

Documentation v0.3 Ð Gestion des demandes et les recherche optimisŽs

Sommaire

- Introduction	2
(Grandes FonctionnalitŽs	2
/ 2.1 Gestion des Demandes dŔntŽr•t & Notifications en Temps RŽel	3
/ 2.2 Recherche de Propositions	4
/ 2.3 Recherche AvancŽe des fvŽnements	5
/ 2.4 Modification et Suppression des fvŽnements	7
(Petites FonctionnalitŽs	8
/ 2.1 Page ŔMes Demandes EnvoyŽesŔ	8
/ 2.2 Notifications en Temps RŽel #	9
/ 2.3 DŽtails dŔun fvŽnement	9
/ 2.4 Gestion des Images des fvŽnements (Frontend)	10
/ 2.5 Filtre par Villes	10
/ 2.6 AmŽliorations UX/UI &	11
9 Impact MŽtier & Valeur AjoutŽe	11
" Tests & Validation	12
Documentation v1.0 Ð Signalement de dangers	12
- Introduction	12
; ; FonctionnalitŽs	12
< Workflow du signalement de danger	13
= DŽtails Techniques	14
> Illustrations	14
9 Comparaison avec les plateformes existantes	14
? Pourquoi nous sommes innovants ?	15
" Tests rŽalisŽs	15
(Conclusion	16
/ Documentation v1.0 Ð Gestion des Projets	16
Description	16
Flux Utilisateur	16
Endpoints Backend	17
Diagramme de SŽquence : Gestion des Projets	18
9 Impact MŽtier & Valeur AjoutŽe	19
(Tests & Validation	19

" Conclusion	19
Documentation Technique Ð Release v1.1 - DevOps2	20
Objectif de la fonctionnalitŽ	20
(NouveautŽ c™tŽ utilisateur	20
@ FonctionnalitŽs livrŽes	20
Authentification Google (OAuth2) Ð IntŽgration initiale	21
Objectif	21
FonctionnalitŽs livrŽes	21
ImplŽmentation technique	21
Migration SQL associŽe	22
Configuration	22
IntŽgration Google Calendar Ð DŽtails techniques	22
1. Authentification OAuth	22
2. Affichage des ŽvŽnements Google (dashboard)	22
3. CrŽation dŽvŽnement ^ la participation	23
4. Backend (routes Node/Express)	23
Architecture technique	23
Probl•mes rencontrŽs	24
BŽnŽfices de cette feature dans la stratŽgie globale de notre application	24
Tests rŽalisŽs	24
Release v1.1 Ð RŽcapitulatif	24

! Introduction

La version v0.3 marque une avancŽe majeure pour l'application en intŽgrant un syst•me de notifications en temps rŽel, une gestion fluide des demandes d'untŽr•t, et une expŽrience utilisateur optimisŽe.

Cette documentation couvre :

1. Les nouvelles fonctionnalitŽs dŽveloppŽes.
2. Les flux utilisateurs pour chaque action clŽ.
3. Les endpoints backend utilisŽs.
4. L'impact mŽtier et la valeur ajoutŽe des amŽliorations.
5. Un diagramme de sŽquence UML pour illustrer le workflow.

" Grandes FonctionnalitŽs

2.1 Gestion des Demandes d'Intérêt & Notifications en Temps Réel

Description

Cette fonctionnalité regroupe tout le flux des demandes d'intérêt, y compris les notifications en temps réel. Les utilisateurs peuvent envoyer et recevoir des demandes d'intérêt, puis être notifiés de l'acceptation ou du refus de la demande en temps réel.

Flux Utilisateur

1. Envoi d'une Demande d'Intérêt

- ! L'utilisateur intéressé clique sur "Demander".
- ! La demande est enregistrée en base (via POST /interests).
- ! Une notification en temps réel est envoyée au proposeur.

2. Consultation des Demandes d'Intérêt

- ! Le proposeur accède à "Mes Intérêts Reçus".
- ! Il voit la demande et peut l'accepter ou la refuser.

3. Notifications et Actions en Temps Réel

- ! Si la demande est acceptée, le demandeur est notifié avec les coordonnées du proposeur.
- ! Si la demande est refusée, le demandeur est informé de la décision.

Endpoints Backend

Méthode	Endpoint	Description
POST	/interests	Créer une demande d'intérêt.
GET	/interests/received/:userId	Récupérer les demandes reçues.
GET	/interests/sent/:userId	Récupérer les demandes envoyées.
PUT	/interests/:id	Accepter ou refuser une demande.
GET	/notifications/:userId	Récupérer toutes les notifications d'un utilisateur.
POST	/notifications	Créer une nouvelle notification.
DELETE	/notifications/:notifId	Supprime une notification spécifique.
DELETE	/notifications/all/:userId	Supprime toutes les notifications d'un utilisateur.

