William Lindblom

GeoFeed Dokumentation

Introduktion

Målgrupp

Målgruppen för appen är individer förenade genom en geografisk plats som vill kommunicera och hålla sig uppdaterade kring det geografiska området individen befinner sig i.

Beskrivning

Har du någonsin besökt en ny stad och undrat var den bästa kebaben finns eller sökt någon i närheten att utmana i en padel match?
Med GeoFeed kan du kommunicera med personer i staden du befinner dig i på ett smidigt sätt. Det krävs ingen registrering eller användarnamn för att använda eller publicera inlägg i applikationen.

Vid publicering av inlägg eller kommentar lagras:

- Namn på stad och stadsdel användaren befinner sig i.
- Användaridentifiering för att användaren ska kunna visa sina egna inlägg.

Betygsambitioner

G

Säkerhet och etik

Även om utvecklingen av applikationen enbart gjordes i utvecklingssyfte och det i skrivande stund inte finns några ambitioner att publicera applikationen till allmänheten så finns det säkerhets- samt etiska aspekter som berör GeoFeed. Till att börja med så använder GeoFeed användarens position och lagrar namnet på staden samt stadsdelen för varje nytt inlägg samt kommentar som publiceras. Dessa lagras tillsammans med ett unikt användar-id.
Även om det är väldigt individuellt huruvida användare tycker detta är känslig data eller inte är det viktigt att användaren är medveten om att denna data lagras då inlägg eller kommentarer publiceras. Användaren kan göras medveten om detta dels genom att den i likhet med andra applikationer som behöver åtkomst till plats låter användaren välja om den vill acceptera detta och dels genom av att i beskrivningen skriva vilken potentiellt känslig data applikationen sparar om användaren.
Vidare finns ur ett säkerhetsperspektiv vissa risker med GeoFeed. Vid en eventuell dataläcka skulle åsikter inom vissa geografiska områden kunna kartläggas och användas på ett sätt som motstrider applikationens syfte. Användarna i GeoFeed är inte anonyma i den mån att ingen information sparas vid inlägg, tvärt om så kan varje inlägg eller kommentar

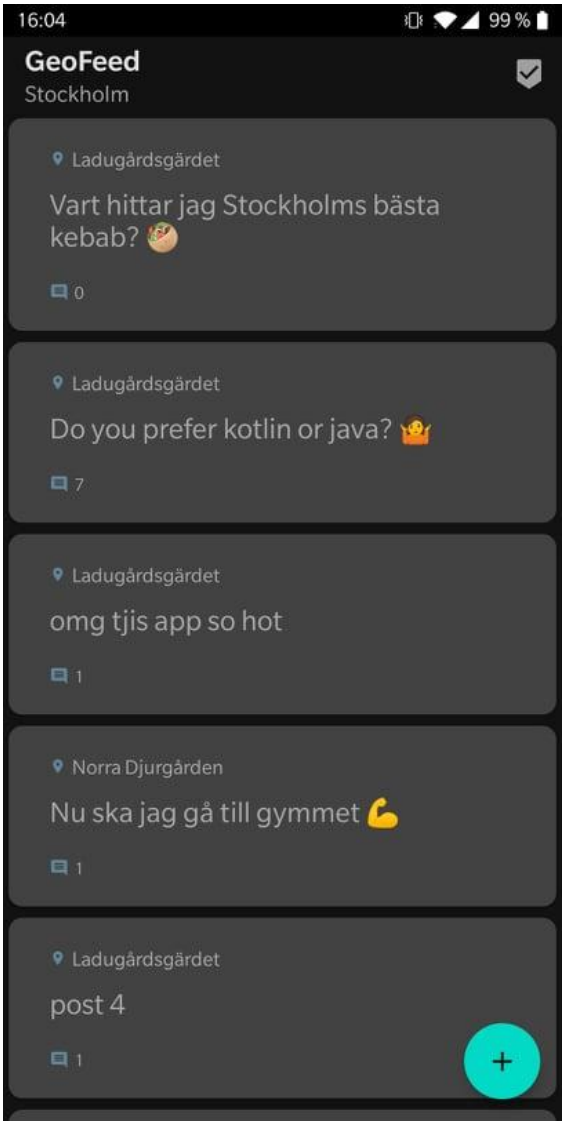
kopplas till en specifik användare. En dataläcka skulle på så sätt kunna användas för att hitta lämpliga individer att påverka i inkräktarens intresse.
 Från användarens perspektiv är dock inläggen och kommentarerna anonyma i den mån att det inte från gränssnittet går att se vem som postat ett inlägg eller kommentar. Detta kan medföra att användare är mer öppna men också risken för användare sprider hatiska kommentarer och håller en hård jargong då de kan publicera i dunklet av skärmen. Om applikationen skulle publiceras till allmänheten hade det därför varit bra att föra en diskussion om någon form av moderering ska ske och hur den ska implementeras för att användingen av applikationen ska följa dess syften.

