



Licence MIASHS Deuxième Année

Rapport de projet Programmation Web et Introduction PHP

Gestionnaire de contacts

Projet réalisé du 22 janvier 2025 au 4 mai 2025



<https://github.com/kevkan75/GestionnaireDeContacts>

Membres du groupe :

BENDJEBBAR Adam 43006224
KAN Kevin 43005517

Remerciements

La réalisation de ce projet n'aurait pas été possible sans le soutien précieux et les conseils avisés de nombreuses personnes, que nous souhaitons ici remercier chaleureusement.

Tout d'abord, nous tenons à exprimer notre gratitude particulière à M. Thibault ANANI, enseignant de Programmation Web et Introduction PHP, pour son encadrement bienveillant, sa disponibilité constante et ses précieux conseils tout au long de ce projet. Ses explications claires durant les cours magistraux ainsi que son accompagnement attentif lors des séances de travaux dirigés nous ont permis de surmonter de nombreuses difficultés, d'aborder avec confiance ce projet ambitieux et de progresser significativement dans nos apprentissages.

Nous souhaitons également remercier chaleureusement nos camarades de promotion, avec qui les échanges réguliers et constructifs ont été particulièrement motivants. Leurs conseils ponctuels, leur esprit d'équipe et leur solidarité ont largement contribué à rendre cette expérience à la fois enrichissante et agréable.

Enfin, nous tenons à remercier sincèrement l'Université Paris Nanterre pour les ressources pédagogiques et matérielles mises à notre disposition tout au long de ce projet. Cet environnement propice à l'apprentissage a grandement facilité le déroulement de notre travail et a permis à notre projet de prendre forme dans les meilleures conditions possibles.

Ce projet représente pour nous une véritable aventure pédagogique et humaine. À toutes celles et ceux qui ont contribué de près ou de loin à sa réalisation. Nous adressons ici l'expression de nos plus sincères remerciements. Votre appui, votre écoute et votre implication ont été essentiels à la réussite de ce projet.

Table des matières

1	Introduction	4
2	Présentation du projet	5
2.1	Objectifs du projet	5
2.2	Fonctionnalités détaillées	6
2.3	Technologies utilisées	9
2.4	Structure de l'application	11
3	Environnement de travail	21
4	Conception du projet	23
4.1	Architecture générale	23
4.2	Conception de la base de données	23
4.3	Conception du front-end (interface utilisateur)	25
4.4	Conception du back-end (logique serveur)	25
4.5	Organisation détaillée des fichiers et répertoires	26
5	Implémentation	27
5.1	Gestion des utilisateurs et des contacts	27
5.2	Module de messagerie individuelle	27
5.3	Gestion des groupes de messagerie	28
5.4	Partage de fichiers et pièces jointes	28
5.5	Tests et validations	29
6	Difficultés rencontrées	30
6.1	Compréhension initiale de PHP et MySQL	30
6.2	Difficultés liées à l'upload de fichiers	30
6.3	Gestion de l'authentification et sécurisation	31
6.4	Débogage et gestion des erreurs récurrentes	31
6.5	Difficulté à définir les fonctionnalités supplémentaires	32
6.6	Synchronisation et collaboration en binôme	32
6.7	Conclusion sur les difficultés rencontrées	32
7	Conclusion	33
8	Webographie	35
9	Annexes	36

1 Introduction

Dans le cadre du cours de Programmation Web et Introduction PHP, nous avons réalisé en binôme un projet ambitieux de développement d'un site web dynamique en PHP et SQL. Il s'agit d'un gestionnaire de contacts enrichi de fonctionnalités de messagerie, développé par deux étudiants débutants passionnés. Ce projet nous tenait à cœur car il représente notre première application web complète conçue avec engagement en tirant parti de l'assistance de l'intelligence artificielle pour nous guider et nous accompagner tout au long du développement.

L'objectif général était de créer une application web permettant à un utilisateur de gérer sa liste de contacts et d'échanger des messages de façon conviviale. Nous avons voulu aller au-delà d'un simple carnet d'adresses en intégrant des fonctionnalités avancées telles que l'envoi de messages (similaire à un webmail ou un mini réseau social), la création de groupes de discussion, le partage de fichiers multimédias et un système de blocage pour la confidentialité. L'interface utilisateur se devait d'être ludique et ergonomique afin de rendre l'expérience agréable et intuitive malgré la richesse fonctionnelle.

Travaillant à deux sur toutes les parties du projet sans division ni programmation des tâches, nous avons pu collaborer étroitement et apprendre ensemble. Cette approche impliquait une coordination constante mais elle nous a permis à chacun de maîtriser l'ensemble du système (front-end et back-end) et de partager équitablement les connaissances acquises. Nous avons parfois sollicité des outils d'IA pour nous guider lorsque nous étions bloqués, tout en veillant à bien comprendre et intégrer chaque solution proposée.

Ce rapport présente de manière structurée notre projet, de sa conception initiale jusqu'à son implémentation et aux difficultés rencontrées. Dans les sections qui suivent, nous décrivons le contexte et les objectifs, les fonctionnalités offertes, les choix techniques (langages, outils, architecture), puis nous détaillons la réalisation effective du site. Enfin, nous faisons le bilan de cette expérience formatrice dans la conclusion.

2 Présentation du projet

Le projet développé intitulé "Gestionnaire de contact" est une application web permettant à un utilisateur de gérer sa liste de contacts personnels et de communiquer avec eux. En somme, il combine les fonctionnalités d'un carnet d'adresses numérique avec celles d'une messagerie interne. Chaque utilisateur peut créer son compte, ajouter d'autres utilisateurs comme contacts (après acceptation) et échanger des messages individuels ou de groupe. Le tout depuis une interface web unique.

Concrètement, l'application offre une gestion avancée des contacts : un utilisateur peut ajouter un nouveau contact (par une demande d'ajout), supprimer un contact existant de sa liste ou encore bloquer un contact indésirable. À cela s'ajoute un module de messagerie complet : il est possible d'envoyer des messages privés à un contact avec un objet et un contenu, un peu comme un email. Nous avons également implémenté la messagerie de groupe où l'utilisateur peut créer des groupes de discussion et y inviter plusieurs contacts afin d'envoyer des messages visibles par tous les membres du groupe (similaire à une conversation de groupe).

Un aspect important de notre projet est le partage de fichiers. Les utilisateurs peuvent joindre des fichiers à leurs messages : qu'il s'agisse de photos ou images, de messages vocaux (audio) ou de documents (fichiers texte, PDF, etc.). L'application est capable de supporter ces pièces jointes et permet aux destinataires de les consulter ou de les télécharger. Cela enrichit les échanges au-delà du texte simple.

Enfin, fidèle au cahier des charges, nous avons accordé une attention particulière à l'interface utilisateur. Le design se veut ludique et ergonomique (simple à prendre en main, navigation intuitive). Notre public cible est tout utilisateur lambda d'un réseau social ou d'un webmail interne : il doit pouvoir utiliser l'application sans formation particulière. Ainsi, l'interface web comprend des icônes évocatrices, des couleurs agréables, un agencement clair des éléments (menus, listes de contacts, fenêtres de message, etc...) et globalement une expérience interactive sympathique.

En résumé, notre projet se positionne comme une petite plateforme de réseau social privé à échelle réduite : chaque utilisateur gère sa liste de contacts amis et peut communiquer aisément avec eux en privé ou en groupe, tout en ayant le contrôle (possibilité de blocage et de suppression). La réalisation de ce projet a été l'occasion de mettre en pratique de nombreuses notions apprises en Programmation Web, tout en développant un produit dont nous sommes fiers.

2.1 Objectifs du projet

- **Implémenter les fonctionnalités requises :** Implémenter les fonctionnalités requises : Réaliser un carnet de contacts amélioré intégrant toutes les fonctionnalités demandées (gestion des contacts, messagerie privée, messagerie de groupe,

partage de fichiers, blocage) de manière opérationnelle. Chaque fonctionnalité devait être aboutie et testée (envoi et réception de message effectivement fonctionnel, etc.).

- **Offrir une interface soignée et intuitive :** Concevoir une interface web agréable (esthétique, ludique) et facile d'utilisation (ergonomie) pour maximiser l'expérience utilisateur. L'objectif était de faire en sorte qu'un utilisateur comprenne naturellement comment ajouter un contact ou envoyer un message, grâce à une présentation claire.
- **Travailler en autonomie et en équipe :** Mener le projet intégralement à deux, sans découpage rigide des tâches, afin que chaque membre participe à tous les aspects (front-end, back-end, base de données). Cela visait à renforcer nos compétences globales en développement web et notre capacité à collaborer (pair programming, échanges continus).
- **Apprendre les technologies web :** Au-delà du résultat fonctionnel, notre objectif principal était d'apprendre et de maîtriser de nouvelles technologies. En particulier, ce projet devait nous faire pratiquer le langage PHP pour la programmation serveur, la gestion d'une base de données en SQL, et l'utilisation du trio HTML/CSS/JavaScript pour le front-end. Nous voulions également découvrir des bonnes pratiques de développement web avec l'architecture MVC, la sécurisation des mots de passe dans la base de données, etc...).
- **Assurer la persistance et la fiabilité des données :** Un objectif technique était de concevoir une base de données robuste stockant toutes les informations (utilisateurs, contacts, messages...) de façon cohérente en évitant les doublons et en respectant l'intégrité référentielle. De plus, nous devions veiller à la sécurité des données (p. ex., chiffrer les mots de passe, prévenir les injections SQL, protéger les sessions utilisateur).
- **Gérer les difficultés et mener le projet à terme :** Nous savions que compte tenu de notre niveau débutant, nous rencontrerions de nombreux défis (bugs, incompréhensions techniques, etc.). Un objectif implicite était donc de développer notre capacité de résolution de problèmes : rechercher des solutions (via la documentation, des forums, l'aide du professeur ou d'une IA), tester, déboguer patiemment et persévérer pour surmonter chaque obstacle. Finir le projet dans les délais impartis et avec toutes les fonctionnalités fonctionnelles a été un moteur tout au long du développement.

Finalement, ce projet visait non seulement à produire une application web répondant à un cahier des charges précis mais aussi à nous faire grandir en tant que développeurs. À la fin du projet, nous espérions avoir acquis une expérience concrète du développement d'une application web complète, du concept initial jusqu'au déploiement local, tout en ayant développé une application web utilisable démontrant nos compétences nouvellement acquises.

2.2 Fonctionnalités détaillées

Notre gestionnaire de contacts amélioré propose un ensemble riche de fonctionnalités.

Tout d'abord l'accès à l'ensemble des services de l'application nécessite la création d'un compte utilisateur sécurisé. Chaque utilisateur peut ainsi créer son compte personnel en utilisant un formulaire simple d'inscription accessible dès la page d'accueil. Durant l'inscription, l'utilisateur fournit des informations essentielles telles que son prénom, son nom, son adresse e-mail (qui sert d'identifiant unique pour se connecter) ainsi qu'un mot de passe sécurisé. Afin d'assurer une protection optimale des données personnelles, nous avons mis en place un système de chiffrement sécurisé des mots de passe en utilisant notamment la fonction `password_hash()` en PHP.

Une fois inscrit, l'utilisateur peut se connecter en utilisant ses identifiants via un formulaire d'authentification classique (e-mail et mot de passe). Après validation réussie de ces identifiants, une session utilisateur sécurisée est initialisée, permettant l'accès aux pages privées du site. L'utilisateur peut également modifier ses informations personnelles, par exemple changer son mot de passe, directement depuis son espace personnel, garantissant ainsi une gestion autonome et sécurisée de ses données.

