RĪGAS 64. VIDUSSKOLA

ĒDIENA UN RECEPŠU REKOMENDĀCIJAS PROGRAMMATŪRA

PROGRAMMATŪRAS DOKUMENTĀCIJA

Darba autori: Toms Valdāts

Darba vadītājs: Edvards Bukovskis

RĪGA 2023

Saturs

Vārdnīca

ERRP	Ēdienu un recepšu rekomendācijas sistēma
URL	Adrese (atsauce) uz interneta resursu (abreviatūra no angļu val. *uniform resource locator*)
API	Programmatūras saskarsne (abreviatūra no angļu val. *application programming interface*)
Python	Augsta līmeņa interpretējama procedurālā, objektorientēta programmēšanas valoda
MVP	Minimālais dzīvotspējīgais produkts (abreviatūra no angļu val. *minimal viable product*)
DBMS	Datubāzes pārvaldības sistēma
CSV	Komatu atdalīts saraksts (abreviatūra no angļu val. *comma seperated list*)
JSON	JavaScript objektu notācija (abreviatūra no angļu val. *JavaScript object notation*)
HTML	Hiperteksta iezīmēšanas valoda (abreviatūra no angļu val. *HyperText markup language*)
Android	Operētājsistēma viedtālruņiem, planšetdatoriem un citām viedierīcēm

1. Ievads

1.1. Aktualitāte

Mūsdienās lielai daļai cilvēku trūkst laika kvalitatīva ēdiena pagatavošanai mājās, tā vietā izvēloties fasētās maltītes vai ātrās ēdināšanas pakalpojumus. Nereti lielākā problēma ēdiena pagatavošanai ir nevis pats pagatavošanas process, bet tieši ideju trūkums par to, ko iespējams pagatavot ar jau esošām sastāvdaļām mājās.

1.2. Nolūks

Ēdienu un recepšu rekomendācijas programmatūra (turpmāk ERRP), izmantojot lietotāja ievadītas sastāvdaļas, iesaka ēdienus, kurus var pagatavot, izmantojot ievadītās sastāvdaļas, un parāda lietotāja izvēlētā ēdiena pagatavošanai visas nepieciešamās sastāvdaļās, ieskaitot tās, kuras lietotājs izvēlējās.

2. Problēmas izpēte un analīze

Modernizētās pasaules ietvaros zūd laiks, ko cilvēki spēj ieguldīt, lai mājās pagatavotu kvalitatīvas, veselīgas un uzturvērtīgas maltītes, tādēļ tiek meklēti ēdiena risinājumi, kas ir laikietilpīgi un finansiāli izdevīgi. Tā rezultātā pēdējo desmitgažu laikā ir plaši attīstījusies ātro ēdināšanas pakalpojumu un fasēto ēdienu kultūra, tomēr šis risinājums ietver to, ka, tāpat kā citu nozaru uzņēmumi, arī šie uzņēmumi cenšas maksimāli vairot peļņu, tādējādi atsakoties no veselības un citiem standartiem.[1]

Cilvēkiem trūkst zināšanas, laiks un idejas, lai varētu pagatavot saturīgu ēdienu no mājās esošajām sastāvdaļām un uzturētu dažādību savā ikdienišķajā ēdiena patēriņā, kas noved pie tā, ka cilvēki pārlieku paļaujas uz fasētiem ēdieniem un ātrās ēdināšanas pakalpojumiem, kas noved pie neveselīga un vienmuļa uztura.[2]

Lai varētu izveidot programmatūru ar īpašībām, kas pēc iespējas vairāk pielāgotos mērķauditorijai, tiktu veikta atvērtā tipa aptauja ar mērķi noteikt, kas šajā jomā ir tipiskākās būtiskās un mazāk būtiskās problēmas cilvēkiem un kādas vēlmes programmatūrai ir jāizpilda, lai lietotāji vēlētos to izmantot.

Aptaujā varētu tikt iekļauti šāda tipa jautājumi:

- “Kāds ir Jūsu vecums?”
- “Cik bieži mēnesī Jūs izvēlaties iegādāties jau gatavu pārtiku tā vietā, lai gatavotu pats/i?”
- “Kas ir biežākie iemesli, kāpēc Jūs negatavojat, bet pērkat gatavu pārtiku?”

Lai aptvertu pēc iespējas plašāku mērķauditorijas loku, tajā pašā laikā atvieglojot pētījuma procesu un datu apkopošanu, aptauja tiktu veidota interaktīvā formā. Līdz mērķauditorijai aptauja tiktu nogādāta caur sociālajiem tīkliem, piemēram, "Facebook" kulinārijas vai kādām citām grupām. Pēc tam tiktu apkopoti aptaujā iegūtie dati un veikti secinājumi, lai izveidotu pēc iespējas cilvēkiem noderīgāku programmatūru.

Ņemot vērā fasēto ēdienu un ātrās ēdināšanas pakalpojumu izplatību, pieaug nepieciešamība pēc risinājuma, kas var nodrošināt ātras un vienkāršas receptes no mājās esošajām sastāvdaļām.[2]

---

[1] Hamid, A. Abdulmumeen, Ahmed N. Risikat and Agboola Sururah. “Food: Its preservatives, additives and applications.” International Journal of Chemical and Biochemical Sciences”, Department of Chemistery and Microbiology, University Of Ilorin. Janvāris 2008. P.M.B. 1515, Ilorin-Nigeria.

[2] Rodgers, S. "Technological innovation supporting different food production philosophies in the food service sectors", International Journal of Contemporary Hospitality Management, 8.februāris 2008. lpp. 19-34. Emeralad Group Publishing Ltd.