Användarhandledning

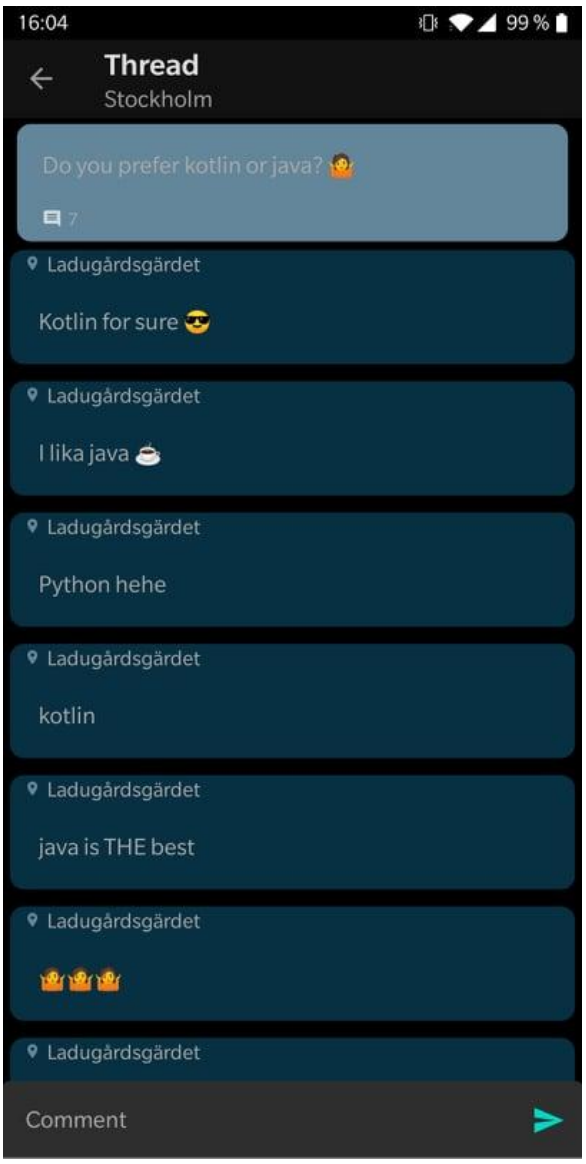
När GeoFeed startas första gången visas en dialog där användaren får acceptera att använda åtkomst till enhetens plats. Om användaren inte accepterar detta är applikationen obrukbar.

När användaren har accepterat åtkomst till enhetens plats visas hemskärmen med inlägg publicerade i staden där användaren befinner sig. Förutom de publicerade inläggen innehåller hemskärmen en action bar högst upp med information (vänster) som visar vilken stad användaren befinner sig i samt en knapp (höger) som tar användaren till en vy för sina

publicerade inlägg. Längst ner i högra hörnet på hemskärmen finns en floating action button (grön knapp med plustecken) som tar användaren till vyn för att skapa ett nytt inlägg.