Chaque utilisateur connecté dispose de sa propre liste de contacts, comparable à un carnet d'adresses personnel ou une liste d'amis. L'ajout d'un nouveau contact s'effectue par l'envoi d'une demande à l'utilisateur concerné. Concrètement, l'utilisateur saisit l'identifiant (adresse e-mail) de la personne qu'il souhaite ajouter, ce qui génère une notification envoyée au destinataire. Celui-ci peut alors choisir d'accepter ou de refuser la demande. En cas d'acceptation mutuelle, chacun voit l'autre apparaître dans sa propre liste de contacts, garantissant ainsi la sécurité et le consentement réciproque des utilisateurs. Une fois le contact ajouté, certaines informations de base (nom, prénom, e-mail, téléphone) deviennent accessibles dans la fiche de contact. Mais à l'inverse s'il refuse, le contact ne sera pas ajouté à la liste. Par la suite la suppression d'un contact est libre et peut être effectuée à tout moment. Lorsqu'un utilisateur retire quelqu'un de sa liste, la suppression prend effet immédiatement et entraîne automatiquement la suppression réciproque du contact.

Dans un souci de protection de la vie privée et de tranquillité des utilisateurs, un système de blocage de contacts a également été mis en place.

Lorsqu'un utilisateur décide de bloquer une personne, celle-ci ne peut plus lui envoyer de messages ni de nouvelles demandes d'ajout. L'action est simple, il suffit de sélectionner un contact indésirable et d'activer l'option **Bloquer**.

Techniquement, le statut de blocage est enregistré de manière unilatérale. Le contact bloqué reste éventuellement visible dans la liste (avec une mention "bloqué"). Mais il devient inactif, c'est-à-dire que tous ses messages ou demandes sont ignorés ou redirigés automatiquement vers un dossier spécifique (comme les messages indésirables). Ce mécanisme permet aux utilisateurs de se prémunir contre les communications non sollicitées, le spam ou le harcèlement. Le blocage est entièrement réversible afin que l'utilisateur puisse à tout moment débloquer un contact depuis les paramètres ou sa liste de contacts bloqués tout en gardant ainsi un contrôle total sur ses interactions.

Le cœur de l'application réside dans la messagerie. En effet l'application intègre une messagerie privée interne au cœur de laquelle se trouvent les échanges directs entre utilisateurs. Chaque message se compose d'un objet et d'un contenu (corps du message) à la manière d'un courrier électronique traditionnel. Depuis une interface de rédaction intuitive, l'utilisateur peut sélectionner un destinataire parmi sa liste de contacts, saisir l'objet du message, choisir de rédiger le

contenu, intégrer un fichier (PDF, images, video, ect...) et/ou enregistrer un message vocal, puis envoyer le tout en un clic.

Une fois le message envoyé, il apparaît immédiatement dans la boîte de réception du destinataire accompagné du nom de l'expéditeur, de l'objet et de la date d'envoi. De son côté, l'expéditeur peut consulter ce message dans sa boîte d'envoi (messages envoyés) afin de s'assurer d'un suivi clair des échanges.

Au-delà des messages privés entre utilisateurs, l'application propose une fonctionnalité de messagerie de groupe, permettant à plusieurs personnes d'échanger simultanément au sein d'un espace de discussion commun.

Un utilisateur peut créer un nouveau groupe de discussion en allant sur **Créer un Crew**. Ensuite il attribue un nom de groupe, sélectionne un ou plusieurs contacts parmi sa liste à inviter.

L'utilisateur à l'origine du groupe en devient automatiquement administrateur, ce qui lui confère des droits particuliers comme l'ajout ou la suppression de membres et la suppression complète du groupe

Pour enrichir les échanges entre utilisateurs, nous avons intégrée une fonctionnalité très appréciable qui est la possibilité de joindre des fichiers aux messages.

Lorsqu'un utilisateur rédige un message privé, il a l'option d'attacher un fichier. Les types de fichiers supportés incluent notamment : des images (photos au format JPG/PNG, etc...), des fichiers audio et vidéo (comme un enregistrement vocal ou des fichiers en MP3/MP4), et des documents bureautiques (PDF, DOCX, etc...).

Par exemple, on peut envoyer une photo en pièce jointe d'un message et/ou enregistrer un message vocal et l'envoyer. L'interface propose un bouton "Joindre un fichier" qui ouvre le sélecteur de fichier du navigateur. L'utilisateur choisit le fichier souhaité sur son appareil et celui-ci est téléchargé vers le serveur lors de l'envoi du message. Côté réception, les messages affichent les pièces jointes de manière appropriée : ainsi, si c'est une image, elle peut être directement affichée sous le texte du message. Si c'est un audio, un petit lecteur audio avec bouton "Lecture" est proposé. Si c'est un document, un lien de téléchargement s'affiche. Nous avons mis en place des limitations pour assurer le bon fonctionnement, par exemple une taille maximale pour les fichiers (afin d'éviter de saturer le serveur, nous avons limité à quelques Mo par fichier) et un contrôle du type de fichier (MIME type ou extension) pour éviter tout fichier potentiellement dangereux (le serveur refuse par exemple un script inconnu).

Le partage de fichiers enrichit considérablement la communication. Ainsi les utilisateurs peuvent échanger des photos souvenirs, envoyer des notes vocales comme dans une messagerie moderne ou partager un document PDF dans un groupe de travail. Le tout au sein même de l'application.

Au-delà des aspects fonctionnels, l'une des forces majeures de l'application réside dans la qualité de son interface utilisateur.

En effet dès la conception, une attention toute particulière a été portée à l'interface utilisateur de notre gestionnaire de contacts. Sa présentation professionnelle mais également chaleureuse et ludique pour offre aux utilisateurs une expérience à la fois conviviale et intuitive. Cela se traduit notamment par une charte graphique cohérente, des couleurs harmonieuses, des icônes intuitives et quelques animations discrètes pour fluidifier l'utilisation quotidienne. L'ergonomie a été spécialement étudiée pour réduire au maximum les clics nécessaires lors des interactions fréquentes : par exemple, écrire un

message ou enregistrer un message vocal ne nécessite que très peu de manipulations depuis la fiche d'un contact.

De plus, l'application a été pensée pour être responsive garantissant ainsi une expérience utilisateur fluide et confortable quel que soit le support utilisé (ordinateur, tablette ou smartphone).

Toutes ces fonctionnalités ont été développées et intégrées de manière cohérente au sein de l'application. Dans les sections suivantes, nous décrivons les aspects techniques et les choix de conception qui nous ont permis de réaliser ce panel de fonctionnalités.

2.3 Technologies utilisées

Pour mener à bien ce projet, nous avons fait appel à plusieurs technologies web complémentaires, conformes aux standards actuels du développement web. Voici les principaux langages, outils et bibliothèques que nous avons utilisés, accompagnés de leur rôle dans le projet :

PHP : Le langage PHP a été utilisé pour tout le développement back-end de l'application. Elle nous a permis d'implémenter la logique serveur avec le traitement des formulaires (ajout de contact, envoi de message, etc...), l'interactions avec la base de données MySQL, la gestion des sessions utilisateurs (authentification), etc... Nous avons choisi PHP car il est bien adapté aux applications web classiques et était l'un des langages enseignés dans le cadre du cours. La version utilisée est PHP 8+, nous donnant accès aux fonctionnalités modernes du langage (comme le typage amélioré, `password_hash()` pour le hachage de mots de passe).

SQL : Nous avons créé une base de données en SQL via l'application PHPMyAdmin (MAMP) pour stocker toutes les données pertinentes de l'application. SQL a servi à créer les tables pour les utilisateurs, contacts, messages, groupes, etc... et à exécuter les requêtes SQL (INSERT, SELECT, UPDATE, DELETE) nécessaires. Le langage SQL a l'avantage d'être robuste et d'offrir un bon support des contraintes d'intégrité référentielles (clés étrangères), ce qui nous a aidés à structurer correctement les relations entre entités (par exemple, lier un message à son expéditeur et son destinataire, ou lier un contact à deux utilisateurs concernés). Nous avons souvent manipulé SQL via PHP (en utilisant l'extension MySQLi/PDO et des requêtes préparées).

Serveur Apache (en local) : L'application a été développée et testée en environnement local avec un serveur Apache (fourni via l'installation de MAMP). Apache a servi le PHP et les pages web durant le développement. Nous avons configuré le serveur localement pour accueillir notre projet (`localhost:8888/projet_php/`). Apache gère aussi la partie routage des URL (nous avons utilisé des fichiers .htaccess pour rediriger certaines requêtes ou pour sécuriser l'accès à certains répertoires comme les fichiers upload). Le choix d'Apache s'est imposé car c'est le serveur web couramment associé à PHP dans les suites type MAMP.

HTML5/CSS3 : Pour le front-end (côté client), nous avons utilisé les langages standards HTML et CSS. HTML5 nous a servi à structurer le contenu des pages (for-

mulaires, tableaux de contacts, sections de navigation, etc.) en tirant parti des éléments sémantiques appropriés (par exemple <header>, <nav>, <section>, <article> pour la boîte de réception, etc...).

CSS3 a été employé pour mettre en forme l'interface de manière attrayante avec le choix des couleurs, des polices, la mise en page en grille ou flexbox, les styles des boutons et des champs, icônes (via fonts ou images CSS) et adaptations responsive. Nous avons veillé à la compatibilité cross-browser en restant dans les spécifications CSS3 largement supportées.

JavaScript : De plus pour le font-end, nous avons également intégré du JavaScript pour améliorer l'interactivité de l'interface. Sans framework lourd, nous avons utilisé du JavaScript pur pour quelques fonctionnalités comme la validation de formulaires en temps réel (vérifier que le champ email est bien formé avant soumission), des effets dynamiques (comme afficher/masquer la fenêtre modale de création d'un groupe ou de composition d'un message), et le rafraîchissement partiel de certaines listes sans recharger toute la page (via des requêtes AJAX basiques pour vérifier les nouvelles demandes d'amis ou messages).

Framework CSS Bootstrap (version 5) : Pour accélérer le développement de l'interface et assurer une base responsive solide, nous avons fait appel au framework CSS Bootstrap. Son système de grille nous a aidés à créer une mise en page bien structurée sans effort (par exemple, diviser l'écran en une colonne menu latéral et une colonne contenu principal). Bootstrap nous a fourni des composants prêts à l'emploi et stylés de manière homogène : des boutons, des formulaires, des modales (fenêtres modales utilisées pour certaines fenêtres contextuelles), des badges de notification, etc. Nous avons personnalisé légèrement le thème (couleurs principales adaptées à notre charte, etc.) pour conserver le côté ludique. L'utilisation de Bootstrap a grandement contribué à l'ergonomie et au design responsive du site, tout en nous faisant gagner du temps sur la partie CSS.

FontAwesome : Pour les icônes illustratives dans l'interface (icône de contact, icône de message, icône de groupe, bouton supprimer, bloquer, etc.), nous avons utilisé la bibliothèque d'icônes FontAwesome. Intégrée via CDN, elle nous a offert une large gamme de pictogrammes vectoriels faciles à utiliser en HTML. Ces icônes renforcent l'aspect visuel ludique et aident à l'intuitivité (ex : une corbeille pour supprimer).