Diagramme de Séquence : Demande d'Intérêt et Notifications

```

@startuml
participant "Utilisateur Intřressř (par l'annonce)" as UI
participant "Frontend (React)" as FE
participant "Backend API" as API
participant "Base de Donnřes" as DB
participant "WebSockets" as WS
participant "Proposeur (de l'annonce)" as P

== ! 1. L'utilisateur envoie une demande d'intřręt ==
UI -> FE: Clique sur "Demander"
FE -> API: **POST** /interests (proposition_id, interested_user_id)
API -> DB: "Vřrifie que la proposition existe"
DB --> API: # OK
API -> DB: $ Enregistre la demande avec statut ***"pending"***
DB --> API: # OK (id_interet)
API -> WS: % **fmet une notification au proposeur**
WS --> P: & "Nouvelle demande reęue"

== ' 2. Le proposeur consulte ses demandes ==
P -> FE: Accęde ř "Mes Intřręts Reęus"
FE -> API: **GET** /interests/received/:userId
API -> DB: "Rřcupęre toutes les demandes associęes ř l'utilisateur"
DB --> API: ( Renvoie les demandes (id, titre, utilisateur intřressř)
API --> FE: )* Affiche la liste des demandes

== # 3A. Le proposeur **accepte** la demande ==
P -> FE: Clique sur "***Accepter***"
FE -> API: **PUT** /interests/:id (status: accepted)
API -> DB: # Met ř jour le statut en ***"accepted"***
DB --> API: # OK
API -> WS: % **fmet une notification avec le statut acceptř**
WS --> UI: & "***+ Votre demande a řtř acceptře ! Voici les contacts ,-***"

== . 3B. Le proposeur **refuse** la demande ==
P -> FE: Clique sur "***Refuser***"
FE -> API: **PUT** /interests/:id (status: rejected)
API -> DB: . Met ř jour le statut en ***"rejected"***
DB --> API: # OK
API -> WS: % **fmet une notification avec le statut refusř**
WS --> UI: & "***. Votre demande a řtř refusře.***"
@enduml

```

2.2 Recherche de Propositions

Description

Cette fonctionnalitř permet aux utilisateurs de rechercher des propositions en fonction de

plusieurs critères : mots-clés, catégorie et distance géographique.

Flux Utilisateur

1. L'utilisateur entre des mots-clés et sélectionne une catégorie de service.
2. Le système effectue une recherche floue sur les titres et descriptions des propositions.
3. Le système filtre les propositions par catégorie sélectionnée.
4. Le système calcule la distance géographique entre l'utilisateur et les propositions.
5. Les résultats sont affichés, triés par proximité géographique.

Endpoints Backend

Méthode	Endpoint	Description
GET	<code>/propositions/search</code>	Recherche des propositions en fonction des mots-clés, catégorie et distance.

Diagramme de Séquence : Recherche de Propositions

```
@startuml
actor "Utilisateur" as User
participant "Frontend (React)" as FE
participant "Backend API" as API
participant "Base de Données" as DB
participant "Fuse.js" as Fuse
participant "WebSocket (si notifications)" as WS

== 1. L'utilisateur effectue une recherche ==
User -> FE: Entre des mots-clés et sélectionne une catégorie
FE -> API: **GET** /propositions/search (mots-clés, catégorie, utilisateur_id)
API -> DB: "Récupère les propositions en fonction de la catégorie"
DB --> API: ( Liste des propositions filtrées par catégorie
API -> Fuse: Utilise Fuse.js pour recherche floue sur 'title' et 'description'
Fuse --> API: ( Liste des propositions correspondant aux mots-clés
API -> DB: "Récupère les coordonnées de l'utilisateur (latitude, longitude)"
DB --> API: ( Coordonnées de l'utilisateur
API -> DB: " Calcule la distance entre l'utilisateur et chaque proposition"
DB --> API: ( Liste des propositions avec distances
API -> FE: )* Affiche les résultats avec distance et pertinence
FE --> User: Montre les propositions filtrées

@enduml
```

2.3 Recherche Avancée des événements

Description

Cette fonctionnalité permet aux utilisateurs de rechercher des Événements en fonction de plusieurs critères : mots-clés, catégorie et ville. Grâce à la bibliothèque Fuse.js, la recherche est floue et permet de retrouver des Événements qui correspondent partiellement aux mots-clés recherchés, même en cas d'erreur de frappe.

Le processus de recherche est optimisé pour une expérience utilisateur fluide :

1. L'utilisateur saisit un mot-clé (et optionnellement, sélectionne une catégorie ou une ville).
2. Le système filtre les Événements en fonction de la catégorie et de la ville sélectionnées.
3. La recherche floue est effectuée sur les titres et descriptions des Événements en utilisant Fuse.js, avec un seuil de pertinence réglable pour affiner les résultats.
4. Les résultats sont retournés et triés par pertinence.

Flux Utilisateur

1. L'utilisateur entre un mot-clé de recherche et, si souhaité, sélectionne une catégorie et/ou une ville.
2. La recherche floue est effectuée dans les titres et descriptions des Événements.
3. Les Événements sont filtrés en fonction de la catégorie et de la ville, si spécifiés.
4. Les résultats de recherche sont retournés, affichés par pertinence.
5. L'utilisateur peut cliquer sur un Événement pour consulter son détail.

Endpoints Backend

Méthode	Endpoint	Description
GET	<code>api /events/search</code>	Recherche des Événements en fonction des mots-clés, catégorie et ville.
GET	<code>api /events/:id</code>	Récupère les détails d'un Événement spécifique.

