Manual for Mina fotavtryck

en applikation för GPS-tracking

Innehållsförteckning

Klasser

För att kunna köra Googles tjänster så krävs en API-nyckel som läggs i AndroidManifest.xml.

Aktiviteter

LoginActivity

Vid uppstart så kollas om det finns någon inloggad användare sen innan och i så fall görs kontrollen mot servern om användaren har en giltig token. Om så är fallet så pushar den nya positioner mot servern och användaren skickas vidare till MainActivity. Annars loggas alla ut och användaren får möjligheten att logga in eller skapa ett nytt konto. Vid inloggning så pushas också nya positioner och en ny token erhålls.

CreateAcccountActivity

Tar emot en epostadress och ett lösenord för att skapa ny användare. Om det är giltiga värden så kallar den på WebHandler för att kontrollera i servern om användaren redan finns. Om inte så skapas en ny.

MainActivity

Härifrån kan användaren starta och stoppa GPS-trackingen, öppna kartan samt logga ut. Nya positioner behandlas i handleNewLocatrion() där det läggs in i den lokala databasen. I fallet där användaren inte tillåtit Mina fotavtryck att använda enhetens GPS så behandlas felmeddelandet i onRequestPermissionResult(). Båda dessa funktionerna kallas på från LocationProvider. När trackingen avslutas så pushas de nya punkterna mot servern.

MapsActivity

Klassen hämtar alla positioner från databasen och begär en lista med hopslagna och avrundade punkter beroende på vilken zoomnivå det är på kartan. När zoomnivån ändrat sig tillräckligt mycket så uppdateras kartan. Kontroll av zoomnivå sker i onCameraIdle() och updateringen sker i update(). I update() går det att bestämma markörernas utseende gällande markörbild, transparens och z-index. Genom att ändra sifforna i if-satserna i onCameraIdle så kan man bestämma när markörernas beteende ska skifta. Alltså vid vilken zoomnivå punkter börjar hamna så tätt att det blir lämpligt att slå samman flera punkter till en tätare punkt. Om ändringar i zoomnivån görs här måste också switch-satsen i roundDown() i ZoomposContainer ändras.

Hjälpklasser

DBHelper

Tar hand om insättningar och hämtningar från den lokala databasen. Vid initiering av ett DBHelper-objekt så kontrolleras om DATABASE_VERSION har ändrats. Den konstanten ska man ändra på när man har gjort ändringar i databasens struktur till exempel lagt till en ny kolumn. Om versionen har ändrats så tas alla gamla tabeller bort och nya skapas. InsertOne() sätter in en ny position i taget och används av MainActivity när trackingen är igång. InserMultiple() används när flera punkter har hämtats från servern. Det går att hämta antingen alla punkter på en gång eller alla punkter efter ett angivet id. GetAllEntries() används av MapsActivity och getAfter() används när de senaste punkterna ska pushas mot servern.

DBUsers

Fungerar på samma sätt som DBHelper men istället för att innehålla en användares alla positioner så innehåller denna databasen användarna själva. Det går att lägga till en ny användare med insert(), logga ut alla användare med logout(), uppdatera till en ny token med updateToken(), kolla om någon är inloggad med isLoggedIn(), logga in en användare med login() samt kolla om en användare redan finns med exist(). Denna klassen används av LoginActivity, CreateAccountActivity och WebHandler.

LocationProvider

Denna klassen sköter tillgången till enhetens GPS-positioner. Den skapar en positionsförfrågan och frågar efter nya positioner så ofta som man angett i UPDATE_INTERVAL och om man har förflyttat sig minst så långt som är angett i UPDATE_EACH_METERS. OnLocationChanged() lyssnar på positionsförändringar och när en sådan sker så skickas den till MainActivity. Om användaren inte har tillåtit Mina fotavtryck att använda GPS så hanteras det i MainActivity, men kontrollen utförs i getPermission().