Outils d'assistance au développement : En dehors du code en lui-même, nous avons utilisé certains outils pour faciliter le développement. L'éditeur de code principal a été Visual Studio Code, avec lequel nous avons écrit nos fichiers PHP, HTML, CSS et JS. Ses fonctionnalités (coloration syntaxique, autocomplétion, debug basique) nous ont fait gagner du temps. Nous avons également utilisé l'outil Git en local pour le versioning de notre code (principalement afin de pouvoir revenir en arrière en cas d'erreur, et pour archiver les différentes étapes du projet).

Enfin, comme mentionné, nous avons ponctuellement fait appel à des IA comme assistant virtuel tel que ChatGPT (OpenAI) pour obtenir de l'aide sur des problèmes de code spécifiques ou des conseils sur les bonnes pratiques (par exemple, comment sécuriser un formulaire de login en PHP, comment structurer une table pour les messages de groupe, etc...) et Grok pour l'interface. L'IA a été un outil parmi d'autres et nous avons toujours validé manuellement les solutions proposées avant de les intégrer.

2.4 Structure de l'application

Avant de plonger dans les détails de conception et d'implémentation, il est utile de présenter la structure globale de l'application web, c'est-à-dire l'organisation des fichiers et des principales pages et sections côté serveur. Cette structuration a été pensée dès le départ comme un facteur de réussite du projet. Un site bien organisé est plus facile à maintenir, à faire évoluer, et à sécuriser.

Notre projet s'articule donc autour de plusieurs modules fonctionnels, chacun correspondant à une page ou un groupe de fichiers dédiés à un aspect spécifique de l'application. Les répertoires sont répartis selon leur rôle (interface, logique serveur, ressources statiques, etc.) et nous avons fait un usage régulier de fichiers inclus (include) afin de mutualiser les composants réutilisables. Voici une présentation détaillée de l'architecture de notre application classée par thématiques fonctionnelles.

L'authentification des utilisateurs constitue le point d'entrée dans notre application. Deux fichiers en particulier structurent cette partie : **login.php** et **register.php**. Le fichier **login.php** est la première page visible par tout utilisateur non connecté. Elle contient un formulaire de connexion classique dans lequel l'utilisateur renseigne son adresse e-mail et son mot de passe. Après soumission, les identifiants sont comparés à ceux présents en base de données. En cas de succès, une session PHP est démarrée avec la commande **session_start()**, stockant l'identifiant utilisateur en mémoire ce qui permet de personnaliser l'expérience de navigation par la suite.

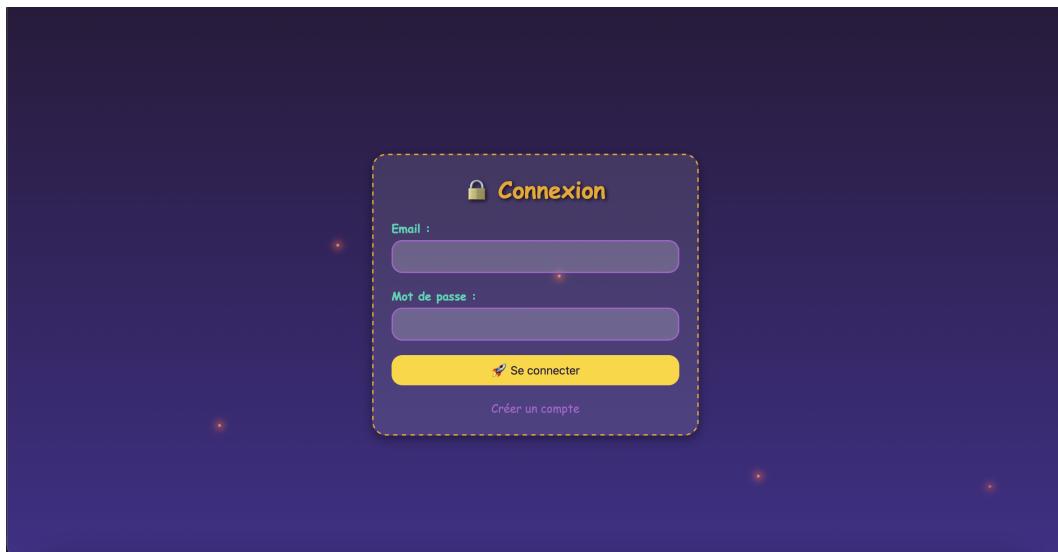


FIGURE 1 – Interface de connexion de l'application

Le fichier **register.php**, quant à lui, permet la création d'un nouveau compte utilisateur. L'inscription passe par un formulaire recueillant des informations personnelles de base (prénom, nom, email, mot de passe), et les données sont vérifiées (unicité de l'email, solidité du mot de passe, validation des champs). Le mot de passe est haché avant insertion en base pour des raisons de sécurité, à l'aide de la fonction **password_hash()** de PHP. Ces deux pages sont essentielles et leur robustesse conditionne l'accès à toutes les autres parties du site.

FIGURE 2 – Interface d’inscription de l’application

□ Éditer Copier Supprimer 10 Un Deux trois@gmail.com \$2y\$10\$UHMinM3YxVLLeZZis6vAMcuJu6EewmEnSr0NBvG9OVYh... 2000-01-01 Homme 2025-04-28 15:03:27

FIGURE 3 – Utilisateur dans la base de données

Une fois connecté, l’utilisateur est redirigé vers **dashboard.php**. Cette page centrale joue le rôle de tableau de bord. Elle donne un aperçu global des fonctionnalités de l’application avec des accès rapides vers les contacts, la messagerie, les groupes, les fichiers reçus ou envoyés, et les paramètres personnels. C’est également depuis cette page que sont affichées certaines notifications dynamiques (par exemple : une demande de contact en attente, ou un nouveau message reçu).



FIGURE 4 – Tableau de bord de l’application

La gestion des contacts représente un cœur fonctionnel de notre application. Cette logique est répartie sur plusieurs fichiers complémentaires.
contacts.php affiche la liste complète des contacts validés d’un utilisateur. Chaque contact est représenté sous forme de carte ou de ligne avec ses informations clés (nom,

prénom, email) et accompagnée de boutons d'action tels que **Envoyer un message**, **Bloquer** ou **Appeler**.

Les pages **demande.php** et **ajouter.php** permettent respectivement de consulter les demandes de contact reçues et d'en initier de nouvelles. Lorsqu'un utilisateur souhaite ajouter un contact, il saisit son adresse e-mail dans **ajouter.php**, ce qui génère une notification de demande à destination du contact. Ce dernier peut alors accepter ou refuser la demande via **demande.php**. En cas d'acceptation, le lien est alors inscrit en base de données de façon réciproque, permettant les échanges.

La suppression d'un contact est une action unilatérale (elle supprime la relation de l'utilisateur initiateur), et peut être réalisée depuis **contacts.php**.



FIGURE 5 – Ajout d'un ami

Éditer Copier Supprimer 32 10 4 2025-04-28 15:20:37

FIGURE 6 – Demande dans la base de données

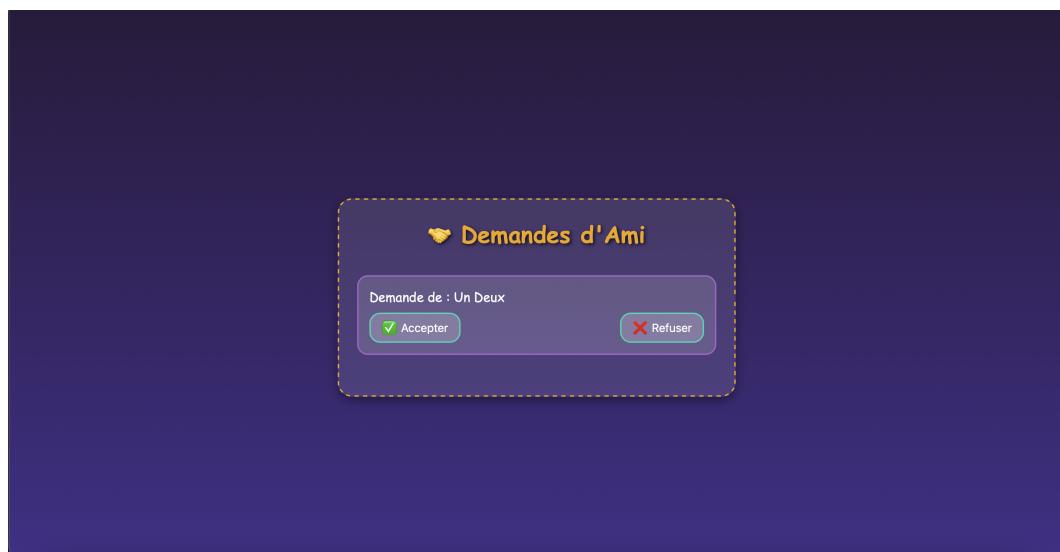


FIGURE 7 – Demandes en ami

Pour préserver la sécurité et le confort des utilisateurs, un système de blocage a été implémenté. Il est accessible via la page **bloque.php**. Celle-ci regroupe la liste des utilisateurs bloqués, tout en offrant la possibilité de débloquer certains profils si nécessaire.

Le blocage est enregistré dans une table dédiée de la base de données et agit comme un filtre dans toutes les interactions. Lorsqu'un utilisateur est bloqué, il ne peut ni envoyer de message, ni ajouter l'autre en contact. Des vérifications sont effectuées à chaque chargement de page sensible pour faire respecter ce contrôle d'accès.

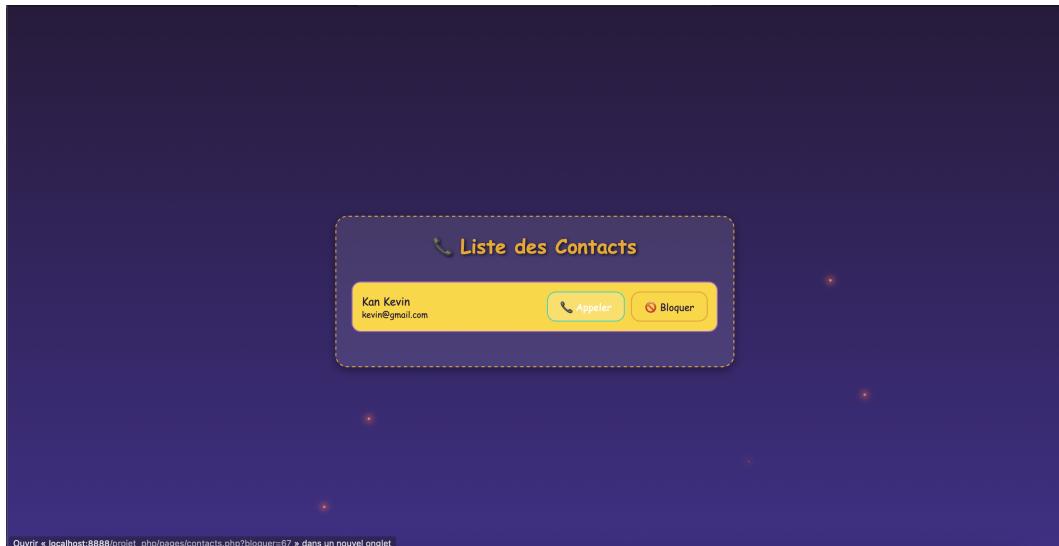


FIGURE 8 – Blocage d'un ami

	← T →	▼	id	utilisateur_id	utilisateur_bloque
<input type="checkbox"/>	Éditer	Copier	Supprimer	12	10

