République Tunisienne
Ministère de l'Enseignement Supérieur de la Recherche Scientifique
Université de Tunis El Manar
Faculté des Sciences de Tunis
Département des Sciences de l'informatique

Rapport de projet

Cycle Ingénieur en Génie logiciel et Système d'Information

Par
Karima Haddad

Mayssa Chikh Ali

Nour Khelifi

VolunteerNow : Conception et développement d'une plateforme web de gestion du volontariat

Année Universitaire : 2025-2026

Table des matières

Introduction

Le projet **VolunteerNow** s'inscrit dans une démarche de digitalisation de l'engagement citoyen. Dans un contexte où les initiatives solidaires se développent, il devient essentiel de proposer une plateforme moderne permettant de connecter efficacement les organisations et les bénévoles. L'objectif principal est de concevoir une application web intuitive offrant la possibilité de découvrir des événements, de s'y inscrire et de suivre son parcours de participation.

Ce mini-rapport présente les différentes étapes de conception et de développement du projet. Il décrit les fonctionnalités essentielles, l'architecture logicielle adoptée, ainsi que les choix technologiques réalisés pour assurer une solution fiable, évolutive et adaptée aux besoins des utilisateurs.

La plateforme VolunteerNow propose plusieurs fonctionnalités innovantes : une carte interactive permettant de localiser les événements, un système de badges pour valoriser l'engagement des bénévoles, un chatbot d'assistance, ainsi qu'un espace d'inscription et de gestion des candidatures. Elle vise à centraliser les initiatives solidaires et à faciliter la participation citoyenne partout en Tunisie.

En résumé, ce projet répond à un besoin réel de modernisation et d'accessibilité dans le secteur du bénévolat. VolunteerNow constitue ainsi une solution complète et évolutive pour promouvoir l'action solidaire de manière simple, transparente et efficace.

Installation du projet

Ce chapitre décrit les différentes étapes nécessaires pour installer et exécuter le projet *VolunteerNow* en local. L'objectif est de fournir un guide clair permettant à tout membre de l'équipe ou futur contributeur de comprendre comment mettre en place l'environnement de développement. Le projet se compose d'un **backend Node.js/Express** et d'un **frontend Angular**, chacun nécessitant une configuration spécifique.

Prérequis

Avant toute installation, plusieurs outils doivent être présents sur la machine :

- **Node.js** et **npm** pour gérer les dépendances et exécuter le serveur ;
- **MongoDB** pour stocker les données de l'application ;
- **Angular CLI** pour compiler et exécuter l'interface utilisateur ;
- **Git** pour récupérer le code source et collaborer efficacement.

Ces outils constituent la base nécessaire pour exécuter correctement le projet.

Récupération du projet

Le code source est hébergé sur GitHub. Pour obtenir une copie locale :

```
git clone https://github.com/.../VolunteerNow.git
cd VolunteerNow
```

Chaque contributeur doit ensuite créer son propre fichier `.env` afin de configurer les variables sensibles, qui ne sont volontairement pas incluses dans le dépôt Git pour des raisons de sécurité.

Installation du backend

Le backend, situé dans le dossier `volunteer-now-backend`, contient l'API, les modèles de données et la logique métier. L'installation des dépendances s'effectue via :

```
npm install
```

Créer ensuite un fichier `.env` contenant les informations suivantes :

```
MONGO_URI=...
JWT_SECRET=...
EMAIL_USER=...
EMAIL_PASS=...
OPENAI_API_KEY=...
```

Ces variables permettent la connexion à MongoDB, la gestion des tokens d'authentification, l'envoi d'e-mails et le fonctionnement du chatbot. Le serveur se lance avec :

```
npm run dev
```

Installation du frontend

Le frontend Angular se trouve dans le dossier `volunteer-now-frontend`. Pour installer les dépendances :

```
npm install
```

Puis lancer l'application :

```
ng serve -o
```

Cette commande ouvre automatiquement l'application dans le navigateur à l'adresse `http://localhost:4200`.

