Projekt - poročilo

Sistemska administracija – Sporttracker

Uvod

Projekt Sporttracker je mišljen kot športna aplikacija, ki zbira podatke o teku uporabnika. Pri tem zbira več podatkov, kot so povprečen srčni utrip, čas teka, lokacija teka oziroma pot teka, meri porabljene kalorije ter pretečeno razdaljo.

Člani projekta

Pri projektu smo bili udeleženi trije. To smo Jakob Polegek, Jan Vrtačnik ter Aljaž Marš. Pri delu smo si sicer pomagali med sabo, vendar je za izdelavo aplikacije za telefon, ki je uporabljena za zbiranje podatkov, bil glavni Jan Vrtačnik. Jakob Polegek in Aljaž Marš pa sta bila zadolžena za izdelavo spletne strani, ki ima glavno vlogo prikaz podatkov. Jakob Polegek je prav tako delal na aplikaciji, ki pridobi iz videa uporabnika, njegov srčni utrip. Aljaž Marš pa je bil zadolžen za ustvarjanje spletnega strežnika, ki bo kodo, za pridobivanje povprečnega srčnega utripa lahko poganjala. Pri izbiri podatkovne baze pa so bili prisotni vsi člani skupine.

Podatkovna baza

Podatke pridobivamo, shranjujemo na Googlovo Firebase platformo. V podatkovno bazo shranjujemo uporabniško ime, višino, težo, starost uporabnika, prav tako pa distanco ki jo je npr. pretekel, čas izvajanja aktivnosti, timestamp (kdaj je bilo vstavljeno v bazo), polje pot, kjer shranjujemo koordinate (latitude in longitude – zemljepisno višini in širino) vsakih 3sekund. Prav tako vstavimo povprečen utrip in število porabljenih kalorij med aktivnostjo (na začetku imata vrednost 0), po procesiranju oz. izračunu na strežniku pa ju posodobimo s pravimi vrednostmi. Za Firebase smo se odločili izključno zaradi obsežne dokumentacije ter vodičev na Youtube. Nekaj alternativ Firebase: Parse, Kinvey, Backendless, Kuzzle, …

Autentikacija uporabnika

Za autentikacijo (registracijo in prijavo) uporabnika uporabljamo Firebase Authentication. Uporabnik se mora za uporabo aplikacije obvezno najprej registrirat, potem potrditi avtomatski e-naslov in nato prijaviti za uporabo aplikacije. Registrira se

z e-poštnim naslovom in geslom. Registracijo in prijavo lahko uporabnik izvede na spletni strani ali na mobilni aplikaciji. Administratorjem uporabnikova gesla niso vidna, viden jim je le način šifriranja in v primeru »nezgode« lahko tudi resetirajo gesla. Administratorjem je prav tako vidno, kdaj se je uporabnik na zadnje prijavil v račun in kdaj je ustvaril ta račun.

Shranjevanje in uporaba videov za računanje srčnega utripa

Uporabnik mora potem, ko konča z aktivnostjo posneti kratek, vsaj 15 sekund dolg, videoposnetek svojega obraza. Ta video se potem naloži na Firebase Storage v formatu: videodocumentid.mp4. Ko se video uspešno naloži na shrambo, se nato sproži http zahtevek z id-jem dokumenta (tega, za katerega je bil posnet video posnetek) na naš API (Python Flask Webserver, gostujoč na Heroku). Prenese se video posnetek iz Firebase storage, ki se potem uporabi v programu za računanje srčnega utripa.

Srčni utrip računamo z pomočjo python programa. Program prejme kot parameter ime lokalne video datoteke (.mp4). V programu se kličejo naslednje skripte:

- preprocessing.py – Kliče funkcijo za branje videoposnetkov iz datoteke ter uporablja Haar kaskadno prepoznavanje obrazov, da izbere regijo interesa (ROI) na vseh okvirjih,
- pyramids.py – Kliče funkcije za generiranje in strnitev slikovnih/video piramid (Gaussian ali Laplacian).
- eulerian.py – Kliče funkcijo za začasni pasovni pas, ki uporablja »Fast-Fourierjevo« transformacijo.
- heartrate.py – Kliče funkcijo za računanje srčnega utripa iz rezultatov FFT (Fast Fourier Transformation) transformacije.