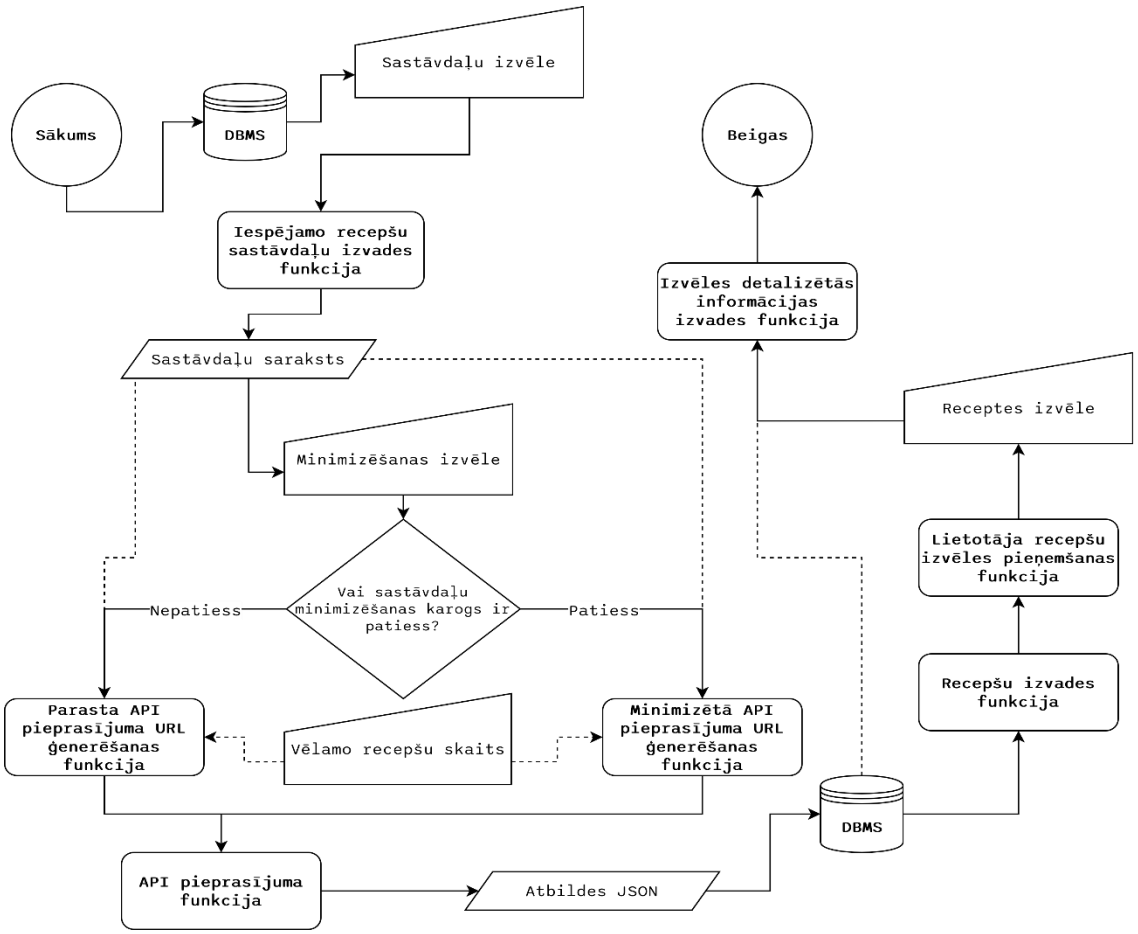
3. Programmatūras prasību specifikācija

3.1. Funkcionālās prasības

ERRP jāspēj ieteikt receptes, kuru sastāvdaļas satur tādas sastāvdaļas, kuras izvēlējies lietotājs. Lai nodrošinātu ERRP piedāvājamo recepšu plašumu un skaitu, gan arī veiktu rekomendācijas funkciju, tiek izmantots *Spoonacular*[3] API, uz kura bāzēts ERRP. ERRP minimālā dzīvotspējīgā produkta (MVP) izstrāde tiek veikta programmēšanas valodā *Python*, kas ļauj ātru programmatūras izstrādi daudzo "zemā līmeņa" datu struktūru abstrakciju dēļ, kā arī ir pieejama uz daudzām platformām. Minimālais dzīvotspējīgais produkts (pirmkodu skatīt 1. pielikumā), ir komandrindas programma, kas ievadi un izvade veic standartizvadē, taču plānotais, pilnais produkts ir *Android* programmatūra ar grafisku lietotājsaskarsni (iespējamo vizualizāciju skatīt 3. pielikumā). Tā kā programmatūrā netiek ievākti un saglabāti personu identificējoši dati, tad ERRP netiek pielietotas kriptogrāfijas metodes.

Programmatūras mērķauditorija:

- Cilvēki studiju gados, jo parasti šajā vecumā cilvēki vēlas izmantot pēc iespējas izdevīgākus variantus, kā paēst, kā arī viņiem nav liela pieredze ēdiena gatavošanā;
- Cilvēki ar nepilngadīgiem bērniem, jo šiem cilvēkiem ir sarežģīti apvienot darbu ar pilvērtīgu vecāku pienākumu izpildi.



1. attēls. ERRP darbības shematiskais attēlojums.

[3] *Spoonacular* API. Pieejams: https://spoonacular.com/food-api/

Galvenās ERRP funkcionālās prasības ir lietotāja ievadīto pieejamo sastāvdaļu pieņemšana (tabula 3.1.1.), atbilstoša API pieprasījuma URL ģenerēšana (tabula 3.1.2.), API pieprasījuma veikšana (tabula 3.1.3.), iespējamo recepšu parādīšana (izvade) lietotājam (3.1.4.), lietotāja ievades pieņemšana, lai tas varētu apskatīt izvēlēto, konkrēto recepti (tabula 3.1.5.), kā arī izvēlētās receptes detalizētās informācijas izvade lietotājam (tabula 3.1.6.). ERRP procesa shematisko attēlojumu skatīt 1. attēlā.

Iespējamo recepšu sastāvdaļas izvēles funkcija (skatīt tabulu 3.1.1.) balstās uz iespējamajām receptēm, kuras spēj pieņemt *Spoonacular* API. Pilns iespējamo ēdienu sastāvdaļu saraksts pieejams *Spoonacular* tīmekļvietnē CSV formātā.[4] ERRP minimālajam dzīvotspējīgam produktam (MVP) jāspēj kā ievadi pieņemt tikai mazu daļu no iespējamajām ēdienu sastāvdaļām, kā arī datubāzes vietā, kas satur sarakstu ar iespējamajām sastāvdaļām, tiks izmantots saraksts ar mazu daļu no sastāvdaļām (iespējamu datu bāzes relāciju modeli skatīt 2. pielikumā).

Tabula 3.1.1.

Iespējamo recepšu sastāvdaļu izvēles funkcija

Mērķis:
Ļauj lietotājam izvēlēties sev pieejamās vai tīkamās sastāvdaļās iespējamo recepšu atlasei
Ievaddati:
Lietotāja izvēle, balstoties uz iepriekš sagatavota sastāvdaļu saraksta, kas nolasīts no datubāzes
Apstrāde:
Atsevišķas sastāvdaļās tiek ieliktas sarakstā
Izvaddati:
Saraksts ar lietotāja izvēlētām sastāvdaļām

Atbilstoša API pieprasījuma URL ģenerēšanā (skatīt tabulu 3.1.2.) tiek izmantots lietotāja izvēlēto ēdienu sastāvdaļu saraksts, kā arī lietotājam ir iespējams izvēlēties, cik daudz iespējamas receptes tas vēlas redzēt un vai tas vēlas prioratizēti redzēt receptas, kuras iespējams pagatavot ar pēc iespējas mazāk sastāvdaļām, kā pamato ņemot izvēlētās sastāvdaļas. Šī recepšu sastāvdaļu minimizēšanas izvēles ievade MVP tiek pieprasīta standartizvadē.

Tabula 3.1.2.

Atbilstoša API pieprasījuma URL ģenerēšanas funkcija

Mērķis:
Ģenerēt atbilstošu API pieprasījuma URL, balstoties uz vairākiem kritērijiem
Ievaddati:
Sastāvdaļu saraksts, recepšu skaits, eksluzivitātes izvēle
Apstrāde:

[4] Iespējamo ēdienu sastāvdaļu pilanis saraksts. *Spoonacular*. Pieejams: https://spoonacular.com/application/frontend/downloads/ingredients.csv

Tiek ģenerēts atbilstošs URL, kas sevī iekļauj sastāvdaļu saraksta pieprasījuma karogu, recepšu skaita karogu un minimizēšanas izvēles karogu
Izvaddati:
Gadījumā, ja ekskluzivitātes karogs ir patiess: ▪ Tiek ģenerēts URL ar pievienotu minimizēšanas karogu Gadījumā, ja ekskluzivitātes karogs nav patiess: ▪ Tiek ģenerēts URL bez minimizēšanas karoga

API pieprasījuma veikšanā (skatīt tabulu 3.1.3.) tiek izmantota ģenerētais pieprasījuma URL, kā arī piekļuves atslēga, kas nepieciešama, lai piekļutu *Spoonacular* API esošajiem datiem. Brīvi pieejamajai API versijai maksimālais atļautais pieprasījumu skaits dienā ir 150 pieprasījumi.[5]

Tabula 3.1.3.

API pieprasījuma funkcija

Mērķis:
Veikt API pieprasījumu, balstoties uz iepriekš ģenerētu pieprasījuma URL
Ievaddati:
Pieprasījuma URL, API piekļuves atslēga
Apstrāde:
API piekļuves atslēga tiek iekļauta Python vārdnīcā, kas tiek ievadīta HTML pieprasījuma metodes bibliotēkas funkcijā
Izvaddati:
Gadījumā, ja pieprasījums ir veiksmīgs (HTML atbildes kods *200*): ▪ Tiek atgriezta un interpretēta iegūtā HTML atbilde JSON formātā Gadījumā, ja pieprasījums ir neveiksmīgs: ▪ Tiek atgriezts attiecīgais HTML kļūmes kods

Iespējamo recepšu izvadei (skatīt tabulu 3.1.4.) pēc API pieprasījuma iegūtie dati tiek parsēti JSON formātā. Tad tiek atrasta konkrētā JSON vērtība, kas satur visas atgrieztās receptes, kuras var pagatavot no izvēlētā recepšu saraksta, kas tiek izvadīta lietotājam saraksta formā, kur katras receptes sākumā ir parādīts tās indekss. ERRP MVP izvade notiek standartizvadē.

