**Osu! Discord Bot**

**(osu-dc-bot)**

**Konceptuālais modelis**

**1.1 versija**

|  |  |  |  |
| --- | --- | --- | --- |
| **Datums** | **Versija** | **Apraksts** | **Autors** |
| **16.12.2023** | **1.0** | **Konceptuālā modeļa dokumenta izveide** | **Dāvis Ē. Adriāns P. Aleksandrs V.** |
| **08.01.2024** | **1.1** | **Objektu relācijas diagrammas atjaunošana** | **Adriāns P., Dāvis Ē.,**  **Aleksandrs V.** |

**Saturs**

[1. Konceptuālais modelis 3](#_Toc155686095)

[1.1. Programmēšanas valoda, izmantotās bibliotēkas 3](#_Toc155686096)

[1.2. Secību diagrammas 4](#_Toc155686097)

[1.3. Objektu relācijas diagramma 9](#_Toc155686098)

# 1. Konceptuālais modelis

## 1.1. Programmēšanas valoda, izmantotās bibliotēkas

Projekta prototips tiek izstrādāts programmēšanas valodā ‘Python’. Šajā valodā izmantotā sintakse ir daudz vienkāršāka, jaunas programmas funkcionalitātes implementēt ir daudz ātrāk, turklāt daudzās trešo partiju bibliotēkas automatizē vairākus procesus, kurus citādi vajadzētu manuāli implementēt ļoti ilgi un neefektīvi.

Prototipēšanai tiek izmantotas sekojošās bibliotēkas un rīki:

* *discord.py* ir *vreperis* oficiālajam Discord API. Bibliotēka automātiski veic zema līmeņa https pieprasījumu komplektus, ļaujot programmētājam veikt vien augsta līmeņa koda pārraudzīšanu. Bibliotēka dod tiešu piekļuvi daudzām botu darbībām, piemēram, sarakstes uzzraudzīšanai vai komandu izpildīšanai.
* *dotenv.py* ļauj piekļūt sistēmas vides mainīgajiem(environment variables) bez sistēmas specifiskām piekļuves shēmām.
* Osu API sesija tiek izveidota un uzturēta manuāli, bet procesa atvieglošanai tiek izmantota bibliotēka *aiohttp. aiohttp* ļauj veikt asinhronuspieprasījumus, kā arī spēj veikt vairākus HTTP datu pieprasījumus vienlaikus.

Galvenā produkta izstrādāšana notiek programmēšanas valodā *TypeScript*. Tādējādi projekta kļūdas tiek noskaidrotas jau kompilēšanas solī caur *tipošanas* principu, programmas darbības ātrums ir daudz lielāks, salīdzinājumā ar *Python* (*Python* tiek interpretēts uz C++, bet *Typescript* – kompilēts), un modulārā struktūra, ko *Typescript* iedrošina izmantot, padara koda testēšanu daudz vienkāršāku.

Produkta izstrādāšanai tiek izmantotas sekojošās bibliotēkas:

* *discord.js* piedāvātās funkcionalitātes ir ļoti līdzīgas *discord.py*, bet šeit bibliotēkas struktūra ir daudz intuitīvāka.
* *osu-api-extended* ir osu! oficiālā API *vreperis*. Tas paslēpj manuālo savienojuma uzturēšanu un ļauj programmētājam uzturēt kodu, kas paredzēts darbībām saistībā ar *osu!* datiem(rezultātu tabulu iegūšana, personalizētu spēļu detektēšana), tikai augstā līmenī.
* *typeorm* atvieglo datubāzes pieprasījumu rakstīšanu, tīra teksta pieprasījumus aizstājot ar OOP(objektu orientētās programmēšanas) līdzīgiem elementiem.

Visa projekta izstrādes laikā tiks izmantots izstrādes rīks Visual Studio Code, projekta failu vēstures un kopīgošanas rīks GitHub un datubāzes menedžmenta sistēma MySql lietotāju un produkta datu glabāšanai.

## 1.2. Secību diagrammas

A screenshot of a computer

Description automatically generated

[1. att.] Darbību secības diagramma lietotāja reģistrēšanās funkcionalitātei

Lietotājs spēlē spēli. DiscordController, kas pasīvi darbojas fonā, detektē, ka lietotājs spēlē spēli, un ziņo BotController par jaunu lietotāju. BotController secīgi pārbauda, vai lietotājs vēl nav reģistrēts/vai lietotājam ir ļauts reģistrēties, vai spēlētā spēle ir produkta atbalstīta, un, ja atbilde uz abiem vaicājumiem ir apstiprinoša, lietotāju reģistrē datubāzē un lietotājs tiek informēts par reģistrāciju. Ja lietotāja reģistrācija nav sekmīga, process tiek pārtraukts un lietotājs par to netiek informēts.