Diagramme de Séquence : Recherche Avancée des événements

```
@startuml
actor "Utilisateur" as User
participant "Frontend (React)" as FE
participant "Backend API" as API
participant "Base de Données" as DB
participant "Fuse.js" as Fuse

== 1. L'utilisateur effectue une recherche ==
User -> FE: Saisit un mot-clé et sélectionne une catégorie ou une ville
FE -> API: **GET** api//events/search (mot-clé, catégorie, ville)
API -> DB: " Récupère tous les Événements en fonction de la catégorie et de la ville
DB --> API: ( Liste des Événements filtrés
API -> Fuse: Recherche floue sur 'title' et 'description'
```

```
Fuse --> API: ( Liste des Événements correspondant aux mots-clés
API -> FE: )* Affiche les résultats de la recherche
FE --> User: Montre les Événements filtrés par pertinence
```

```
== 2. L'utilisateur consulte un Événement ==
User -> FE: Clique sur un Événement
FE -> API: **GET** api/events/:id
API -> DB: " Récupère les détails de l'événement avec l'ID
DB --> API: ( Détails de l'événement
API -> FE: )* Affiche les détails de l'événement
FE --> User: Montre les détails de l'événement
```

@enduml

2.4 Modification et Suppression des événements

Description

Les utilisateurs peuvent désormais modifier ou supprimer leurs événements à partir de l'interface. Cela permet une gestion complète des événements, incluant l'actualisation ou la suppression de données obsolètes.

Flux Utilisateur

1. Modification

- ! L'utilisateur ouvre les détails de son événement.
- ! Il clique sur le bouton "Modifier".
- ! Un formulaire pré-rempli s'affiche avec les informations actuelles.
- ! Après modification, il clique sur "Enregistrer" pour sauvegarder les modifications.

2. Suppression

- ! L'utilisateur ouvre les détails de son événement.
- ! Il clique sur le bouton "Supprimer".
- ! Une confirmation s'affiche avant suppression définitive.

Endpoints Backend

Méthode	Endpoint	Description
PUT	/api/events/:id	Met à jour un événement existant.
DELETE	/api/events/:id	Supprime un événement spécifique.

Diagramme de Séquence : Modification et Suppression des événements

```

@startuml
actor "Utilisateur" as User
participant "Frontend (React)" as FE
participant "Backend API" as API
participant "Base de DonnŽes" as DB

== 1. Modification ==
User -> FE: Ouvre les dŽtails de l'ŽvŽnement
FE -> API: **GET** /api/events/:id
API -> DB: RŽcup•re les donnŽes de l'ŽvŽnement
DB --> API: Renvoie les donnŽes de l'ŽvŽnement
API --> FE: Affiche les dŽtails
User -> FE: Clique sur "Modifier" et enregistre les modifications
FE -> API: **PUT** /api/events/:id (modifications)
API -> DB: Met ^ jour l'ŽvŽnement
DB --> API: Confirme la mise ^ jour
API --> FE: Notifie le succ•s de la modification

== 2. Suppression ==
User -> FE: Clique sur "Supprimer"
FE -> API: **DELETE** /api/events/:id
API -> DB: Supprime l'ŽvŽnement
DB --> API: Confirme la suppression
API --> FE: Notifie le succ•s de la suppression
@enduml

```

" Petites Fonctionnalités

2.1 Page "Mes Demandes EnvoyŽes" \$

Description

Ajout d'une nouvelle section permettant aux utilisateurs de suivre leurs demandes et voir si elles sont acceptŽes ou refusŽes.

Flux Utilisateur

1. L'utilisateur consulte la section "Mes demandes envoyŽes".
2. Il voit toutes ses demandes avec leur statut actuel.
3. Si la demande est acceptŽe, il acc•de aux coordonnŽes du proposeur.

Endpoints Backend

MŽthode	Endpoint	Description
---------	----------	-------------

GET	/interests/sent/:userId	Retourne les demandes envoyées par l'utilisateur.
PUT	/interests/:id	Mettre à jour le statut d'une demande.

2.2 Notifications en Temps Réel

Description

Les notifications sont envoyées en temps réel à l'utilisateur lorsqu'une action importante se produit (acceptation/refus d'une demande, etc.). Cela permet une interaction fluide et réactive avec l'application.

Flux Utilisateur

1. L'utilisateur effectue une action qui génère une notification.
2. Une notification apparaît instantanément dans le panneau des notifications.
3. L'utilisateur peut la consulter et la supprimer.

Endpoints Backend

Méthode	Endpoint	Description
POST	/notifications	Créer une nouvelle notification.
GET	/notifications/:userId	Récupérer toutes les notifications d'un utilisateur.
DELETE	/notifications/:notificationId	Supprimer une notification spécifique.
DELETE	/notifications/all/:userId	Supprimer toutes les notifications d'un utilisateur.

2.3 Détails d'un Événement

Description

Les utilisateurs peuvent désormais visualiser les détails d'un événement. Cette page affiche les informations complètes de l'événement sélectionné, comme son titre, sa description, sa date, son lieu, sa catégorie, et son image associée.

