorganisées facilement par un simple glisser-déposer.
- **Sécurité et fiabilité** : protéger les données sensibles des utilisateurs en mettant en place des mécanismes d'authentification et de gestion des mots de passe, tout en garantissant une persistance fiable grâce à un stockage local et une API sécurisée.

À travers ces objectifs, **PlanIt** ambitionne de devenir un compagnon simple mais efficace du quotidien, capable d'améliorer la gestion du temps et d'augmenter la productivité de ses utilisateurs.

Problèmes identifiés et solutions apportées

Dès les premières réflexions, j'ai identifié plusieurs problèmes courants liés à la gestion de tâches. Le premier concerne la **complexité des outils existants** : de nombreuses applications ajoutent des fonctionnalités superflues, ce qui surcharge l'interface et complique l'expérience utilisateur. J'ai choisi une approche opposée en privilégiant la **sobriété fonctionnelle** : seules les actions essentielles (ajout, modification, suppression, réorganisation) sont présentes, ce qui garanti une prise en main immédiate.

Un second problème réside dans la **sécurité des données personnelles**. Les informations liées à l'organisation d'un utilisateur peuvent être sensibles, et un stockage non sécurisé pourrait entraîner des risques d'exploitation. Pour répondre à cet enjeu, j'ai intégré des pratiques de sécurité éprouvées : hachage et salage des mots de passe, authentification par **JWT** avec expiration, et gestion stricte des droits d'accès à l'API.

Enfin, la **fragmentation des usages** entre ordinateurs et mobiles représente un défi supplémentaire. Beaucoup d'outils manquent de fluidité sur les petits écrans. J'ai donc relevé ce défi en adoptant une conception **responsive** dès le départ, en utilisant des techniques modernes de mise en page (flexbox, grid, unités dynamiques). Ainsi, l'application offre la même qualité d'expérience, qu'elle soit utilisée sur un poste de travail ou sur un smartphone.

Grâce à ces solutions, **PlanIt** répond non seulement aux besoins d'organisation hebdomadaire, mais le fait de manière sécurisée, intuitive et accessible.

CAHIER DES CHARGES

Besoins

Le besoin principal à l'origine de ce projet est celui d'un outil simple et efficace pour organiser sa semaine. Les utilisateurs recherchent souvent un support qui leur permette d'avoir une vision claire de leurs activités quotidiennes, sans devoir naviguer dans des interfaces complexes ou des applications trop chargées.

Ainsi, les besoins fonctionnels identifiés sont les suivants :

- **Créer et gérer des tâches** : chaque utilisateur doit pouvoir ajouter une tâche en lui attribuant un titre, un jour de la semaine et éventuellement une description.
- **Organiser visuellement la semaine** : les tâches doivent être affichées de manière structurée, jour par jour, dans un tableau clair et lisible.
- **Réorganiser facilement** : il est nécessaire de pouvoir déplacer une tâche d'un jour à l'autre ou de réordonner celles d'une même journée via un système intuitif de glisser-déposer.
- **Authentification sécurisée** : chaque utilisateur doit disposer de son propre compte afin que ses données restent privées et protégées.
- **Accessibilité multiplateforme** : l'outil doit être utilisable aussi bien sur ordinateur que sur smartphone, afin de correspondre aux usages modernes.

Sur le plan non fonctionnel, le besoin se traduit également par une exigence de **sécurité et de fiabilité**. Les données doivent être protégées (hachage des mots de passe, sessions sécurisées par JWT), et l'interface doit rester fluide et ergonomique, même pour un public peu habitué aux outils numériques.

En résumé, **PlanIt** doit répondre au besoin de disposer d'un **agenda numérique hebdomadaire accessible, pratique et sécurisé**, qui remplace ou complète efficacement l'agenda papier ou les applications trop complexes.

Objectifs

Les objectifs du projet découlent directement des besoins exprimés. Ils visent à offrir une solution concrète et adaptée aux attentes des utilisateurs.

Le premier objectif est de fournir une **interface utilisateur claire et minimaliste**, centrée uniquement sur l'essentiel : l'organisation de la semaine. L'utilisateur doit pouvoir comprendre immédiatement comment interagir avec l'application, sans tutoriel ni apprentissage complexe.

Un deuxième objectif est de garantir une **expérience fluide et cohérente** sur tous types d'appareils. Grâce à une conception responsive, l'application doit s'adapter aux ordinateurs, tablettes et téléphones, pour que les utilisateurs puissent gérer leurs tâches sur n'importe quel support.

Le troisième objectif concerne la **sécurisation des données personnelles**. Le système d'authentification doit être robuste, garantissant que seules les personnes autorisées ont accès à leurs informations. De plus, les tâches créées doivent être associées à chaque utilisateur, ce qui implique une gestion rigoureuse des sessions et des droits d'accès.

Enfin, le projet a pour objectif d'offrir une **base solide et évolutive**. *PlanIt* doit être conçu de manière à pouvoir accueillir, à l'avenir, des améliorations telles que le partage de plannings entre utilisateurs, l'ajout de rappels ou encore l'intégration avec des services tiers (Google Calendar, Outlook, etc.).

En somme, les objectifs de *PlanIt* ne se limitent pas à fournir une application fonctionnelle, mais visent à construire un outil **pérenne, fiable et évolutif**, répondant aux usages actuels tout en anticipant les besoins futurs.

Cibles

J'ai conçu **PlanIt** comme un outil simple et accessible à un large public. Cependant, certaines catégories d'utilisateurs se révèlent particulièrement adaptées à l'usage de ce type d'application. Pour illustrer cela, j'ai identifié trois **personas** qui incarnent les profils types des utilisateurs cibles.



Claire, l'étudiante organisée

- **Âge** : 21 ans
- **Statut** : Étudiante en licence de psychologie
- **Contexte** : Claire jongle entre ses cours, ses révisions, ses petits jobs et ses activités associatives. Elle a besoin d'un outil clair pour planifier ses journées et ne pas se laisser déborder.
- **Objectifs** :
 - Suivre ses devoirs et ses révisions par matière.
 - Planifier ses créneaux de travail et ses rendez-vous personnels.
 - Avoir une vision simple et globale de sa semaine.
- **Problèmes rencontrés** : Les applications d'agenda qu'elle connaît (Google Calendar, Outlook) sont trop complexes pour son usage quotidien, et elle se perd dans les nombreuses fonctionnalités.
- **Comment PlanIt l'aide** : Une interface épurée, avec uniquement une vue hebdomadaire et la possibilité d'ajouter des notes rapides, lui permet de gérer son temps sans stress.



Lucas, le salarié multitâches

- **Âge** : 35 ans
 - **Statut** : Cadre en entreprise
 - **Contexte** : Marc doit jongler entre ses réunions professionnelles, ses dossiers clients et sa vie familiale. Il recherche un outil pour **séparer clairement ses tâches personnelles et professionnelles**, et avoir une vision rapide de ses priorités.
- **Objectifs** :
 - Organiser ses réunions et ses tâches personnelles dans une seule interface.
 - Déplacer facilement une tâche en cas de changement de planning.
 - Maintenir un équilibre entre vie pro et vie perso.
 - **Problèmes rencontrés** : Ses outils professionnels sont trop orientés entreprise et ne lui permettent pas de mélanger efficacement les deux sphères de sa vie.
 - **Comment PlanIt l'aide** : Grâce au glisser-déposer, il peut réorganiser sa semaine en un clic, et la simplicité de PlanIt lui évite la surcharge cognitive liée aux logiciels d'agenda professionnels.