Tabula 3.1.4.

Recepšu izvades funkcija

Mērķis:
Parādīt lietotājam iespējamās receptes, kuras iespējams pagatavot, izmantojot sniegto sastāvdaļu sarakstu
Ievaddati:
API atbilde JSON formātā
Apstrāde:
No JSON tiek atlasītas konkrētās vērtības (recepšu nosaukumi)

[5] *Spoonacular* API cenas. https://spoonacular.com/food-api/pricing

Izvaddati:
Gadījumā, ja ir receptes, kuras var pagatavot ar lietotāja sniegtu sastāvdaļu sarakstu: ▪ Tiek izvadīts lietotājam saraksts ar iespējamo recepšu nosaukumiem Gadījumā, ja nav neviens receptes, kuras atbilst recepšu sarakstam: ▪ Tiek izvadīts lietotājam paziņojums, ka nav nevienas iespējamās receptes

Lietotāja recepšu izvēles pieņemšanai MVP (skatīt tabulu 3.1.5.) no lietotāja tiek pieņemta ievade kā standartievade, kur lietotājam ir jāievada izvēlētās receptes indekss, lai to varētu apskatīt detalizētāk.

Tabula 3.1.5.

Lietotāja recepšu izvēles pieņemšanas funkcija

Mērķis:
Ļaut lietotājam izvēlēties konkrētu recepti, par kuru vēlas iegūt detalizētāku informāciju, kas ieķlauj visas nepieciešamās sastāvdaļas, sastāvdaļu daudzumus un to nianses
Ievaddati:
Receptes indekss
Apstrāde:
Lietotāja atbilde (receptes indekss) tiek pieņemta no standartievades.
Izvaddati:
Lietotāja ievadītais izvēlētās receptes indekss tiek saglabāts mainīgajā

Izvēlētās receptes detalizētās informācijas izvadei (skatīt tabulu 3.1.6.), izmantojot lietotāja ievadīto receptes indeksu, no JSON atbildes tiek parsēta konkrētā receptes detaļu vērtība, kuru iespējams iegūt ar indeksu. MVP tā tiek izvadīta un noformatēta standartizvadē.

Tabula 3.1.6.

Izvēlētās receptes detalizētās informācijas izvades funkcija

Mērķis:
Parādīt lietotājam detalizētu informāciju par konkrētu, izvēlēto recepti.
Ievaddati:
Izvēlētās receptes indekss, API atbilde JSON formātā
Apstrāde:
Izmantojot receptes indeksu, tiek atrasta attiecīgā JSON vērtība, kas satur detalizēto informāciju par konkrēto recepti
Izvaddati:
Receptes detalizētās informācijas izvade lietotājam

3.2. Nefunkcionālās prasības

Ņemot vērā ERRP risināto problēmu ēdienu pagatavošanas procesā ar recepšu izvēli un to ērtu nepieejamību, tad, pirmkārt, ERRP programmatūrai jābūt lietotājiem ērtai. Tas nozīmē, ka tai ir jābūt ātrai, intuitīvi lietojamai, kā arī visai programmatūras sniegtajai informācijai ir jābūt skaidrai un viegli uztvertamai.

4. Programmatūras izstrādes plāns

ERRP izstrādei piemērotam programmatūras izstrādes modelim jāspēj, pirmkārt, balstīties uz izstrādātās programmatūras dokumentāciju, taču tajā pašā laikā tam jāspēj ļaut pakāpeniskus uzlabojumus un programmatūras paplašinājumus, lai ERRP pakāpeniski no minimāla, vienkāršota minimālā dzīvotspējīga produkta kļūtu par pilnvērtīgu produktu, ko iespējams palaist lietošanā patērētājam, tādēļ vispiemērotākais programmatūras izstrādes modelis būtu iteratīvais modelis, jo ir skaidri definētas galvenās prasības, taču pastāv nepieciešamība un varbūtība mazākiem uzlabojumiem un atjauninājumiem produkta dzīves cikla laikā.

ERRP iteratīvās izstrādes vienu ciklu var aprakstīt 4 soļos:

1. ERRP programmatūras laiduma prasību plānošana un dizains,
2. laiduma izstrāde un vienību testēšana,
3. laidumam izvirzīto prasību atbilstības pārbaude un akcepttestēšana,
4. palaišana tirgū un esošo versiju uzturēšana, kamēr tās nav aizstātas ar jaunākām versijām.

1. cikla posmā vispirms tiktu definētas funkcionālās pamatprasības, no kurām izrietu dizaina prasības. ERRP izstrādē šajā posmā tiktu izstrādāta programmatūras pamatspecifikācija, kurā tiktu aprakstītas nepieciešamās funkcijas, piemēram. Ja šis iteratīvais cikls nav pirmais programmatūras dzīves ciklā, tad noteikti būtu jāapsver iepriekšējā ciklā sastaptās problēmas un lietotāju atsauksmes, piemēram, vai ir kādas dizaina nepilnības, vai, piemēram, ir lietotāju pieprasījums savu recepšu publicēšanai un pieejamībai ERRP. Tad tiktu izveidotas vairākas iespējamās dizaina skices, no kurām tiktu atlasītā funkcionāli visatbilstošākā un mērķauditorijai pievilcīgākā. Kopumā posmam tiktu atvēlētas 2 līdz 3 nedēļas.

2. cikla posmā tiktu izstrādāts funkcionējošs ERRP laidums, kas balstīts uz 1. posmā izvirzītajām prasībām un izveidotajām dizaina skicēm, kā arī paralēli tiktu veikta vienību testēšana, lai nodrošinātu, kā programmatūras funkcijas iekļaujas definētajos ierobežojumos, un visas iespējamās kļūmes situācijas tiktu pārtvertas un apstrādātas. Posmam tiktu atvēlētas 2 nedēļas. Ja 3. cikla posmā būtu atrasti kādi trūkumi vai nepilnības, tad izstrādes posms tiktu pagarināts.

3. cikla posmā izveidotas laidums tiktu izvērtēts, vai tas atbilst izvirzītajām prasībām, kā arī tiktu veikta akcepttestēšana, lai pārbaudītu programmatūras funkcionalitāti. Šeit būtu nepieciešama speciāla cilvēku komanda, kas veic šo pārbaudi. Ja tiek atrastas nepilnības, tad vai nu cikls atgriežas iepriekšējā, programmatūras izstrādes posmā, vai arī programmatūras uzlabošana tiktu veikta paralēli 3. posmam. Šim posmam tiktu atvēlētas 2 nedēļas laika.

4. cikla posmā izstrādātais ERRP fināla laidums tiktu publicēts. Tā kā vienlaikus ir iespējams, ka tirgū ir vairākas gan jaunākas, gan vecākas ERRP versijas, tad vecākā versija zaudētu izstrādātāja atbalstu. Paralēli palaistu laidumu atbalstam, produkta pārvaldības un dizaina grupa ievāktu lietotāju atsauksmes un kritiku, lai varētu veidot nākošās prasības nākamajam, iteratīvajam izstrādes ciklam. Šim posmam paredzētais laiks, galvenokārt, ir atkarīgs no lietotāju atgriezeniskās saites un produkta pārvaldības grupas idejām jaunām ERRP funkcijām un papildinājumiem, taču tam nevajadzētu aizņemt vairāk par 3 līdz 4 nedēļam, lai programmatūra nestagnētu un nezaudētu novitāti un lietotājus.

5. Atkļūdošanas un akcepttestēšanas pārskats

5.1. Vienību testēšanas pārskats

Tabula 5.1.1.