A screenshot of a computer

Description automatically generated

[2. att.] Darbību secības diagramma lietotāja informācijas atjaunošanas funkcionalitātei

Lietotājs produktam aplikācijā veic informācijas maiņas pieprasījumu, čata formātā uzrakstot paroli autentifikācijai, informācijas lauka nosaukumu un jauno informāciju. BotController secīgi pārbauda, vai parole sakrīt ar lietotāju saistīto datubāzē, vai informācijas lauka nosaukums ir korekts un vai informācija ir laukam atbilstoša. Ja atbilde uz visiem vaicājumiem ir apstiprinoša, datubāzē lietotāja informācija tiek nomainīta un lietotājs par veiksmīgo maiņu tiek informēts. Ja kāds vaicājums ir noliedzošs, process tiek pārtraukts un lietotājs saņem kļūdas paziņojumu.

A screenshot of a computer

Description automatically generated

[3. att.] Darbību secības diagramma lietotāju labāko nedēļas rezultātu apkopošanas funkcionalitātei

Lietotājs produktam aplikācijā veic nedēļas rezultātu atjaunināšanas pieprasījumu BotController, vai to automātiski inicializē iknedēļas taimeris. BotController pieprasa sarakstu ar visiem lietotājiem no datubāzes, un katram lietotājam veic pieprasījumu OsuController, atpakaļ iegūstot apstrādātu apkopojumu. Pēc procesa izdarīšanas šo sarakstu padod DiscordController, kas informāciju sakārto un publicē serverī.

A screenshot of a computer

Description automatically generated

[4. att.] Darbību secības diagramma karšu aptaujas uzsākšanas funkcionalitātei

Lietotājs produktam aplikācijā veic karšu aptaujas uzsākšanas pieprasījumu. BotController no datubāzes pieprasa sarakstu ar lietotājiem. Katram lietotājam tiek noskaidrots tiešsaistes statuss un 3 iemīļotākās kartes. Pēc tam OsuController iegūst piecas iecienītākās kartes, un šo sarakstu padod DiscordController, kas izveido aptauju un publicē to serverī.

A diagram with text and images

Description automatically generated with medium confidence

[5. att.] Darbību secības diagramma lietotāja informācijas apskatīšanas funkcionalitātei

Lietotājs produktam aplikācijā veic informācijas pieprasījumu, čata formātā uzrakstot informācijas lauka nosaukumu. BotController secīgi pārbauda, vai lietotāja identifikators eksistē datubāzē un vai informācijas lauka nosaukums ir korekts. Ja atbilde uz abiem vaicājumiem ir apstiprinoša, no datubāzes tiek pieprasīta lietotāja informācija un lietotājs to saņem čata formātā. Ja kāds vaicājums ir noliedzošs, process tiek pārtraukts un lietotājs saņem kļūdas paziņojumu.

## 1.3. Objektu relācijas diagramma

A diagram of a product

Description automatically generated

“DiscordUser” tabulā tiek glabāts reģistrēto Discord lietotāju saraksts. Katram lietotājam tiek saglabāts unikāls ID datubāzē, izmantotais lietotājvārds Discordā, diskriminators lietotājvārdam, reģistrācijas datums produktam, pēdējās izmaiņas lietotājam datubāzē, kā arī izdzēšanas statuss.

“OsuUser” tabulā tiek glabāts reģistrēto osu! lietotāju saraksts. Katram lietotājam tiek saglabāts unikāls ID datubāzē, izmantotais lietotājvārds osu!, osu! profile ID, reģistrācijas datums produktam, pēdējās izmaiņas lietotājam datubāzē, kā arī izdzēšanas statuss.

“ProductUser” tabulā tiek glabāti mākslīgie Discord lietotāju un osu! lietotāju pārojumi. Katram abstraktajam lietotājam tiek saglabāts unikāls ID datubāzē, atsauce uz Discord lietotāja ID, atsauce uz osu! lietotāja ID, lietotāja lietotājvārds, šifrētās paroles šifrs un šifra beigas.

“GameStats” tabulā tiek glabāti rezultāti katram lietotājam par kādu reģistrētu sasniegumu kādā spēlē. Tiek saglabāts unikāls sasnieguma ID, atsauce uz abstraktā lietotāja ID, spēlētās spēles ID, sasnieguma skaitliskais rezultāts, sasnieguma rangs starp citiem, sasnieguma tips un sasnieguma sasniegšanas laiks(datums un pulksteņlaiks).

DiscordUser un ProductUser – one-to-one, vienam produkta lietotājam var būt viens saistīts Discord konts, lai gan saistītais konts var mainīties.

OsuUser un ProductUser - one-to-one, vienam produkta lietotājam var būt viens saistīts osu! konts, lai gan saistītais konts var mainīties.

ProductUser un GameStats – one-to-many, vienam produkta lietotājam var būt vairāki saistīti sasniegumi.

Piebilde – lai gan vienam cilvēkam ir aizliegts spēlēt uz vairākiem osu! kontiem un vairākiem cilvēkiem ir aizliegts spēlēt uz viena osu! konta, vienam cilvēkam var piederēt vairāki Discord konti, līdz ar to ar osu! kontu asociētais Discord konts var mainīties bieži.