Karine, la maman active

- **Âge** : 42 ans
 - **Statut** : Mère de famille, à temps partiel comme assistante administrative
 - **Contexte** : Sophie doit gérer son emploi du temps au bureau, les activités scolaires et extrascolaires de ses enfants, ainsi que les rendez-vous familiaux (médecin, sorties, etc.).
- **Objectifs** :
 - Centraliser les rendez-vous familiaux et personnels.
 - Visualiser rapidement les journées chargées pour éviter les oublis.
 - Simplifier l'organisation des activités quotidiennes.
 - **Problèmes rencontrés** : Les carnets de notes papier ne sont pas assez pratiques et finissent souvent perdus ou incomplets. Les applications trop complexes la découragent rapidement.
 - **Comment PlanIt l'aide** : En quelques clics, elle enregistre les rendez-vous importants et peut consulter son planning de la semaine sur son téléphone, partout et à tout moment.

User Stories

Les user stories décrivent les besoins principaux des utilisateurs de **PlanIt** et la valeur qu'ils retirent de l'application.

Gestion du compte utilisateur

En tant qu'utilisateur,
je veux créer un compte sécurisé puis me connecter facilement à mon espace personnel,
afin de retrouver mes tâches sauvegardées et gérer mon planning de manière confidentielle.

Gestion des tâches

En tant qu'utilisateur,
je veux ajouter, modifier, déplacer ou supprimer des tâches pour chaque jour de la semaine,
afin de organiser mon emploi du temps de façon claire et flexible.

Ajout rapide et ergonomie

En tant qu'utilisateur pressé,
je veux ajouter une tâche rapidement sans remplir trop de détails,
afin de gagner du temps et noter l'essentiel immédiatement.

Vue hebdomadaire et personnalisation

En tant qu'utilisateur,
je veux visualiser mes tâches sur une semaine entière avec un affichage clair et un message personnalisé,
afin de avoir une vision globale de mes priorités et une expérience plus agréable.

Sécurité et confidentialité

En tant qu'utilisateur,
je veux que mes données personnelles et mes tâches soient sécurisées,
afin de pouvoir utiliser **PlanIt** en toute confiance.

Le MVP de PlanIt

Le projet a été pensé dès le départ autour d'un **MVP (Minimum Viable Product)**, c'est-à-dire une version simplifiée mais fonctionnelle permettant de répondre immédiatement aux besoins essentiels des utilisateurs. L'objectif était de proposer rapidement une solution utilisable, tout en gardant la possibilité d'ajouter des fonctionnalités plus avancées par la suite.

Dans cette première version, le MVP inclut :

- La **création et la connexion** sécurisée d'un compte utilisateur.
- La **visualisation hebdomadaire** des tâches, pour offrir une vue claire et synthétique.
- L'**ajout de tâches** (classique et rapide) avec titre, jour et description optionnelle.
- La **modification de l'organisation** par glisser-déposer et la **suppression** de tâches.
- Un **système de persistance des données** pour que chaque utilisateur retrouve son planning personnalisé.

Ce noyau fonctionnel constitue une base solide pour valider l'idée auprès d'utilisateurs réels. Il permet de vérifier que l'application répond bien aux besoins d'organisation personnelle, tout en ouvrant la voie à des évolutions futures (rappels, notifications, synchronisation multi-appareils, etc.).

Fonctionnalités détaillées

L'application **PlanIt** propose un ensemble de fonctionnalités permettant à l'utilisateur de gérer efficacement ses tâches sur une base hebdomadaire. Chaque fonctionnalité a été pensée pour répondre à un besoin précis, tout en maintenant une interface simple et ergonomique.

1. Authentification sécurisée

- **Inscription** : un utilisateur peut créer un compte en renseignant une adresse e-mail et un mot de passe (6 caractères minimum). Un prénom optionnel peut également être ajouté.
- **Connexion** : un utilisateur peut se connecter avec ses identifiants pour retrouver son espace personnel et ses données.
- **Déconnexion** : un bouton permet de fermer la session à tout moment, garantissant la confidentialité des informations.
- **Persistance de session** : grâce au stockage local du token, la session reste active même après un rechargement de la page.

2. Tableau de bord hebdomadaire

- **Vue par jours** : l'écran principal présente un calendrier composé de 7 colonnes, correspondant aux jours de la semaine (du lundi au dimanche).
- **Affichage dynamique** : chaque colonne affiche le nombre de tâches et la liste détaillée associée au jour sélectionné.
- **Indication visuelle** : lorsqu'aucune tâche n'est planifiée, l'information « Aucune tâche » est affichée pour éviter toute confusion.

3. Gestion des tâches

- **Ajout classique** : via une fenêtre de dialogue, l'utilisateur peut créer une tâche en précisant :
 - un titre obligatoire,
 - une description optionnelle,
 - le jour de la semaine associé.
- **Ajout rapide** : un formulaire simplifié permet d'ajouter une tâche en une seule étape (titre + jour).
- **Suppression** : chaque tâche peut être supprimée via un bouton dédié, avec confirmation de l'action.
- **Réorganisation** : les tâches peuvent être déplacées d'un jour à un autre par glisser-déposer (drag & drop).
- **Mise à jour automatique** : l'ordre des tâches est sauvegardé après chaque réorganisation.

4. Expérience utilisateur enrichie

- **Message d'accueil personnalisé** : un message adapté au moment de la journée (Bonjour, Bon après-midi, Bonne soirée, Bonne nuit) est affiché à l'utilisateur.
- **Interface intuitive** : la navigation est simplifiée grâce à l'utilisation d'onglets pour basculer entre la connexion et l'inscription.
- **Compatibilité multi-supports** : l'interface s'adapte aux écrans d'ordinateurs, tablettes et smartphones.

5. Sécurité et stockage des données

- **Sécurisation des mots de passe** : les mots de passe sont stockés sous forme hachée et salée côté serveur.
- **Authentification par token (JWT)** : chaque utilisateur est identifié par un jeton unique garantissant la confidentialité des échanges.
- **Sauvegarde locale** : les informations essentielles de session sont stockées localement (localStorage) de manière sécurisée.
- **Gestion des erreurs** : des messages clairs et précis sont affichés en cas d'erreur (connexion échouée, inscription déjà existante, session expirée, etc.).

Wireframes

Page d'accueil connexion/création de compte desktop & mobile :

This wireframe represents a desktop version of a login and registration page. At the top, there is a header bar containing a blue square labeled 'Logo' and the text 'PlanIt' next to a horizontal line. Centered below the header is a light gray rectangular box. Inside this box, at the top, are two tabs: 'Connexion' and 'Créer un compte'. Below the tabs are three input fields: 'Mail', 'Mdp', and a third unlabeled field. At the bottom of the box is a blue button labeled 'Créer compte'.

This wireframe represents a mobile version of the login and registration page. It features the same header as the desktop version. The central light gray box contains the same 'Connexion' and 'Créer un compte' tabs. However, the input fields are arranged differently: 'Prénom' is the first field, followed by 'Mail', and then 'Mdp'. The blue 'Créer compte' button remains at the bottom of the form area.



Connexion Créer un compte

Mail

Mdp

Créer compte



Connexion Créer un compte

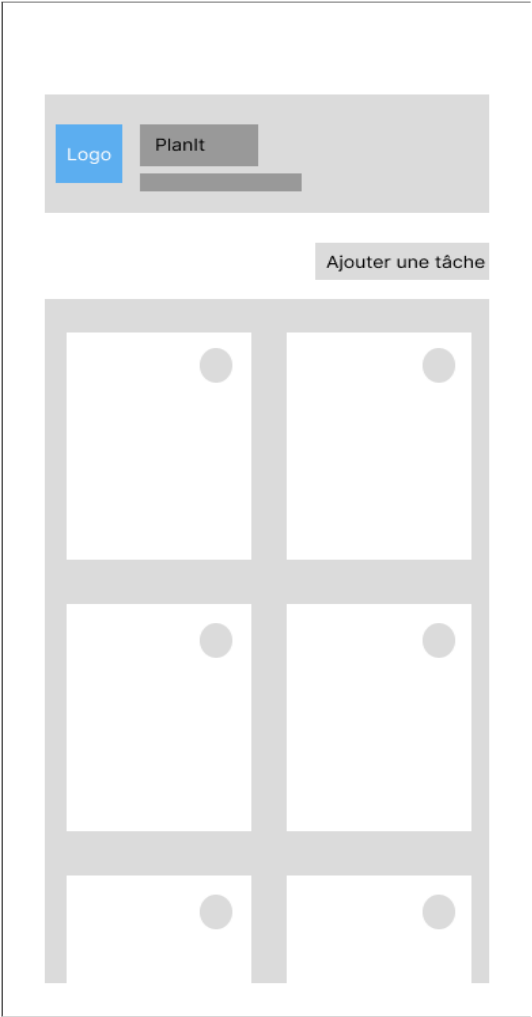
Prénom

Mail

Mdp

Créer compte

Page principal calendrier version desktop & mobile :



Charte graphique

La charte graphique de **PlanIt** a été pensée pour refléter la simplicité, la modernité et la clarté indispensables à un outil de planification efficace. Elle vise à offrir une expérience visuelle à la fois agréable et fonctionnelle, adaptée à une utilisation quotidienne sur ordinateur et smartphone.