Flux Utilisateur

1. L'utilisateur clique sur un événement dans la liste des événements.
2. Une fenêtre modale s'affiche, contenant les détails complets de l'événement.

Méthode	Endpoint	Description
GET	<code>/api /events/: id</code>	Récupérer les détails d'un événement spécifique.

2.4 Gestion des Images des événements (Frontend)

Description

La prise en charge des images d'événements a été ajoutée dans : - Le formulaire de création et de modification des événements. - La page de détails des événements.

Les utilisateurs peuvent visualiser une image par défaut (si aucune image n'est fournie) ou une image personnalisée associée à l'événement.

Flux Utilisateur

1. Lors de la création ou modification d'un événement, l'utilisateur peut spécifier l'URL d'une image.
2. Si l'utilisateur ne renseigne pas d'image, une image par défaut est utilisée.
3. La page de détails affiche l'image associée à l'événement.

Endpoints Backend

Méthode	Endpoint	Description
GET	<code>/api /events/: id</code>	Récupérer les détails de l'événement, y compris l'URL de l'image.
POST	<code>/api /events</code>	Permet de créer un événement avec une image associée.
PUT	<code>/api /events/: id</code>	Permet de modifier l'image associée à un événement.
GET	<code>/api /val i date-image</code>	Permet de vérifier si une URL d'image est valide.

2.5 Filtre par Villes

Description

Un filtre par villes a été ajouté pour permettre aux utilisateurs de rechercher des événements en fonction de leur localisation.

Flux Utilisateur

- 1. L'utilisateur sélectionne une ville dans la liste déroulante des filtres.
- 2. Les événements affichés sont automatiquement filtrés pour correspondre à la ville sélectionnée.

Endpoints Backend

Méthode	Endpoint	Description
GET	/cities	Récupère les villes disponibles pour les événements.

Note : Les filtres sont appliqués côté frontend en combinant les critères de recherche pour offrir une expérience utilisateur optimale.

2.6 Améliorations UX/UI &

L'application a été remaniée graphiquement pour une meilleure expérience utilisateur :

- Nouvelle navbar fixe avec navigation fluide.
- Popup de notifications stylisé avec mise en forme propre.
- Suppression du bleu flashy et adoption d'un design plus pur.
- Animations CSS pour un rendu plus dynamique.
- Espacement et marges ajustés pour une meilleure lisibilité.

' Impact Métier & Valeur Ajoutée

Fonctionnalité	Valeur Ajoutée
# Notifications en temps réel	Permet aux utilisateurs d'être informés instantanément des actions importantes.
\$ Gestion des demandes d'inscription	Simplifie l'interaction entre utilisateurs, rendant le processus plus intuitif.
% Suivi des demandes envoyées	Apporte de la transparence sur l'état des interactions.
& Expérience utilisateur améliorée	Favorise l'adoption de la plateforme grâce à une interface plus intuitive et agréable.
' Recherche avancée des événements	Permet une recherche rapide et précise des événements grâce à la recherche floue, même avec des erreurs typographiques.

(Tests & Validation

¥ Notifications en temps réel : Fonctionnent sans latence.

¥ Gestion des statuts (pending, accepted, rejected) : Bien mise à jour en base.

¥ UI et UX fluides : Interface réactive et intuitive.

Documentation v1.0 ¶ Signalement de dangers

v1.0, Février 2025 :toc: :toc-title: Sommaire

! Introduction

La fonctionnalité de signalement de dangers permet aux utilisateurs de remonter en temps réel des incidents dans leur quartier. Cette feature repose sur un workflow rapide et efficace pour assurer une réactivité maximale.

(Objectif : Offrir une plateforme où les résidents peuvent signaler instantanément des problèmes de sécurité et autres nuisances, avec des notifications en temps réel via WebSockets.

Pourquoi cette feature ? -) Faciliter la communication locale : les utilisateurs peuvent informer leurs voisins d'un danger potentiel. - * Réactivité immédiate : les signalements sont visibles immédiatement et les dangers critiques envoient une notification. - + Amélioration de la sécurité : plus de transparence et de réactivité sur les incidents urbains.

) * Fonctionnalités

1. Section Signalement rapide -Via un formulaire dédié, les utilisateurs peuvent signaler un problème en quelques clics : - Sélection d'une catégorie parmi : * + Dangers & Sécurité (vol, bagarre, accident) * , Problèmes Urbains (routes endommagées, lampadaires HS) * - Nuisances Sonores (fête bruyante, klaxons) * . Problèmes de stationnement (véhicule gênant, parking saturé) - Description courte et zone du quartier concernée. - Option , Critique : Si activé par l'utilisateur lors de la saisie du formulaire, on envoie une notification immédiate aux résidents.

2. Section pour l'affichage des signalements - - Les 5 derniers signalements sont visibles sur le Dashboard, mis à jour en temps réel. - . A l'aide d'un bouton "voir plus", l'utilisateur peut voir en détail tous les signalements qui ont été faits, sur la page dédiée aux signalements.