FIGURE 9 – Bloqués dans la base de données

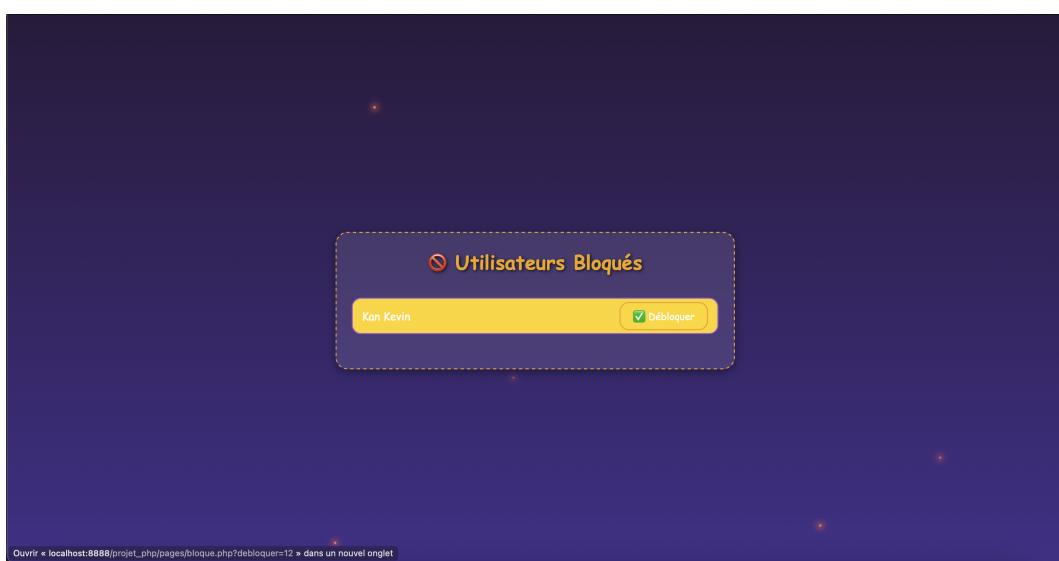


FIGURE 10 – Déblocage d'un ami

L'une des fonctionnalités principales de l'application repose sur une messagerie interne entre utilisateurs. Ce système est réparti sur plusieurs fichiers.

La page **envoyer_message.php** est la page de sélection du contact à qui envoyer un message, elle permet de choisir un destinataire. Puis **message.php** permet de saisir un objet, un corps de message, et éventuellement de joindre un fichier et/ou un audio. Cette page intègre également le système d'upload (téléchargement).



FIGURE 11 – Sélection du destinataire

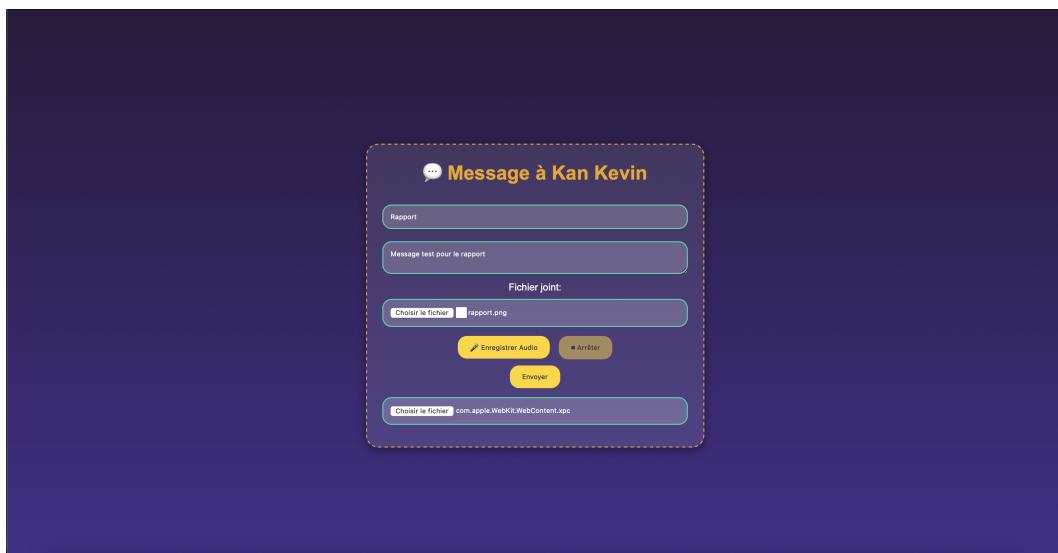


FIGURE 12 – Rédaction du message

messages_envoyes.php répertorie tous les messages envoyés par l'utilisateur courant avec la date, le destinataire et l'objet. Il est également possible de consulter et modifier chaque message individuellement.

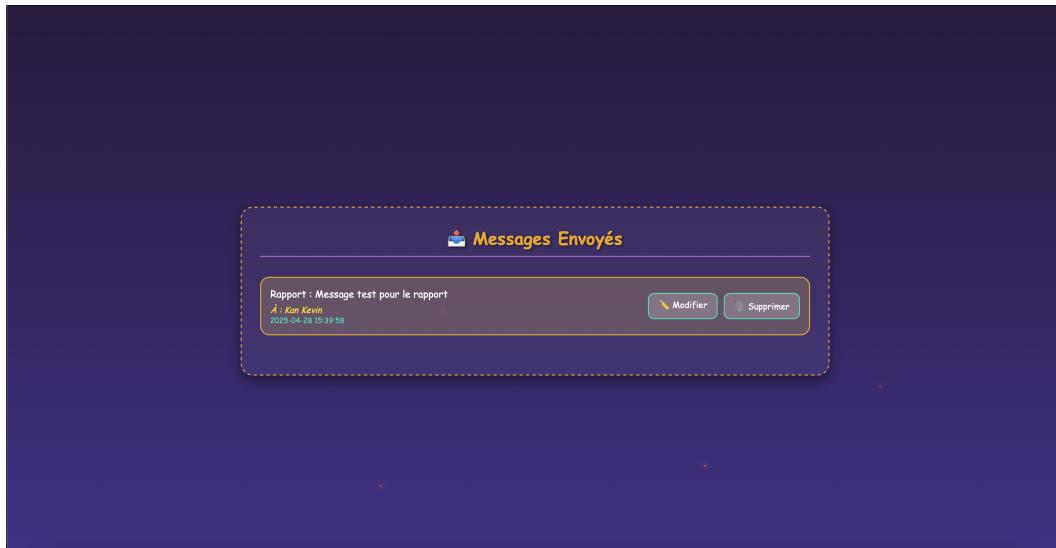


FIGURE 13 – Messages envoyés

message_detail.php est la page d'affichage complète d'un message reçu. Elle contient toutes les métadonnées (expéditeur, destinataire, date, objet, message texte et pièce jointe) et permet d'y répondre.



FIGURE 14 – Boîte de réception

Chaque message est enregistré en base de données et des vérifications de relation entre les utilisateurs sont effectuées pour éviter les abus.

Éditer Copier Supprimer 67 10 69 Message test pour le rapport 2025-04-28 15:39:58 Rapport /Applications/MAMP/htdocs/projet_php/pages/.. /fich... /Applications/MAMP/htdocs/projet_php/pages/.. /fich...

FIGURE 15 – Message enregistré dans la base de données

Afin de permettre les échanges à plusieurs, nous avons développé un module de groupes de discussion, accessible via **mes_groupes.php** et **cree_groupe.php**.

La page **cree_groupe.php** permet à un utilisateur de créer un nouveau groupe en lui attribuant un nom et en y ajoutant des membres sélectionnés parmi ses contacts. L'utilisateur devient alors administrateur du groupe, avec un pouvoir de gestion (ajout ou retrait de membres, suppression du groupe, etc...).