1. Identité visuelle et logo

Le logo de **PlanIt** repose sur un **symbole simple et reconnaissable** associé à une typographie moderne et lisible. L'icône centrale reprend un **signe étoilé**, symbolisant l'organisation, la mise en avant et la priorisation des tâches.

Le logo peut être décliné en plusieurs versions :

- Icône seule (utilisée pour la favicon ou les notifications),
- Logo complet avec le nom *PlanIt*, utilisé pour l'en-tête et les supports de communication.

Cette identité graphique met l'accent sur l'équilibre entre sérieux (gestion d'organisation) et dynamisme (couleurs modernes et formes arrondies).

2. Palette de couleurs

La palette de **PlanIt** se compose de **couleurs sobres rehaussées de teintes vives**, permettant de distinguer rapidement les éléments importants.

- **Couleur principale** : Bleu (#5764f8) → symbolise la confiance et la productivité.
- **Couleur secondaire** : Turquoise (#24c7d6) → apporte une touche de dynamisme et de fraîcheur.
- **Couleur d'accent** : Orange (#ffb347) → met en avant les éléments interactifs ou prioritaires.
- **Couleurs neutres** : Blanc cassé (#f5f6fb) et gris foncé (#1b2559) pour assurer lisibilité et contraste.

L'ensemble est utilisé de manière cohérente dans l'application :

- Les boutons et actions principales utilisent le **dégradé bleu-turquoise**,
- Les informations secondaires apparaissent en **gris atténué**,
- Les notifications et alertes sont renforcées par des couleurs contrastantes (rouge pour les erreurs, vert/bleu pour les succès).

3. Typographie

J'ai utilisé la typographie **Inter**, une police moderne, lisible et parfaitement adaptée au web. Elle assure une bonne hiérarchie visuelle grâce à ses variantes (400 à 700).

- **Titres (h1, h2, h3...)** : police en gras (600/700), pour donner du relief et structurer le contenu.
- **Texte courant** : corps en régulier (400), garantissant une bonne lisibilité.
- **Sous-titres et annotations** : en semi-bold (500/600), pour distinguer les métadonnées et informations secondaires.

Cette uniformité permet une **expérience utilisateur cohérente** tout en donnant un aspect professionnel.

4. Style et mise en forme des composants

L'interface repose sur des **formes arrondies** et des **effets d'ombre légers**, renforçant la convivialité et la modernité :

- **Boutons** : arrondis (radius important), avec un effet de survol et des dégradés dynamiques,
- **Cartes (tasks, calendrier)** : ombres douces pour donner de la profondeur,
- **Dialogues et formulaires** : arrière-plans semi-transparents avec effet *blur* pour une meilleure hiérarchie visuelle.

5. Iconographie et visuels

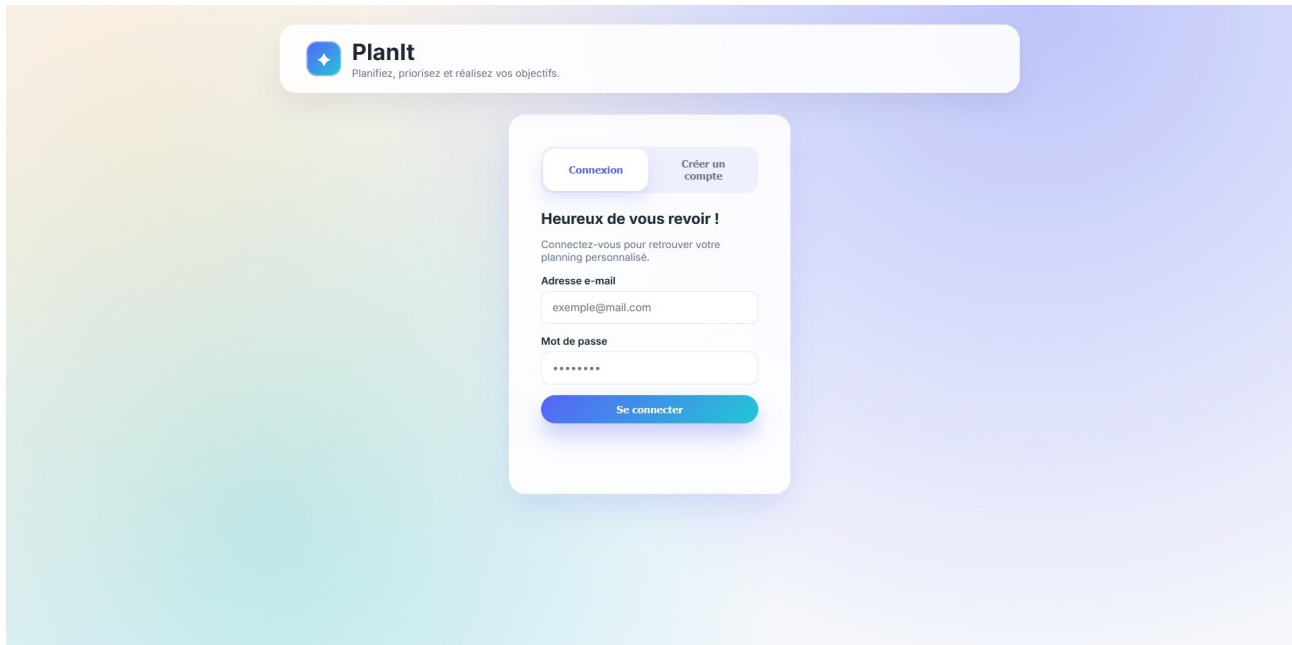
Les icônes utilisées sont simples et épurées, centrées sur l'action :

- « + » pour ajouter une tâche,
- « X » pour supprimer,
- Icônes circulaires et minimalistes pour conserver l'harmonie visuelle.

