Szegedi Tudományegyetem

Informatikai Intézet

SZAKDOLGOZAT

Révész Gergő László

2022

Szegedi Tudományegyetem

Informatikai Intézet

Általános feladatütemező szolgáltatás

Szakdolgozat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Készítette: |  | Témavezető: |  |
|  | Révész Gergő László |  | Prágai József |  |
|  | programtervező informatikus szakos hallgató |  | asd |  |

Szeged

2022

*Feladatkiírás*

A szakdolgozat keretein belül egy általános feladat ütemező készült, amely segítségével időszakosan végrehajtandó funkciók futtathatók, előre definiált időközönként.

A szakdolgozatomat a GriffSoft Informatikai Zrt.-nél készítettem. A cég fő terméke a Forrás ügyviteli rendszer, amelyet főként Önkormányzatok, Költségvetési intézmények és középvállalatok használnak.

Az ügyviteli rendszer használata során többször felmerül az igény, hogy szükség van adott időközönként rendszer- vagy egyéb ügyviteli funkció futtatására. Ilyen lehet egy e-mail küldés vagy akár a NAV számára küldendő Online számla bevallás.

Ennek megoldására készítettem egy központi ütemező alkalmazást, amelybe ezeket az időszakos futást igénylő funkciókat könnyen beilleszthetjük, egyedileg kezelhetjük, paraméterezhetjük.

Az ütemezőben definiált feladatok végrehajtásának figyelésére, egy böngésző alapú alkalmazás is fejlesztésre került, ami által egyszerűen nyomon követhetjük az aktív feladatok futási státuszát.

## Tartalmi összefoglaló

* ***A téma megnevezése:***

*Általános feladatütemező szolgáltatás*

* ***A megadott feladat megfogalmazása:***

*A feladat egy olyan alkalmazás készítése, amely dinamikusan, futásidőben tud Windows szolgáltatáshoz funkciókat hozzáadni és törölni, valamint egy felhasználói felület készítése ezen funkciók állapotainak megfigyelésére.*

* ***A megoldási mód:***

*Egy, a keretrendszer által nyújtott technológia, a Reflection segítségével futásidőben tud a program változtatni saját szerkezetén, és ezt kihasználva készítettem egy újra indítást nem igényló, de szerkezetileg folyamatosan változtató programot.*

* ***Alkalmazott eszközök, módszerek:***

*Az alkalmazást a Visual Studio nevű fejlesztői környezetben készítettem el, C# nyelven.*

*A fejlesztés Windows szolgáltatás részéhez a .NET Core, a felhasználói felület részéhez a Blazor keretrendszereket használtam. Az alkalmazást több eszközön, különböző verziószámokkal teszteltem.*

* ***Elért eredmények:***

*Az alkalmazást a Griffsoft Informatikai Zrt. rendszere használja.*

* ***Kulcsszavak:***

***//TODO csak 4-5 kell***

*számítógép, program, assembly, futtatás, aszinkron, appDomain,*

***Tartalomjegyzék***

Feladatkiírás 2

Tartalmi összefoglaló 3

Tartalomjegyzék 4

Bevezetés 5

1. Használt technológiák, eszközök bemutatása 6
   1. Programozási nyelv 6
   2. Fejlesztői környezet 7
   3. Keretrendszer 8
2. Asd //TODO
   1. Windows service
   2. Blazor
3. Rendszerterv, a megvalósításhoz szükséges összetevők specifikálása
   1. Funkciók specifikálása
   2. Felhasználói felületek

…

Megvalósítás menete

Irodalomjegyzék 7

Nyilatkozat 8

**Bevezetés**

A Griffsoft Informatikai Zrt. fő terméke a Forrás ügyviteli rendszer, amit Önkormányzatok, Költségvetési intézmények és középvállalatok is használnak adminisztrációs célokra. A Forrás használata közben számos alkalommal felmerül az igény bizonyos funkciók előre meghatározott időközönként történő lefuttatására. Az ilyen funkciók pl. egy e-mail küldés vagy a NAV számára küldendő Online számla bevallás.