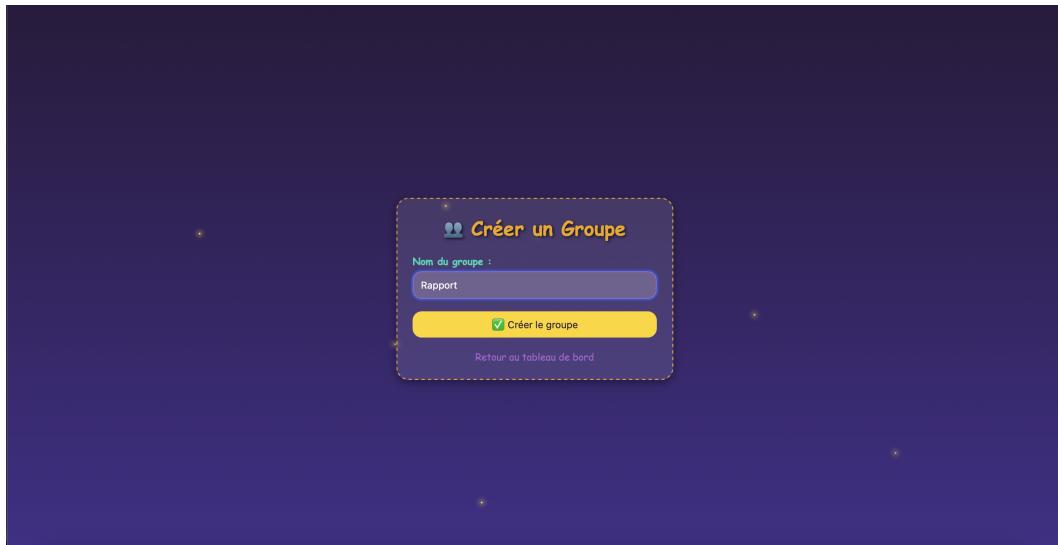


FIGURE 16 – Crédit à la création du groupe

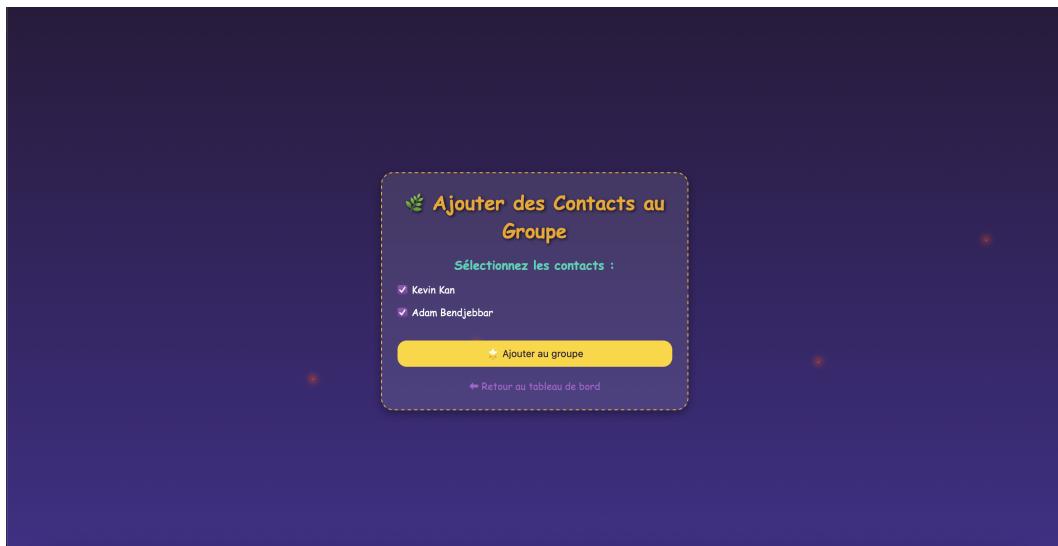


FIGURE 17 – Ajout des membres du groupe avant création

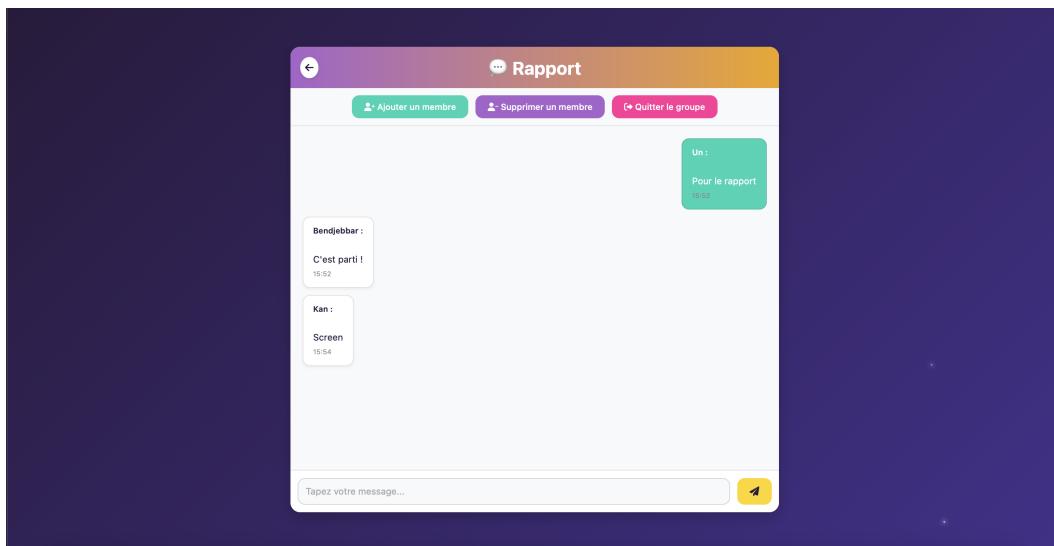


FIGURE 18 – Messages dans le groupe

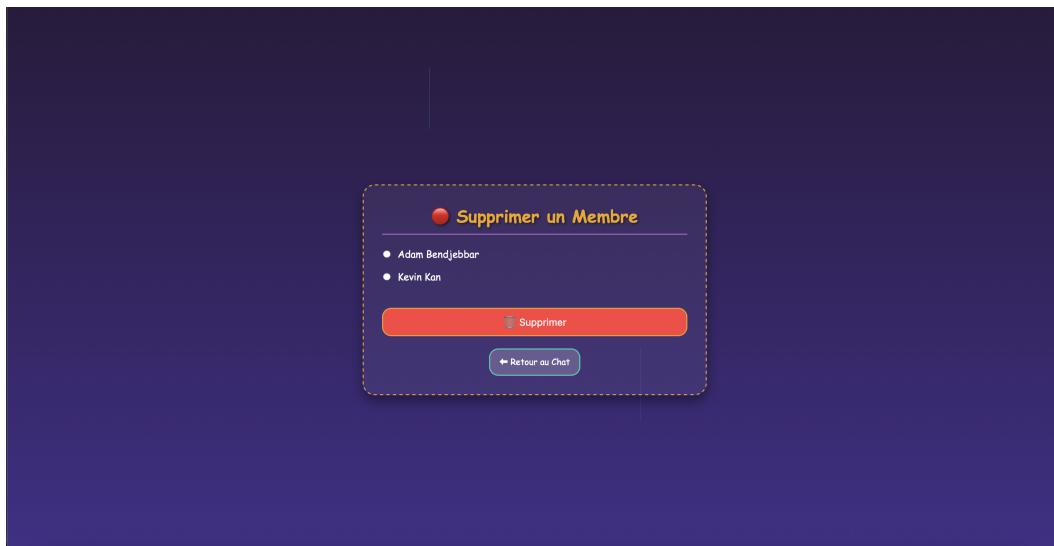


FIGURE 19 – Suppression membre du groupe

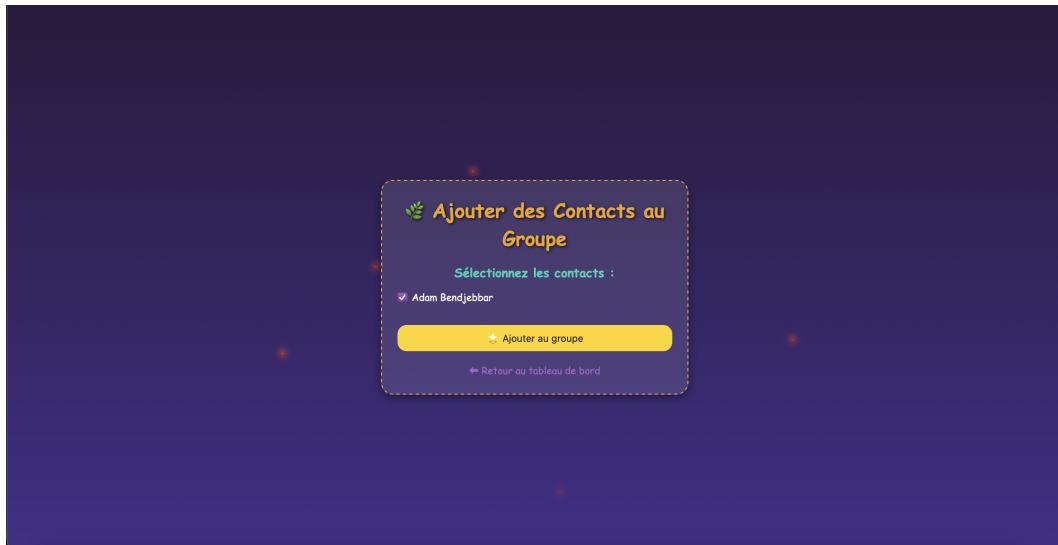


FIGURE 20 – Ajout membre du groupe après création

La page **mes_groupes.php** liste tous les groupes dont l'utilisateur est membre. Chaque groupe ouvre vers un fil de discussion au format chat dans lequel les utilisateurs peuvent envoyer des messages simples.



FIGURE 21 – L'ensemble des groupes où est l'utilisateur

Ce système de groupes reproduit une logique communautaire et collaborative, particulièrement utile pour les échanges en contexte scolaire, familial ou professionnel.

L'une des fonctionnalités qui enrichit considérablement l'expérience utilisateur est la possibilité d'envoyer des fichiers en pièce jointe.

Le système repose sur un script serveur qui vérifie :

- Le type MIME du fichier (image, audio, document bureautique autorisé)
- La taille maximale autorisée (quelques Mo pour éviter les abus)
- L'absence de code malveillant via des vérifications sur l'extension.
- Les fichiers sont enregistrés dans un dossier `textbf{fuploads}`. Des dossiers tel que `fichier_vocal` et `img`, gèrent toute la logique d'upload, de stockage et de sécurisation des fichiers.

En réception, les pièces jointes sont affichées de façon intuitive : petites d'images, lecteur audio intégré pour les vox ou un lien de téléchargement, des liens de téléchargement pour les PDF ou autres documents.

Enfin, nous avons structuré notre projet autour de composants réutilisables centralisés dans un dossier `includes` qui inclut :

database.php qui centralise les informations de connexion à la base de données et les variables globales.

```
1  <?php
2  $host = "localhost";
3  $dbname = "ad"; // NOM DE LA BASE DE DONNÉES
4  $username = "KARIM"; // NOM D'UTILISATEUR
5  $password = "test"; // MOT DE PASSE
6
7  try {
8      $pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username, $password);
9      $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
10     //echo "✓ Connexion réussie à la base de données !"; // Test de connexion
11 } catch (PDOException $e) {
12     die("✗ Erreur de connexion : " . $e->getMessage());
13 }
14 ?>
```

FIGURE 22 – database.php

Cette approche modulaire garantit une meilleure maintenance et nous a permis de travailler progressivement, en ajoutant chaque fonctionnalité indépendamment tout en assurant la cohérence du tout.

En résumé, l'application est structurée en plusieurs pages PHP dédiées aux grandes fonctionnalités (contacts, messages, groupes) reliées entre elles par un menu de navigation. Des dossiers séparés contiennent les ressources partagées (includes pour les composants communs, assets pour les fichiers statiques, uploads pour les fichiers utilisateurs). Cette organisation claire nous a aidés à développer en parallèle différentes parties tout en gardant une architecture logique.

3 Environnement de travail

Le développement de notre projet s'est déroulé dans un environnement local maîtrisé, pensé pour garantir à la fois souplesse, rapidité de test et confort de travail. Travaillant exclusivement à deux, nous avons organisé notre collaboration de manière simple mais efficace en nous appuyant sur des outils modernes, adaptés à notre niveau et à nos objectifs.

Nous avons tous deux travaillé sur des ordinateurs personnels sous macOS. Ce qui nous a permis d'uniformiser totalement notre environnement technique. Pour simuler un serveur web en local, nous avons utilisé MAMP, un outil simple et efficace qui regroupe Apache, PHP 8 et MySQL 8 dans un même package. Grâce à MAMP, nous avons permis faire fonctionner notre application sur localhost de façon fluide pendant tout le développement. L'exécution locale offrait plusieurs avantages : un accès immédiat aux fichiers, un recharge rapide des pages et aucune dépendance à une connexion internet ou à un serveur distant. Cela nous a permis de tester chaque modification en temps réel en toute autonomie.

Pour l'édition du code, nous avons opté pour Visual Studio Code (VS Code), un éditeur moderne, léger et puissant. Il s'est avéré particulièrement adapté à notre projet qui combinait plusieurs technologies (PHP, HTML, CSS, JavaScript). Nous avons installé plusieurs extensions utiles comme PHP IntelliSense pour bénéficier de l'autocomplétion et des suggestions de code intelligentes ou encore Live Server pour le rechargement automatique du front-end lors des modifications HTML/CSS. L'interface de VS Code nous a permis de travailler de manière lisible et ordonnée. En mettant en valeur la structure du projet grâce à l'arborescence intégrée et aux outils de recherche rapide. Cette cohérence visuelle a facilité notre travail en binôme, même lorsque nous étions sur des fichiers complexes ou interconnectés.

Ne disposant pas d'un serveur Git collaboratif ou d'un dépôt distant initialement, nous avons adopté une méthode de travail centrée sur la communication constante et le partage de fichiers localement. Une bonne partie du projet a été développée en pair programming, c'est-à-dire à deux sur le même poste, pour coder ensemble une fonctionnalité. Cela nous a permis de confronter nos idées en temps réel, de repérer plus rapidement les erreurs et d'avancer efficacement sur les points techniques délicats.

Lorsque nous travaillions séparément, nous nous répartissions les tâches de manière souple (par exemple : l'un sur le back-end d'une fonctionnalité, l'autre sur la mise en page ou l'interface). Nous synchronisions régulièrement nos avancées en échangeant des archives compressées du projet ou via un dépôt Git local partagé ponctuellement sur GitHub, ce qui nous a permis de garder une trace des versions et d'éviter de perdre du code. Avant chaque fusion, nous nous assurions que les modifications s'intégraient sans conflit, en les testant immédiatement en local.

Cette organisation souple mais structurée nous a permis d'alterner phases de travail individuel et collaboration étroite, tout en nous obligeant à lire, comprendre et respecter le code de l'autre, ce qui a été très formateur.

Tous les tests ont été effectués directement en local sur le serveur Apache fourni par

MAMP. Nous avons utilisé notre base de données en SQL pour injecter des données de test, et créé plusieurs comptes fictifs à nos noms, du type adambendjebbar@gmail.com ou kevin@gmail.com afin de simuler de vraies interactions utilisateurs.

