Neumann János Egyetem GAMF Műszaki és Informatikai Kar

Java alkalmazások előadás

Java alkalmazások előadás

fejlesztői és felhasználói dokumentáció

Készítette:

Nagy Antal (RYEEHN) mérnökinformatikus hallgató

Habony Zoltán (RYEEHN) mérnökinformatikus hallgató

Kecskemét, 2022

Tartalomjegyzék

Fejlesztői dokumentáció

com.examle.wpa_alpha

A com.examle.wpa_alpha csomag tartalmazza az alkalmazásunk által használt komponenseket, amit további logikai csoportokra bontottunk és külön csomagokba szerveztük egymástól elkülönítve. Emellett tartalmazza az alkalmazásunknak a fő belépési pontját is a WindPowerApplication.java-t.

WindPowerApllication.java

A WindPowerApllication.java az alkalmazás belépési pontja. Két függvényt tartamaz az osztály egy `public void start(Stage stage)`, ami az ablak megjelenítéséért felel és egy `public static void main(String[] args)`, ami az alkalmazás indításáért felel. A `start(Stage stage)`, metódusban példányosítunk egy `MainLayoutView mainLayoutView` nézetet és a hozzátartozó `MainLayoutController mainLayoutController`, ami a nézet beállítását végzi. Ezen kívül alkalmazunk még az ablak testre szabásához egyéb metódusokat is.

com.examle.wpa_alpha.Views

A com.examle.wpa_alpha.Views csomag tartalmazza az alkalmazás keretéhez használt nézetet.
Tartalma egy `MainLayoutView.java` osztály.

MainLayoutView.java

A MainLayoutView.java osztály tartalmaz egy `public MainLayoutView()` paraméter nélküli konstruktort, mely az alkalmazás ablakának létrehozásáért felel. Emellett definiáltunk egy `public BorderPane createBorderPane()`, ami az alkalmazás keretét építi fel. Az ablakot három fő részre osztja. Az ablak tetején megjelenő topPane-ra, ami a menü sávot tartalmazza. A közepén megjelenő centerPane-re, ami az egyes menüpontokhoz tartozó nézeteket jeleníti meg. Az ablak alján lévő bottomPane-re, ami a készítők nevét és egy copyright-ot tartalmaz. Ezen kívül az osztály tartalmaz egy `public MenuBar createMenuBar()`, ami a menüt hozza létre egy kulcs-értékpár tároló `HashMap<String,HashMap<String,String>>` és `MenuBlueprint menuBlueprint` generáló segédosztály segítségével. A `MenuBlueprint` osztálynak van egy `public MenuBlueprint(HashMap<String,HashMap<String,String>> Menu)` paraméteres konstruktora, ami paraméterként átveszi a kulcs-értékpár tárolós változót(megmondja, hogy az adott almenünek mi a főmenüje és az almenü azonosítóját). A `public Menu createMenus(String menu)` és a `public MenuItem createMenuItem(String text, String id)` metódusok segítségével létrehozza a főmenüket és a hozzájuk tartozó almenüpontokat.

com.examle.wpa_alpha.Controllers

A com.examle.wpa_alpha.Controllers csomag tartalmazza a kontrollert, ami a nézet és a modellek kapcsolatának megteremtéséért felel. Itt reagálunk a menüpont választás eseményre és töltsük be a kiválasztott menüponthoz tartozó nézetet. Tartalma egy `MainLayoutController.java` osztály.

MainLayoutController.java

A MainLayoutController.java osztály tartalmaz egy `public MainLayoutController (MainLayoutView view) { setView(view);}` paraméteres konstruktort, ami paraméterként átvesz egy nézetet és a `public void setView(MainLayoutView view)` metódusával beállítja a nézet megjelenítését. Az alkalmazás menüpontjaihoz hozzárendel egy `setOnAction (actionEvent -> {})` eseménykezelőt, ami által meghatározható, hogy melyik menüpont melyik nézetét kell megjeleníteni.

com.examle.wpa_alpha.Modells

A com.examle.wpa_alpha.Modells csomag tartalmazza a menüpontokhoz tartozó nézeteit, vezérlőelemeit és az ezekhez tartozó üzleti logikát, az adatokat.