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
RecipeData. **get_minimized**	Sastāvdaļu saraksts ar derīgām sastāvdaļām, derīgs (>0) recepšu skaits	Metode atgriež parsētu JSON objektu, kas satur receptes	Izpildās
Apraksts	Sastāvdaļu saraksts ar nederīgām sastāvdaļām, derīgs (>0) recepšu skaits	Metode atgriež kļūdu	Izpildās
Tiek pārbaudīta minimizētā API pieprasījuma darbība	Sastāvdaļu saraksts ar derīgām sastāvdaļām, nederīgs (≤0) recepšu skaits	Metode atgriež kļūdu	Izpildās

Tabula 5.1.2.

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
RecipeData. **get_normal**	Sastāvdaļu saraksts ar derīgām sastāvdaļām, derīgs (>0) recepšu skaits	Metode atgriež parsētu JSON objektu, kas satur receptes	Izpildās
Apraksts	Sastāvdaļu saraksts ar nederīgām sastāvdaļām, derīgs (>0) recepšu skaits	Metode atgriež kļūdu	Izpildās
Tiek pārbaudīta parastā API pieprasījuma darbība	Sastāvdaļu saraksts ar derīgām sastāvdaļām, nederīgs (≤0) recepšu skaits	Metode atgriež kļūdu	Izpildās

Tabula 5.1.3.

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
RecipeData. **generate_partial_url**	Derīgs (>0) recepšu skaits	Metode atgriež formatētu URL *string* ar skaita karogu	Izpildās
Apraksts	Nederīgs (≤0) recepšu skaits	Metode atgriež formatētu URL *string* bez skaita karoga	Izpildās
Tiek pārbaudīta parastā API pieprasījuma darbība			

Tabula 5.1.4.

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
UserInterface. **output_possible_ingredients**	Metodes konstruktorā definēts iespējamo sastāvdaļu saraksts	Metode standartizvadē izvada definēto sastāvdaļu sarakstu	Izpildās
Apraksts			
Tiek pārbaudīta sastāvdaļu saraksta izvade standartizvadē			

Tabula 5.1.5.

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
UserInterface. **prompt_ingredient_selection**	Visa ievadītā sastāvdaļu virkne satur derīgas sastāvdaļas	Metode atgriež sarakstu ar ievadītajām sastāvdaļām	Izpildās
Apraksts	Daļa no ievadītā sastāvdaļu virknes satur derīgas sastāvdaļas	Metode izvada paziņojumu, ka konkrētā, ievadītā sastāvdaļa nav derīgo sarakstā	Izpildās
Tiek pārbaudīta ievadīto sastāvdaļu parsēšana	Ievade satur dažādus simbolus, kas varētu izjaukt parsēšanu (,.@,&^# utml.)	Metode izvada paziņojumu, ka ievade neatbilst regeksā definētajam formātā	Izpildās
	Ievade ir “q”	Programma beidz darbību	Izpildās

Tabula 5.1.6.

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
UserInterface. **prompt_minimization**	Ievade satur tikai “jā”, “ja” vai “j”	Metode atgriež *True*	Izpildās
Apraksts	Ievade satur tikai “nē”, “ne” vai “ne”	Metode atgriež *False*	Izpildās
	Ievade ir “q”	Programma beidz darbību	Izpildās
Tiek pārbaudīta minimizācijas opcijas izvēles ievade	Ievade satur kaut citu	Metode izvada paziņojumu, ka ievade neatbilst formātam	Izpildās

Tabula 5.1.7.

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
UserInterface. **prompt_recipe_count**	Ievade satur veselu, pozitīvu skaitli	Metode atgriež ievadīto skaitli	Izpildās
Apraksts	Ievade ir “q”	Programma beidz darbību	Izpildās

Tiek pārbaudīta recepšu skaita ievade	Ievade satur kaut ko citu	Metode izvada paziņojumu, ka ievade neatbilst formātam	Izpildās

Tabula 5.1.8.

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
UserInterface. **output_recipe_list**	Minimizācijas opcija ir *True*	Metode izsauc minimizētā API pieprasījuma metodi un izdrukā pieejamās receptes	Izpildās
Apraksts	Minimizācijas opcija ir *False*	Metode izsauc parastā API pieprasījuma metodi un izdrukā pieejamās receptes	Izpildās
Tiek pārbaudīta iegūto, pieejamo recepšu saraksta izvade atkarībā no minimizācijas opcijas			

Tabula 5.1.9.

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
UserInterface. **prompt_recipe_selection**	Ievade satur veselu, pozitīvu skaitli, kas ≤ par pieejamo recepšu skaitu	Metode atgriež ievadīto skaitli	Izpildās
Apraksts	Ievade satur veselu, pozitīvu skaitli, kas > par pieejamo recepšu skaitu	Metode izvada paziņojumu, ka izvelētā recepte nav sarakstā	Izpildās
Tiek pārbaudīta receptes izvēles indeksa ievade	Ievade ir “q”	Programma beidz darbību	Izpildās
	Ievade nav vesels, pozitīvs skaitlis	Metode izvada paziņojumu, ka ievade neatbilst formātam	Izpildās

Tabula 5.1.10.

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
UserInterface. **output_detailed_recipe_info**	JSON receptē nav trūkstošās sastāvdaļas	Metodes izvades trūkstošo sastāvdaļu sadaļa ir tukša	Izpildās
Apraksts	JSON receptē ir trūkstošās sastāvdaļas	Metodes izvades trūkstošo sastāvdaļu sadaļā ir trūkstošās sastāvdaļas	Izpildās
Tiek pārbaudīta konkrētas receptes detalizētās informācijas izvade			

5.2. Akcepttestēšanas pārskats

5.2.1. Funkcionālo prasību izpildījums

Tabula 5.2.1.1.

Funkcionālā prasība	Izpildījums	Komentāri
Iespējamo recepšu sastāvdaļu izvēles funkcija	Izpildās	
Atbilstoša API pieprasījuma URL ģenerēšanas funkcija	Izpildās	Gan parasta pieprasījuma, gan minimizēta pieprasījuma gadījumos parciālais URL paliek vienāds. Minimizēšanas karogs tiek pievienots minimizētā pieprasījuma funkcijā, vienkāršojot sistēmu.
API pieprasījuma funkcija	Izpildās	Kodā šī funkcija ir sadalīta divās metodes: viena parastā pieprasījuma veikšanai, otra minimizētā pieprasījuma veikšanai.
Recepšu izvades funkcija	Izpildās	
Lietotāja recepšu izvēles pieņemšanas funkcija	Izpildās	
Izvēlētās receptes detalizētās informācijas izvades funkcija	Izpildās	

5.2.2. Nefunkcionālo prasību izpildījums

Tabula 5.2.2.1.

Nefunkcionālā prasība	Izpildījums	Komentāri
Intuitīvitāte	Daļēji izpildās	Vizuāla lietotājsaskarsne padarītu ERRP intuitīvāku lietošanai, kā arī pieprasītu mazāk tehniskās priekšzināšanas komandrindas vidē programmatūras lietošanā.
Ātrums	Izpildās	
Informācijas skaidrums un pieejamība	Izpildās	Lietotājam jābūt pieradušam pie teksta lietotājsaskarsnēm.

6. Koda metožu definīcijas un apraksti

6.1. *recipe_data* modulis

6.1.1. *RecipeData.**get_minimized**(self, ingredients_list: list[str], recipe_output_count: int) -> any*

Veic HTTP pieprasījumu *Spoonacular* API, atgriežot interpretētu JSON atbildi, kurā tiek prioratizēts iegūt receptes ar pēc iespējas mazāku sastāvdaļu skaitu (tiek minimizēts).

6.1.2. *RecipeData.**get_normal**(self, ingredients_list: list[str], recipe_output_count: int) -> any*

Veic HTTP pieprasījumu *Spoonacular* API, atgriežot interpretētu JSON atbildi, kurā netiek prioratizētas minimālas recepšu sastāvdaļas.

6.1.3. *RecipeData.**generate_partial_url**(self, recipe_output_count: int) -> str*

Ģenerē parciālu pieprasījuma URL, kas satur nepieciešamās pieprasījuma opcijas, balstoties uz lietotāja izvēlētajām sastāvdaļām un vēlamo, API atgriezto recepšu skaitu.