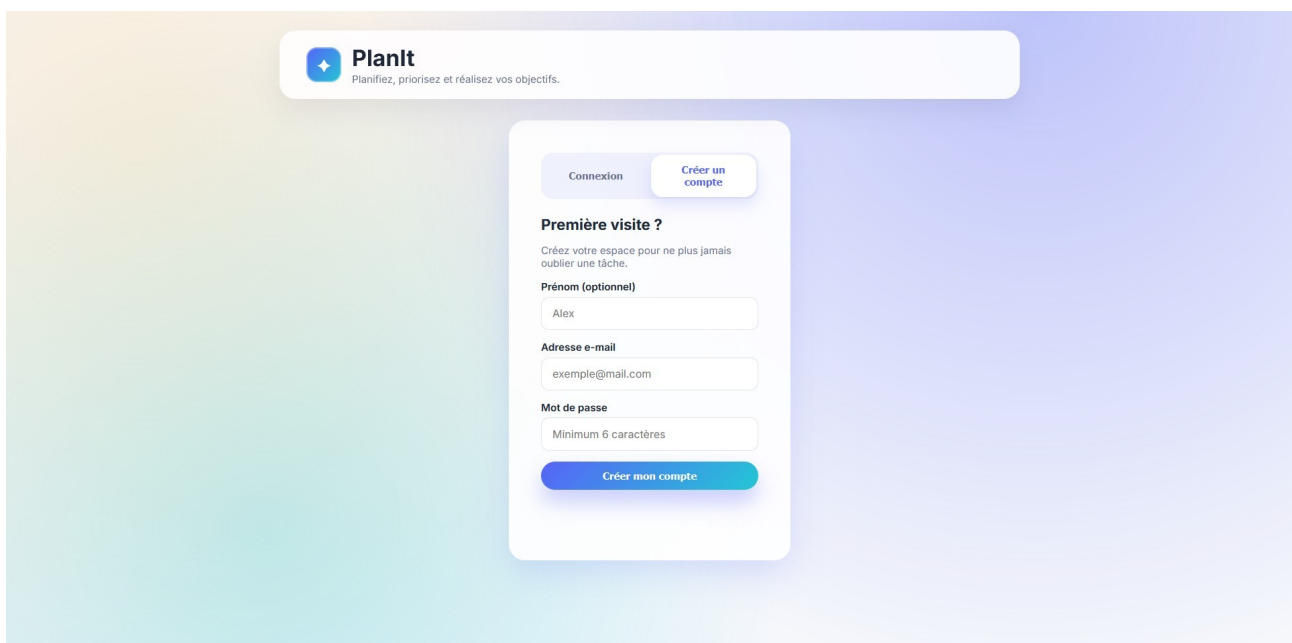
L'approche minimaliste limite la surcharge visuelle et renforce la **lisibilité immédiate** des fonctionnalités.

Maquettes

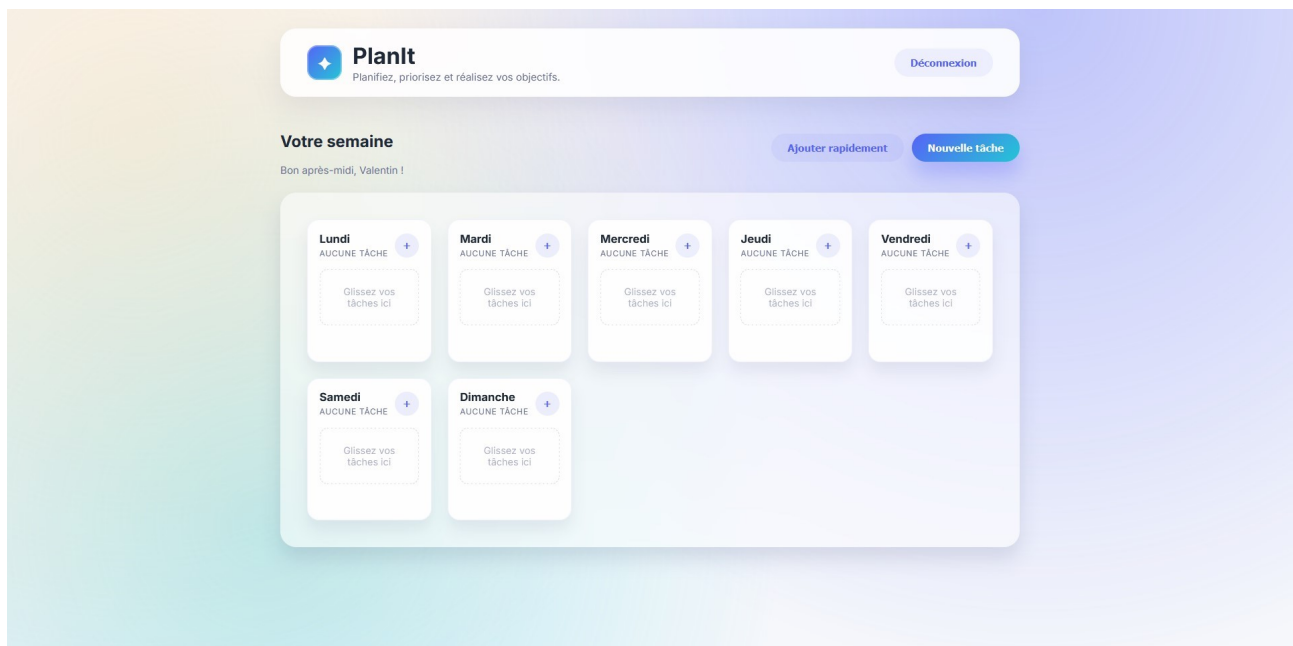
J'ai conçu plusieurs maquettes interactives avec Figma², me permettant d'explorer différentes mises en page, d'affiner l'ergonomie et d'assurer une cohérence visuelle avec l'identité du projet.



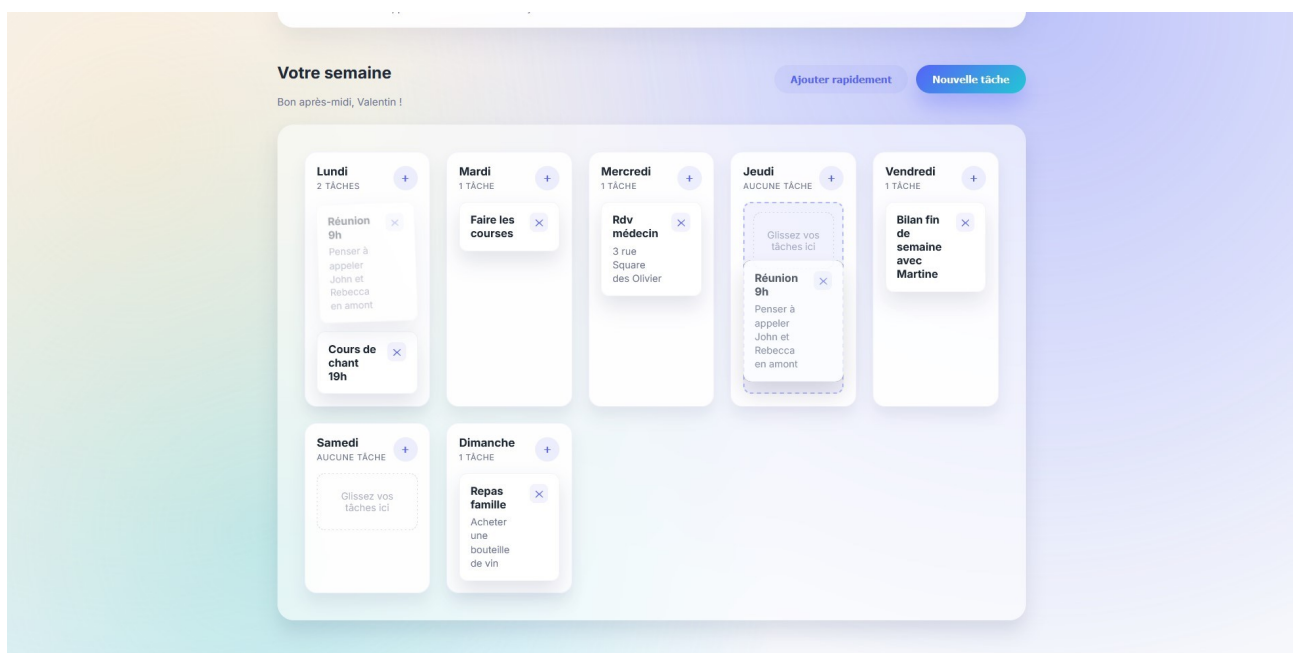
Page d'accueil avec option «Connexion» & «Créer un compte»