3. Ajout de notifications WebSockets pour signaler le danger - Si le signalement est critique, une notification en temps réel est envoyée à tous les utilisateurs. - Mise à jour automatique du compteur de notifications. - Pas besoin de recharger la page : le signalement et les notifs sont instantanément visibles, ce qui permet à l'utilisateur de recevoir l'information sans faire d'effort particulier.

4. Section Mes signalements - Les utilisateurs peuvent consulter tous leurs signalements passés. - Marquer un signalement comme résolu pour indiquer que le problème a été traité. - Synchronisation avec la liste globale : Si l'alerte est résolue, elle apparaît aussi comme résolue pour tous.

/ Workflow du signalement de danger

```
@startuml
participant "Utilisateur" as UI
participant "Frontend (React)" as FE
participant "Backend API" as API
participant "Base de Données" as DB
participant "WebSockets" as WS
participant "Autres utilisateurs" as USERS

== ! 1. Signalement d'un danger ==
UI -> FE: Remplit le formulaire et valide
FE -> API: **POST** /signalements (catégorie, description, critiqueÉ)
API -> DB: " Enregistre le signalement
DB --> API: # OK

== / 2. Notification en temps réel si critique ==
API -> WS: % **mettre une notification ^ tous les utilisateurs**
WS --> USERS: & **Notification "Problème signalé"***
USERS -> FE: **Mise ^ jour immédiate du compteur de notifications**

== 0 3. Mise ^ jour du tableau de bord ==
API -> WS: **Mise ^ jour "Derniers signalements"***
WS --> FE: % Mettre ^ jour **sans recharger** 1

== 2 4. Gestion des signalements ==
UI -> FE: Accède ^ "3 Mes signalements"
FE -> API: **GET** /signalements/utilisateur/{user_id}
API -> DB: " Récupère les signalements de l'utilisateur
DB --> API: ( Renvoie la liste
API --> FE: Affichage des signalements

== # 5. Marquer un signalement comme résolu ==
UI -> FE: Clique sur "4* Marquer comme résolu"
FE -> API: **PUT** /signalements/:id/resoudre
API -> DB: # Met ^ jour le statut "Résolu"
DB --> API: **OK**

@enduml
```

0 DŽtails Techniques

/ Base de donnŽes - Table `signalements` : * `id` (INT, PRIMARY KEY) * `user_id` (INT, FOREIGN KEY vers `users`) * `categorie` (ENUM) * `description` (TEXT) * `critique` (BOOLEAN) * `quartier` (TEXT) * `resolu` (BOOLEAN, DEFAULT FALSE) * `date_creation` (DATETIME, DEFAULT CURRENT_TIMESTAMP)

Ź Table `notifications` (ajout du type `danger_alert`)

- ! `id`
- ! `user_id`
- ! `type` (ENUM)
- ! `message`
- ! `related_entity_id`
- ! `created_at`

/ Backend API (Node.js, Express, MySQL) - POST `/signalements` O CrŽe un nouveau signalement - GET `/signalements` O RŽcup•re tous les signalements - PUT `/signalements/:id/resoudre` O Marque un signalement comme rŽsolu - WebSockets : Notification temps rŽel via `io.emit("notification-global", {É})`

/ Frontend (React) - Composants * `SignalmentForm.jsx` O Formulaire de signalement * `SignalementsList.jsx` O Affichage des signalements * `Dashboard.jsx` O IntŽgration des signalements rŽcents * `Notifications.jsx` O Gestion des alertes en temps rŽel

1 Illustrations

/ Wireframe image::images/wireframe_signalement.png[]

/ Capture d'Žcran du site image::images/signalements_dashboard.png[]

' Comparaison avec les plateformes existantes

Notre solution se distingue par son approche temps rŽel et son interface ultra-rŽactive. Voici comment elle se positionne face aux alternatives existantes :

Plateforme	Type de signalement	Instantanéité des mises à jour	Notifications aux résidents	Suivi des signalements
AlloVoisins / Nextdoor	Discussions entre voisins, annonces de services	1 Non (les publications sont statiques)	1 Non (les notifications concernent uniquement des interactions sociales)	1 Non (pas de suivi des incidents)
DansMaRue (Paris)	Signalements urbains (voirie, éclairage public, etc.)	1 Non (validation requise par la mairie)	1 Non (aucune notification directe aux citoyens)	" Oui (suivi possible après traitement)
FixMyStreet	Problèmes d'infrastructure (routes, mobilier urbain)	1 Non (mises à jour manuelles)	1 Non (seules les autorités locales reçoivent les alertes)	" Oui (gestion par les services municipaux)
Notre application (Dangers, nuisances et incidents du quotidien	" Oui (mise à jour automatique en temps réel)	" Oui (alerte immédiate aux résidents en cas de danger critique)	" Oui (gestion et résolution directe par les utilisateurs)

2 Pourquoi nous sommes innovants ?

2 Rapidité & Instantanéité Notre solution utilise les WebSockets pour une mise à jour immédiate des signalements et une notification instantanée aux résidents.

(Autonomie des utilisateurs L'utilisateur peut signaler, suivre et clôturer un incident sans intervention administrative.

Notifications intelligentes Seuls les signalements critiques déclenchent une alerte pour éviter le spam tout en maintenant un haut niveau de réactivité.

