**IndexOutOfRange Csapat**

**Klaszter menedzser program dokumentáció**

A dokumentáció 3 részre tagolható:

1. **Bevezetés:** Tartalmazza a program rövid bemutatását, a feladat ismertetését, és a csapattagok bemutatkozását.
2. **Felhasználói dokumentáció:** Tartalmazza a program használatához szükséges összes információt. Részletesen ismerteti a program működését és funkcióit, és leírja a lehetséges felhasználói eseteket.
3. **Fejlesztői dokumentáció:** Ismerteti a program fejlesztésének menetét, a szerkezeti felépítését, valamint egyes funkciók technikai működését.

**Bevezetés**

**A programról**

Ez a program azzal foglalkozik, hogy menedzseli egy megadott klaszteren futó folyamatokat. A felhasználó, miután megadta a klaszter elérési útvonalát, létrehozhat és törölhet számítógépeket és programokat is futtathat ezeken a gépeken a klaszteren belül. Továbbá használhat egy klaszter kezelőt, illetve egy algoritmus a futó alkalmazások rendezésére.

**Extra funkciók**

Az alap elváráson kívüli extra funkciók:

* **Klaszter kezelő:** Egy drag-and-drop felület, amin a létrehozott számítógépeket átadhatjuk a klaszterbe.
* **Programok szétosztása:** Egy felhasználható algoritmus, ami egy optimális módon szétosztja a futó alkalmazásokat az aktív számítógépek között.
* **Animált jelzések:** Animált jelzések a felületen, amikor nem fut elegendő, vagy túl sok példány fut egy adott programból
* **Számítógép tulajdonságai:** Megjelenik egy ablak a számítógép tulajdonságairól, amikor rá kattintunk a gombjára a Klaszter kezelőben.

**Feladat ismertetése**

**Verseny**

Ez a program a 2025-ös *Dusza Árpád Országos Programozói Emlékverseny* programozói kategóriájának döntőjébe készült. A verseny regionális fordulója során a 2-3 fős csapatoknak néhány óra alatt kellett egy asztali alkalmazást elkészíteniük. A döntőbe jutott csapatoknak ezt az alkalmazást kellett továbbfejleszteniük, majd a zsűri előtt bemutatniuk.

**Feladat**

A versenyzőknek olyan programot kellett készíteniük, amely alkalmas klaszteren futó folyamatok kezelésére, vagyis képes fájlokban tárolt adatok alapján számítógépeket, illetve programokat szimulálni a klaszteren belül. A döntő során ezt a programot kedvünk szerint egészíthettük ki extra funkciókkal egy grafikus felületen.

**A csapatunk**

Az **IndexOutOfRange** csapat a debreceni *DSZC Mechwart András Gépipari és Informatikai Technikum* 13.B osztályos tanulóiból áll.

**Beke Tamás**

A projektmunka során, én implementáltam a regionális fordulón már elkészült metódusokat, illetve az akkor el nem készült metódusokat én kódoltam le, és implementáltam.

**Gajdos Csanád**

//Majd meglátjuk

**Fejlesztői dokumentáció**

**Fejlesztés menete**

**Verziókezelés**

A verziókezeléshez Git-et használtunk. A Github repónkban a fő branch-en dolgoztunk, mivel eltudtuk különíteni a munkáinkat, hogy amikor egyszerre dolgozunk akkor mindig külön fájlba legyen a munkánk.

**Kommunikáció**

A munka közbeni kommunikáció személyesen, valamint Discord-on keresztül történt. Általában hanghívásos megbeszélést tartottunk, amikben egyeztettünk a projekt jelenlegi állapotával kapcsolatban, átbeszéltük a további feladatokat, illetve segítettünk egymásnak amennyiben elakadtunk valahol.

**Program futtatása**

**Rendszerkövetelmények**

Operációs rendszer: Windows 10 vagy újabb

Felbontás: 1920x1080 vagy nagyobb felbontás

Fejlesztői követelmények:

* Visual Studio 2022 vagy újabb
* DotNet 8.0 vagy újabb

**Program indítása**

A program a Dusza\_WPF.exe fájl megnyitásával futtatható.

