BairroSeguro – Prompt para Agente de IA (MVP v0.1)

1. Visão Geral do Projeto

Crie um aplicativo Android nativo para alertas comunitários de segurança em tempo real, destinado a bairros ou condomínios no Brasil. O MVP deve permitir que moradores criem/recebam alertas, visualizem um feed com mapa, e contem com moderação básica – tudo com custo zero em infraestrutura até o app atingir escala paga.

2. Stack de Tecnologias

Camada	Tecnologia
Mobile	Kotlin 1.10.x + Jetpack Compose + Hilt + MVVM
Offline	Room DB (histórico local), WorkManager (sync)
Auth	Firebase Authentication (e■mail link; SMS opcional p/ moderador)
Realtime & Push	Cloud Firestore (modo native) + Cloud Functions (Node 20) + Firebase Cloud Messaging
Mapas	Google Maps SDK for Android (100 k tiles/mês grátis)
Infra	Firebase Hosting (landing + painel web leve), GitHub Actions (CI/CD)
Analytics & Crash	Firebase Analytics, Crashlytics

3. Funcionalidades MVP (Must■Have)

• Registro via e■mail magic■link (Firebase Auth).

• Entrar em grupo por geofencing (GPS) ou código/QR gerado por moderador.

• Botão único “Alerta!” com 4 categorias: Crime, Emergência Médica, Perigo Ambiental, Animal Perdido.

• Envio de push imediato a todos no grupo usando FCM topic `group_{groupId}`.

• Feed cronológico (últimas 24 h) e mapa com pins agregados.

• Moderação: apagar alerta, bloquear usuário por 24 h.

• Histórico offline: últimos 30 alertas armazenados em Room DB.

• Configuração de privacidade LGPD: optar por precisão reduzida (arredondamento 100 m).

4. Requisitos Não Funcionais

• Latência ponta■a■ponta < 5 s para entrega de push.

• Escalar até 5 000 usuários ativos por bairro sem custos além do free■tier.

• Criptografia em repouso (Firestore e Room).

• Uptime 99 % horário comercial; fallback local se API cair.

• Compatibilidade TalkBack (acessibilidade).

• minSdkVersion 26 (Android 8).

5. Modelo de Dados Firestore

```
Coleções principais:

1. /groups/{groupId}
  - name: string
  - geoHash: string // GeoHash do centro
  - centerLat, centerLon: number
  - radiusMeters: number (ex. 500)
  - createdAt: timestamp
  - createdBy: uid
  - moderatorIds: array
  - memberCount: number

2. /users/{uid}
  - displayName, email, phone
  - groups: array // groupIds
  - roles: array // ["basic"] ou ["moderator"]
  - createdAt, lastSeen

3. /alerts/{alertId}
  - groupId: string
  - category: string // "crime" | "medical" | "hazard" | "animal"
  - message: string?
  - lat, lon: number
  - roundedLat, roundedLon: number // 2 dec. p/ privacidade
  - createdBy: uid
  - createdAt: timestamp
  - status: string // "approved" | "pending" | "removed"
```

6. Cloud Functions (Node 20)

```
1. sendAlert (trigger onCreate /alerts)
  - Se alert.status == "approved": envia FCM topic group_{groupId}.

2. moderateAlert (callable)
  - Somente moderatorIds podem alterar status para "approved" ou "removed".

3. cleanupOldAlerts (scheduled daily)
  - Apaga documentos /alerts com createdAt < now() - 30 dias.

4. joinGroup (callable)
  - Adiciona userId em groups, incrementa memberCount.

5. createGroup (callable)
  - Cria doc /groups, define usuário como moderador.
```

7. Regras de Segurança Firestore (trecho)

```
rules_version = '2';
service cloud.firestore {
 match /databases/{database}/documents {

  match /groups/{groupId} {
   allow read: if request.auth != null && groupId in request.auth.token.groups;
   allow create: if request.auth != null;
   allow update, delete: if request.auth != null &&
    request.auth.uid in resource.data.moderatorIds;
  }

  match /alerts/{alertId} {
   allow read: if request.auth != null &&
    resource.data.groupId in request.auth.token.groups;

   allow create: if request.auth != null &&
    request.resource.data.groupId in request.auth.token.groups &&
    request.auth.uid == request.resource.data.createdBy;

   allow update: if request.auth != null &&
    request.auth.uid in
get(/databases/$(database)/documents/groups/$(request.resource.data.groupId)).data.moderatorIds;
  }
 }
```

```
}
```

8. Estrutura do App Android

```
Root Gradle
■■■ app/        // UI (Compose), Activities
■■■ core/       // models, utils
■■■ data/       // Firebase impl., Room
■■■ domain/      // use-cases
■■■ cloud/       // Functions client wrapper
■■■ test/       // unit tests
Arquitetura: Clean Architecture (MVVM) + Hilt DI + Kotlin Coroutines + Flow.
```

9. Dependências Gradle sugeridas

```
implementation("androidx.compose.ui:ui:1.7.0")
implementation("androidx.activity:activity-compose:1.9.0")
implementation("com.google.dagger:hilt-android:2.52")
kapt("com.google.dagger:hilt-compiler:2.52")
implementation("com.google.firebase:firebase-auth-ktx:23.0.0")
implementation("com.google.firebase:firebase-firestore-ktx:26.0.0")
implementation("com.google.firebase:firebase-messaging-ktx:24.1.0")
implementation("com.google.android.gms:play-services-maps:18.2.0")
implementation("androidx.room:room-runtime:2.7.0")
kapt("androidx.room:room-compiler:2.7.0")
implementation("androidx.work:work-runtime-ktx:2.9.0")
testImplementation("junit:junit:4.13.2")
androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
```

10. Estratégia de Testes

• Unitários: ViewModels, Repositórios (Mockito/Kotlin Coroutines Test).

• Room DAO tests com in■memory DB.

• Instrumentados (Espresso): fluxo de onboarding e alerta.

• Cloud Functions: emulador Firebase para testes de regras e chamadas.

11. Pipeline CI/CD (GitHub Actions)

```
jobs:
 build:
  runs-on: ubuntu-latest
  steps:
   - uses: actions/checkout@v4
   - uses: gradle/gradle-build-action@v3
    with:
     arguments: clean lint testDebug assembleDebug
   - name: Upload APK
    uses: actions/upload-artifact@v4
    with:
     name: app-debug
     path: app/build/outputs/apk/debug/app-debug.apk

 deploy-functions:
  needs: build
  runs-on: ubuntu-latest
  steps:
   - uses: actions/checkout@v4
   - uses: w9jds/firebase-action@v13
    with:
     args: deploy --only functions
    env:
     FIREBASE_TOKEN: ${{ secrets.FIREBASE_TOKEN }}
```

12. Checklist de Entregas

- [ ] Configurar projeto Firebase (Auth, Firestore, FCM)
- [ ] Provisionar Map SDK key
- [ ] Setup repositório e CI
- [ ] Tela de onboarding + Auth
- [ ] Criação/entrada em grupo
- [ ] Publicar alerta + push
- [ ] Feed + Mapa
- [ ] Moderação
- [ ] LGPD/privacidade
- [ ] Testes unitários & UI
- [ ] Deploy Functions
- [ ] Landing page + Play Console beta