Ces comptes nous ont permis de valider l'ensemble des fonctionnalités de l'application dans des cas concrets. L'envoi et réception de messages, l'acceptation ou refus de demandes de contact, le blocage et déblocage d'un utilisateur, la création d'un groupe de messagerie avec plusieurs membres, les test d'affichage des pièces jointes (images, vocaux, documents), etc...

À chaque nouvelle fonctionnalité implémentée, nous vérifiions son bon fonctionnement en effectuant plusieurs manipulations croisées depuis deux navigateurs ou deux sessions ouvertes.

Nous avons également testé l'interface sur plusieurs navigateurs (Safari, Chrome, Firefox) afin de nous assurer de sa compatibilité et du bon rendu CSS, quelle que soit la plateforme.

En fin de projet, pour simuler un déploiement réel et permettre une démonstration dynamique, nous avons transféré les fichiers sur un petit serveur distant de test. Nous avons utilisé FileZilla pour le transfert des fichiers, en adaptant les chemins et paramètres de connexion MySQL à l'environnement du serveur. Cette étape nous a permis de vérifier que notre application fonctionnait bien hors de l'environnement local et en conditions proches d'un usage réel multi-utilisateur. C'était aussi l'occasion de valider la portabilité de notre code, un aspect important dans tout projet web.

Tout au long du projet, nous nous réunissions régulièrement souvent de manière informelle afin de faire un point d'étape et discuter des prochaines actions. Cette communication continue a été essentielle car nos tâches étaient souvent interconnectées. Il était donc crucial que chacun sache ce que l'autre faisait et d'éviter les redondances ou incohérences de logique entre les fichiers.

En tant que débutants, nous avons souvent rencontré des blocages techniques ou des erreurs difficiles à comprendre. Dans ces cas, nous avons d'abord consulté les ressources officielles : la documentation de PHP sur w3schools.com pour les fonctions serveur, dev.mysql.com pour la structure SQL, et les forums comme Stack Overflow pour résoudre des cas concrets. Lorsque cela ne suffisait pas, nous avons eu recours à l'intelligence artificielle, principalement ChatGPT et Grok, pour obtenir des explications pédagogiques ou des suggestions de code. Cette aide a été précieuse notamment pour comprendre certaines erreurs PHP subtiles ou pour valider nos choix techniques. Bien entendu, toutes les réponses générées par l'IA ont été vérifiées, adaptées et testées par nous-mêmes. Enfin, M. Thibault ANANI, notre enseignant encadrant, nous a aussi soutenus à plusieurs reprises lors des séances de travaux pratiques, en nous orientant sans jamais nous donner directement la solution. Cela a renforcé notre autonomie et notre capacité à apprendre par nous-mêmes.

En définitive, notre environnement de travail nous a permis de développer dans de bonnes conditions à la fois techniques et humaines. Le choix du local via MAMP, l'usage d'outils professionnels comme VS Code, la bonne entente dans notre binôme et l'organisation rigoureuse que nous avons su mettre en place ont grandement facilité le bon déroulement du projet. Cette expérience nous a permis de découvrir des méthodes de travail proches du monde professionnel, tout en consolidant nos bases en développement web collaboratif.

4 Conception du projet

La phase de conception a été déterminante pour structurer nos idées, définir clairement l'organisation technique du projet et établir une base solide avant de commencer le développement proprement dit. Nous avons articulé cette phase essentielle autour de plusieurs axes majeurs que nous détaillons ci-dessous en commençant avec l'architecture générale de l'application puis la conception minutieuse de la base de données et de l'interface utilisateur Enfin la logique serveur du back-end.

4.1 Architecture générale

Pour notre application web, nous avons retenu une architecture classique organisée en plusieurs couches distinctes. Le front-end pour la gestion de l'interface utilisateur, le back-end PHP pour la logique applicative côté serveur et une couche de stockage représentée par notre base de données SQL.

Cette séparation nette des responsabilités nous a permis de répartir efficacement les tâches tout en assurant une cohérence globale de l'application. Concrètement, l'utilisateur interagit avec le front-end (HTML/CSS/JavaScript), lequel envoie des requêtes HTTP vers les scripts PHP du serveur. Ces scripts traitent ensuite les requêtes reçues en interrogeant la base de données pour récupérer ou modifier les données puis génèrent dynamiquement des réponses HTML qui sont renvoyées au navigateur pour affichage. Cette architecture simple et efficace nous a semblé parfaitement adaptée à un projet étudiant de cette envergure.

4.2 Conception de la base de données

La base de données occupe une place centrale dans notre application. En effet, elle gère toutes les informations essentielles aux échanges entre utilisateurs. Nous avons opté pour un modèle relationnel rigoureux et clairement défini à travers plusieurs tables interconnectées.

La table **Utilisateurs** constitue la base du système. Elle stocke les informations personnelles des utilisateurs (identifiant unique, nom, prénom, email sécurisé, mot de passe crypté, etc.). À partir de cette table, nous avons dérivé plusieurs autres tables pour gérer spécifiquement les relations et les interactions entre les utilisateurs.

La table **Contacts** permet de gérer les relations entre utilisateurs sous forme de contacts personnels validés mutuellement. Chaque entrée indique clairement qui est en relation avec qui, permettant d'établir simplement des listes d'amis ou de contacts professionnels.

Pour gérer efficacement les demandes d'ajout de contacts, nous avons prévu une table spécifique nommée **Demandes_ami**. Elle stocke temporairement les demandes en

attente d'approbation, ce qui garantit que les contacts soient toujours validés explicitement par les deux parties concernées avant d'être confirmés dans la table principale **Contacts**.

La gestion des messages échangés entre utilisateurs est prise en charge par la table **Messages** qui contient des informations précises comme l'expéditeur, le destinataire, l'objet du message, le contenu texte, ainsi que des pièces jointes éventuelles (images, fichiers audio ou documents). Nous avons choisi une structure qui permet de stocker simplement le chemin relatif des fichiers attachés, facilitant ainsi leur gestion sur le serveur.

Pour permettre aux utilisateurs de créer des conversations collectives, nous avons conçu la table **Groupes** associée à une table **Messages_groupe**. La table **Groupes** enregistre les informations principales de chaque groupe (nom, créateur, date de création), tandis que **Messages_groupe** gère les échanges textuels internes à ces groupes. Enfin, la table associative **Membres_groupe** relie les utilisateurs à leurs groupes respectifs, tout en précisant le rôle de chacun (administrateur ou membre standard).

Nous avons également inclus une table **Bloque** permettant à un utilisateur de bloquer les communications indésirables provenant d'autres utilisateurs. Ce mécanisme est simple et efficace pour protéger les utilisateurs contre les messages non sollicités ou le harcèlement potentiel.



FIGURE 23 – L'ensmeble des tables de la base de données

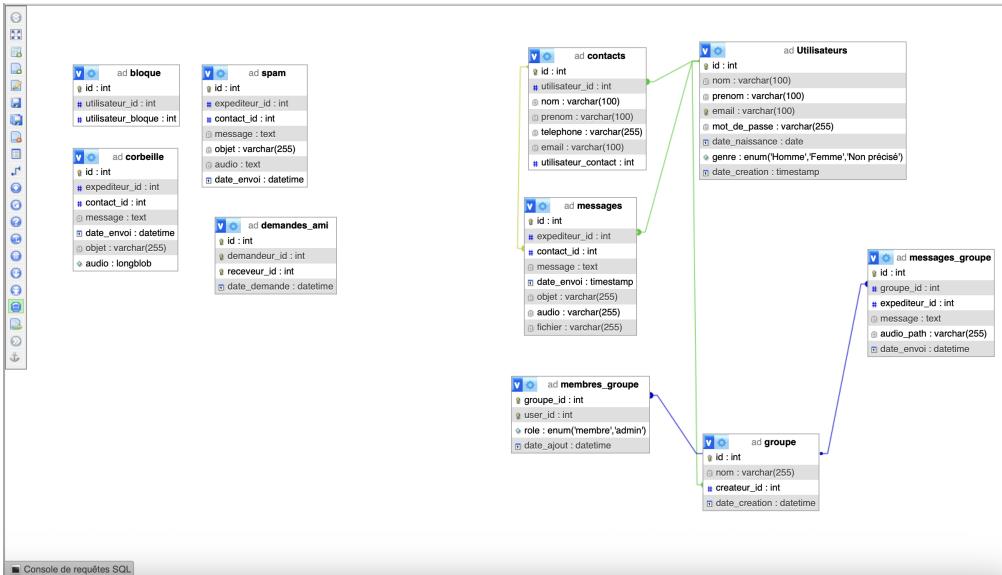


FIGURE 24 – Schéma de la base de données

Toutes ces tables sont interconnectées grâce à des clés étrangères clairement définies, ce qui garantit l'intégrité constante des données et facilite la gestion automatique des opérations complexes telles que la suppression en cascade d'informations liées.

4.3 Conception du front-end (interface utilisateur)

L'interface utilisateur représente un aspect fondamental du projet car elle conditionne directement l'expérience et le confort d'utilisation. Dès le départ, nous avons opté pour une ergonomie claire, intuitive et conviviale. Pour atteindre cet objectif, nous avons défini plusieurs principes clés d'interface : simplicité, lisibilité et cohérence.

Nous avons conçu une présentation visuelle unifiée à travers l'utilisation intensive de composants réutilisables en PHP, tels que le **header** (**includes/header.php**) et le **footer** (**includes/footer.php**), assurant ainsi une homogénéité graphique et fonctionnelle sur toutes les pages.

Chaque page importante, comme les pages de contacts, messages ou groupes, présente une mise en page structurée avec un menu latéral fixe permettant un accès rapide aux différentes sections du site.

Nous avons également soigné l'affichage des informations critiques comme les listes de contacts, les messages reçus ou les groupes créés par les utilisateurs. Chaque élément est clairement identifié et des actions simples comme l'ajout, la suppression ou l'envoi d'un message sont immédiatement accessibles grâce à des boutons ou icônes explicites. Ce choix de conception garantit que chaque utilisateur, même débutant, puisse rapidement comprendre et utiliser aisément l'ensemble des fonctionnalités offertes par l'application.

4.4 Conception du back-end (logique serveur)

La logique applicative côté serveur a été minutieusement pensée et structurée afin de traiter efficacement toutes les interactions utilisateur. Chaque script PHP joue un rôle

précis, en traitant des requêtes spécifiques liées à une fonctionnalité donnée. Nous avons mis en place une gestion rigoureuse des formulaires avec validation systématique des données reçues (format, taille, cohérence), afin de sécuriser totalement les interactions utilisateur contre des injections SQL, des attaques XSS ou autres abus potentiels.

Concernant la gestion des fichiers téléchargés par les utilisateurs (pièces jointes), nous avons conçu une logique stricte de validation des fichiers, notamment en termes de taille maximale autorisée et de types MIME acceptés (JPEG, PNG, MP3, PDF, etc.). Une fois validés, les fichiers sont renommés systématiquement afin d'éviter les collisions de noms et sont stockés de manière sécurisée sur le serveur, leur chemin d'accès étant enregistré en base pour permettre une récupération facile ultérieurement.

La sécurité globale de l'application est renforcée par une gestion centralisée des sessions PHP. Chaque accès à une page privée requiert la vérification préalable d'une session utilisateur valide. Cette approche garantit que seules les personnes authentifiées puissent accéder aux ressources personnelles ou sensibles.