Az ellátandó feladatok megnövekedett száma miatt a dolgozók munkáját megkönnyítené, ha ezeket egy számítógépes program végezné el automatikusan. Ezen probléma megoldása miatt esett a választásom egy olyan alkalmazás elkészítésére, ami ezeknek a feladatoknak eleget tesz.

A szakdolgozatom célja egy olyan Windows operációs rendszeren futó szolgáltatás programozása, amit mindamellett, hogy menedzseli a szükséges funkciók futtatását, lehetőséget nyújt újabb feladatok hozzáadására és meglévők törlésére egyaránt anélkül, hogy meg kellene szakítani a program futását.

//TODO Blazor rész

A kész alkalmazás egy könnyen használható Windows szolgáltatás, aminek továbbfejlesztése kevés programozói tudást igényel. A dinamikus megvalósításnak köszönhetően bármennyi ideig képes futni nagyon kevés erőforrás használatával, és kezeli a funkciók futtatása közben keletkező esetleges hibákat.

//TODO bővíteni

1. **Használt technológiák, eszközök bemutatása**

**1.1. Programozási nyelv**

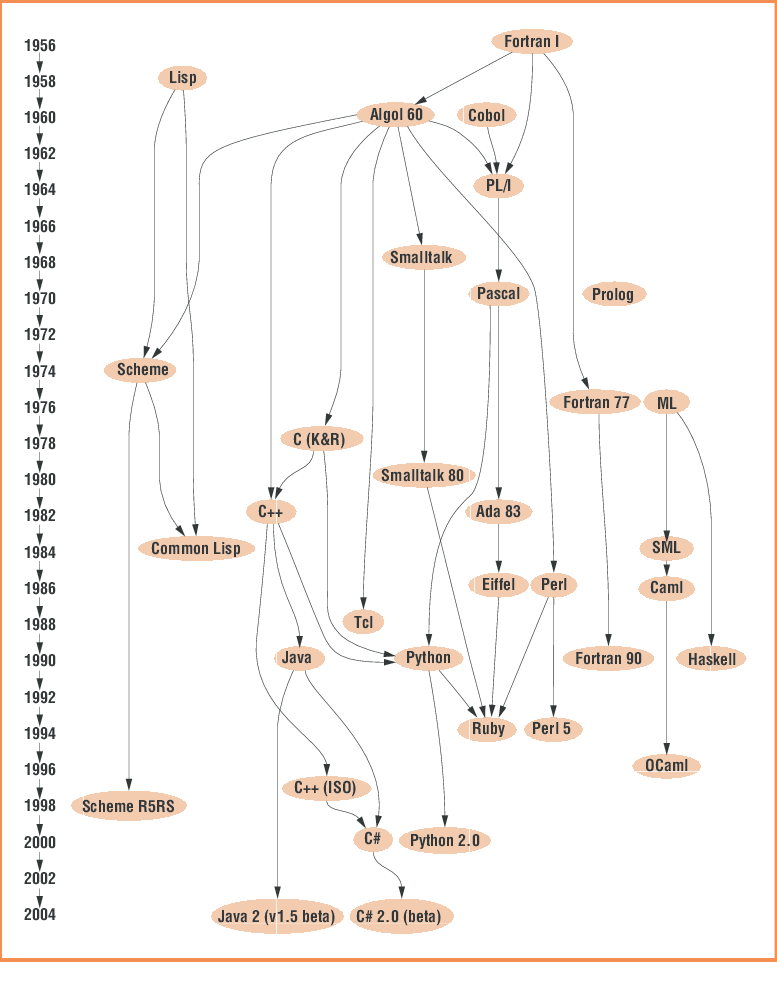
**//TODOref**:https://www.researchgate.net/figure/A-brief-history-of-high-level-programming-languages-from-1956-to-2004\_fig1\_3248243

Egy mai számítógép az emberek számára követhetetlen számolási sebességgel bír, és ez miatt jó ötletnek tűnhet, ha az egyes emberi feladatokat rá tudnánk bízni egy számítógépre. Ezek „agya” egyelőre még nem ért emberi nyelven, de akkor mégis hogyan tudnánk megmondani egy számítógépnek, hogy mit csináljon?

