BOTANIAI

Projektna dokumentacija

2022/2023

UM FERI

Lahovnik, Borak, Drozg, Jevšnik

Kazalo vsebine

[Kazalo slik 1](#_Toc136101987)

[Uvod 3](#_Toc136101988)

[Namen 3](#_Toc136101989)

[Izbrane tehnologije 3](#_Toc136101990)

[Razdelitev dela 3](#_Toc136101991)

[Organizacija dela 3](#_Toc136101992)

[Predvideni uporabniki 4](#_Toc136101993)

[Struktura podatkovne baze 6](#_Toc136101994)

[Podatki o rastlini 6](#_Toc136101995)

[Podatki o uporabniku 7](#_Toc136101996)

[Glavne aktivnosti uporabnikov 8](#_Toc136101997)

[Klasifikacija rastline 8](#_Toc136101998)

[Virtualni vrt 9](#_Toc136101999)

[Urejanje podatkov rastline v virtualnem vrtu 10](#_Toc136102000)

[Izbris rastline iz virtualnega vrta 11](#_Toc136102001)

[Pregled podrobnosti rastline 12](#_Toc136102002)

[Beleženje zalivanja rastlin 13](#_Toc136102003)

[Rezultati in produkti 14](#_Toc136102004)

[Napovedni model 14](#_Toc136102005)

[Metrike napovednega modela 14](#_Toc136102006)

[Validacijska točnost in izguba 16](#_Toc136102007)

[Uporabljeni sloji 17](#_Toc136102008)

[Zaledni sistem 18](#_Toc136102009)

[Mobilna aplikacija 18](#_Toc136102010)

[Izzivi, na katere smo naleteli tekom razvoja 18](#_Toc136102011)

[Nekompatibilnost knjižnjic v pythonu 18](#_Toc136102012)

[Nekompatibilnost knjižnjic v mobilni aplikaciji 18](#_Toc136102013)

# Kazalo slik

[Slika 1: Prvotni diagram primerov uporabe 4](#_Toc136102014)

[Slika 2: Prilagojen diagram primerov uporabe 5](#_Toc136102015)

[Slika 3: Struktura dokumenta Plant 6](#_Toc136102016)

[Slika 4: Struktura dokumenta User 7](#_Toc136102017)

[Slika 5: Diagram aktivnosti klasifikacija rastline 8](#_Toc136102018)

[Slika 6: Diagram aktivnosti virtualni vrt 9](#_Toc136102019)

[Slika 7: Diagram aktivnosti urejanje podatkov rastline 10](#_Toc136102020)

[Slika 8: Diagram aktivnosti izbris rastline 11](#_Toc136102021)

[Slika 9: Diagram aktivnosti pregled podrobnosti rastline 12](#_Toc136102022)

[Slika 10: Diagram aktivnosti beleženje zalivanja rastline 13](#_Toc136102023)

[Slika 11: Končno število učnih primerkov glede na razred 14](#_Toc136102024)

[Slika 12: Metrike napovednega modela 15](#_Toc136102025)

[Slika 13: Matrika zmede testnih podatkov 15](#_Toc136102026)

[Slika 14: Pravilna vrsta Slika 15: Napovedana vrsta 16](#_Toc136102027)

[Slika 16: Graf točnosti in izgube 16](#_Toc136102028)

[Slika 17: Graf točnosti in izgube nad neprilagojenimi učnimi podatki 16](#_Toc136102029)

[Slika 18: Metrike kakovosti nad neprilagojenimi podatki 16](#_Toc136102030)

[Slika 19: Zaporedje slojev v napovednem modelu 17](#_Toc136102031)

# Uvod

## Namen

Ideja je nastala zaradi pomembnosti rastlin in njihove pravilne skrbi. Rastline nas obkrožajo povsod na svetu, zaradi česar je pomembno, da vemo katere rastline nas neposredno obdajajo in ali nas lahko kakšne rastline potencialno življenjsko ogrožajo. S tem namenom smo ustvarili aplikacijo, ki omogoča prepoznavo različnih vrst rastlin glede na fotografijo. Na podlagi fotografije nato klasifikacijski model napove vrsto rastline in o tej vrsti si lahko preberemo več informacij. Hkrati omogoča shranjevanje rastlin v svoj virtualni vrt, kjer si lahko tudi izbere na koliko dni je potrebno rastlino zaliti in prvi dan zalivanja, na podlagi česar bo nato uporabnik prejemal potisna obvestila (push notifications). Aplikacija je torej namenjena vsem ljubiteljem rastlin, vsem, ki želijo izvedeti več informacij o razičnih vrstah rastlin ter vsem, ki se želijo naučiti več o pravilni skrbi za posamezno rastlino.