FakeRestReadModell.java

A FakeRestReadModell.java osztály tartalmaz egy `public FakeRestReadModell(String title, String taskDescription)` konstruktort, ami átveszi a nézethez tartozó címet és a feladat leírását. A `public void createFilterBox()` létrehoz egy beviteli felületet, ami egy azonosító szám megadásával a `FakeRestClient.java` osztály segítségével lekérdezhetünk adatokat a https://gorest.co.in rest szerveréről. És a JSON-ben kapott választ megjelenítjük az alkalmazás ablakában. Az azonosító szám ellenőrzéséhez `public boolean isNumeric(String value)` készített számtípust ellenőrző metódust alkalmaztunk. Nem létező erőforrás elérését is kezdeményezhetjük, de akkor hibaüzenetet kapunk!

FakeRestWriteModell.java

A FakeRestWriteModell.java osztály tartalmaz egy `public FakeRestWriteModell(String title, String taskDescription)` konstruktort, ami átveszi a nézethez tartozó címet és a feladat leírását. A `public void createFilterBox()` létrehoz egy beviteli felületet. A beviteli mezőben kötelező megadni, a felhasználó nevét, nemét, emailcímét, státuszát (elérhető/nem elérhető). Minden egy es beviteli mezőnél ellenőriztük, hogy a bemenet nem lehet nulla és a várt értékeknek megfelelő típust adtunk-e meg. Az email cím ellenőrzéséhez külön reguláris kifejezést, `Pattern pattern = Pattern.compile("^[A-Za-z0-9+_.-]+@(.+)$")` adtunk meg. A művelet végrehajtásáról tájékoztatjuk a felhasználót egy üzenettel.

FakeRestModifyModell.java

A FakeRestModifyModell.java osztály tartalmaz egy `public FakeRestModifyModell (String title, String taskDescription)` konstruktort, ami átveszi a nézethez tartozó címet és a feladat leírását. A `public void createFilterBox()` létrehoz egy beviteli felületet. A beviteli mezőben kötelező megadni a felhasználó azonosítóját, amit szerkeszteni szeretnénk, a felhasználó nevét, nemét, emailcímét, státuszát (elérhető/nem elérhető). Minden egy es beviteli mezőnél ellenőriztük, hogy a bemenet nem lehet nulla és a várt értékeknek megfelelő típust adtunk-e meg. Az email cím ellenőrzéséhez külön reguláris kifejezést, `Pattern pattern = Pattern.compile("^[A-Za-z0-9+_.-]+@(.+)$")` adtunk meg. A művelet végrehajtásáról tájékoztatjuk a felhasználót egy üzenettel. Továbbá megjelenítjük, az régi és az új állapotot is.

FakeRestDeleteModell.java

A FakeRestDeleteModell.java osztály tartalmaz egy `public FakeRestDeleteModell (String title, String taskDescription)` konstruktort, ami átveszi a nézethez tartozó címet és a feladat leírását. A `public void createFilterBox()` létrehoz egy beviteli felületet, ami egy azonosító szám megadásával a `FakeRestClient.java` osztály segítségével elvégzi a törlés. A felhasználónak megjelenítjük a törölt elem adatait is. Az azonosító szám ellenőrzéséhez `public boolean isNumeric(String value)` készített számtípust ellenőrző metódust alkalmaztunk. Nem létező erőforrás azonosítóját is megadhatjuk, de akkor hibaüzenetet kapunk!

ParalellModell.java

A ParalellModell.java osztály tartalmaz egy `public ParalellModell(String title, String taskDescription)` konstruktort, ami átveszi a nézethez tartozó címet és a feladat leírását. A konstruktor továbbá a felület létrehozását is elvégzi. A `public void make()` metódus hozza létre a szálakat a párhuzamos programozás végrehajtásához. A metódusban lévő `public void run()` függvényben kezeljük le a létrehozás során keletkező hibákat és állítjuk be a szálhoz tartozó értékeket pl: késleltetés és a felületen megjelenő változásokat leíró `Platform.runLater(new Runnable() { firstThreadLabel.setText(thread)}` metódust.