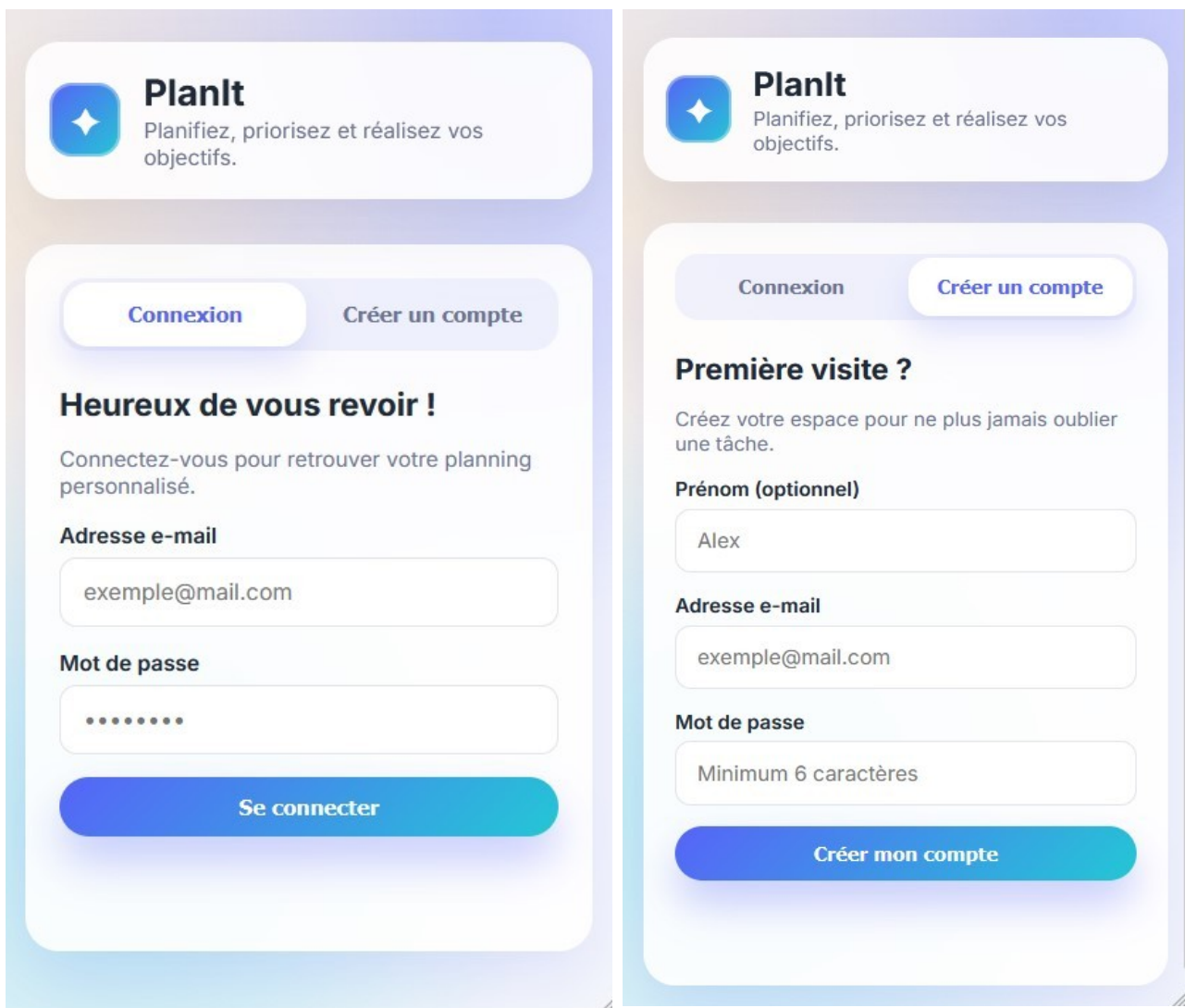


2 Figma est un éditeur de graphiques vectoriels et un outil de prototypage

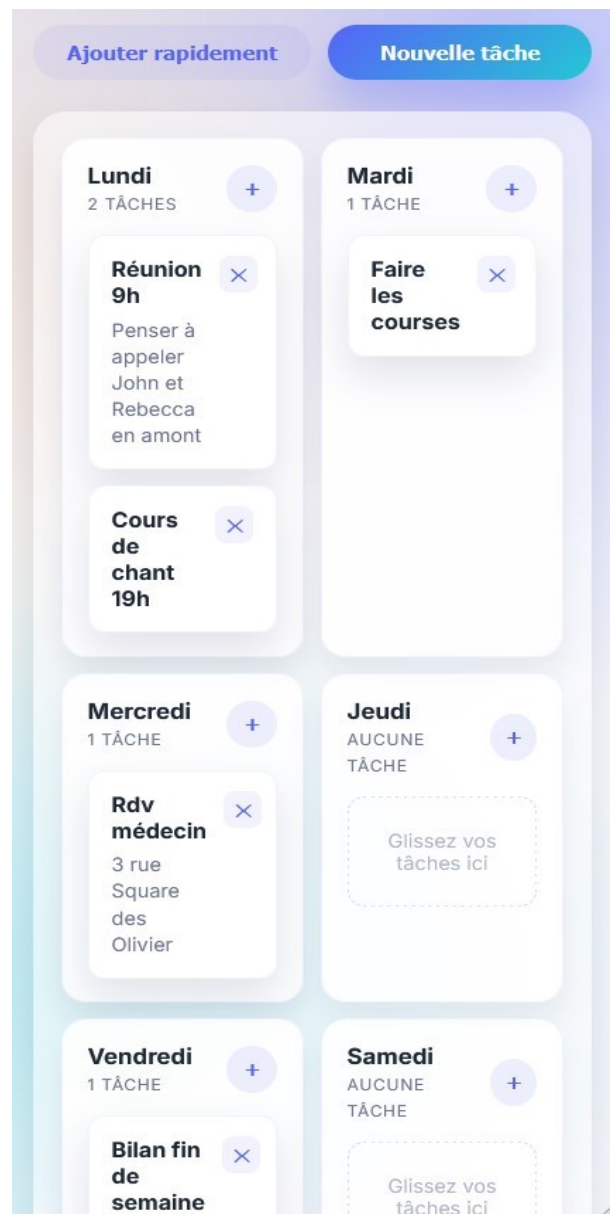


Page principale et fonctionnalité drag & drop





Page connexion et création de compte version mobile



Calendrier version mobiles

SPÉCIFICATIONS TECHNIQUES

J'ai voulu que ce projet repose sur une architecture simple et efficace, pensée pour allier accessibilité, rapidité et évolutivité.

1. Technologies utilisées

- **Front-end :**

Le côté client est développé en **HTML5**, **CSS3** et **JavaScript Vanilla (ES6)**, garantissant une compatibilité large sans dépendre de frameworks lourds.

Le design repose sur une approche **responsive** pour une adaptation automatique sur ordinateurs, tablettes et smartphones.

- **Back-end :**

Le serveur est construit avec **Node.js** (sans framework supplémentaire, uniquement le module http).

La gestion des utilisateurs et des tâches est assurée via des **endpoints REST** simples et sécurisés (authentification JWT, hashage des mots de passe avec crypto).

- **Base de données :**

Une base de données locale en **JSON** (data.json) est utilisée pour stocker les utilisateurs et leurs tâches. Ce choix léger correspond à un MVP, mais peut évoluer vers une base relationnelle (MySQL, PostgreSQL) ou NoSQL (MongoDB).