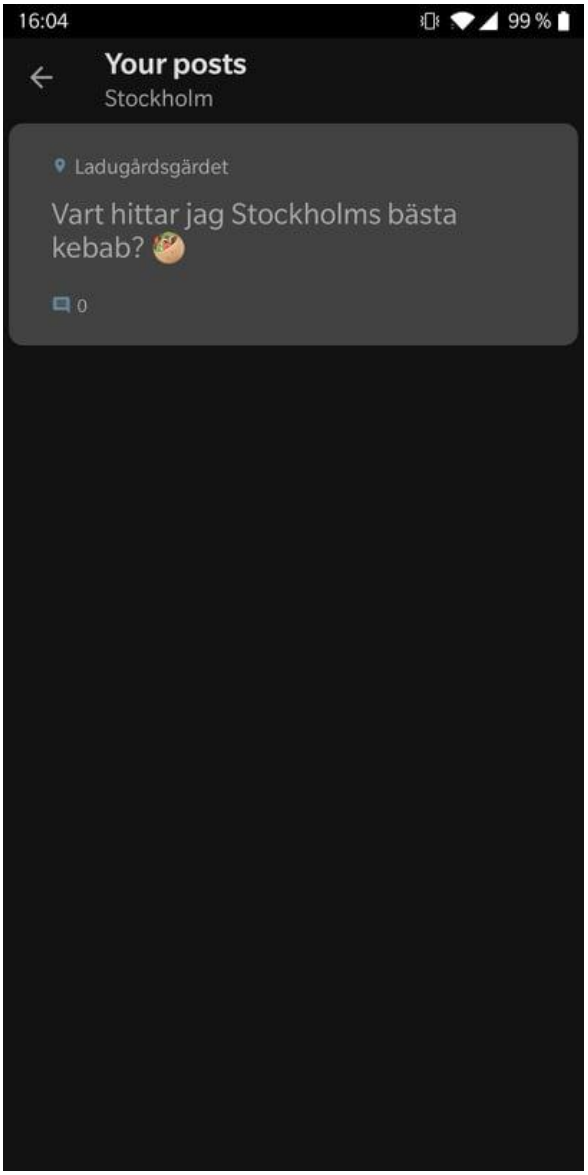


Genom av att klicka på ett av inläggen i listan navigerar sig användaren till vyn för att visa det specifika inlägget.



Ovan visas vyn för att visa ett specifikt inlägg. Högst upp finns en action bar där användaren kan navigera sig tillbaka till tidigare vy. Med ljusare nyans visas här texten i inlägget och hur många som har kommenterat inlägget. I mörkare nyans visas kommentarer till inlägget tillsammans med stadsdelen användaren som kommenterar befann sig i när kommentaren skrevs.

Längst ner på skärmvyn finns ett textinmatningsfält där användaren kan skriva en kommentar till inlägget. Därefter kan användare ladda upp kommentaren genom av att klicka på skicka knappen som återfinns längst till höger vid textinmatningsfältet.