6.2. *user_interface* modulis

6.2.1. *UserInterface.**main**(self) -> None*

Galvenā metode *UserInterface* klasei, kas izpilda teksta lietotājsaskarsnes algoritmu.

6.2.2. *UserInterface.**output_possible_ingredients_list**(self) -> list[str]*

Izvada standartizvadē visas recepšu sastāvdaļas, kuras lietotājam iespējams izvēlēties.

6.2.3. *UserInterface.**prompt_ingredient_selection**(self, possible_ingredient_list: list[str]) -> list[str]*

Pieņem lietotāja ievadi standartievadē vēlamo sastāvdaļu izvēlē.

6.2.4. *UserInterface.**prompt_minimization**(self) -> bool*

Pieņem lietotāja ievadi standartievadē ievadīto sastāvdaļu minimizācijas opcijai.

6.2.5. *UserInterface.**prompt_recipe_count**(self) -> int*

Pieņem lietotāja ievadi standartievadē, lai lietotājs varētu izvēlēties vēlamo recepšu skaitu, ko redzēt.

6.2.6. *UserInterface.**output_recipe_list**(self, possible_ingredient_list: list[str]) -> None*

Izvada standartizvadē receptes, kuras iespējams pagatavot ar izvēlētajām sastāvdaļām.

6.2.7. *UserInterface.**prompt_recipe_selection**(self, possible_ingredient_list: list[str]) -> int*

Pieņem lietotāja ievadi standartievadē, lai lietotājs varētu izvēlēties konkrētu recepti, par kuru iegūt detalizētāku informāciju.

6.2.8. *UserInterface.**output_detailed_recipe_info**(self, possible_ingredient_list: list[str]) -> None*

Izvada standartizvadē detalizētu informāciju par konkrētu recepti.

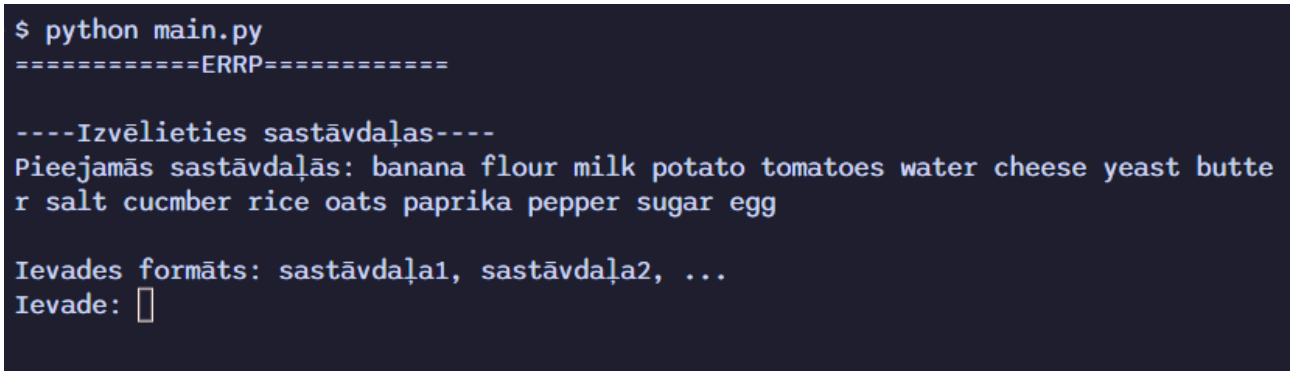
6.2.9. *UserInterface.**newline**(self, count: int) -> None*

Utilītmetode n-to jaunu rindu izdrukai standartizvadē.

7. Lietotāja ceļvedis

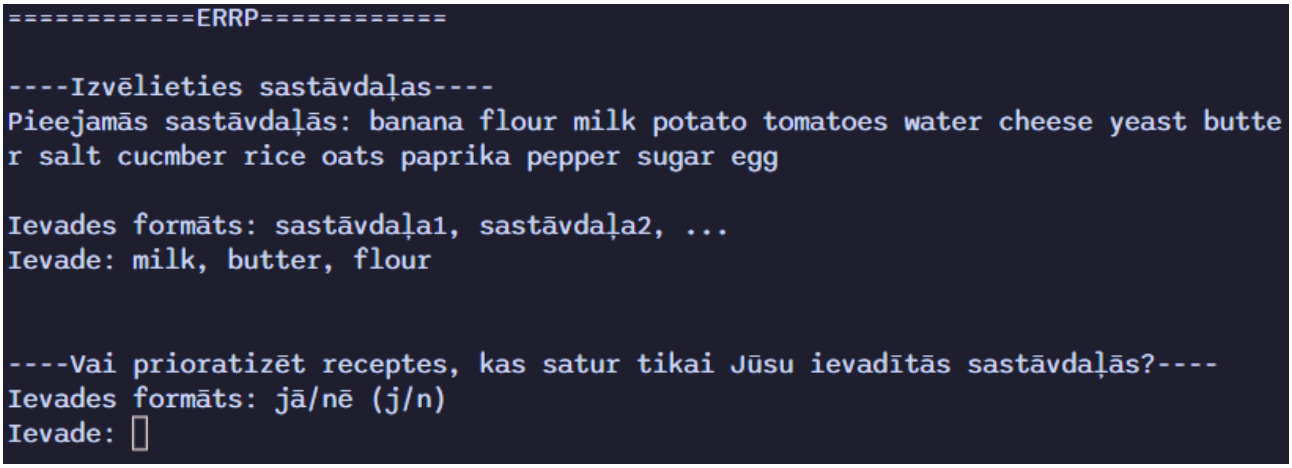
ERRP spēj darboties uz *Windows*, *MacOS*, *Linux* un *BSD* platformām, kurās ir pieejams *Python* programmēšanas valodas interpretētājs (minimālā versija 3.10 vai jaunāka), kā arī papildus vēl nepieciešams lejupielādēt *Python requests* bibliotēku attiecīgajai *Python* versijai. Lai lietotu programmatūru, nepieciešama piekļuve komandrindas videi.

Lai sāktu programmu nepieciešams atrasties mapē ar 3 programmas skriptiem. Lai palaistu ERRP, jāizmanto komanda `python main.py`.



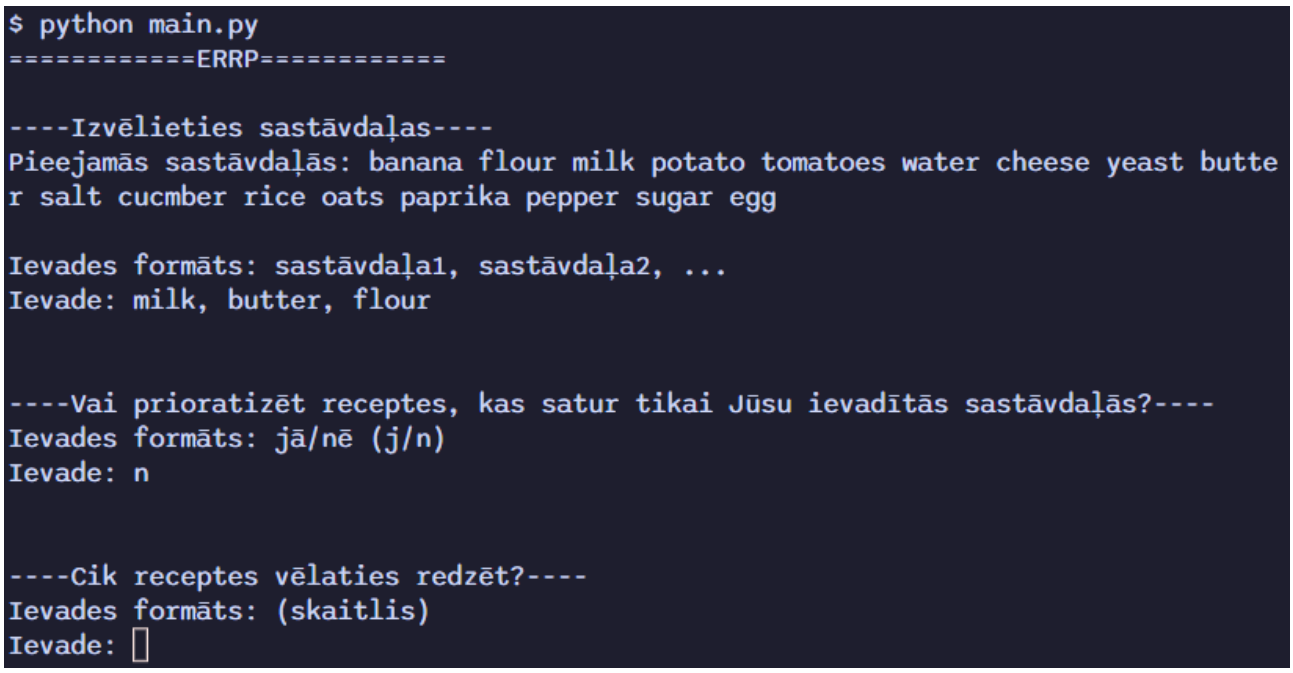
2. attēls. Komanda `python main.py`.