RawPosition

En RawPosition är vad en position består av. Den innehåller ett id, longitud, latitud, när den skapades samt till vilken session den tillhör. En session varar från att användaren klickar på starta tracking tills stoppa tracking klickas. Session nyttjas inte i applikationen med stödet finns.

User

User håller koll på vilken användare som använder applikationen just nu på den lokala enheten. En användare identifieras med dess epost. Från User kan även hämtas aktuell token samt id på senast insatta positionen på serversidan, för att veta vilka positioner som behöver pushas.

WebHandler

All kommunikation mot servern går via WebHandler. Det finns fyra metoder och de är createUser(), login(), pull() och push(). Alla funktioner skapar ett JSON-objekt och skickar det till varsin webbsida vars adresser anges med PUSH_URL, PULL_URL, LOGIN_URL och CREATEUSER_URL. Servern svarar med en JSON som parsas och behandlas. Applikationen behöver vara seriell för att saker och ting behöver ske i en viss ordning och därför har inte trådar använts i andra klasser. Det nätverks-API som använts krävde dock trådanvändning så därför används trådar i WebHandler och en extra tråd används när positioner pushas eftersom det inte behöver ske seriellt.

ZoomPos

En ZoomPos är en avskalad RawPosition. Den lagrar en longitud och latitud fast med färre decimaler beroende på vilken zoomnivå den skapats för. Den klarar även av att lagra flera likadana punkter med hjälp av occurance.

ZoomPosContainer

ZoomPosContainer kallas på från MapsAcitivty när nya markörer behövs. Konstruktorn tar emot en lista med RawPosition och skapar en ny lista med ZoomPos. ZoomPos får olika många decimaler beroende på vilken zoomnivå som angetts. Avskalandet av decimaler sker i rounDown() och om gränsvärden för zoomnivåer ändras i MapActivity så måste de också ändras här. När konstruktorn kallas så skalas först varje position av och läggs in i den nya listan. Sedan sorteras listan så att likadana punkter hamnar bredvid varandra för då går det snabbare att söka igenom listan och slå samman positioner som är likadana. Efter sammanslagningen har det bildats hål i listan så därför defragmenteras den som en sista åtgärd.

Webserver

Installation

Webservern har testats på apache2, mysql och php5. Uppgifter för databasen kan läggas i mysql_config_example.json som parsas från db_config.php. Det går att konfigurera db_config direkt men om man har de känsliga uppgifterna i json-filen istället så kan den läggas ovanför root och man slipper oroa sig över att känsliga uppgifter hamnar i git-repot. En databas behöver skapas med det namn som man anger i konfigurationsfilen och sen behöver setup.php köras för att skapa tabellen med användare. När nya användare registrerar sig så skapas det automatiskt en ny tabell för varje användare med deras positioner. Livslängden på en token går att ändra i filen db_connect.php i konstanten TOKENEXPIRATION. Glöm inte att ändra URL-konstanterna i WebHandler för att peka mot servern.

Metoder

WebHandler kallar på de fyra filerna push, pull, login och createuser som i som tur gör databasförfrågningar genom db_connect. Nedan gås de viktigaste metoderna i db_connect igenom.

authenticate

Tar emot en epost och en token som den kollar mot serverns databas. Om token är för gammal så returneras -2 och om eposten inte hittas eller om det är fel token så returneras -1.

login

Om epost och lösenord stämmer så genereras en ny token som returneras. Annars returneras "wrongpassword".

pull

Alla positioner efter lastId hämtas från databsen och returneras i en array.

push

Alla nya positioner kommer in en array och läggs till i databasen. Kontrollen på vilka positioner som är nya görs lokalt och inte på servern.

createUser

En kontroll görs för att se om användaren redan existerar och returnerar i så fall "alreadyexists". Annars läggs en ny användare till i users-tabellen och en ny tabell med användarens positioner skapas. En token genereras även och returneras.