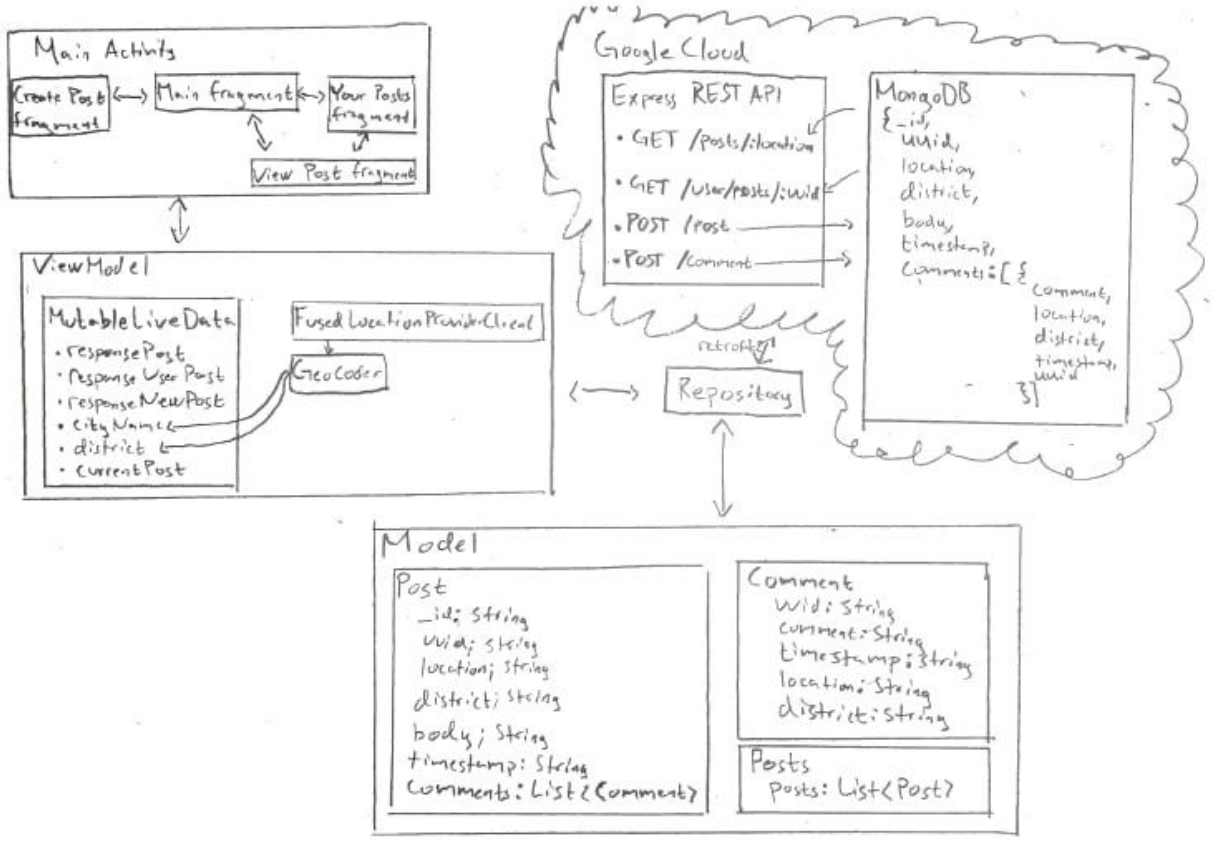


Då användaren befinner sig på hemskärmen och klickar på knappen högst upp till höger kommer denne till skärmvyn ovan. Den här skärmvyn visar inlägg som har gjorts av användaren tidigare oavsett vilken stad användaren befann sig i när dessa publicerades. I likhet med hemskärmen kan användaren här klicka på ett inlägg för att visa det specifika inlägget med dess kommentarer. Om användaren klickar på bakåtknappen högst upp till vänster tar denne sig tillbaka till hemskärmen.

Om användaren från hemskärmen klickar på applikationens floating action button (grön knapp med plustecken) visas skärmvyn ovan. Skärmvyn finns för att användaren ska kunna skapa ett nytt inlägg. Högst upp på skärmvyn finns en action bar med en bakåtknapp längst till vänster som tar användaren tillbaka till hemskärmen. Därefter visas titeln med underrubriken med staden användaren befinner sig i. Längst till höger i action baren finns en knapp där användaren publicerar sitt inlägg. Nedaför skärmvyns action bar finns ett textinmatningsfält som tar upp resterande del av skärmen, det är i den användaren skriver sitt inlägg denne vill publicera.

William Lindblom

Lösningens uppbyggnad



Lösningen består av två delar, Android applikationen skriven i Kotlin samt en backend bestående av en server med ett REST API i express.js samt MongoDB som databas.