2. Compatibilité et déploiement

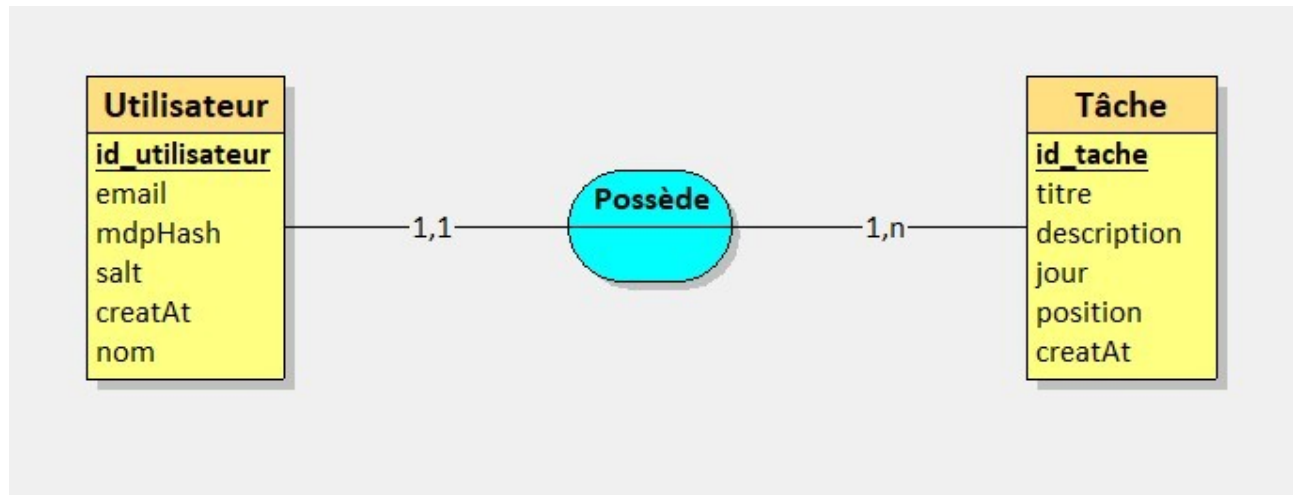
- **Navigateurs compatibles :** Google Chrome, Mozilla Firefox, Microsoft Edge, Safari (versions récentes).

- **Déploiement :**

- Localement via Node.js (npm start).
- Hébergement possible sur un VPS, un serveur cloud (AWS, GCP, Azure) ou via des services comme **Heroku** ou **Render**.
- Le front peut également être hébergé sur **GitHub Pages**, avec un back-end accessible en ligne via une API déployée.

BASES DE DONNÉES

MCD



MLD



Utilisateur(id_utilisateur, email, nom, mdpHash, salt, createdAt)

Tâche(id_tache, titre, description, jour, position, createdAt, id_utilisateur)

Dictionnaire des données

Table	Attribut	Type	Description
Utilisateur	id	UUID	Identifiant unique de l'utilisateur
	email	VARCHAR(255)	Adresse e-mail (unique)
	name	VARCHAR(100)	Prénom ou nom affiché (optionnel)
	passwordHash	VARCHAR(255)	Mot de passe hashé
	salt	VARCHAR(255)	Sel utilisé pour sécuriser le hash
	createdAt	DATETIME	Date de création du compte
Tâche	id	UUID	Identifiant unique de la tâche
	userId	UUID (FK)	Référence vers l'utilisateur propriétaire
	title	VARCHAR(80)	Titre court de la tâche
	description	TEXT	Notes ou détails supplémentaires (optionnel)
	day	INTEGER (0-6)	Jour de la semaine (0 = Lundi, 6 = Dimanche)
	position	INTEGER	Position de la tâche dans la journée (ordre d'affichage, drag & drop)
	createdAt	DATETIME	Date de création de la tâche

Endpoints

Authentication

- **POST** /api/auth/register
→ Crée un nouveau compte utilisateur.
 - Entrée : { "email": "...", "password": "...", "name": "..." }
 - Réponse : { "token": "...", "user": {...} }
- **POST** /api/auth/login
→ Connecte un utilisateur existant.
 - Entrée : { "email": "...", "password": "..." }
 - Réponse : { "token": "...", "user": {...} }
- **GET** /api/auth/me
→ Retourne les informations de l'utilisateur connecté (via JWT).
 - Entrée : header Authorization: Bearer <token>
 - Réponse : { "user": {...} }

Tâches

- **GET** /api/tasks
→ Récupère toutes les tâches de l'utilisateur connecté.
 - Réponse : { "tasks": [{...}, {...}] }
- **POST** /api/tasks
→ Crée une nouvelle tâche.
 - Entrée : { "title": "...", "description": "...", "day": 0 }
 - Réponse : { "task": {...} }
- **PUT** /api/tasks/:id
→ Met à jour une tâche existante (titre, description, jour, position).
 - Entrée : { "title": "...", "description": "...", "day": 2, "position": 1 }
 - Réponse : { "task": {...} }
- **DELETE** /api/tasks/:id
→ Supprime une tâche.
 - Réponse : 204 No Content
- **PUT** /api/tasks/reorder
→ Met à jour l'ordre des tâches (drag & drop).
 - Entrée : { "tasks": [{ "id": "...", "day": 0, "position": 0 }, ...] }
 - Réponse : { "success": true }

Routes Front-End

L'application **PlanIt** ne repose pas sur un framework de routing complexe (comme React). Elle utilise une **navigation monopage (SPA)** gérée directement en **JavaScript**.

Il n'y a donc pas de vraies routes côté client, mais plutôt des **vues conditionnelles** affichées/masquées selon l'état de l'application.

1. Vue Authentification

- **Section HTML** : `<section class="auth-view" data-auth>`
- **Contenu** :
 - Formulaire de connexion (`#login-form`)
 - Formulaire de création de compte (`#register-form`)
- **Logique JS** :
 - Affichée par défaut si aucune session n'est active.
 - Navigation par **onglets** (Connexion ↔ Créer un compte).
 - Quand un utilisateur s'authentifie avec succès, la vue est masquée et le tableau de bord est affiché.

2. Vue Tableau de bord (Board)

- **Section HTML** : `<section class="board-view hidden" data-board>`
- **Contenu** :
 - En-tête avec salutation personnalisée (`data-user-greeting`).
 - Boutons d'action :
 - "Nouvelle tâche" (ouvre une fenêtre modale complète).
 - "Ajouter rapidement" (ouvre une fenêtre modale simplifiée).
 - Un calendrier hebdomadaire (`[data-calendar]`) divisé en **7 colonnes (Lundi → Dimanche)**.
- **Logique JS** :
 - Chargement des tâches depuis l'API (`/api/tasks`).
 - Ajout/suppression/modification de tâches.
 - Drag & drop pour réordonner les tâches.
 - Bouton **Déconnexion** (retour à la vue Authentification).

3. Vues Dialogues modaux

- **Nouvelle tâche** : `<dialog class="task-dialog" data-task-dialog>`
 - Permet de saisir un titre, un jour et une description.
- **Ajout rapide** : `<dialog class="quick-task" data-quick-task>`
 - Saisie simplifiée : titre + jour.
- **Logique JS** :
 - Ouverts via des boutons dans le tableau de bord.
 - Fermeture via un bouton ou en annulant le formulaire.
 - Envoi du formulaire → création de la tâche via `/api/tasks`.

PlanIt

EXEMPLES DE CONCEPTION



Authentification et gestion de session

- **Problème à résoudre** : Permettre à un utilisateur de créer un compte, se connecter et rester identifié de manière sécurisée.
- **Solution** :
 - Côté **back-end**, génération d'un token JWT et stockage dans localStorage côté client.
 - Côté **front-end**, restauration de session automatique si un token valide est trouvé.