Finalement, pour optimiser les performances et la réactivité de notre application, nous avons soigneusement conçu les requêtes SQL exécutées côté serveur en minimisant leur nombre et en utilisant des jointures efficaces. Des index ont été appliqués sur les champs fréquemment sollicités afin d'accélérer les opérations et d'offrir à l'utilisateur une expérience de navigation rapide et agréable.

4.5 Organisation détaillée des fichiers et répertoires

Pour garantir une clarté maximale et une maintenance facilitée du projet, nous avons adopté une organisation précise et cohérente des fichiers et répertoires. Le fichier **login.php** représente le point d'entrée principal de l'application. Chaque page ou fonctionnalité spécifique est isolée dans des fichiers PHP dédiés, comme **login.php**, **dashboard.php**, **contacts.php**, **messages_envoyes.php** ou **cree_groupe.php**, placés logiquement au sein du projet.

Les ressources statiques (CSS, JavaScript, images) sont regroupées dans des dossiers clairement identifiés (**css**, **js**, **img**), ce qui facilite grandement leur gestion. Cette structuration détaillée rend le projet lisible, facile à comprendre et à adapter par la suite.

Cette conception méthodique et structurée a permis une réalisation fluide et efficace de notre application, tout en posant des bases solides pour son évolution future.

5 Implémentation

L'implémentation de notre projet s'est articulée autour de la mise en œuvre des fonctionnalités principales de la plateforme, en veillant à la sécurité, à la performance et à l'ergonomie. Chaque composant a été développé et testé minutieusement pour offrir une expérience utilisateur fluide et cohérente.

5.1 Gestion des utilisateurs et des contacts

La gestion des utilisateurs constitue le socle de notre application. Elle englobe l'inscription, la connexion, la gestion des contacts et le système de blocage.

- **Inscription et connexion** : Lors de l'inscription, les mots de passe des utilisateurs sont hachés à l'aide de la fonction `password_hash()` de PHP, qui utilise l'algorithme `bcrypt` par défaut. Cela garantit une sécurité renforcée des données sensibles.
Lors de la connexion, la fonction `password_verify()` est utilisée pour comparer le mot de passe saisi avec le hachage stocké en base de données. Les sessions PHP permettent de maintenir l'état de connexion de l'utilisateur entre les pages.
- **Gestion des contacts** : Les utilisateurs peuvent envoyer des demandes d'ajout de contact. Une fois la demande acceptée, une entrée est créée dans la table `Contacts`. La suppression d'un contact entraîne la suppression de l'entrée correspondante dans cette table.
- **Système de blocage** : Un utilisateur peut bloquer un autre utilisateur, ce qui est enregistré dans la table `Bloque`. Lorsqu'un blocage est en place, toutes les interactions entre les deux utilisateurs sont empêchées par des vérifications systématiques lors des opérations pertinentes.

Chaque algorithme a été modifié pour enregistrer ses étapes intermédiaires dans un format lisible. Ces étapes sont transmises au serveur, qui les relaie au client pour affichage. Ce mécanisme garantit une visualisation détaillée et éducative du fonctionnement interne des algorithmes.

5.2 Module de messagerie individuelle

Le module de messagerie permet aux utilisateurs d'échanger des messages privés.

Envoi de messages : Les utilisateurs peuvent envoyer des messages via un formulaire HTML comportant des champs pour le destinataire, l'objet et le contenu du message. Le traitement côté serveur, réalisé en PHP, valide les entrées et enregistre le message dans la table `Messages`.

Gestion des pièces jointes : Les utilisateurs peuvent joindre des fichiers à leurs messages. Le traitement des fichiers uploadés est effectué à l'aide de la fonction `move_uploaded_file()`

de PHP qui vérifie que le fichier a bien été téléchargé via HTTP POST avant de le déplacer vers un répertoire sécurisé sur le serveur. Des contrôles supplémentaires tels que la vérification de l'extension et de la taille des fichiers sont mis en place pour renforcer la sécurité.

5.3 Gestion des groupes de messagerie

La plateforme offre la possibilité de créer des groupes de discussion pour faciliter la communication entre plusieurs utilisateurs.

Création de groupes : Un utilisateur peut créer un groupe en sélectionnant des membres via une interface conviviale. Le groupe est enregistré dans la table `Groupe`, et les membres sont ajoutés dans la table `Membres_groupe`.

Messagerie de groupe : Les messages envoyés dans un groupe sont stockés dans la table `Messages_groupe` avec une référence au groupe concerné. L'interface affiche les messages du groupe en ordre chronologique et un formulaire permet d'ajouter de nouveaux messages à la conversation.

5.4 Partage de fichiers et pièces jointes

Le partage des fichiers constitue une fonction clé de la plateforme.

Le téléchargement des fichiers se fait grâce à des formulaires web en HTML et le traitement côté serveur utilise la fonction `move_uploaded_file()` pour déplacer les fichiers vers un répertoire sécurisé.

Des mesures de sécurité, telles que la vérification de l'extension, de la taille et du type MIME des fichiers, sont mises en place pour prévenir les risques liés aux fichiers malveillants.

L'accès aux fichiers est protégé par des contrôles d'authentification et des restrictions d'accès au niveau du serveur.

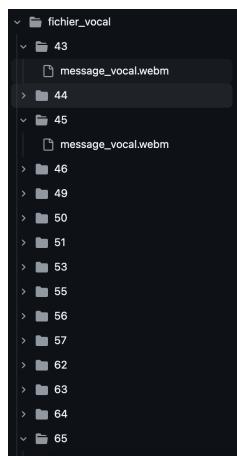


FIGURE 25 – Sous répertoires des fichiers vocaux contenu dans fichier_vocal

5.5 Tests et validations

Afin de garantir la robustesse et la fiabilité de notre plateforme, nous avons consacré une étape importante aux tests et aux validations. Ces tests se sont déroulés tout au long du projet, parallèlement aux phases d'implémentation afin d'identifier rapidement les éventuels dysfonctionnements et de les corriger efficacement.

Dans un premier temps, nous avons effectué des tests unitaires manuels sur chaque fonctionnalité isolée : création de comptes, connexion sécurisée, gestion des contacts et envoi de messages. Chaque fonctionnalité a été testée sous plusieurs cas d'utilisation, notamment avec des entrées valides et non valides, afin de vérifier la robustesse des validations côté serveur.

Ensuite, des scénarios d'intégration plus complexes ont été réalisés, simulant une utilisation réaliste de l'application. Par exemple, nous avons testé l'échange de messages entre deux utilisateurs avec des comptes fictifs, vérifié le bon fonctionnement du système de blocage en essayant d'envoyer des messages après avoir bloqué un utilisateur, et validé la création et la gestion des groupes avec plusieurs utilisateurs simultanés.

Nous avons également prêté une attention particulière à la gestion des fichiers joints, en réalisant des tests approfondis pour s'assurer que les restrictions sur les tailles et types de fichiers fonctionnaient correctement, et que l'accès aux fichiers était bien sécurisé.

Enfin, l'interface utilisateur a été testée sous plusieurs navigateurs (Chrome, Firefox, Safari) afin de s'assurer de sa compatibilité, de sa réactivité et de sa lisibilité en toutes circonstances. Ces tests variés et rigoureux nous ont permis d'obtenir une plateforme stable, sécurisée et performante, assurant ainsi une expérience utilisateur optimale.

6 Difficultés rencontrées

La réalisation de ce projet de plateforme web messagerie et gestion de contacts nous a confrontés à des nombreuses difficultés majeures, techniques, méthodologiques et organisationnelles. En tant que débutants en programmation web, ces défis éprouvants ont demandé une très grande persévérance, ainsi qu'un usage fréquent de ressources externes comme l'aide de notre professeur et l'intelligence artificielle. En dépit de ceux-ci, de telles défis ont finalement été une précieuse opportunité d'apprentissage qui a non seulement permis des progrès techniques, mais aussi amélioré notablement le niveau de la qualité du projet.

Notre solution offre un cadre flexible, intuitif et efficace au développement d'emplois du temps personnalisés, allant ainsi chercher à relever les grandes demandes du planification sur les écoles et les associations professionnelles. Toutefois, bien qu'avec tant de ces bénéfices, notre système présente certains aspects et directions potentielles de la progression vers la perfection. Pour instance, les conflits de planification pourrait être rééquilibré pour servir des solutions d'améliorations accrues dans quelques conditions particulières. Par ailleurs, fournir davantage d'options de personnalisation des plans selon le penchant individuel de l'utilisateur aurait des possibilités importantes pour améliorer son expérience totale.

6.1 Compréhension initiale de PHP et MySQL

Dès les premières séances de développement, l'une des difficultés notables a été la prise en main des langages PHP et SQL. Si les bases semblaient accessibles grâce à nos cours magistraux, la mise en pratique directe s'est avérée complexe. Les premiers scripts réalisés étaient souvent incorrects ou inefficaces, notamment concernant l'interaction entre PHP et la base de données MySQL. Nous avons dû refaire plusieurs fois certains formulaires, tels que ceux pour l'inscription ou la gestion des contacts, en raison de requêtes SQL mal construites ou d'erreurs de syntaxe en PHP.

Pour pallier ce problème, nous avons intensifié nos recherches et nos essais, tout en nous appuyant fortement sur des ressources en ligne comme la documentation officielle de PHP et MySQL, ainsi que des forums spécialisés. L'accompagnement régulier de notre enseignant, M. Thibault ANANI, a été essentiel pour clarifier les concepts difficiles à saisir initialement, notamment la gestion des sessions utilisateur et la sécurisation des données échangées.

En solution, l'équipe a utilisé des ressources en ligne comme la documentation officielle et des tutoriels spécialisés. Des tests répétés sur des exemples simples ont également permis une prise en main progressive de ces outils.

6.2 Difficultés liées à l'upload de fichiers

Une autre difficulté majeure s'est présentée avec la gestion de l'upload de fichiers (images, documents audio, et fichiers bureautiques). Les premières tentatives ont systématiquement rencontré des erreurs, principalement à cause des limites de taille des

fichiers autorisées par PHP ou des vérifications insuffisantes des types MIME. Ce qui permettait parfois l'upload de fichiers inappropriés ou mal sécurisés.

La gestion de ces fichiers nécessitait des vérifications minutieuses côté serveur pour garantir la sécurité et la cohérence. Plusieurs heures de recherche, d'essais et de débogage furent nécessaires pour maîtriser la fonction `move_uploaded_file()` de PHP. L'assistance de ChatGPT nous a particulièrement aidés pour comprendre les subtilités de la gestion des fichiers, notamment en nous proposant des exemples de code sécurisés et efficaces.

Finalement, après de multiples tests rigoureux et corrections, cette fonctionnalité fut stabilisée et sécurisée de manière satisfaisante.

6.3 Gestion de l'authentification et sécurisation

La sécurisation des formulaires et des interactions utilisateur constituait une difficulté importante, particulièrement sensible étant donné la nature du projet. Au début, nos formulaires présentaient plusieurs vulnérabilités aux attaques par injection SQL ou XSS. Nous avions du mal à comprendre comment protéger efficacement les entrées utilisateur, ce qui rendait notre application fragile.

Mais heureusement l'aide de notre enseignant et l'IA (ChatGPT) ont été extrêmement utiles dans ce contexte, en nous guidant sur les bonnes pratiques à adopter : une utilisation des requêtes préparées, une validation stricte des entrées utilisateur, un échappement systématique des données affichées.