Android applikationen består av en aktivitet med fyra fragments (Se bilder under Användarhandledning). För att navigera mellan fragments används Navigation component. Samtliga fragments delar ViewModel där data, så som rensponse objekt, plats relaterad data samt Post objekt, lagras för att senare visas upp i gränssnittet. Detta sker genom av att varje fragment observerar den data den behöver och därefter uppdaterar gränssnittet. Uppdatering av datan i applikationens ViewModel varierar beroende på vilken typ av data det är. För plats relaterade data uppdateras den genom GoogleAPIs FusedLocationProviderClient som lyssnar på nya koordinater, därefter används Geocoder för att få fram stad samt stadsdel. För http response objekt uppdateras denna genom requests till servern. Dessa requests sker med hjälp av biblioteket retrofit2 samt Kotlins korutiner. För Post objektet, som innehåller det inlägget användaren har klickat på för att ta sig till skärmvyn där det specifika inlägget visas, så kan det uppdateras på två sätt. Antingen genom klicket där användaren tar sig till skärmvyn, det uppdateras då internt i applikationens ViewModel med det medskickade objektet. Det andra sättet det kan ske på är om användaren publicerar en ny kommentar som då triggar en post request vars response

innehåller det uppdaterade Post objektet med den nya kommentaren tillagd vilket sedemera uppdaterar Post objektet i applikationens ViewModel.

Reflektion

Arbetet med uppgiften varierade och det var ingen självklar väg till lösningen. Till att börja med var tanken att varje skrämvy skulle vara en egen aktivitet men då jag senare kom fram till att jag behövde ha en delad ViewModel mellan olika skärmvyer togs beslutet att återimplementera dessa som fragments istället. Detta då, efter vad jag förstått, det inte går att ha en delad ViewModel mellan olika aktiviteter. Samtidigt baserades valet att använda sig av ViewModel delvis på att ha en central lagring av data abstraherad från skärmvyerna och delvis på att datan skulle behöva uppdateras från olika skärmvyer. Ett exempel på när datan behöver uppdateras från olika skärmvyer är när en användare publicerar ett nytt inlägg. Vid en lyckad publicering navigeras användaren från skärmvyn för att skapa nya inlägg till hemskärmen, i samband med detta vill jag på ett smidigt sätt kunna uppdatera inläggen på hemskärmen vilket då sker genom en metod i ViewModel klassen. Med det sagt så är ViewModel klassen relativt stor med mycet ansvar och hade eventuellt kunnat delas upp i ytterligare klasser.

Eftersom att användare skulle kunna kommunicera med varandra krävdes någon form av backend implementation. Valet av den backend stack som användes baserades huvudsakligen på min tidigare erfarenhet. Vad som dock blev problematiskt var problem som uppstod med Google Cloud konfigurering vilket sedemera tog tid från Android utvecklingen. Ett alternativ hade här varit att använda firebase istället men då jag inte har någon tidigare erfarenhet av detta och det inte hör till kursen tror jag inte jag hade vunnit något på det.

Från början var tanken att användare även skulle gilla och ogilla inlägg och att det i applikationen således skulle gå att sortera inläggen dels baserat på tid (som det är nu) och dels baserat på flest gilla-markeringar. På grund av tidsbrist var jag dock tvungen att begränsa mig och detta implementerades därför inte.

Utöver det så är jag i sin helhet nöjd med min lösning och kan inte komma på något uppenbart som jag borde gjort annorlunda. Detta kan dock eventuellt bero på att jag är i början på mitt lärande inom android utveckling och därför har svårare att se ett bredare perspektiv och bästa praxis för olika ändamål. Vidare har jag utveklats mycket av projektet och lärt mig mycket mer om både Android utveckling, hur platssensorer kan användas, Kotlin med dess korutiner samt kommunikation med extern API.