Egy számítógép utasításokat hajt végre. Egy utasítás lehet például egy szorzás, ami szétbontható összeadások összegére. Egy összeadást is tovább lehet egyszerűsíteni több, de egyszerűbb műveletre, és ezt addig ismételni, amíg annyira elemeire bontott utasítást nem kapunk, hogy azt egymás után írt egyesekkel és nullákkal (bitekkel) reprezentálni lehet. Az így kapott bitek sorozata egy számítógép számára már értelmezhető, de az ember számára szinte érthetetlenné váló utasítás. Már a legegyszerűbb matematikai műveleteket is nagyon hosszadalmas és macerás munka ilyen kis elemi részekre bontani annak érdekében, hogy elvégeztethessük egy számítógéppel, nem beszélve a komplexebb programokról, amik az előbb említett bonyolultabb utasítások egymás utáni sorozatából állnak.

Ennek a problémának a megoldására születtek meg a programozási nyelvek. A programozási nyelvek mindegyike, csak úgy, mint egy emberi nyelv, rendelkezik saját szabályrendszerrel. Egy szabályrendszer lehetőséget ad az emberek számára egységes módon megfogalmazni egy utasítást, vagy annak sorozatát, egy programot. Egy program a nyelv szabályait követve mindamellett, hogy az más, a szabályokat szintén ismerő ember számára is olvashatóvá és érthetővé válik, a programozási nyelv a programnak el tudja készíteni a számítógép számára érthető megfelelőjét. Ezt a folyamatot, amikor egy ember által olvasható programból egy gép által olvasható, bitekből álló gépi kód készül, fordításnak nevezik. A fordítást a programozási nyelv saját fordítója csinálja, ami csakugyan egy program, ezért természetesen a legelső ilyen fordítót programozási nyelv hiányában nehéz volt „megírni”.

Attól függetlenül, hogy mindegyik végeredménye egy gép számára érthető és végrehajtható utasítás sorozat, a ma létező programozási nyelvek között érdemes lehet választani az alapján, hogy milyen célra szeretnénk programot készíteni, ugyanis ettől függően változhat a program megírásának nehézsége és minősége is. Én a szakdolgozatomat a Microsoft által fejlesztett C# programozási nyelven készítettem el.



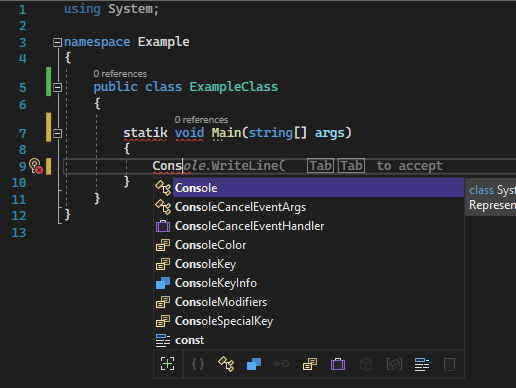
1.1. ábra: magas szintű programozási nyelvek története

**1.2. Fejlesztői környezet**

**//TODO ref**: https://ak-akademia.hu/fejlesztoi-kornyezetek/

Programot írni bárhova tudunk, ahol lehet betűket tárolni, beleértve egy papírlapot, de még egy barlang falát is. Ahhoz viszont, hogy elérjük, hogy ezeket az utasításokat egy számítógép hajtsa végre, először is kell a programozási nyelvnek előbb említett fordítójára, másodszor pedig egyes programok esetében még szükségünk lehet egy futtatókörnyezetre is.