Connexion entre frontend et backend

L'interface doit communiquer avec l'API. Cette configuration se fait dans le fichier `environment.ts` :

```
export const environment = {
  apiUrl: "http://localhost:5000/api"
};
```

Cette étape est indispensable pour permettre l'authentification, la gestion des événements et l'interaction utilisateur.

Architecture et arborescence du projet

Le projet *VolunteerNow* adopte une architecture web structurée autour de deux couches principales : un **backend** développé en Node.js/Express et un **frontend** construit avec Angular. Cette séparation des responsabilités garantit une organisation claire du code, une meilleure maintenabilité et une grande évolutivité du système.

Architecture du backend

Le backend représente le coeur logique de l'application. Il assure la gestion des utilisateurs, des événements, des candidatures, des badges, de la communication avec la base de données ainsi que l'intégration du chatbot. L'architecture interne suit un modèle proche du MVC :

- **Routes** : elles définissent les points d'entrée de l'API et reçoivent les requêtes HTTP.
- **Contrôleurs** : ils contiennent la logique métier associée à chaque fonctionnalité.
- **Modèles** : ils définissent les schémas MongoDB pour les entités (User, Event, Candidature...).
- **Middleware** : ils assurent des traitements transversaux comme l'authentification JWT.
- **Services et utilitaires** : gestion des emails, génération de tokens, intégration OpenAI.

La base de données MongoDB stocke les informations sous forme de documents JSON, offrant une grande flexibilité et facilitant l'évolution des structures de données.

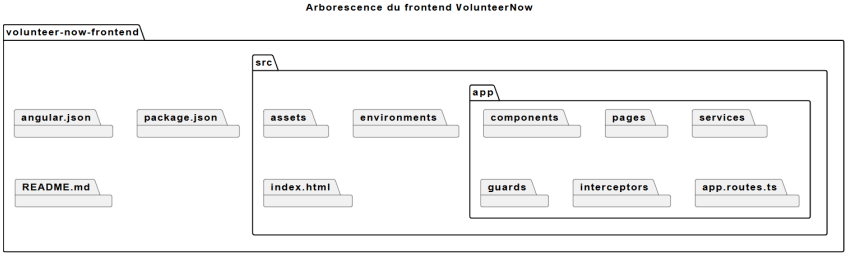

Figure 1 – Architecture du backend

Architecture du frontend

Le frontend Angular constitue l'interface utilisateur. Organisé de manière modulaire, il repose sur :

— **Composants** : éléments visuels réutilisables (boutons, cartes, formulaires).
— **Pages** : vues complètes (Accueil, Événements, Profil).
— **Services** : modules assurant la communication avec le backend via HTTP.
— **Intercepteurs** : ajout automatique du token JWT dans les requêtes sécurisées.
— **Guards** : protection des routes nécessitant une authentification.
— **Modules externes** : Leaflet pour la carte interactive, OpenAI pour le chatbot.

Cette structure modulaire permet une expérience fluide, une navigation rapide et une grande facilité de maintenance.

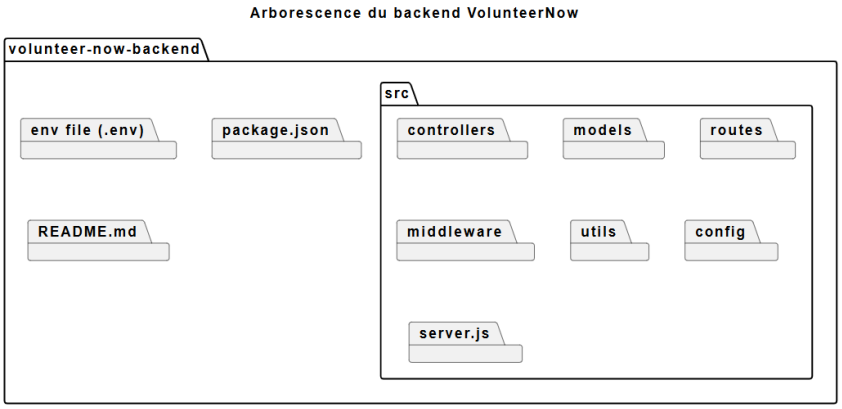

Figure 2 – Architecture du frontend

Communication entre frontend et backend

La communication repose sur une API REST. Le processus se déroule ainsi :

1. le frontend envoie une requête HTTP à l'API ;
2. le backend vérifie l'authentification via un middleware ;
3. la logique métier s'exécute et la base MongoDB est interrogée ;
4. une réponse JSON est renvoyée au frontend ;
5. l'interface se met à jour en fonction du résultat.