3 Expérience utilisateur optimisée Interface fluide, ergonomique et conçue pour une utilisation rapide depuis un mobile ou un desktop.

Notre application comble un manque majeur dans la gestion des signalements en quartiers : l'instantanéité et l'autonomie des citoyens.

4 Conclusion : Contrairement à d'autres plateformes, notre application offre une communication rapide, directe et communautaire.

(Tests réalisés

¥ Tests unitaires : Vérification du bon enregistrement d'un signalement en base.

¥ Tests d'intégration : Simulation d'une notification critique et validation de son affichage en WebSockets.

¥ Tests REST API (Postman) :

! Envoi d'un signalement O 200 OK

! Marquer un signalement comme résolu O 200 OK

! Récupération des notifications en temps réel O (Fonctionnel

" Conclusion

- Bilan de la feature : - Instantanéité & efficacité avec WebSockets. - Expérience utilisateur fluide (mise à jour automatique des signalements et notifications). - Modularité & évolutivité (possibilité d'ajouter des filtres par quartier, historique des signalements).

5 Prochaines améliorations possibles : - Ajouter une cartographie interactive des signalements. - Permettre aux utilisateurs de commenter et réagir aux signalements. - Statistiques sur les types de signalements les plus fréquents. - Ajouter le temps réel pour dire à tous les utilisateurs qu'un signalement est désormais terminé.

(Feature livrée avec succès ! 6

Documentation v1.0 0 Gestion des Projets

Description

Cette fonctionnalité introduit la gestion complète des projets au sein de l'application. Les utilisateurs peuvent créer, modifier et supprimer des projets communautaires, voter pour un projet et suivre leur évolution. Les projets sont rattachés aux quartiers pour favoriser des initiatives locales et renforcer l'engagement des résidents.

Flux Utilisateur

1. Création d'un Projet

! L'utilisateur clique sur + Créer un projet.

! Il remplit un formulaire comprenant : titre, description, catégorie, date limite.

! Le projet est automatiquement associé au quartier de l'utilisateur.

! Une fois validé, le projet apparaît dans la liste des projets de son quartier.

2. Affichage des Projets

! Par défaut, seuls les projets du quartier de l'utilisateur sont affichés.

! Une case à cocher "Afficher tous les projets" permet de voir l'ensemble des projets disponibles.

3. Détails d'un Projet

! Un utilisateur peut cliquer sur un projet pour voir ses détails complets (créateur, description, votes, date limite).

! Si l'utilisateur est le créateur du projet, il peut le modifier ou le supprimer.

4. Modification d'un Projet (seulement pour le créateur)

! L'utilisateur accède aux détails de son projet et clique sur "Modifier".

! Un formulaire pré-rempli lui permet de mettre à jour les informations.

! Après validation, les modifications sont enregistrées en base et affichées en temps réel.

5. Suppression d'un Projet (seulement pour le créateur)

! L'utilisateur clique sur "Supprimer".

! Une confirmation s'affiche pour éviter toute suppression accidentelle.

! Le projet est définitivement supprimé.

6. Votes sur un Projet

! Les utilisateurs peuvent voter pour ou contre un projet (3 Upvote ou 4 Downvote).

! Un utilisateur ne peut pas voter pour son propre projet.

! Les votes sont mis à jour en temps réel sans rechargement de la page.

! Une fois la période de votes terminée, un projet est accepté ou rejeté en fonction du nombre de votes positifs/négatifs.

Endpoints Backend

Méthode	Endpoint	Description
POST	/api/projects	Créer un projet
GET	/api/projects	Récupérer tous les projets (avec option quartier/tous les projets)
GET	/api/projects/:id	Récupérer les détails d'un projet
PUT	/api/projects/:id	Modifier un projet (seulement si l'utilisateur est le créateur)
DELETE	/api/projects/:id	Supprimer un projet (seulement si l'utilisateur est le créateur)
POST	/api/projects/:id/vote	Voter pour un projet (3 / 4)

Diagramme de S quence : Gestion des Projets

```
@startuml
actor "Utilisateur" as User
participant "Frontend (React)" as FE
participant "Backend API" as API
participant "Base de Donn es" as DB

== 5 1. Cr ation d un Projet ==
User -> FE: Clique sur "Cr er un projet"
FE -> API: **POST** /api/projects (titre, description, cat gorie, deadline, quartier_id)
API -> DB: " V rifie les donn es et ins re le projet
DB --> API: # OK (id_projet)
API --> FE: Confirme la cr ation et met ^ jour la liste des projets

== 5 2. Affichage des Projets ==
User -> FE: Acc de ^ la page "Projets"
FE -> API: **GET** /api/projects?quartier_id=X
API -> DB: " R cup re les projets du quartier
DB --> API: ( Liste des projets filtr s
API --> FE: Affichage des projets

== 5 3. Modification d un Projet ==
User -> FE: Ouvre son projet et clique sur "Modifier"
FE -> API: **PUT** /api/projects/:id (nouvelles valeurs)
API -> DB: # Met ^ jour le projet
DB --> API: ( Confirme la mise ^ jour
API --> FE: Affichage des nouvelles valeurs

== 5 4. Suppression d un Projet ==
User -> FE: Clique sur "Supprimer"
FE -> API: **DELETE** /api/projects/:id
API -> DB: . Supprime le projet
DB --> API: # Suppression confirm e
API --> FE: Met ^ jour la liste des projets

== 5 5. Vote sur un Projet ==
User -> FE: Clique sur "6" ou "7"
FE -> API: **POST** /api/projects/:id/vote (vote=up/down, user_id)
API -> DB: " V rifie si l utilisateur a d j ^ vot 
DB --> API: # OK
API -> DB: $ Met ^ jour le vote
DB --> API: ( Retourne le nouveau compteur de votes
API --> FE: Affichage des votes mis ^ jour

@enduml
```