StreamReadModell.java

A StreamReadModell.java osztály tartamaz egy `public StreamReadModell(String title, String taskDescription, Database database)` konstruktort, ami átveszi a nézethez tartozó címet,a feladat leírását és az adatbázist reprezentáló osztályt. A `public void createFilterBox()` létrehoz egy beviteli felületet. A beviteli mezők adatainak megadása opcionális. Lehetőség van a települések nevére, a torony azonosítójára, a torony teljesítményére és a megye nevére. A `public List<Torony> collectData()` metódus a felel a mezőkben megadott adatok alapján a szűrések elvégzésére. A listákra alkalmazott `stream().filter()` metódussal használatával végeztük a szűrést, lambdakifejezések megadásával. A `public TableView createTableView()` metódus segítségével megjelenítjük a szelektált adatokat táblázatos formában.

MLDecisionTreeModell.java

Az MLDecisionTreeModell.java osztály tartamaz egy `public MLDecisionTreeModell(String title, String taskDescription)` konstruktort, ami átveszi a nézethez tartozó címet és a feladat leírását. Emellett felépíti a megfelelő nézetet. A felületen egy gomb lenyomásával legenerálódik a Döntési fa.txt, amit a C meghajtón belül a MachineLearning mappában hoz létre. A tanításhoz az erre a célra készített `MachineLearningClass machineLearningClass` osztályt használtuk. A tanítás folyamatának ismertetését a MachineLearning csomag résznél részletezzük. Ha a tanítás sikeres volt és a fájlt is sikerült létrehozni, akkor egy Sikeres tanítás üzenetet jelenítünk meg a felhasználónak. Ha a tanítás sikertelen, akkor egy hibaüzenetet jelenítünk meg.

MLMoreAlgorithmModell.java

Az MLMoreAlgorithmModell.java osztály tartamaz egy `public MLMoreAlgorithmModell(String title, String taskDescription)` konstruktort, ami átveszi a nézethez tartozó címet és a feladat leírását. Emellett felépíti a megfelelő nézetet. A felületen egy gomb lenyomásával legenerálódik az összes tanult gépi tanulási algoritmus egy Gépitanulás.txt fájlban. A tanításhoz az erre a célra készített `MachineLearningClass machineLearningClass` osztályt használtuk. A fájlt a C meghajtón belül a MachineLearning mappában hozza létre. A felületen továbbá megjelenik a legjobb algoritmus és a hozzá tartozó Correctly Classified Instance értéke.

MLMoreAlgorithmListModell.java

Az MLMoreAlgorithmListModell.java osztály tartamaz egy `public MLMoreAlgorithmListModell(String title, String taskDescription)` konstruktort, ami átveszi a nézethez tartozó címet és a feladat leírását. Emellett felépíti a megfelelő nézetet. A felületen egy lenyíló listából választhatjuk ki, hogy melyik algoritmust szeretnénk használni. A kiválasztás után az algoritmus futtatása gombra kattintva kiváltunk egy esemény, ami a listából választott elem alapján. Azt, hogy melyik algoritmus hajtódjon végre egy switch-case utasítással döntjük el. A tanításhoz az erre a célra készített `MoreMachineLearningClass moreMachineLearningClass` osztályt használtuk. Az algoritmus futtatása és a tanítás befejezése után, a felhasználói felületen jelenik meg a megoldás.

com.example.wpa_alpha.MachineLearning

A com.example.wpa_alpha.MachineLearning csomag tartalmazza a gépi tanuláshoz szükséges osztályokat. Ezeket az alkalmazás gépi tanulás menüpontjához tartozó nézeteknél használtuk.