V programu smo uporabili knjižnice OpenCV (alternativa Dlib), NumPy (alternative tinynumpy napisana v celoti v pythonu - počasnejše izvajanje) in SciPy (alternativa SymPy, ki ne potrebuje prevajanja). Za OpenCV smo se odločili, ker predstavlja nek standard za delo in razvijanje na področju računalniškega vida ter ker je napisan v C++ (hitro izvajanje). Numpy in SciPy smo uporabljali za matematične funkcije in ker podpira velika več dimenzionalna polja in matrike.

Spletna aplikacija

Spletno aplikacijo smo ustvarili v React-u. Namen aplikacije je prikaz aktivnosti (teka/sprehoda/kolesarjenja…). Na strani prikazujemo uporabnikovo uporabniško ime,

težo, višino, število porabljenih kalorij med aktivnostjo, utrip po aktivnosti, razdaljo ki jo je opravil med aktivnostjo, ter čas izvajanja aktivnosti. Spodaj, pod omenjenimi podatki pa prikazujemo tudi mapo (openstreetmap), kjer prikazujemo začetek uporabnikove aktivnosti (marker) in pot ki jo je opravil. Za prikaz mape smo uporabili dodatek Leaflet oz. React Leaflet. Zanj smo se odločili zaradi veliko lažje implementacije markerjev in risanje poti na zemljevidu oz. mapi kot pa z Google Maps (react-google-maps).

Python Web Server(API)

Python API

V Sporttracker aplikaciji, imamo potrebo po uporabi Python skript, ki pa jih ne moremo kar tako izvesti iz nič. Za to smo se odločili za uvedbo Python API strežnika. Le ta čaka na novo ustvarjeno aktivnost v aplikaciji za zbiranje podatkov in se sproži ob prejetem HTTP zahtevku, ki je metode post. V tem zahtevku se pošlje ime dokumenta, na katerem se bo Python skripta izvedla. Ko strežnik to zahtevo dobi, izlušči ime dokumenta in s podatkovne baze Firebase, kjer je shranjen video, ki bo uporabljen v tej skripti, vzame video, ki je shranjen pod imenom "video" + ime dokumenta + ".mp4". Po tem, ko video pridobi iz podatkovne baze, zažene Python skripto, ki bo iz videa pridobila trenuten srčni utrip. Poleg tega še skripta izračuna še porabljene kalorije. Te podatke po izračunu shrani v dokument, ki je v Firebase-u, od koder se potem prikažejo podatki na spletni strani.

Uporabljene knjižnice

Pri Python API strežniku, je za delovanje bilo potrebnih kar nekaj knjižnic. Med glavnimi so bile Firebase_Admin, Flask in Pyerbase.

Flask

Knjižnica, se uporablja za obdelavo skupnih virov (CORS), ki omogoča AJAX zahtevke. Knjižnica omogoča preprosto ustvarjanje spletnih strani ter njihov razvoj. Prav tako je Flask sam po sebi že spletni strežnik tako da ni potrebno nalagati dodatnih storitev, ki bi bile potrebne za prikazovanje in delovanje spletne strani. Prav tako Flask ne omejuje izbire podatkovnih baz in je to popolnoma prepuščeno uporabniku.

Za Flask smo se odločili zaradi preprostosti knjižnice za uporabo, saj ponuja vse kar potrebujemo in je hkrati knjižnica, ki podpira ostale knjižnice zelo dobro. Prav tako pa je ogromno dokumentacije za to knjižnico kar nam je še dodatno pomagalo pri izbiri.

Alternativ Flask-a pa je še zelo veliko. Med najbolj uporabljenimi so FastAPI, Django in web2py.

Firebase admin

Je knjižnica, ki omogoča delo in uporabo Firebase podatkovne baze. Knjižnica omogoča zelo preprosto povezavo na Firebase podatkovno bazo, saj jo razvija Google, ki je razvijalec Firebase podatkovne baze. To knjižnico uporabljamo iz pridobivanje podatkov iz Firestore baze in za spreminjanje podatkov vanj.

Za Firebase_admin knjižnico smo se odločili prav zaradi zgoraj omenjene preproste uporabe ter zelo dobre podpore v Firebase podatkovni bazi.

Ena izmed alternativ Firebase admin bi bila Pyrebase, ki jo prav tako uporabljamo.