Extrait de code (back-end – Node.js)

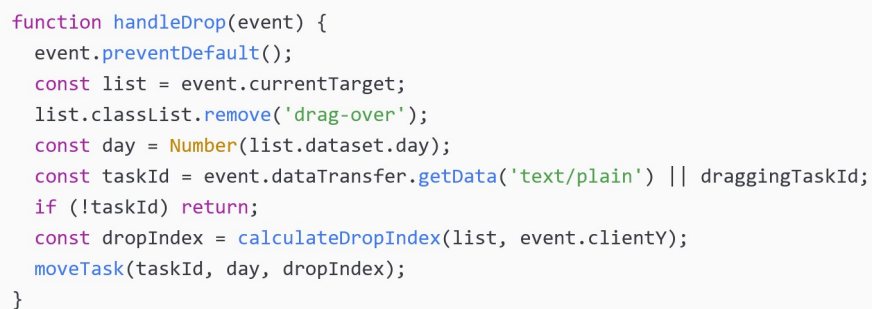
```
function generateToken(payload) {  
  const header = base64UrlEncode(JSON.stringify({ alg: 'HS256', typ: 'JWT' }));  
  const exp = Math.floor(Date.now() / 1000) + 60 * 60 * 24 * 30; // 30 jours  
  const body = { ...payload, exp };  
  const encodedPayload = base64UrlEncode(JSON.stringify(body));  
  const signature = sign(`${header}.${encodedPayload}`);  
  return `${header}.${encodedPayload}.${signature}`;  
}
```

Extrait de code (front-end – JS)

```
async function restoreSession() {  
  const savedToken = localStorage.getItem(STORAGE_TOKEN_KEY);  
  if (!savedToken) return;  
  state.token = savedToken;  
  try {  
    const data = await api('/api/auth/me');  
    if (data?.user) {  
      state.user = data.user;  
      await enterBoard();  
    } else {  
      clearSession();  
    }  
  } catch {  
    clearSession();  
  }  
}
```


Drag & Drop des tâches dans le calendrier

- **Problème à résoudre** : Permettre aux utilisateurs de réorganiser leurs tâches visuellement par simple glisser-déposer.
- **Solution** : Utilisation des événements dragstart, dragover, drop pour déplacer une tâche et recalculer son ordre.



```
function handleDrop(event) {  
  event.preventDefault();  
  const list = event.currentTarget;  
  list.classList.remove('drag-over');  
  const day = Number(list.dataset.day);  
  const taskId = event.dataTransfer.getData('text/plain') || draggingTaskId;  
  if (!taskId) return;  
  const dropIndex = calculateDropIndex(list, event.clientY);  
  moveTask(taskId, day, dropIndex);  
}
```

Création rapide de tâche

- **Problème à résoudre** : Ajouter une tâche en une seule étape, sans ouvrir la boîte de dialogue détaillée.
- **Solution** : Un formulaire simplifié en bas de page, qui ne demande que le titre et le jour de la semaine.

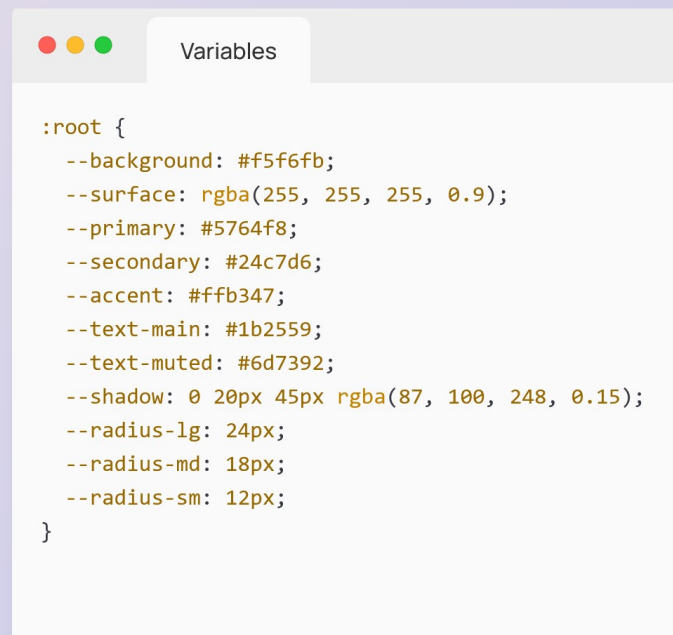


```
async function handleQuickTaskSubmit(event) {
  event.preventDefault();
  const formData = new FormData(quickTaskForm);
  const title = (formData.get('title') || '').trim();
  const day = Number(formData.get('day'));
  if (!title) return;
  const task = { title, day };
  const data = await api('/api/tasks', { method: 'POST', body: task });
  if (data?.task) {
    addTaskToBoard(data.task);
    renderBoard();
  }
  quickTaskDialog.close();
}
```

Exemples de conception CSS

Utilisation des variables CSS pour une charte graphique cohérente

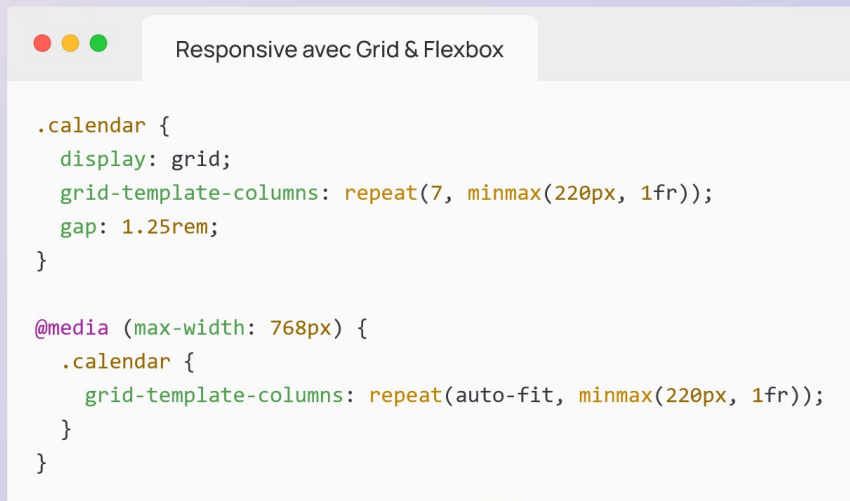
- **Problème à résoudre** : Avoir une identité visuelle homogène et facilement modifiable.
- **Solution** : Définition de variables globales (:root) pour les couleurs, ombres et rayons de bordure.



```
:root {
  --background: #f5f6fb;
  --surface: rgba(255, 255, 255, 0.9);
  --primary: #5764f8;
  --secondary: #24c7d6;
  --accent: #ffb347;
  --text-main: #1b2559;
  --text-muted: #6d7392;
  --shadow: 0 20px 45px rgba(87, 100, 248, 0.15);
  --radius-lg: 24px;
  --radius-md: 18px;
  --radius-sm: 12px;
}
```

Design responsive avec Grid et Flexbox

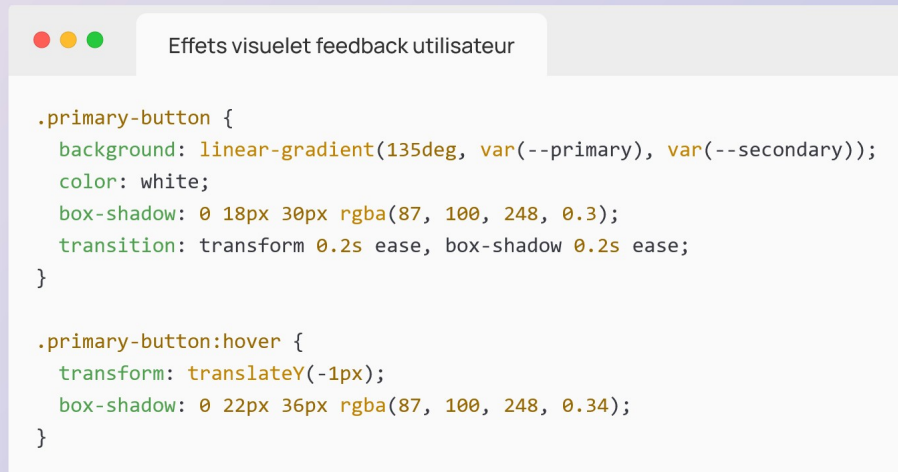
- **Problème à résoudre** : Adapter l'application aux écrans de bureau, tablettes et mobiles.
- **Solution** : Utilisation combinée de flex et grid, avec des media queries pour adapter l'affichage.



```
.calendar {  
  display: grid;  
  grid-template-columns: repeat(7, minmax(220px, 1fr));  
  gap: 1.25rem;  
}  
  
@media (max-width: 768px) {  
  .calendar {  
    grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));  
  }  
}
```

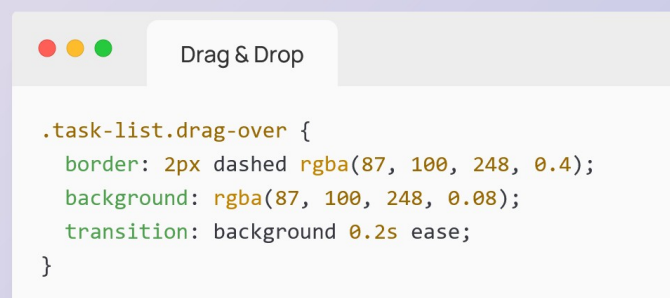
Effets visuels et feedback utilisateur

- **Problème à résoudre** : Rendre l'interface agréable et donner des retours visuels clairs.
- **Solution** : Ombres, transitions et états de survol (hover) pour améliorer l'UX.