' Impact MŽtier & Valeur AjoutŽe

FonctionnalitŽ	Valeur AjoutŽe
) Projets rattachŽs aux quartiers	Favorise les initiatives locales et renforce le lien social.
" Gestion compl•te (CRUD)	Permet aux utilisateurs de crŽer, modifier et supprimer leurs projets en toute autonomie.
78 Votes en temps rŽel	Donne un retour direct sur lŽntŽr•t du projet aupr•s de la communautŽ.
5 VisibilitŽ optimisŽe	Les projets sont mis en avant selon leur popularitŽ et leur pertinence.

(Tests & Validation

Ÿ Tests unitaires :

! CrŽation, modification et suppression dŽun projet O " OK

! Votes sur un projet O " OK

Ÿ Tests dŽintŽgration :

! Validation de lŽaffichage des projets filtrŽs par quartier O " Fonctionnel

! Test de lŽoption "Afficher tous les projets" O " Fonctionnel

Ÿ Tests REST API (Postman) :

! POST /api/projects O 201 Created

! GET /api/projects (avec quartier_id) O 200 OK

! PUT /api/projects/:id (modification) O 200 OK

! DELETE /api/projects/:id O 200 OK

! POST /api/projects/:id/vote O 200 OK

" Conclusion

- Bilan de la feature : - " CrŽation et gestion des projets simple et fluide - " Filtrage intelligent des projets selon le quartier - " Syst•me de votes participatif pour la validation des projets - " Interface optimisŽe et ergonomique

5 Prochaines amŽliorations possibles : - Ajout dŽune gestion des tŽches par projet (Kanban). - Syst•me de commentaires sur les projets. - Ajout dŽun statut de projet (En cours, TerminŽ, etc.).

(Feature livrŽe avec succ•s ! 6

Documentation Technique Ð Release v1.1 - DevOps2

Objectif de la fonctionnalité

Permettre aux utilisateurs de :

1. Se connecter à leur compte Google via un bouton dédié.
2. Visualiser leurs Événements Google Calendar dans le dashboard de la plateforme.
3. Ajouter automatiquement un Événement à leur Google Calendar lorsqu'ils cliquent sur le bouton "Participer" à un Événement.

Cette intégration offre une expérience fluide et connectée, évitant aux utilisateurs d'avoir à gérer manuellement leur emploi du temps après s'être inscrits à un Événement.

" Nouveauté côté utilisateur

Avant cette release, l'utilisateur ne pouvait ni s'inscrire à un Événement, ni le quitter, et aucune synchronisation n'existait avec son agenda personnel.

Avec cette intégration, il peut désormais :

- ¥ Participer ou quitter un Événement local directement depuis l'interface KnockNShare ;
- ¥ Ajouter automatiquement cet Événement à son propre Google Calendar (avec lieu, date, heure, description) ;
- ¥ Et surtout, visualiser en temps réel ses Événements Google, y compris ceux ajoutés via KnockNShare, depuis le dashboard de l'application.

Cette avancée rapproche la plateforme d'un véritable assistant de vie communautaire connecté, conforme à notre vision de simplification des interactions sociales au sein des quartiers.

5 Fonctionnalités livrées

- ¥ Connexion OAuth2.0 à Google (frontend) avec affichage des Événements à venir.
- ¥ Ajout automatique d'un Événement Google Calendar lors du clic sur "Participer".
- ¥ Conservation du token d'accès dans un contexte React (`GoogleAuthContext`) avec mise à jour automatique.
- ¥ Bouton "Participer" fonctionnel : interaction avec la base de données + appel API Google Calendar.

Authentication Google (OAuth2) & Intégration initiale

Objectif

Permettre aux utilisateurs de se connecter à KnockNShare via leur compte Google, sans avoir à créer un compte ou à renseigner un mot de passe. Cette étape est également un prérequis technique à l'intégration du calendrier Google.

Fonctionnalités livrées

- ¥ Redirection de l'utilisateur vers la page d'authentification Google.
- ¥ Décodage du `id_token` pour obtenir les données de base (`email`, `name`, `google_id`).
- ¥ Vérification de l'existence de l'utilisateur en base, création automatique si inexistant.
- ¥ Génération d'un JWT signé, transmis au frontend via redirection.
- ¥ Stockage du `userId` et de l'`access_token` dans le `localStorage` (clé `googleAccessToken`) pour les requêtes vers l'API Calendar.