Vispirms, lai varētu atlasīt vēlamās receptes, ir nepieciešams izvēlēties sastāvdaļas, kuras Jūs vēlaties izmantot recepšu pagatavošanā (skatīt 2. attēlu). Zem teksta "Izvēlieties sastāvdaļas" var redzēt visas pieejamās sastāvdaļas. Lai veiktu izvēli, jāieraksta vēlamās sastāvdaļas. Ja ir vairākas vēlamās sastāvdaļas, tad tās ir jāatdala ar komatu, piemēram, ja vēlaties lietot pienu, sviestu un miltus, ievades teksts izskatītos šādi "milk, butter, flour". Kad vēlamās sastāvdaļas ir izvēlētas, nospiediet *Enter* taustiņu uz tastatūras.



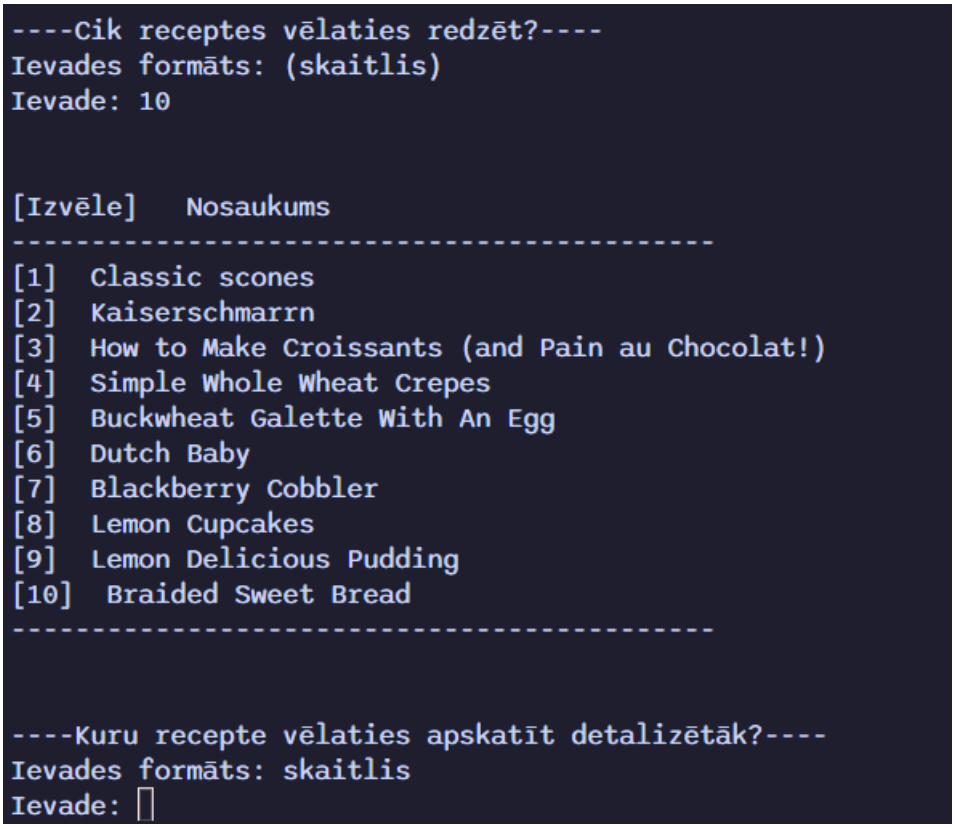
3. attēls. Recepšu prioratizēšanas dialogs.

Tālāk nepieciešams atzīmēt, vai vēlaties prioratizēti redzēt receptes, kuras sastāv tikai no Jūsu izvēlētajām sastāvdaļām (skatīt 3. attēlu). Ja piekrītat, tad ievadiet "jā" vai "j". Ja nepiekrītat, tad "nē" vai "n". Lai turpinātu, spiediet *Enter*.



4. attēls. Recepšu skaita dialogs.

Šeit nepieciešams ievadīt, cik daudz recepšu piedāvājumu Jūs vēlaties redzēt (skatīt 4. attēlu). Maksimālais piedāvājumu skaits ir 100 receptes. Lai turpinātu, spiediet *Enter*.



5. attēls. Receptes izvēles dialogs.

Tagad Jūs varat redzēt recepšu piedāvājumus, kas satur Jūsu izvēlētas sastāvdaļas. Lai apskatītu konkrētu recepti detalizētāk, nepieciešams ievadīt tās skaitli. Lai turpinātu, spiediet *Enter*.

```
----Kuru recepte vēlaties apskatīt detalizētāk?----
Ievades formāts: skaitlis
Ievade: 2

------------------------------------------------------------
Recepte: Kaiserschmarrn
Izvēlētas sastāvdaļas: milk butter flour
Attēls pieejams: https://spoonacular.com/recipeImages/994607-312x231.jpg
--------------------Esošās sastāvdaļas----------------------
375.0 ml: milk
          3.5%
          ( fat)
37.0 g: butter
          unsalted
------------------Trūkstošās sastāvdaļas--------------------
5.0 small: eggs
------------------------------------------------------------
```

6. attēls. Pabeigta ERRP darbība.

Visbeidzot, Jūs varāt ieraudzīt detalizētāk informāciju par izvēlēto recepti, ieskaitot tās attēlu. Esošās sastāvdaļas ir tās sastāvdaļās, kuras Jūs ievadījāt, bet trūkstošās sastāvdaļas ir tās sastāvdaļas, kas nepieciešamas papildus ievadītajām, lai pagatavotu konkrēto recepti.

8. Piemērotās licences pamatojums

ERRP programmatūra tiks izplatīta, kā atvērtā pirmkoda programmatūra zem GNU Vispārējās publiskās licences 3. versijas (GPLv3).

Galvenās priekšrocības, atverot ERRP pirmkodu ikvienam, ir, pirmkārt, lietotāju un entuziastu iesaiste programmatūras attīstībā un nepieciešamo funkciju papildināšanā, jo iespējams ērti veikt ieteikumus, pat iesakot kodu, ko pievienot. Otrkārt, pat ja programmatūru pārstāj uzturēt galvenais izstrādātājs, tad jebkurš cits lietotājs var pārņemt programmatūras izstrādi, pirmkodu atzarojot, tādejādi nodrošinot ERRP ilgtspēju un aktualitāti nākotnē.

Izvēloties GPLv3 licenci, tiek nodrošināts, ka nekad nākotnē ERRP pirmkodu neviens nevarēs aizvērt, bez visu koda papildinātāju piekrišanas, kas nozīmē, ka programmatūra vienmēr būs pieejama visiem, tomēr tas neliedz ERRP komercializēt jeb pārdot kompilētas programmatūras versijas par maksu, jo brīvi pieejamam ir jābūt pirmkodam, nevis kompilētiem bināriem failiem, kuri tiek izplatīti.