Pyrebase

Knjižnica Pyrebase, deluje na zelo podoben način kot Firebase admin, edina razlika je, da je ni ustvaril Google. Ampak kljub temu deluje tako dobro kot zgoraj opisan Firebase admin.

Za Pyrebase smo se odločili, zaradi preprost dela z datotekami v Firebase podatkovni baze.

Requests

Knjižnica, ustvarjena za prejemanje in pošiljanje HTTP zahtevkov. Omenjeno delo izjemno poenostavi in v našem primeru, kjer želimo izluščit le en podatek iz zahtevka za nadaljnjo delo s podatki.

Za to knjižnico smo se odločili zaradi preprostosti uporabe, saj nismo potrebovali nič bolj naprednega in sem mnenja, če bi do tega prišlo, da bi to knjižnica lahko pomagala.

Alternative te knjižnice pa so še na primer urlib3, httplib2 ter grequests. Alternativ je še več, vendar so bile naštete le najbolj znane.

Android

Del našega projekta je Android aplikacija, ki je glavna za pridobivanje podatkov, katere potrebujemo za našo storitev. Za to, da našo mobilno aplikacijo naredimo za Android operacijski sistem smo se odločili zato, ker se nam zdi, da so telefoni z Android operacijskim sistemom bolj razširjeni med ljudmi medtem ko njegov

največji tekmec iOS, bolj uporabljen med mlajšim denarno zadovoljenim osebam. Eden izmed večjih razlogov, da smo se odločili za Android je tudi to, da smo bili z njim in programom Android Studio seznanjeni na fakulteti zato smo že imeli nekaj podlage za izdelavo aplikacije. Medtem, ko je izdelava Android aplikacij po navadi zahtevnejša za ustvarjanje in hkrati zanjo porabimo več časa je pa zato podprta na več napravah, ker je dosti več različnih Android kot pa iOS naprav. Na koncu smo hoteli, da je naša aplikacija podprta na več sistemih in pa tudi naših sistemih ker sami uporabljamo Android telefone smo ostali pri tem operacijskem.

Za okolje v katerem smo ustvarili aplikacijo smo izbrali Android Studio. Ta IDE je najbolj optimiziran za ustvarjanje Android aplikacij, ima veliko dokumentacije in je prav tako uradni Googlov IDE za Android. Z Android Studiem smo delali na fakulteti in je bila zato to najbolj logična izbira. Medtem ko drugi IDE kot so Eclipse in IntelliJ IDEA so bolj priljubljena za razne Java aplikacije Android Studio zadošča vsem našim potrebam za našo aplikacijo. IntelliJ je eden izmed najbolj priljubljenih IDE trenutno, najbolj zaradi enostavnosti, pregleda, pomoči pri pisanju kode. Ker pa je Android Studio bil ustvarjen prav za izdelavo Android aplikacij, in je to razmeroma enostavno smo uporabili to okolje.

Za jezik v katerem smo ustvarili aplikacijo smo se odločili za Javo ker smo jo že poznali, ima tudi več knjižnic in dokumentacije na spletu. Kotlin pa je vedno bolj uporabljen za Android aplikacije, ker je bolj robusten, lažje za razumevanje ter omogoča lažje reševanje problemov, ki nastajajo pri ustvarjanju aplikacije. Kotlin naj bi tudi imel bolj enostavno in razumljivo sintakso. Kotlin pa lahko tudi normalno deluje z knjižnicami ki so napisane v Javi. Zaradi tega ker uporablja manj kode za iste stvari kot Java je potrebno manj časa in pa tudi manj možnosti napak.

Ena izmed bolj pomembnih knjižnic, ki smo jo potrebovali je od Google Location Services API s katero smo lahko na določen interval prejemali lokacijo uporabnika in jo pošiljali na internetno stran kjer smo prikazali pot. Ta knjižnica je del Google Play Services zato smo potrebovali API key da smo pridobivali želene podatki in bi jih lahko tudi prikazovali na Google Maps. Knjižnica je izboljšana alternativa Androidovi knjižnici Location. Ima več uporabnih funkcij ter drugih priročnih dodatkov ter vnaprej pripravljenih funkcij.

Za pošiljanje na bazo in branje podatkov iz baze smo uporabili Firebase, katerega samo dodamo preko »dependencies« in lahko pišemo na bazo razne dokumente ter pošiljamo datoteke.