Cette architecture assure une séparation nette des responsabilités, une sécurité renforcée grâce à l'utilisation des tokens JWT, et une grande évolutivité pour l'ajout de nouvelles fonctionnalités.

Description des fonctionnalités

La plateforme *VolunteerNow* propose un ensemble de fonctionnalités destinées à faciliter l'interaction entre les bénévoles et les organisations. Ce chapitre présente les modules principaux développés dans le cadre du projet.

Authentification et gestion des utilisateurs

La plateforme intègre un système d'authentification sécurisé reposant sur les tokens JWT. Les utilisateurs peuvent créer un compte, se connecter, modifier leur profil, réinitialiser leur mot de passe et se déconnecter. Un contrôle des rôles permet de différencier les fonctionnalités accessibles aux bénévoles et aux organisations.

Gestion des événements

Les organisations peuvent créer, modifier et supprimer des événements. Chaque événement comprend un titre, une description, une date, une localisation et un statut. Les bénévoles peuvent consulter la liste des événements disponibles, afficher les détails et visualiser leur position sur une carte.

Participation et gestion des candidatures

Les bénévoles peuvent soumettre une candidature pour participer à un événement, suivre l'état de leur demande (en attente, acceptée, refusée) et recevoir des notifications. Les organisations, quant à elles, disposent d'une interface leur permettant d'accepter ou refuser les candidatures reçues.

Carte interactive

L'application intègre une carte interactive basée sur Leaflet. Celle-ci affiche l'ensemble des événements géolocalisés, permettant aux utilisateurs de repérer facilement les actions proches de leur position.

Système de badges

Un système de badges valorise l'engagement des bénévoles. Les badges Bronze, Argent, Or et Platine sont attribués selon le nombre d'événements auxquels le bénévole a participé.

Chatbot intégré

La plateforme inclut un chatbot alimenté par l'API OpenAI. Celui-ci permet aux utilisateurs d'obtenir des réponses rapides à leurs questions, d'être guidés dans l'utilisation du site et de bénéficier d'un support permanent.

Système de notifications

Les utilisateurs reçoivent des notifications concernant la validation de leurs candidatures, la mise à jour du statut d'un événement ou d'autres actions importantes liées à leur activité.

Choix techniques

Le développement du frontend de la plateforme *VolunteerNow* repose sur l'utilisation de plusieurs mécanismes clés d'Angular permettant de gérer efficacement les formulaires, la réactivité de l'interface, l'accès au DOM et la communication entre les composants. Dans ce chapitre, ces mécanismes ne sont pas seulement décrits, mais également **justifiés** afin de mettre en évidence les raisons de leur choix.

Two-Way Binding avec `ngModel`

La directive `ngModel` a été utilisée pour les formulaires simples tels que la connexion, l'inscription ou le chatbot. Elle permet une synchronisation automatique entre la vue et le composant, simplifiant la gestion des champs.

```
<input [(ngModel)]="userInput">
```

Ce choix a été privilégié pour les formulaires courts et peu complexes, où la mise en place de Reactive Forms aurait été plus lourde que nécessaire. Le *two-way binding* via `ngModel` offre un gain de temps de développement et une lisibilité accrue du code pour les cas où les validations restent simples. L'objectif n'était donc pas d'utiliser l'approche la plus « avancée », mais la plus **adaptée** au contexte d'usage.