//Kép

A programhoz mellékeltünk egy alapértelmezett klasztert, amely tartalmaz 2 számítógépet, illetve néhány alkalmazást.

**Használt technológiák**

**Program**

**C#**

A program elkészítésére a .NET keretrendszert választottuk. A C# erős típusú nyelv, ami csökkenti a fejlesztés során keletkező hibákat. A nyelv objektum-orientált felépítése lehetővé teszi az adatszerkezetek rendezett kezelését, valamint megkönnyíti a program részegységekre bontását. Ezen kívül a LINQ eszközökkel az adatainkat hatékonyan szűrhetjük és alakíthatjuk, megkönnyítve a kimutatások elkészítését.

**Windows Presentation Foundation**

A grafikus felület a Windows Presentation Foundation keretrendszer (rövidebb nevén WPF) segítségével készült. A WPF különböző elrendezési lehetőségeket kínál, például tábla elrendezést vagy rács elrendezést, amelyek segítségével könnyen kialakítható az alkalmazás felhasználói felülete. Erős adatkötési funkciókkal rendelkezik, amelyek lehetővé teszik az adatok dinamikus megjelenítését és frissítését a felhasználói felületen. Továbbá lehetővé teszi a stílusok és sablonok alkalmazását, amelyek segítségével egyszerűen testre szabható az alkalmazás megjelenése.

**Fejlesztői eszközök**

**Git**

A számos verziókezelő rendszer közül a Git-et választottuk, mivel ez a legelterjedtebb megoldás, és mindannyian ismertük. A Git lehetővé teszi, hogy minden változtatást nyomon kövessünk és dokumentáljunk. A különböző ágak használata megkönnyítette a párhuzamos munkavégzést, és segített elkerülni, vagy szükség esetén elhárítani a konfliktusokat.

**Visual Studio 2022**

A fejlesztés főként a Visual Studio fejlesztői környezetben történt. A visual studio számos eszközt kínál, amely megkönnyíti a C# kód írását. A beépített Designer ablak gyorsabbá teszi a felhasználói felület elkészítését. Ezen kívül lehetővé teszi a hibák keresését, és a program teljesítményének tesztelését.

**Program Szerkezeti Felépítése**

**Klaszter (Klaszter.cs)**

//Kép

A KiIratas() metódus kiírja a klaszteren lévő programok neveit, memória, és millimag igényüket, illetve hogy mennyi kell belőlük futni.

**Példány elemek:**

* ProgramName: A klaszteren lévő program neve.
* MennyiActive: Mennyi példányban kell futni az adott programnak.
* Millimag: Menny millimag kell a program futtatásához.
* Memoria: Mennyi memória szükséges a program futtatásához.

**Program folyamatok (ProgramFolyamatok.cs)**

//Kép

**Példány elemek:**

* InditasIdeje: Tárolja a program indításának idejét.
* IsActive: Tárolja, hogy a program aktív e.
* ProcesszorEroforras: Tárolja, hogy mennyi processzor erőforrást igényel a program.
* MemoriaEroforras: Tárolja, hogy mennyi memóriát igényel a program.
* FajlNeve: Tárolja a programfájl nevét.

**Számítógép Konfigurációk (SzamitogepConfig.cs)**

//Kép

A KiIratas() metódus kiírja a klaszteren lévő programok neveit, memória, és millimag igényüket, illetve hogy mennyi kell belőlük futni.

**Példány elemek:**

* Millimag: A számítógép maximális millimag kapacitása.
* Memoria: A számítógép maximális memória kapacitása.
* Eleres: A számítógép mappa elérési útvonala.
* ProgramPeldanyAzonositok: A szamítógépen futú program példányok azonosítói.
* JelenlegiMemoria: A számítógép jelenlegi memória kapacitása.
* JelenlegiMag: A számítógép jelenlegi millimag kapacitása.
* PozicioX: A számítógép X kordinátája a Klaszter canvasban.
* PozicioY: A számítógép Y kordinátája a Klaszter canvasban.