MachineLearningClass.java

A MachineLearningClass.java tartalmaz egy `public MachineLearningClass(String fileName, int classIndex)` konstruktort, ami paraméterként átveszi a tanításhoz használt adathalmazt (.arff fájl) és azt az indexet, ami alapján a tanítást el kell végeznünk. Egy `try-catch` blokkban egy `BufferedReader bufferedReader = new BufferedReader(new FileReader(fileName))` példányt hozunk létre amiben tároljuk az adatokat, majd létrehozzuk az adathalmaz alapján a vizsgálandó példányokat `Instances instances = new Instances(bufferedReader)`. Ezután kiválasztjuk a tanítóhalmazt, kiértékelő halmazt és `J48 classifier` nevű osztályozót, ami döntési fa módszerét fogja alkalmazni a tanítás során. Az eredmények eltárolásához létrehoztunk egy függvényt ami ezt megvalósítja `void writeToFile(String input)`. Paraméteként átvesz egy fájlnevet és egy adott mappában eltárolja azt. Ha a mappa létezik, akkor nem hozza újra létre.

MoreMachineLearningClass

A MoreMachineLearningClass.java tartalmaz egy public MoreMachineLearningClass(String fileName, int classIndex, Classifier classifier), ami paraméterként átveszi a tanításhoz használt adathalmazt (.arff fájl), az indexet, ami alapján a tanítást el kell végeznünk és végül egy osztályozási módszert. Egy `try-catch` blokkban egy `BufferedReader bufferedReader = new BufferedReader(new FileReader(fileName))` példányt hozunk létre amiben tároljuk az adatokat, majd létrehozzuk az adathalmaz alapján a vizsgálandó példányokat `Instances instances = new Instances(bufferedReader)`. Ezt minden egyes algoritmusra lefuttatjuk. Az eredmények eltárolásához létrehoztunk egy függvényt, ami ezt megvalósítja `void writeToFile(String input)`. Paraméteként átvesz egy fájlnevet és egy adott mappában eltárolja azt. Ha a mappa létezik, akkor nem hozza újra létre. A legjobb algoritmus tárolásához az osztályban létrehoztunk osztályszintű privát változókat, `private static String bestAlgorithm` és `private static Double precision = 0.0.` Emelett az osztálynak továbbá feladata a kiválasztott algoritmus megjelenítése is `public static String showChoosedAlgorithm(String fileName, int classIndex, Classifier classifier).`

com.example.wpa_alpha.Modells.Stream

A com.example.wpa_alpha.Modells.Stream tartalmazza az adatbázis táblázatait reprezentáló a szöveges fájlok alapján elkészített modelleket. Ezek alapján végeztük el a szűréséket és tároltuk el benne az adatokat.

com.example.wpa_alpha.DataAccessObjects

A com.example.wpa_alpha.DataAccessObjects csomag tartalmazza az adatbáziskapcsolatot megvalósító `DatabaseConn.java` osztályt és az adatbázishoz tartozó műveleteket leíró osztályokat. Az adatbázis kapcsolat létrehozása interface-en keresztül történik `class DatabaseConn<T>`. Ezért a későbbi fejlesztések során, ha mögöttes adatbázist le akarjuk cserélni a jelenlegi Msql-ről, akkor ez könnyen megtehető a laza kapcsolat miatt. A programszintű adatbázis műveleteket végrehajtó osztályok: `HelyszinDAO.java`, `MegyeDAO.java`, `ToronyDAO.java`.

com.example.wpa_alpha.RestClient

A com.example.wpa_alpha.RestClient csomag tartalmazza a Kliensoldalhoz tartozó metódusokat.