## Izbrane tehnologije

Za programske jezike smo izbrali python za klasifikacijo, JavaScript za zaledni sistem in TypeScript za mobilno aplikacijo. Na področju klasifikacije smo si pomagali z uporabo knjižnjic, kot so Tensorflow, Numpy, Scikit-learn, matplotlib in seaborn. V sklopu zalednega sistema smo izbrali ogrodje ExpressJS, s pomočjo različnih knjižnjic, kot je Mongoose za povezavo na MongoDB podatkovno bazo, dotenv za okoljske spremenljivke in Swagger za dokumentacijo API-ja. V sklopu mobilne aplikacije smo uporabili ogrodje React Native, s pomočjo različnih knjižnjic, kot je Expo, zaradi česar je bilo možno aplikacijo uporabljati v ExpoGo emulatorju, sicer smo pa uporabili nekatere druge pomembne knjižnjice, kot je expo-camera za uporabo kamere v aplikaciji in firebase, za avtentikacijo in uporabo firebase functions.

Za te tehnologije smo se odločili, zato ker jih vsi vsaj nekoliko poznamo, kar je olajšalo razvoj in zaradi česar je bilo reševanje težav, ki so nastopile lažje.

## Razdelitev dela

Ena oseba se je posvetila razvoju zalednega sistema, ena oseba se je posvetila razvoju klasifikacijskega modela, preostali dve osebi pa sta se posvetili razvoju mobilne aplikacije. Takšno razdelitev oziroma organizacijo znotraj skupine smo izbrali, saj je bilo največ dela ravno pri razvoju mobilne aplikacije.

## Organizacija dela

Pred začetkom razvoja projekta smo se v skupini dogovorili kdo bo delal na katerem delu projekta in katere tehnologije bodo izbrane. Med razvojem smo se na začetku sestali in izvedli uvoden sestanek, kjer smo šli skozi morebitne pomisleke in razložili morebitne nejasnosti. Tekom razvoje smo se vsak teden sestali na tedenskih Scrum sestankih, sicer so sestanki potekali po potrebi. V kolikor je bilo potrebnih več sestankov na teden smo jih izvedli. Za dogovarjanje smo uporabili aplikacijo Discord, kjer smo ustvarili skupinski pogovor in izvajali skupinske klice.

# Predvideni uporabniki

Predvideni uporabniki, funkcionalnosti aplikacije in integracije oziroma zunanje storitve so prikazani v naslednjem diagramu primerov uporabe, kot je bilo prvotno zamišljeno.

A picture containing text, diagram, line, drawing

Description automatically generated

Slika : Prvotni diagram primerov uporabe

Ker pa smo tekom razvoja naleteli na nekompatibilnost nekaterih knjižnjic, ni bilo možno vgraditi klasifikacijskega modela, zaradi česar je bilo potrebno diagram primerov uporabe nekoliko prilagoditi.

A picture containing text, diagram, line, drawing

Description automatically generated

Slika : Prilagojen diagram primerov uporabe

# Struktura podatkovne baze

## Podatki o rastlini

Podatki o rastlinah so zbrani v kolekciji imenovani »Plant« in imajo naslednjo strukturo:

* \_id – unikatni identifikator,
* latin – latinsko ime rastline,
* common – navadno ime rastline,
* description – daljši opis rastline,
* watering – na koliko časa bi naj rastlino zalivali,
* sunlight – kolikšni količini sončne svetlobe bi naj bila izpostavljena,
* plantingTime – v katerih letnih časih rastlino sadimo,
* soil – v kakšnih prsteh rastlina uspeva,
* wateringDetail – podrobneje opisan način zalivanja,
* fertilization – na kakšen način lahko gnojimo rastlino,
* toxicity – ali je rastlina strupena in za koga,
* folder\_num – številka direktorija, v katerem se nahajajo primerki te rastline.

A screenshot of a computer

Description automatically generated with medium confidence

Slika : Struktura dokumenta Plant

## Podatki o uporabniku

Podatki o uporabnikih so zbrani v kolekciji imenovani »User« in imajo naslednjo strukturo:

* \_id – unikatni identifikator,
* name – ime uporabnika,
* surname – priimek uporabnika,
* email – e-poštni naslov uporabnika,
* notifications – ali želi uporabnik prejemati potisna obvestila ali ne,
* history – zbrani rezultati klasifikacijskega modela, slika uporabljena za napoved, datum napovedi in razultat klasifikacijskega modela,
* personalGarden – zbrane rastline uporabniškega virtualnega vrta in podatki, ki so potrebni za skrb posamezne rastline.