Utilisation de ChangeDetectorRef

Certaines opérations, comme l'ajout d'un message dans le chatbot ou le défilement automatique, nécessitent une mise à jour forcée de la vue. Pour cela, `ChangeDetectorRef` a été utilisé afin de déclencher manuellement la détection de changements.

```
this.cdr.detectChanges();
```

Ce choix répond à une contrainte pratique : lorsque de nouveaux messages sont ajoutés, la vue n'est pas forcément mise à jour au moment où l'on souhaite effectuer le défilement vers le bas. Forcer la détection de changements permet de garantir que le DOM est à jour avant d'appliquer le *scroll*. L'utilisation de `ChangeDetectorRef` illustre ainsi la capacité à aller au-delà du fonctionnement « automatique » d'Angular pour résoudre un problème précis d'interface utilisateur.

Accès au DOM avec ViewChild

Le décorateur `ViewChild` permet d'accéder à un élément du DOM. Il a été utilisé notamment pour gérer le défilement automatique de la zone de discussion du chatbot.

```
@ViewChild('chatBody') chatBody!: ElementRef;
```

L'accès direct au DOM n'a pas été utilisé partout, mais uniquement dans des cas ponctuels où il est nécessaire de manipuler des propriétés spécifiques de l'élément (par exemple `scrollTop` et `scrollHeight`). Ce choix est justifié par le besoin d'un contrôle fin sur le comportement de la zone de discussion, afin d'offrir une expérience utilisateur fluide. Il illustre également la capacité à utiliser les outils d'Angular pour combiner **liaison de données** et **manipulation ciblée du DOM**.

Cycle de vie des composants

Les hooks `ngOnInit` et `ngAfterViewInit` ont été utilisés pour initialiser les données et manipuler le DOM une fois la vue rendue. Cela garantit un affichage correct et une gestion cohérente des événements asynchrones.

Le choix de ces hooks n'est pas arbitraire : `ngOnInit` est utilisé pour charger les données et configurer l'état initial du composant, tandis que `ngAfterViewInit` est réservé aux opérations qui nécessitent que la vue soit totalement construite, comme le défilement du chatbot. Cela permet de respecter le cycle de vie d'Angular et d'éviter des erreurs liées à l'accès prématuré au DOM. Ce choix montre une compréhension des bonnes pratiques liées à la gestion du cycle de vie des composants.

Directives Angular

Les directives `*ngFor`, `*ngIf` et `ngClass` ont été utilisées pour générer dynamiquement les éléments visuels, afficher ou masquer des sections et appliquer des styles selon le statut des candidatures.

Ces directives ont été choisies car elles permettent d'exprimer clairement la logique d'affichage directement dans le template : `*ngFor` pour parcourir des listes (événements, messages du chatbot, candidatures), `*ngIf` pour afficher des éléments sous conditions (utilisateur connecté, rôle, présence de résultats), et `ngClass` pour changer l'apparence en fonction d'un état (statut accepté/refusé/en attente). Ce choix renforce la séparation entre la logique métier (dans le TypeScript) et la logique de présentation (dans le HTML).

Routing et guards

Le module `RouterModule` permet une navigation fluide dans l'application. `AuthGuard` protège les routes réservées aux utilisateurs authentifiés.

Le choix d'utiliser des guards est motivé par des considérations de sécurité et d'architecture : plutôt que de vérifier manuellement dans chaque composant si l'utilisateur est connecté, la protection est centralisée au niveau du routing. Cela garantit que certaines pages (profil, tableau de bord des organisations, gestion des événements) ne sont accessibles qu'aux utilisateurs autorisés, tout en gardant les composants plus simples et plus réutilisables.

Services Angular

Les services (*AuthService*, *EventService*, *CandidatureService*) centralisent la communication avec le backend. Ils utilisent `HttpClient` pour envoyer des requêtes HTTP et récupérer les données.

Ce choix découle d'un principe fondamental d'Angular : la **séparation des responsabilités**. En plaçant la logique d'appel API dans des services, les composants restent centrés sur l'affichage et les interactions utilisateur. Cette approche facilite la maintenance, les tests unitaires et l'évolution du code (par exemple, modifier un endpoint ne nécessite pas d'intervenir dans plusieurs composants).