Pielikumi

1. pielikums
ERRP pirmkods

Datne *main.py*

```
import recipe_data
import user_interface

def main() -> None:
    ui = user_interface.UserInterface()
    # Palaiž ERRP galveno algoritmu
    ui.main()

if __name__ == "__main__":
    main()
```

Datne *user_interface.py*

```
import re
import sys
import recipe_data as rd


#=====================ERRP procesa un lietotājsaskarsnes klase=================
class UserInterface:
    def __init__(self):
        self.recipe_count: int
        self.defined_possible_ingredient_list: list[str] = ["banana", "flour",
"milk", "potato",
                                                            "tomatoes", "water",
"cheese", "yeast",
                                                            "butter", "salt",
"cucmber", "rice", "oats",
                                                            "paprika", "pepper",
"sugar", "egg"]
        self.minimization_option: bool
        self.ingredient_selection: list[str]
        self.recipe_selection: int
        self.json_dati: any

#====================Galvenā izvades un lietotāja ievades metode==============
    def main(self) -> None:
        print("============ERRP============")
        self.newline(1)
        possible_ingredient_list: list[str] =
self.output_possible_ingredient_list()
        self.newline(1)
        self.ingredient_selection =
self.prompt_ingredient_selection(possible_ingredient_list)
        self.newline(2)
        self.minimization_option= self.prompt_minimization()
        self.newline(2)
        self.recipe_count = self.prompt_recipe_count()
        self.newline(2)
        self.output_recipe_list()
        self.newline(2)
        self.recipe_selection = self.prompt_recipe_selection()
        self.newline(1)
        self.output_detailed_recipe_info()

#====================Sastāvdaļu ievades metode=================
    def prompt_ingredient_selection(self, possible_ingredient_list: list[str]) -
> list[str]:
        while True:
            check_err = False
```

```
            ingredient_selection = []
            print("Ievades formāts: sastāvdaļa1, sastāvdaļa2, ...")
            print("Ievade: ", end="")
            ievade: str = input().lower().replace(" ", "")

            # Iziet
            if ievade == "q":
                sys.exit(0)

            # Regex filtrs, vai ievadītais virspusēji atbilst ievades formātam
            ievade_pattern: re.Pattern[str] = re.compile("[^,]+")
            if ievade_pattern.match(ievade):
                # Noņem atstarpes un sadala atsevišķās sastāvdaļās
                ievade_sadalits: list[str] = ievade.split(",")

                # Ja sastāvdaļā ir iespējamo sastāvdaļu sarakstā,
                # tad tā tiek pievienota beigu sastāvdaļu sarakstam.
                # Ja tā nav, tad tiek izmesta kļūda, kurā parādīta neiespējamā
sastāvdaļā.
                for ingredient in ievade_sadalits:
                    if ingredient in possible_ingredient_list:
                        ingredient_selection.append(ingredient)
                    else:
                        check_err = True
                        print(f"Sastāvdaļa '{ingredient}' nav iespējamo
sastāvdaļu sarakstā")
                        break
                if not check_err:
                    return ingredient_selection
            else:
                print("Ievade neatbilst formātam")


#====================Minimizēšanas opcijas ievades metode==================
    def prompt_minimization(self) -> bool:
        while True:
            print("----Vai prioratizēt receptes, kas satur tikai Jūsu ievadītās
sastāvdaļās?----")
            print("Ievades formāts: jā/nē (j/n)")
            print("Ievade: ", end="")
            ievade: str = input().lower().replace(" ", "")

            # Iziet
            if ievade == "q":
                sys.exit(0)

            # Ja ievade ir "jā" vai "j", tad metode atgriež True, ja nē, tad
attiecīģi False.
            # Ja ievadē ir kaut kas cits, tad tiek izmests kļūdas paziņojums
            if ievade in ["jā", "ja", "j"]:
                return True
            elif ievade in ["nē", "ne", "n"]:
                return False
            else:
                print(f"Nepareizs ievades formāts. Ievadei jābūt vai nu jā/nē,
vai j/n. Jūsu ievade: {ievade}")

#====================Recepšu skaita ievades metode==================
    def prompt_recipe_count(self) -> int:
        while True:
            print("----Cik receptes vēlaties redzēt?----")
            print("Ievades formāts: (skaitlis)")
            print("Ievade: ", end="")
```

```
            ievade: str = input().lower().strip()

            # Iziet
            if ievade == "q":
                sys.exit(0)


            # Ja ievadītais nav vesels skaitlis, tad tiek izmests kļūdas
paziņojums
            if not ievade.isdigit():
                print(f"Nepareizs ievades formāts. Ievadei jābūt veselam
skaitlim. Jūsu ievade: {ievade}")
            else:
                break

        return int(ievade)

#====================Konkrētās receptes izvēles metode=================
    def prompt_recipe_selection(self) -> int:
        while True:
            print("----Kuru recepte vēlaties apskatīt detalizētāk?----")
            print("Ievades formāts: skaitlis")
            print("Ievade: ", end="")

            ievade: str = input().lower().strip()

            # Iziet
            if ievade == "q":
                sys.exit(0)
            # Ja ievade nav skaitlis, tad tiek izmests kļūdas paziņojums
            if not ievade.isdigit():
                print(f"Nepareizs ievades formāts. Ievadei jābūt veselam
skaitlim. Jūsu ievade: {ievade}")

            # Ja izvēlētās receptes indekss ir lielāks par recepšu skaitu, tad
tiek izmests kļūdas paziņojums
            elif int(ievade) > self.recipe_count:
                print("Nepareiza ievade. Izvēlētā recepte nav sarakstā")
            else:
                break

        return int(ievade)

#====================Sastāvdaļu izvades metode==================
    def output_possible_ingredient_list(self) -> list[str]:
        # MVP ietvaros definētais iespējamo sastāvdaļu saraksts
        ingredient_list = self.defined_possible_ingredient_list
        print("----Izvēlieties sastāvdaļas----")
        # Izdrukā standartizvadē iespējamās sastāvdaļas
        print("Pieejamās sastāvdaļās: ", end="")
        for ingredient in ingredient_list:
            print(ingredient + " ", end="")
        print("")

        return ingredient_list

#====================Iegūto recepšu saraksta izvades metode=================
    def output_recipe_list(self) -> None:
        # Ja minimizēšanas opcija ir patiesa, tad tiks veikts minimizētais API
pieprasījums
        if self.minimization_option:
            # Veic minimizētu HTTP pieprasījumu API
            recipe_data = rd.RecipeData()
```