Nous avons dû revoir entièrement nos scripts de connexion et d'inscription, ce qui a nécessité un important investissement en temps mais a conduit à une compréhension beaucoup plus approfondie des aspects sécuritaires du développement web.

6.4 Débogage et gestion des erreurs récurrentes

Tout au long du cycle de développement, nous avons été confrontés à une multitude de bugs récurrents, aussi frustrants soient-ils. Que ce soit le blocage des fonctionnalités utilisateur ou d'affichage des messages générant fréquemment des erreurs logiques complexes à déboguer.

À plusieurs reprises, des fonctionnalités apparemment correctement codées ne fonctionnaient pas comme elles le devraient, ce qui a nécessité de nombreuses heures de débogage et de recherche.

Nous avons dû apprendre à utiliser efficacement des outils de débogage tels que `var_dump()`, `error_log()` et des fonctionnalités intégrées à Visual Studio Code, afin d'identifier précisément les causes profondes des problèmes. Ces pratiques nous ont permis de développer une approche plus méthodique de la recherche d'erreurs, facilitant ainsi la suppression progressive de bugs complexes.

La correction de chaque bug a constitué une véritable réussite, renforçant notre compréhension technique et notre confiance dans notre capacité à concevoir une application web fiable.

6.5 Difficulté à définir les fonctionnalités supplémentaires

Une difficulté moins technique, mais tout aussi impactante, fut celle liée à la définition même des fonctionnalités à intégrer. À plusieurs reprises, nous avons atteint un stade où nous étions incertains des fonctionnalités supplémentaires à implémenter pour enrichir le projet et le rendre véritablement utile et attractif. Cette incertitude a ralenti considérablement notre progression à certains moments.

Pour surmonter cet obstacle, nous avons sollicité l'avis de notre professeur, M. ANANI, ainsi que de l'intelligence artificielle ChatGPT. Ces consultations nous ont aidés à identifier de nouvelles fonctionnalités pertinentes telles que la création de groupes de messagerie ou l'ajout d'une interface ludique et ergonomique pour les utilisateurs. Ce processus nous a permis d'élargir considérablement l'étendue et l'intérêt du projet tout en conservant une cohérence globale.

6.6 Synchronisation et collaboration en binôme

Enfin, travailler à deux sur un projet aussi dense présentait aussi des défis de synchronisation et de cohérence du travail. Malgré notre organisation et notre utilisation d'outils collaboratifs comme GitHub, des conflits d'intégration ou des doublons de code sont apparus régulièrement. Ces difficultés nous ont poussés à affiner notre manière de collaborer avec des échanges régulières, séances de pair programming et vérifications systématiques du travail de chacun mais surtout un travail sérieux tout au long du projet que ce soit lors les séances de travaux dirigés ou lors du travail personnel chez soi.

Grâce à ces améliorations méthodologiques, nous avons réussi à harmoniser efficacement nos contributions individuelles, rendant notre travail collaboratif productif et enrichissant.

6.7 Conclusion sur les difficultés rencontrées

Toutes ces difficultés, bien que conséquentes, nous ont offert une occasion unique d'apprentissage approfondi. Elles ont significativement renforcé nos compétences techniques et organisationnelles, tout en améliorant notre capacité à résoudre des problèmes complexes de manière autonome. Le soutien constant de notre professeur et l'appui précieux de l'IA ont été déterminants pour surmonter ces défis et mener à bien notre projet.

7 Conclusion

La réalisation de ce projet a représenté une étape majeure et très enrichissante dans notre parcours universitaire. Bien qu'étant débutants en programmation web au démarrage, ce projet nous a permis d'approfondir significativement nos compétences techniques. Notamment en PHP et MySQL, en réalisant une application web complète, fonctionnelle et interactive.

Tout au long de ce projet, nous avons pu aborder concrètement de nombreuses problématiques rencontrées dans la conception d'une plateforme numérique complexe : la gestion rigoureuse d'une base de données relationnelle, l'implémentation de mécanismes d'authentification sécurisés, la gestion efficace des interactions utilisateurs ainsi que la prise en charge d'échanges multimédias (photos, fichiers audio, documents). Nous avons ainsi découvert et appliqué concrètement les bonnes pratiques en sécurité web, allant de la sécurisation des mots de passe à la prévention des injections SQL et des attaques XSS, renforçant ainsi nos compétences pratiques et théoriques.

Ce travail collectif nous a également permis de développer de précieuses compétences en gestion de projet et en travail d'équipe. Chaque étape, de la conception initiale à l'implémentation détaillée, a nécessité une collaboration constante et efficace entre nous. Nous avons appris à gérer notre temps, à nous répartir équitablement les tâches, et à prendre en charge chacun les responsabilités techniques nécessaires à l'avancement du projet.

L'un des moments les plus intéressants a été la confrontation aux problèmes techniques et organisationnels qui nous ont conduit à rechercher activement des solutions. Nous sommes confrontés à beaucoup de challenges, que ce soit à des problèmes liés à la sécurisation des fichiers et à la résolution des problèmes d'upload, à la complexité d'interaction entre front-end et back-end, ou encore à des bugs cycliques compliqués à détecter. Confrontés à ce type d'obstacles, les ressources documentaires disponibles en ligne, les outils débogageurs mais également la guidance de notre enseignant, M. Thibault ANANI. De plus les outils d'intelligence artificielles (notamment les utilisations de ChatGPT et Grok) nous ont permis progressivement à chaque fois de résoudre chacun de ces problèmes. L'aide précieuse de ces ressources extérieures nous a offerte la chance d'apprendre d'expérience et d'accorder une résistance durable à nos capacités résolutive de solutions techniques complexes.

Au-delà des considérations purement techniques, ce projet nous tenait particulièrement à cœur !

Dès le départ, nous voulions créer une application qui ne se contente pas de fonctionner, mais qui soit aussi agréable à utiliser. On a vraiment pris le temps de réfléchir à une interface simple, fluide, intuitive. On a réellement pris le temps de réfléchir à une interface simple, fluide, intuitive et sommes fiers du résultat. Travailler sur l'ergonomie et l'expérience utilisateur nous a permis de comprendre à quel point ces éléments sont essentiels dans le développement web.

Au fil du projet, on a aussi confirmé quelque chose de plus personnel : notre véritable intérêt pour la programmation web. On s'est investis à fond, et voir que le résultat final dépasse nos attentes nous motive encore davantage à continuer dans cette voie.

Chaque étape, chaque difficulté surmontée, nous a permis de progresser et de renforcer nos compétences.

Ce projet a été un vrai tournant. Il nous a permis de passer de la théorie à une pratique concrète avec toutes les responsabilités que cela implique. On en garde une expérience marquante, valorisante mais surtout très formatrice. C'est une étape importante dans notre parcours à l'Université Paris Nanterre et une preuve concrète de ce qu'on est capables de réaliser quand on s'en donne les moyens.

8 Webographie

Références

- [W3Schools] <https://www.w3schools.com/php/>
- [W3Schools] <https://www.w3schools.com/sql/>
- [ChatGPT] chatgpt.com
- [Github] github.com
- [Visual Studio Code] code.visualstudio.com
- [MAMP] <https://www.mamp.info/en/mac/>
- [MDN Web Docs] <https://developer.mozilla.org/fr/>
- [Bootstrap Documentation officielle] <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
- [Grok] <https://grok.com>
- [Stackoverflow] <https://stackoverflow.com/>
- [PHP Official Documentation] <https://www.php.net/manual/fr/>
- [MySQL Documentation officielle] <https://dev.mysql.com/doc/>
- [PHP Sessions - Sécurité et Bonnes Pratiques] <https://www.php.net/manual/fr/features.sessions.security.php>
- [OWASP Sécurité Web] <https://owasp.org/www-project-top-ten/>
- [PHP Sécurité (OpenClassrooms)] <https://openclassrooms.com/fr/courses/1665806-programmez-en-orientee-objet-en-php>
- [Guide MIME Types (IANA)] <https://www.iana.org/assignments/media-types/media-types.xhtml>
- [Font Awesome Icons] <https://fontawesome.com/docs>
- [Apache HTTP Server Documentation] <https://httpd.apache.org/docs/current/>
- [FileZilla] <https://filezilla-project.org>

9 Annexes

Manuel Utilisateur

Version macOS / Version Windows

Étape 1 : Installation de MAMP

- Téléchargez MAMP gratuitement sur : [MAMP macOS](#).
- Lancez le fichier .pkg téléchargé et suivez les instructions d'installation.
- Une fois installé, démarrez MAMP depuis votre dossier Applications.

Étape 2 : Configuration de MAMP

- Dans l'interface MAMP, cliquez sur l'onglet **Preferences**.
- Allez dans l'onglet **Ports** et choisissez :
 - Apache Port : 8888
 - MySQL Port : 8889
- Cliquez ensuite sur l'onglet **Web Server** :
 - Cliquez sur **Select** et choisissez le dossier dans lequel vous souhaitez installer votre projet (par exemple `htdocs` ou votre dossier personnel).
- Cliquez sur **OK** pour sauvegarder.

Étape 1 : Installation de WAMP

- Téléchargez gratuitement WAMP depuis : [WAMP Windows](#).
- Lancez l'installation et suivez les instructions à l'écran (installation recommandée avec les paramètres par défaut).
- À la fin de l'installation, démarrez WAMP depuis le menu démarrer ou via l'icône sur votre bureau.

Étape 2 : Configuration de WAMP

- Une fois lancé, WAMP affiche une icône verte dans la barre des tâches (près de l'heure), indiquant qu'il est opérationnel.
- Cliquez sur l'icône verte de WAMP puis sélectionnez :
 - **www directory** pour afficher le dossier racine de votre serveur web.

Étape 3 : Installation du projet PHP

- Dézippez le dossier de votre projet dans le répertoire sélectionné (par exemple : `/Applications/MAMP/htdocs/projet_php`).
- Votre projet devrait maintenant être accessible via : http://localhost:8888/projet_php.

Étape 3 : Installation du projet PHP

- Copiez le dossier décompressé de votre projet (`projet_php`) dans le répertoire ouvert précédemment (`C:\wamp64\www\`).
- Votre projet devrait être accessible via l'adresse : `http://localhost/projet_php`.

Étape 4 : Configuration de la base de données

- Allez dans votre navigateur et tapez : `http://localhost:8888/phpMyAdmin`.
- Connectez-vous avec les identifiants par défaut de MAMP :
 - Utilisateur : `root`
 - Mot de passe : `root`
- Cliquez sur l'onglet **Importer**, puis sélectionnez votre fichier `ad.sql`.
- Cliquez sur **Exécuter** en bas pour importer votre base de données.

Étape 5 : Modification des paramètres du projet PHP

- Ouvrez le fichier `config/database.php` du projet dans Visual Studio Code ou tout autre éditeur de texte.
- Vérifiez que les informations sont correctes :

```
php
CopierModifier
$host = 'localhost';
$dbname = 'nom_de_votre_base'; // nom indiqué dans votre fichier .sql
$user = 'root';
$password = 'root';
```

Étape 6 : Accès au projet

- Lancez votre navigateur internet.
- Allez sur `http://localhost:8888/projet_php`.
- Votre site web est maintenant lancé et prêt à l'emploi.

Étape 6 : Accès au projet

- Ouvrez un navigateur (Chrome, Firefox, Edge).
- Rendez-vous à l'adresse suivante : `http://localhost/projet_php`.
- Votre site est maintenant opérationnel et prêt à être utilisé.