Chatbot et manipulation dynamique de l'UI

Le chatbot repose sur :

— un tableau de messages géré avec `*ngFor`,
— une simulation d'écriture via des délais asynchrones,
— le défilement automatique (`ViewChild + ChangeDetectorRef`).

Ces choix techniques ont été faits pour offrir une expérience utilisateur proche d'une vraie conversation : les messages sont affichés dynamiquement, le chatbot simule un temps de saisie et la vue se met automatiquement à jour pour afficher le dernier message. Le recours combiné à `*ngFor`, `ViewChild` et `ChangeDetectorRef` illustre la manière dont Angular peut être utilisé pour construire des interfaces **réactives**, **vivantes** et adaptées aux usages modernes.

Captures d'écran

Authentification

FIGURE 3 – *
Page de connexion

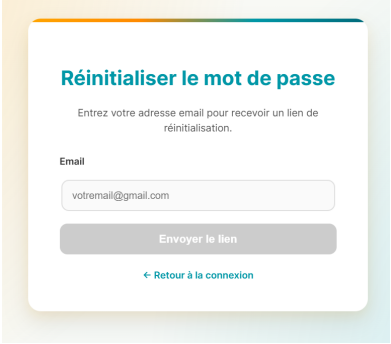

FIGURE 4 – *
Mot de passe oublié

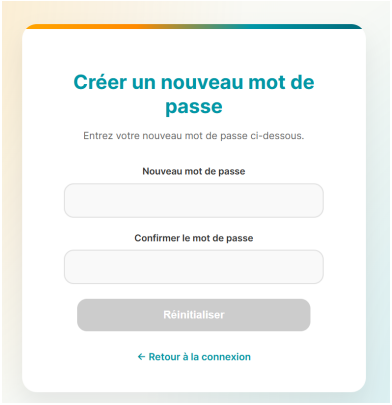

FIGURE 5 – *
Réinitialisation du mot de passe

Gestion des événements

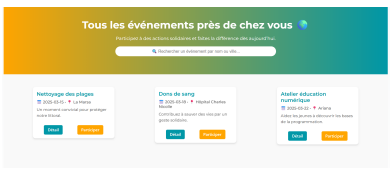

FIGURE 6 – *
Liste des événements

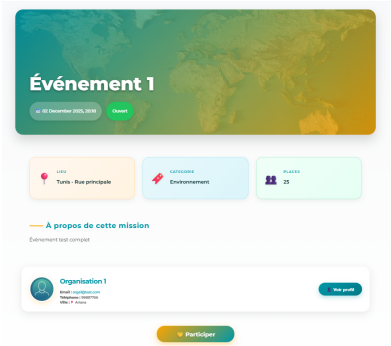

FIGURE 7 – *
Détails d'un événement

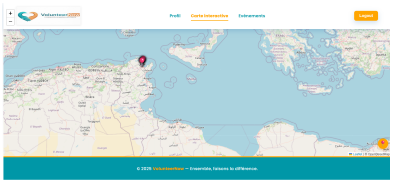

FIGURE 8 – *
Carte interactive

Interfaces profils

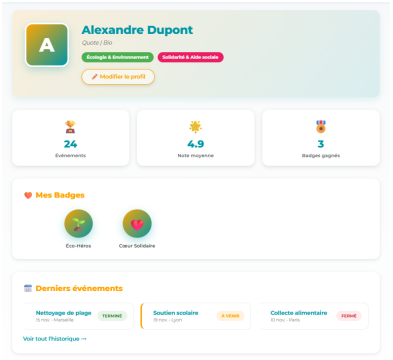

FIGURE 9 – *
Espace bénévole

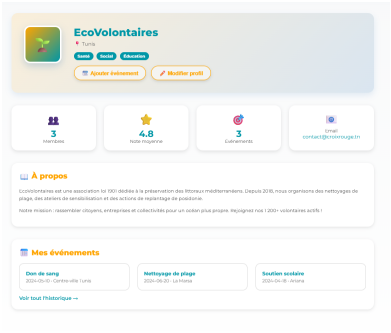

FIGURE 10 – *
Espace organisation

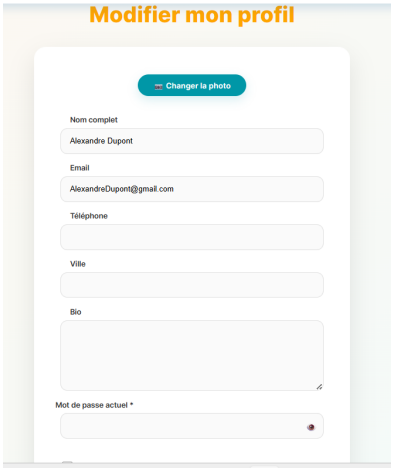

FIGURE 11 – *
Modification du profil

Inscription et Chatbot

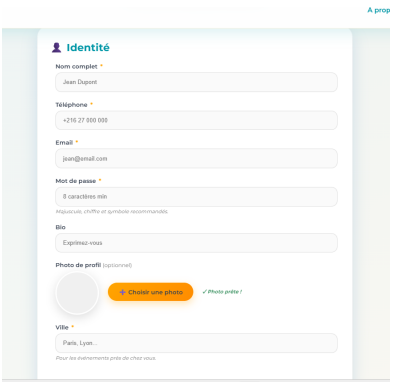

FIGURE 12 – *
Page d'inscription

FIGURE 13 – *
Page À propos

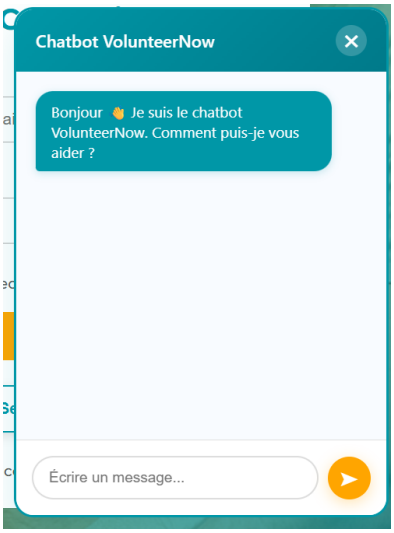

FIGURE 14 – *
Chatbot intégré

Conclusion et limites du projet