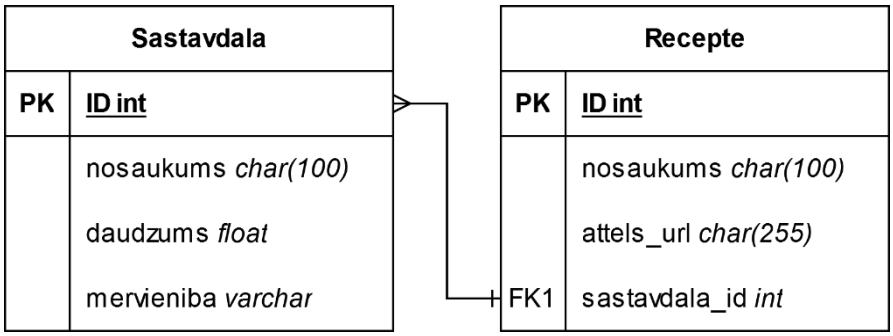
```
            self.json_dati =
recipe_data.get_minimized(self.ingredient_selection, self.recipe_count)

            # Gadījumā, ja neeksistē receptes ar izvēlētajām sastāvdaļam, tiek
izdrukāts kļūdas paziņojums
            try:
                self.json_dati[0]
            except:
                print("Nav recepšu ar izvēlētajām sastāvdaļām!")
                sys.exit(0)

            # Izdrukā receptes
            print("[Izvēle]   Nosaukums")
            print("-------------------------------------------")
            for indekss, nosaukums in enumerate(self.json_dati):
                print("[{}]  {}".format(indekss + 1, nosaukums["title"]))
            print("-------------------------------------------")

        # Ja nē, tad tiks veikts parastais API pieprasījums
        else:
            # Veic parastu HTTP pieprasījumu API
            recipe_data = rd.RecipeData()
            self.json_dati = recipe_data.get_normal(self.ingredient_selection,
self.recipe_count)

            # Gadījumā, ja neeksistē receptes ar izvēlētajām sastāvdaļam, tiek
izdrukāts kļūdas paziņojums
            try:
                self.json_dati[0]
            except:
                print("Nav recepšu ar izvēlētajām sastāvdaļām!")
                sys.exit(0)

            # Izdrukā receptes
            print("[Izvēle]   Nosaukums")
            print("-------------------------------------------")
            for indekss, nosaukums in enumerate(self.json_dati):
                print("[{}]  {}".format(indekss + 1, nosaukums["title"]))
            print("-------------------------------------------")

#=============Receptes detalizētās informācijas izvades metode==================
    def output_detailed_recipe_info(self) -> None:
        self.recipe_selection -= 1

        # Izveido string ar izvēlētajām sastāvdaļām
        izveletas_receptes_izvade: str = "Izvēlētas sastāvdaļas: "
        for i in self.ingredient_selection:
            izveletas_receptes_izvade += i
            izveletas_receptes_izvade += " "


        print("-------------------------------------------------------------")
        print("Recepte: " + self.json_dati[self.recipe_selection]["title"])
        print(izveletas_receptes_izvade)
        print("Attēls pieejams: " +
self.json_dati[self.recipe_selection]["image"])
        print("--------------------Esošās sastāvdaļas---------------------")
        # Izdrukā izmantotās sastāvdaļās, kas receptē netrūkst
        for izmantotas_sastavdalas in
self.json_dati[self.recipe_selection]["usedIngredients"]:
            print("{} {}: {}".format(izmantotas_sastavdalas["amount"],
                                     izmantotas_sastavdalas["unit"],
                                     izmantotas_sastavdalas["name"]))
```

```
            # Papildus piezīmes sastāvdaļām
            if izmantotas_sastavdalas.get("meta", 0):
                for papildus_detalas in izmantotas_sastavdalas["meta"]:
                    print(f"           {papildus_detalas}")

        print("-------------------Trūkstošās sastāvdaļas--------------------")

        # Izdrukā izmantotās sastāvdaļās, kas receptes pagatavošanai trūkst
        for trukstosas_sastavdalas in
self.json_dati[self.recipe_selection]["missedIngredients"]:
            print("{} {}: {}".format(trukstosas_sastavdalas["amount"],
                                     trukstosas_sastavdalas["unit"],
                                     trukstosas_sastavdalas["name"]))

            # Papildus piezīmes sastāvdaļām
            if trukstosas_sastavdalas.get("meta", 0):
                for papildus_detalas in trukstosas_sastavdalas["meta"]:
                    print(f"           {papildus_detalas}")
        print("-------------------------------------------------------------")

#====================Rindstarpu ģenerācijas utilītfunkcija=================
    def newline(self, count: int) -> None:
        for _ in range(count):
            print("")
```

Datne *recipe_data.py*

```
import requests
import json
from datetime import datetime

#====================Recepšu datu klase==================
class RecipeData:
    def __init__(self) -> None:
        self.query_ingredients_list: list[str]
        self.request_data: str
        self.save_requests: bool = False

        self.API_KEY: str = "################################"
        self.BASE_API_URL: str =
"https://api.spoonacular.com/recipes/findByIngredients"

#=================Minimizētā API pieprasījuma veikšanas metode==================
    def get_minimized(self, ingredients_list: list[str], recipe_output_count:
int) -> any:
        self.query_ingredients_list: list[str] = ingredients_list
        # Ģenerē parciālu API pieprasījuma URL ar nepieciešamajām opcijām
        options_url: str = self.generate_partial_url(recipe_output_count) +
"&ranking=2"

        # Veic HTTP pieprasījumu
        headers: dict[str, str] = {"x-api-key": self.API_KEY}
        request: requests.Response = requests.request("GET", self.BASE_API_URL +
options_url, headers=headers)

        # Parsē atgrieztos pieprasījuma datus uz JSON
        self.request_data: any = request.json()

        return self.request_data


#====================Parastā API pieprasījuma veikšanas metode==================
    def get_normal(self, ingredients_list: list[str], recipe_output_count: int)
-> any:
```

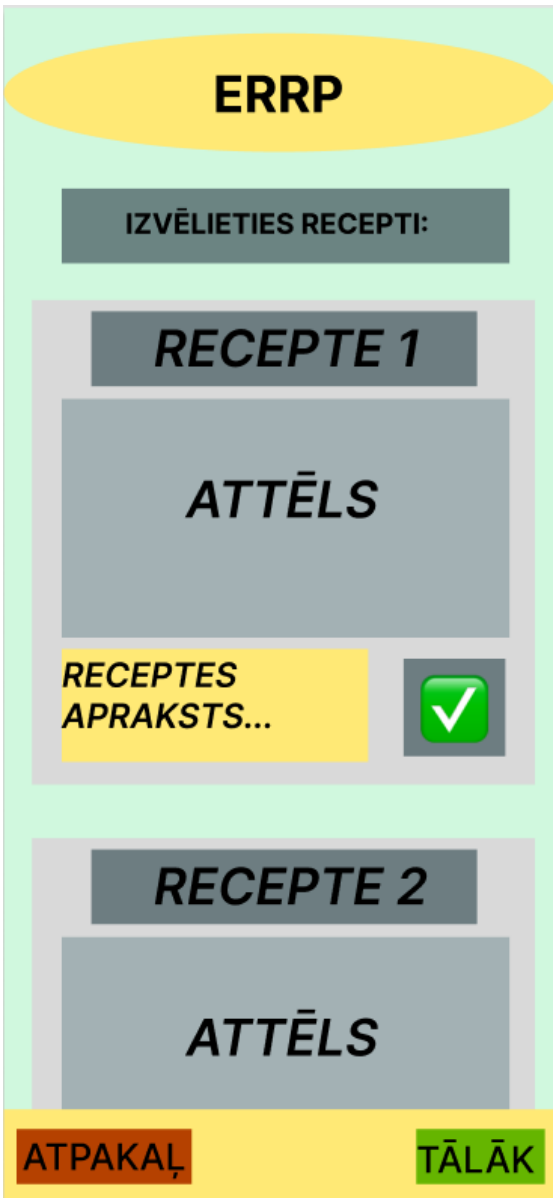
```
        self.query_ingredients_list: list[str] = ingredients_list
        # Ģenerē parciālu API pieprasījuma URL ar nepieciešamajām opcijām
        options_url: str = self.generate_partial_url(recipe_output_count)

        # Veic HTTP pieprasījumu
        headers = {"x-api-key": self.API_KEY}
        request = requests.request("GET", self.BASE_API_URL + options_url,
headers=headers)

        # Parsē atgrieztos pieprasījuma datus uz JSON
        self.request_data: any = request.json()

        return self.request_data

#====================Parciālā URL ģenerēšanas metode==================
    def generate_partial_url(self, recipe_output_count: int) -> str:
        options: str = "?ingredients="
        options += self.query_ingredients_list[0]

        # Pievieno visas sastāvdaļas
        for i in range(len(self.query_ingredients_list)):
            if i > 0:
                options += ",+"
                options += self.query_ingredients_list[i]

        # Pievieno vēlamo recepšu skaitu
        if recipe_output_count > 0:
            options += f"&number={recipe_output_count}"

        return options
```



1. attēls. Iespējamais datu bāzes relāciju modelis.

Funkcionāli ERRP vienai sastāvdaļai nevar būt vairākas receptes.

1. attēls. Lietotājsaskarsnes pirmais skats.



2. attēls. Lietotājsaskarsnes otrais skats.

3. attēls. Lietotājsaskarsnes trešais skats.