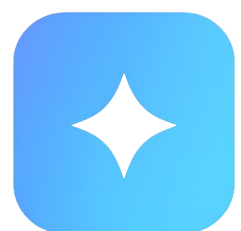
Mise en avant du Drag & Drop avec états visuels

- **Problème à résoudre** : Indiquer clairement où l'utilisateur peut déposer une tâche.
- **Solution** : Styles appliqués lors du survol (drag-over).



PlanIt

TESTS



TESTS UNITAIRE

Afin de garantir la fiabilité et la maintenabilité du projet **PlanIt**, des tests unitaires ont été envisagés pour vérifier le bon fonctionnement des principales fonctionnalités de l'application. Les tests unitaires permettent de valider, de manière isolée, chaque fonction critique du code afin de détecter rapidement d'éventuelles régressions lors d'évolutions futures.

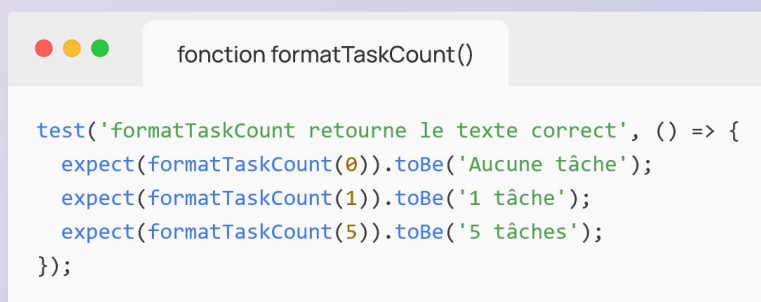
Objectifs des tests unitaires

- Vérifier que les fonctions de manipulation des tâches (ajout, suppression, réorganisation) se comportent correctement.
- Garantir que la logique métier du back-end (authentification, gestion des droits, persistance des données) reste fiable.
- Assurer que l'application réagit correctement aux cas d'erreurs (ex. identifiants incorrects, champ vide, etc.).

Ces exemples illustrent comment il est possible de tester les fonctions critiques de l'application, aussi bien côté **front-end** (JavaScript) que côté **back-end** (API). Bien que le projet PlanIt ait été conçu comme un **MVP**, la mise en place de tests unitaires constitue une étape essentielle pour fiabiliser le code et préparer les futures évolutions.

Vérification de la fonction formatTaskCount()

Cette fonction permet d'afficher le nombre de tâches par jour avec une formulation adaptée.



```
fonction formatTaskCount()

test('formatTaskCount retourne le texte correct', () => {
  expect(formatTaskCount(0)).toBe('Aucune tâche');
  expect(formatTaskCount(1)).toBe('1 tâche');
  expect(formatTaskCount(5)).toBe('5 tâches');
});
```

Vérification de l'ajout d'une tâche

Ce test isole la logique d'ajout d'une tâche au tableau de bord.



```
test('addTaskToBoard ajoute correctement une tâche', () => {
  const board = createEmptyBoard();
  const task = { id: '1', title: 'Nouvelle tâche', day: 0, position: 0 };
  state.board = board;
  addTaskToBoard(task);

  expect(state.board[0].length).toBe(1);
  expect(state.board[0][0].title).toBe('Nouvelle tâche');
});
```

Vérification du login côté API

Je m'assure qu'un utilisateur existant peut bien se connecter avec les bons identifiants.



```
test('handleLogin accepte un utilisateur valide', async () => {
  const credentials = { email: 'test@mail.com', password: 'secret' };
  const response = await api('/api/auth/login', {
    method: 'POST',
    body: credentials
  });

  expect(response).toHaveProperty('token');
  expect(response.user.email).toBe(credentials.email);
});
```


CONCLUSION

La réalisation de **PlanIt** a été une expérience enrichissante qui m'a permis de mettre en pratique l'ensemble des compétences acquises durant ma formation de développement web et web mobile. Ce projet m'a confronté aux défis du développement d'une application complète, allant de la conception initiale à l'implémentation technique, en passant par les choix d'architecture et les tests.

Un choix important de ce projet a été de ne pas utiliser de framework front-end comme React, Vue ou Angular. Ce choix est volontaire : il m'a permis de renforcer ma maîtrise des fondamentaux du web (HTML, CSS, JavaScript Vanilla) et de mieux comprendre les mécanismes internes d'une application monopage (SPA). Travailler sans framework a aussi simplifié le projet dans le cadre d'un MVP (Minimum Viable Product), tout en garantissant une application rapide et légère. Dans un contexte professionnel, l'ajout d'un framework reste possible et permettrait d'améliorer la modularité et la maintenabilité du code.

Un autre axe central du projet a été de mettre en avant l'importance de l'**UI/UX** (expérience utilisateur et interface utilisateur). J'ai porté une attention particulière à la simplicité des parcours utilisateurs (connexion, ajout rapide de tâches, organisation visuelle des colonnes) ainsi qu'à la clarté de l'interface (contrastes, lisibilité, hiérarchie visuelle). Le système de drag & drop et les feedbacks immédiats (messages, couleurs, animations) visent à rendre l'application non seulement fonctionnelle mais aussi agréable à utiliser.

Enfin, **PlanIt** n'est pas une fin en soi mais un **point de départ**. Son architecture simple et évolutive laisse place à de nombreuses améliorations possibles : passage à une véritable base de données relationnelle, ajout de fonctionnalités collaboratives, intégration d'outils de productivité ou encore internationalisation.

À travers ce projet, j'ai donc non seulement validé mes acquis techniques en front-end et back-end, mais j'ai aussi confirmé mon attrait pour l'**UI/UX design**, domaine dans lequel je souhaite orienter la suite de mon apprentissage. Mon objectif futur sera de développer des applications qui ne se contentent pas de répondre à un besoin technique, mais qui offrent également une **expérience utilisateur fluide, intuitive et plaisante**.

REMERCIEMENTS

La réalisation de *PlanIt* a été une expérience enrichissante, rendue possible grâce aux contributions de nombreuses personnes.

Je tiens à exprimer ma gratitude envers l'équipe pédagogique de l'**AFPA de Niort** et particulièrement à **Fatahou SOULA**, dont l'encadrement et les conseils m'ont permis d'approfondir mes compétences et de structurer ce projet.

Un immense merci à **Roland De TOMASI**, qui m'a accueilli lors de mon stage de formation. Sa bienveillance, son expertise et son accompagnement m'ont permis d'appliquer concrètement mes connaissances, de progresser dans un cadre professionnel et de me faire découvrir le métier D'UI/UX designer.

Enfin, mes sincères remerciements aux testeurs et contributeurs, dont les retours ont permis d'améliorer l'ergonomie et les fonctionnalités de ce projet.

Ce projet n'est qu'une étape, et j'espère qu'il continuera d'évoluer au fil du temps.