A fejlesztői környezetek olyan programozók számára készült szoftverek, amik a fordítót, a futtatókörnyezetet és még egy szövegszerkesztőt is tartalmaznak annak érdekében, hogy a programozónak a rendelkezésére álljon minden egy helyen. Egy ilyen fejlesztői környezet nagyon sokat tud gyorsítani egy program fejlesztésén. Léteznek azonban olyan fejlesztői környezetek is, amik mindamellett, hogy tudják ezek mindegyikét, még adott programozási nyelvek formai szabályait is „ismerik”, ami miatt még több segítséget tudnak nyújtani. Képesek például a programkód gépelése közben vétett hibákat jelezni a fejlesztő számára, felugró listával segíteni befejezni a nyelv kulcs szavait, képes velük a fejlesztő a program futása közben belelátni a számítógép memóriájának bizonyos részeibe (debugolás), és még ezeken kívül is rengetek hasznos funkcióra tehet szert egy fejlesztő a használatukkal. Ezeket a fejlesztői környezeteket integrált fejlesztői környezeteknek hívják, és manapság már a leggyakrabban használt programozási nyelvek mindegyikéhez létezik ilyen.



1.2. ábra: a Visual Studio egy hibajelzése és egy kiegészítő javaslata

A fejlesztői környezet tehát egy olyan szoftver, ami a programozóknak hatalmas segítséget nyújt a programok fejlesztésében, javításában. A szakdolgozatom programkódját a C# programozási nyelv egy szintén Microsoft által készített integrált fejlesztői környezetét, a Visual Studio-t használva készítettem el.

**1.3. Keretrendszer**

//TODO ref: <https://hackr.io/blog/what-is-frameworks>

//TODO ref: https://hu.wikipedia.org/wiki/Szoftverkeretrendszer

Egy nagy projekt megírása sok profi programozónak is rengeteg időbe telik, azonban léteznek módszerek, amikkel fejlesztés ideje lerövidíthető.

A programozók valamilyen hálózaton, például az interneten keresztül meg tudják osztani egymással elkészített munkáikat, ily módon ezt egy másik programozó már fel tudja használni programjában. Jó példa erre a majdnem minden programozási nyelvben megtalálható Math könyvtár, ami matematikai függvényeket tartalmaz. Ennek köszönhetően, ha egy programozó programjának fejlesztése során például hatványozni szeretne, nem kell a nulláról leprogramoznia egy hatványozás függvényt, használhatja a mások által megírt Math könyvtárat.

A könyvtáraknál még nagyobb segítséget tudnak nyújtani a keretrendszerek. Ezek szintén programozók által készített, de a könyvtáraknál jóval bővebb rendszerek, amik sok más programozást segítő eszközt is tartalmaznak, ilyenek például az API-k (//TODO RÉSZLETEZZEM??). A keretrendszerek általános eszköztárat biztosítanak, amik segítségével a fejlesztőnek nincs szükség mindent a nulláról felépíteni, helyette a lényegi részre, a funkciók implementálására fókuszálhat nagyban lerövidítve ezzel a projekt fejlesztésének idejét. Egy web alkalmazás esetében például nem kell azzal foglalkozni, hogyan jeleníthetjük meg azt egy böngésző ablakban, helyette az időnket a lényegi részre, az alkalmazás funkcióinak fejlesztésére fordíthatjuk.

//TODO blazor

Nagy különbség a kettő között, hogy könyvtárak esetében a programozó munkája lesz kibővítve máséval, a keretrendszerek viszont a programozó munkájával kiegészítve építenek fel egy működő programot.

A keretrendszerek használatának elsajátítása nagyon meggyorsíthatja egy program fejlesztését, szükségtelen használata viszont csak ront a program performanciáján. A szakdolgozatomhoz a C# programozási nyelv keretrendszerét, a .NET keretrendszert használtam, ami a nagy eszköztárának köszönhetően meggyorsította munkámat.

***Irodalomjegyzék***

***Nyilatkozat***

Alulírott Révész Gergő László, Programtervező informatikus szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Szoftverfejlesztés Tanszékén készítettem, Programtervező informatikus BSc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Informatikai Intézet könyvtárában, a helyben olvasható könyvek között helyezik el.

Szeged, 2022. január 17.

aláírás