1. Slaids

Labdien! Mani sauc Artis Pauniņš un šajā studiju semestrī es izstrādāju kvalifikācijas darbu, kurā tiek veidots .NET izsaukumu adapteris WebAppOS videi kā arī starpkoda ģeneratoru, kurš ģenerē C# kodu.

1. Slaids

Kas ir WebAppOS? Tā ir starpplatformu vide, kas ļauj izsaukt kodu, kas rakstīts dažādās programmēšanas valodās, un kas atrodas dažādos tīkla mezglos jeb serveros.

WebAppOS ir tīmekļa procesori, kas atbild par funkciju izsaukšanu un procesorā ir izsaukumu adapteri, kas nodrošina koda izsaukšanu konkrētā programmēšanas valodā, tāpēc adapterus veido katrai programmēšanas valodai atsevišķi.

1. Slaids

Šajā shēma ir vienkāršoti izskaidrots, kā darbojas WebAppOS. Tajā ir tīmekļa atmiņa, kurā glabājas objekti un kas sinhronizē WebAppOS. Vēl ir serveri, kuros glabājas funkcijas, kuras var izsaukt WebAppOS vidē, un lai šīs funkcijas izsauktu, ir nepieciešams adapteris konkrētai programmēšanas valodai. Kā redzams pašlaik servera pusē ir adapteri Javā un lua valodās bet nav C# adaptera.

1. Slaids

Papildus adapteriem, WebAppOS ir RAAPI, kurš ir zema abstrakcijas līmeņa API tīmekļa atmiņas datu piekļūšanai, bet šo abstrakcijas līmeni vajag augstāku.

Kā pieminēju tīmekļa atmiņā glabājas objekti, bet pašlaik nav iespējams strādāt ar šiem atmiņas objektiem kā ar kādas dotas programmēšanas valodas objektiem.

Tādējādi kopā ir trīs problēmas: C# tīmekļa procesora adaptera trūkums, vēlme pēc augstāka līmeņa API tīmekļa atmiņas piekļūšanai un iespēja strādāt ar atmiņas objektiem kā ar C# objektiem.

1. Slaids

Tāpēc būs nepieciešams tīmekļa procesora adapteris valodai C#, kurš strādā uz .NET 5, augstāka abstrakcijas līmeņa API un jaunas “definēšanas” valodas gramatika un kompilators, kas no jaunās valodas koda, kuru turpmāk saukšu par jaunkodu, ģenerēs starpkodu C# valodā. Ar Šī starpkoda palīdzību varēs strādāt ar tīmekļa atmiņas objektiem kā ar C# objektiem. Šis starpkods apraksta tīmekļa atmiņas datus, tāpēc tajā tiks izmantos iepriekš minētais API.

1. Slaids

Visiem risinājumiem šajā darbā tiek izmantota programmēšanas valoda C# ar .NET 5, jo .NET 5 ir atbalsts uz vairākām platformām (Platforma = OS + Procesora tips).

Jaunās valodas gramatikas un kompilatora veidošanai apguvu un izmantoju ANTLR parsētāja ģeneratoru, kurš ir labi zināms ģenerators ar atbalstu vairākās labi zināmās programmēšanas valodās, piemēram, C#, Java, JavaScript. Tiek izmantota jaunākā versija – ANTLR4.

Ģeneratoram ir prasība ģenerēt katru definēto klasi savā failā, lai padarītu starpkodu uzturamāku un lai ieverotu OOP principus.

1. Slaids

Šeit ir redzams neliels piemērs jaunkoda sintaksei, kurā ir parādīts, kā tiek definētas klases, tās atribūti un metodes, kā arī asociācijas.

Kā redzams klasēm, atribūtiem un metodēm ir līdzīga sintakse kā C# valodai.

Asociācijām ir atslēgvārds *association* un iekavās ir asociācijas definīcija, kurā ir avota un mērķa galapunkltu definīcija. Galapunktā ir lomas vārds un aiz kola ir klases vārds.

1. Slaids

Šajā shēmā tiek parādīts kompilatora un ģeneratora darbības pamatprincips, kurš ir ļoti vienkāršs. Tiek padots vārdtelpas nosaukums un jaunkoda fails. Vispirms tiek pārbaudīts, vai vārdtelpa ir pareizā formātā un tad tiek kompilēts jaunkods. Ja pēc vārtelpas pārbaudes un jaunkoda kompilēšanas nav konstatētas kļūdas, tad tiek ģenerēts starpkods.

1. Slaids

Šeit ir attēlota struktūra, kas tiek izmantota jaunkoda informācijas glabāšanai. Pašā kompilatorā glabājas jaunkodā aprakstītās klases, asociācijas un jaunkodā pieļautās kļūdas.

Katrā klasē glabājas atribūti, metodes, asociāciju galapunkti, virsklase un tās apakšklases.