Conclusion

Le projet *VolunteerNow* a permis de concevoir une plateforme web destinée à faciliter la mise en relation entre les bénévoles et les organisations. Il a atteint l'ensemble de ses objectifs principaux, notamment la centralisation et la gestion des événements, la simplification du processus de participation, ainsi que la valorisation de l'engagement grâce à un système de badges. Le développement a également intégré des fonctionnalités interactives telles qu'une carte Leaflet et un chatbot, améliorant ainsi l'expérience utilisateur. Sur le plan technique, le projet a permis d'exploiter des technologies modernes comme Angular, Node.js, Express et MongoDB, tout en mettant en pratique des notions fondamentales du développement web, parmi lesquelles les API REST, l'architecture client–serveur, la sécurité via JWT, la gestion de formulaires complexes et l'intégration de services externes. Ce travail représente ainsi une expérience complète et enrichissante, combinant conception fonctionnelle et implémentation technique dans un cadre réaliste.

Limites du projet

— Absence d'un système de messagerie interne.
— Les candidatures nécessitent un rafraîchissement manuel ; une mise à jour automatique via WebSockets serait un ajout pertinent.
— La sécurité, bien que présente, pourrait être renforcée :
— Certaines fonctionnalités (notifications, tableau de bord avancé, statistiques) pourraient être étoffées.

Perspectives

— Intégration de la messagerie en y intégrant le temps réel.
— Mise en place d'une suite de tests automatisés.
— Renforcement de la sécurité et de la performance.
— Développement d'une application mobile associée.