A screen shot of a computer program

Description automatically generated with low confidence

Slika : Struktura dokumenta User

# Glavne aktivnosti uporabnikov

## Klasifikacija rastline

A picture containing text, diagram, screenshot, parallel

Description automatically generated

Slika : Diagram aktivnosti klasifikacija rastline

## Virtualni vrt

A picture containing text, diagram, screenshot, plan

Description automatically generated

Slika : Diagram aktivnosti virtualni vrt

## Urejanje podatkov rastline v virtualnem vrtu

A picture containing text, screenshot, diagram, parallel

Description automatically generated

Slika : Diagram aktivnosti urejanje podatkov rastline

## Izbris rastline iz virtualnega vrta

A picture containing text, screenshot, diagram, parallel

Description automatically generated

Slika : Diagram aktivnosti izbris rastline

## Pregled podrobnosti rastline

A picture containing text, screenshot, diagram, parallel

Description automatically generated

Slika : Diagram aktivnosti pregled podrobnosti rastline

## Beleženje zalivanja rastlin

A picture containing text, screenshot, diagram, parallel

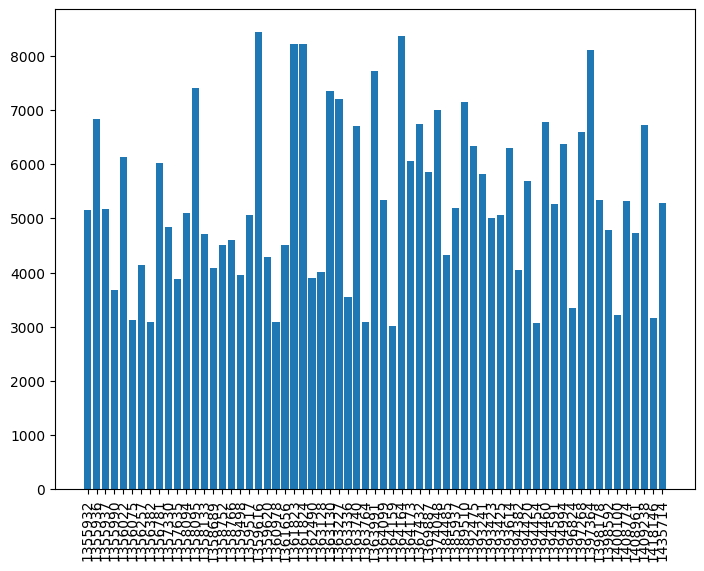
Description automatically generated

Slika : Diagram aktivnosti beleženje zalivanja rastline

# Rezultati in produkti

## Napovedni model

Končna verzija modela je bila naučena na Augmented podatkih, kar pomeni, da smo podatkovno množico umetno prilagodili. Tiste vrste, ki so imele manj kot 3000 primerkov (slik), smo umetno prilagodili. Vsako sliko smo vertikalno in horizontalno zrcalili in shranili kot nove primerke. Nato smo umetno prilagodili še podatke, ki so imeli manj kot 4000 podatkov, ampak še vrsta ni bila umetno prilagojena. V tem primeru smo vsako sliko zgolj vertikalno zrcalili. Na ta način smo se lahko tudi izognili underfit-tanju podatkov, kar se največkrat zgodi, ker imamo premalo primerkov za posamezno vrsto. Število vrst, ki smo jih uporabili za učenje modela, je 65.



Slika : Končno število učnih primerkov glede na razred

### Metrike napovednega modela

Končna verzija napovednega modela je dosegla 70% točnost nad testnimi primerki. Hkrati je dosegel 0,73 povprečno preciznost, 0,71 povprečen priklic in 0,71 povprečno F-mero. S testiranjem različnih parametrov, slojev, zaporednjem slojev, smo se poskusili čim bolj približati najboljšim vrednostim. Najboljše vrednosti bi bile: 100% za točnost in 1 za vse ostale metrike.

A picture containing text, screenshot, font, number

Description automatically generated

Slika : Metrike napovednega modela

Napovedi nad testnimi podatki lahko ovrednotimo tudi na podlagi matrike zmede. Vidimo lahko, da naš model ni delal preveliko napak. Kjer pa so bile prisotne napake, pa je bilo največkrat zaradi velike podobnosti različnih vrst – podobne so si bile tako v barvi kot obliki.

A picture containing screenshot, colorfulness, purple, line

Description automatically generated

Slika : Matrika zmede testnih podatkov

Primer, kjer je pravilna vrsta 1394454 (*lat.* *Pelargonium inquinans*), napovedana pa je bila vrsta 1435714 (*lat. Pelargonium zonale*).

A close-up of a pink flower

Description automatically generated with low confidence A close up of a flower

Description automatically generated with medium confidence

Slika : Pravilna vrsta Slika : Napovedana vrsta

Kot lahko vidimo iz slik, sta si vrsti precej podobni. Na podlagi imena pa lahko sklepamo, da tudi spadata v isto družino rastlin.