Asociācijās glabā divus asociācijas galapunktus – mērķi un avotu. Katram galapunktam ir lomas vārds un klase, kurai tas ir piesaistīts.

Klasē tiek definēti lauki, kas ir atribūti un metodes un tiem ir jāglabā aizsardzības līmeni, datu tipu un vārdu, bet metodēm vēl papildus glabā anotācijas, kas apraksta metodes un URL, kas ir kā norāde uz metodes kodu.

1. Slaids

Iepriekš tika pieminēts kas ir RAAPI un kam tas ir domāts. Tajā ir funkcijas, kuras var izmantot, lai piekļūtu tīmekļa atmiņā esošajiem objektiem. Šīs RAAPI funkcijas kalpo par pamatu augstā līmeņa API realizācijai. Papildus šīm funkcijām vēl tiek izmantotas LINQ iebūvētās funkcijas, ar kuras palīdzību var iegūt datus un pārveidot tos par objektiem.

1. Slaids

Šeit ir attēlota augstā līmeņa API struktūra.

Šeit galvenās struktūras ir tīmekļa atmiņa, kurā glabājas tīmekļa klases un tīmekļa klase, kurā glabājas klases objekti, asociāciju galapunkti un atribūti.

Atribūtos glabājas datu tips un vārds, asociāciju galapunktos glabājas lomas vārds un klases, objektos jeb instancēs glabājas atribūtu vērtības un tiem saistītie objekti.

1. Slaids

Izsaukumu adapteris tika realizēts izmantojot .NET iebūveto bibliotēku *reflection* jeb refleksiju, ar kuru var ne tikai peikļūt .NET klasēm un atribūtiem, bet arī iegūt funkcijas un tās dinamiski izsaukt. Kā minēju katrai funkcijai ir URL un tas tiek rakstīts formātā *protokols:lokācija:ceļš,* kur protokols ir programmēsanas valoda, kurā tiek rakstīts izsaucamais kods, lokācija, kas apraksta to, kur atrodas izsaucamais kods (vai tas atrodas lokali uz datora vai serverī) un ceļš norāda, kur koda atrodas metode (katrai programmēšanas valodai ir citādāk).

Refleksīvajā izsaukšanā vispirms bija nepieciešams sadalīt URL, lai atsevišķi iegūtu informāciju par protokolu, lokāciju un ceļu. C# gadījumā ceļu vajadzēja sadalīt vēl divās daļās, kur pirmo daļu izmanto klases atrašanai un otro daļu – metodes atrašanai. Kad visi dati ir iegūti, tad tiek izsaukts klases konstruktors, lai pēc tam varētu izsaukt izsaucamo metodi.

1. Slaids

Rezultātā ir iegūtas divas lietas. Pirmkārt ir izstrādāts kompilators, kurš kompilē jaunizveidotas valodas kodu un no šī koda ģenerē starpkodu C# vidē.

Kā arī ir iegūta iespēja WebAppOS vidē izsaukt funkcijas ar .NET adapteri un var darboties ar tīmekļa atmiņas objektiem kā ar C# objektiem izmantojot ģenerēto starpkodu.

Lai viss strādātu tika izmantotas trīs tehnoloģijas – Java, lai darbotos WebAppOS, .NET 5, lai darbotos izstrādātie risinājumi un WebAppOS kernell, lai izmantotu RAAPI funkcijas.

Tā kā šie risinājumi tika veidoti WebAppOS, tad tie tiks iekļauti WebAppOS distribūcijā.

1. Slaids

Nedaudz par to, kāds bija darba tapšanas process. Tad projekts strādāja pēc spirālveida paradigmas. Tas nozīmē, ka tika izrunātas noteiktas prasības, tad tika veidots kods un kad kods darbojas, tad tiek runātas par jaunām prasībām un tās tiek kodētas. Kad kods strādā, tad tās risinājums tika dokumentēts. Testēšana notika pēc visu prasību kodēšanas.

Katras dienas veiktās porkgrammkoda izmaniņas vai jauno funkciju pievienošana tika atjaunota atsevišķā GitHub repozitorijā, īsi pieminot, kas tika veikts.

1. Slaids

Rēķinot darbietilpību tika pieņemts, ka varu saražot 15 funkcijpunktus koda. Tad tika noteikts, kas ir jābūvē katram konkrētam risinājumam un kam ir jāpievērš uzmanība. Veicot šo darbību, rezultātā tika iegūts, ka ģeneratoram ir jāsaražo aptuveni 21 funkcijpunkts, bet adapterim un augstā līmeņa API – 24 funkcijpunkti.

Izdalot iegūtos funkcijpunktus ar pieņemto vērtību var secināt, ka darbs iekļaujas vismaz trīs cilvēkmēnešos un vēl tika saskaitīts, ka visiem risinājumiem tika rakstītas aptuveni 2700 koda rindiņas.