Implémentation technique

Backend (Node.js/Express)

- ¥ Ajout des routes suivantes :

```
GET /api/auth/google // redirection vers Google
GET /api/auth/google/callback // traitement du code + création/utilisateur
```

- ¥ Ajout du scope :

```
scope: "openid profile email https://www.googleapis.com/auth/calendar.readonly"
```

- ¥ Décodage du `id_token` avec `jsonwebtoken`, génération d'un JWT interne avec `userId`, `email`, etc.
- ¥ Enregistrement des utilisateurs Google en base (table `users`) avec `google_id`, sans mot de passe.

Frontend (React)

- ¥ Ajout d'un bouton "Se connecter avec Google" sur la page de connexion (`LoginPage.jsx`).
- ¥ Ajout d'une page `OAuthSuccess.jsx` qui :
- ¥ lit le `token` et l'`access_token` dans l'URL,
- ¥ les stocke dans `localStorage`,

- met à jour l'`AuthContext`,
- redirige l'utilisateur vers le dashboard.

Migration SQL associée

Ajout de la colonne `google_id` dans la table `users` et passage de `password` en nullable :

```
ALTER TABLE users ADD COLUMN google_id VARCHAR(255);
ALTER TABLE users MODIFY COLUMN password VARCHAR(255) NULL;
```

Fichier de migration : `update_users_schema.sql`

Configuration

Ajout des variables suivantes dans le fichier `.env` :

```
GOOGLE_CLIENT_ID=...
GOOGLE_CLIENT_SECRET=...
GOOGLE_CALLBACK_URL=http://localhost:3000/api/auth/google/callback
```

Le `access_token` Google est désormais stocké dans le navigateur et peut être utilisé par les autres fonctionnalités (ex : synchronisation Calendar).

Intégration Google Calendar & Détails techniques

1. Authentification OAuth

Le composant `GoogleAuthProvider.jsx` initialise et configure `gapi.auth2` :

- récupère et stocke le token.
- écoute les connexions/déconnexions avec `auth.isSignedIn().listen`.
- expose `signIn()` et `token` via `GoogleAuthContext`.

Scope utilisé :

```
const SCOPES = "https://www.googleapis.com/auth/calendar.events";
```

2. Affichage des Événements Google (dashboard)

Dans `Dashboard.jsx` :

¥ Le bouton Connecter Google Calendar lance `signIn()`.

¥ Si un `token` est présent, les événements sont récupérés via :

```
gapi.client.calendar.events.list({...})
```

¥ Le composant `DashboardCalendar.jsx` affiche ces événements.

3. Création d'un événement et la participation

Dans `EventPage.jsx`, lors du clic sur Participer :

¥ Ajout du participant via :

```
POST /api/events/participe
```

¥ Création d'un événement Google Calendar :

```
POST https://www.googleapis.com/calendar/v3/calendars/primary/events
Headers: Authorization: Bearer access_token
```

Payload envoyé :

```
{
  "summary": "Titre",
  "description": "Description",
  "location": "Adresse",
  "start": { "dateTime": "...", "timeZone": "Europe/Paris" },
  "end": { "dateTime": "...", "timeZone": "Europe/Paris" }
}
```

4. Backend (routes Node/Express)

```
POST /api/events/participe // ajoute des participants
DELETE /api/events/leave // supprime des participants
```

Gestion de la base MySQL avec vérification des doublons.

Architecture technique

Composant/Fichier	Rôle
<code>GoogleAuthProvider.jsx</code>	Gère l'authentification OAuth et expose le token

Composant/Fichier	R��le
Dashboard.j s x	Affiche les ��v��nements et d��clenche la connexion
DashboardCalendar.j s x	Affiche les ��v��nements Google Calendar
EventPage.j s x	G��re la logique de participation + ajout Google Calendar
/api /events/partici pate	Backend : enregistre la participation
/api /events/leave	Backend : supprime la participation

Probl  mes rencontr  s

-    Expiration du token : contourner avec `isSignedIn` dans le contexte
-    Client ID multiple : risque de conflit,    s  curiser dans un `.env`.
-    Perte de session inter-composant : r  solue avec `GoogleAuthContext`.

B  n  fices de cette feature dans la strat  gie globale de notre application

Cette fonctionnalit   am  liore consid  rablement l'exp  rience utilisateur :

-    Meilleure exp  rience pour l'utilisateur : automatisation de l'ajout d'  v  nements, la participation est plus simple et rapide, et appara  t dans le calendrier personnel de l'utilisateur
- Gain de temps O plus besoin de noter l'  v  nement dans un agenda externe.
- Moins d'oubli O les rappels automatiques Google s'appliquent.
- Notre plateforme KnockNshare devient un vrai outil d'organisation de vie communautaire.

Tests r  alis  s

-    Connexion OAuth fonctionnelle.
-    Visualisation des   v  nements dans le calendrier.
-    Ajout d'un   v  nement fonctionnel avec token valide.
-    D  connexion/reconnexion g  r  e automatiquement.

Release v1.1    R  capitulatif

-    Int  gration compl  te OAuth (Google Calendar)
-    Ajout automatique d'  v  nements
-    Visualisation des   v  nements Google
-    Reconnexion et gestion de session