### Validacijska točnost in izguba

Na naslednjih grafih lahko vidimo, kako se je spreminjala točnost in izguba nad učnimi in validacijskimi podatki skozi epohe. Vidimo lahko, da se je validacijska točnost višala in izguba manjšala, kar tudi iščemo.

A picture containing line, screenshot, plot, text

Description automatically generated

Slika : Graf točnosti in izgube

Napovedni model je bil naučen na 20 epohah in prilagojenih učnih podatkih. Na koncu je dosegel 70% točnost, kar je 5% več, kot je dosegel enak model na 30 epohah in navadnih, neprilagojenih učnih podatkih.

A picture containing line, plot, diagram, screenshot

Description automatically generated

Slika : Graf točnosti in izgube nad neprilagojenimi učnimi podatki



Slika : Metrike kakovosti nad neprilagojenimi podatki

### Uporabljeni sloji

Uporabljeni so bili različni konvolucijski, združevalni, zgoščevalni in Dropout sloji. Konvolucijski sloji so namenjeni procesiranju (delov) slike, kjer se jedro sloja postopoma premika po sliki. Združevalni sloji so namenjeni zmanjševanju dimenzionalnosti in števila uteži. Dropout sloj se uporablja za preprečevanje prenatreniranosti modela. Temu sloju povemo kolikšen delež uteži z najmanjšim doprinosom naj izpusti oziroma pozabi. Na koncu uporabimo še Dense zgoščevalni sloj s številom razredov kot parameter.

A screenshot of a computer

Description automatically generated with medium confidence

Slika : Zaporedje slojev v napovednem modelu

## Zaledni sistem

Zaledni sistem je bil ustvarjen s pomočjo ogrodja ExpressJS. Zaledni sistem je povezan z NoSQL podatkovno bazo v oblaku (MongoDB Atlas). Za povezavo na bazo je bila uporabljena knjižnica Mongoose. Zaledni sistem ponuja različne metode za pridobivanje podatkov o rastlinah ali uporabnikovih podatkih. Za dostop do podatkov o določenem uporabniku je potreben veljaven žeton jwtToken tega uporabnika, ki nam ga ustvari in priskrbi Firebase ob prijavi. Na ta način vsaj do neke mere zavarujemo podatke uporabnikov, ki jih hranimo (ime, priimek, e-poštni naslov). Zaledni sistem gostimo pri ponudniku Render, ki omogoča povezavo z git repozitorijem in proženje novega grajenja zalednega sistema ob vsakem novem commit-u na izbrano vejo repozitorija.

## Mobilna aplikacija

Mobilna aplikacija je bila ustvarjena s pomočjo ogrodja React Native in Expo. Expo nam omogoča emulacijo aplikacije v ExpoGo aplikaciji, ki si jo naložimo na telefon. Uporabili smo tudi različne knjižnjice, ki olajšajo implementacijo in uporabo aplikacije, kot je axios za pošiljanje http(s) zahtev, firebase za avtentikacijo in uporabo Firebase Functions in expo-camera za uporabo kamere v aplikaciji.

# Izzivi, na katere smo naleteli tekom razvoja

## Nekompatibilnost knjižnjic v pythonu

Tekom razvoja smo naleteli na nekompatibilnost nekaterih knjižnjic. Nekompatibilna je bila knjižnjica tensorflowjs, zaradi česar je bil potreben »downgrade« nekaterih drugih knjižnjic – tensorflow, numpy, protobuf. Po spremembi verzij ostalih knjižnjic je knjižnjica tensorflowjs postala kompatibilna.

## Nekompatibilnost knjižnjic v mobilni aplikaciji

Tekom razvoja smo naleteli na nekompatibilnost nekaterih knjižnjic. Nekompatibilna je bila knjižnjica @tensorflow/tfjs-react-native, ki je nujno potrebna za vgraditev že naučenega modela. Poskusili smo zmanjšati verzije preostalih knjižnjic, s katerimi ni bila kompatibilna, vendar je to vodilo v izgubo funkcionalnosti knjižnjic, ki pa smo jih nujno potrebovali. Zaradi tega že naučenega modela nismo uspeli vgraditi v aplikacijo. Sicer bi lahko model zgradili tudi v aplikaciji, ampak potem bi se model moral učiti v realnem času z viri, ki mu jih ponuja sama mobilna naprava, kar pa je precej slabše kot učenje modela na računalnikih, saj imajo mobilne naprave v splošnem manjše zmogljivosti. Hkrati sta učna in validacijska množica veliki (26,1Gb in 1,92Gb), za kar pa marsikdo nima prostora na mobilni napravi. Zato model gostimo kot Firebase oziroma Google Function, ki je dostopna preko http zahteve (Functions as a Service).