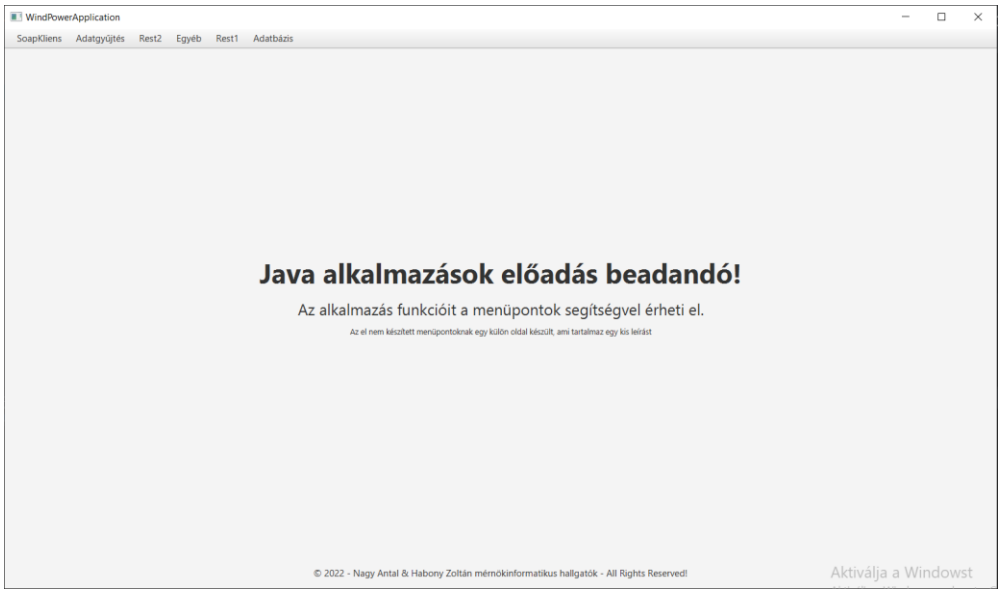
FakeRestClient.java

A FakeRestClient tartalmazza a REST kéréseket megvalósító `GET`, `POST`, `PUT`, `DELETE` műveleteket. Emellett megvalósítja a REST szerverrel való kapcsolatot. A kapcsolathoz szükség van egy API-kulcsra (token) is, amit egy osztályszintű tagváltozóban tárolunk. Ezt minden egyes kérésnél el kell küldeni a szervernek, hogy legyen jogosultságunk az adatok hozzáférésére.

Felhasználói dokumentáció

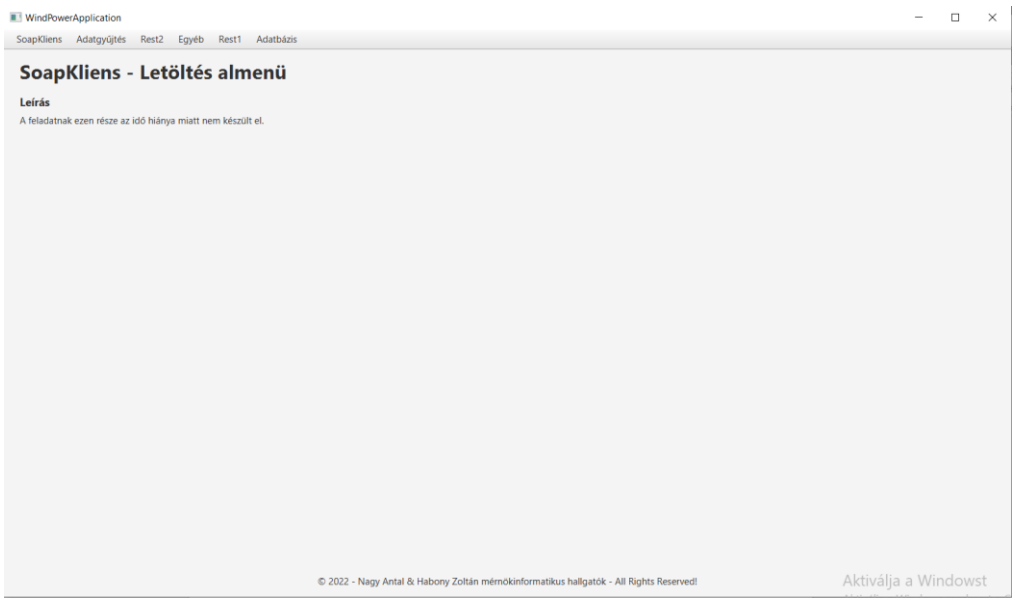
Főoldal

Az alkalmazás neve WindPowerApplication. Az alkalmazás megnyitása után egy fő ablak fogja fogadni. Erről az oldalról, a menüpontok segítségével válthat át egy másikra.



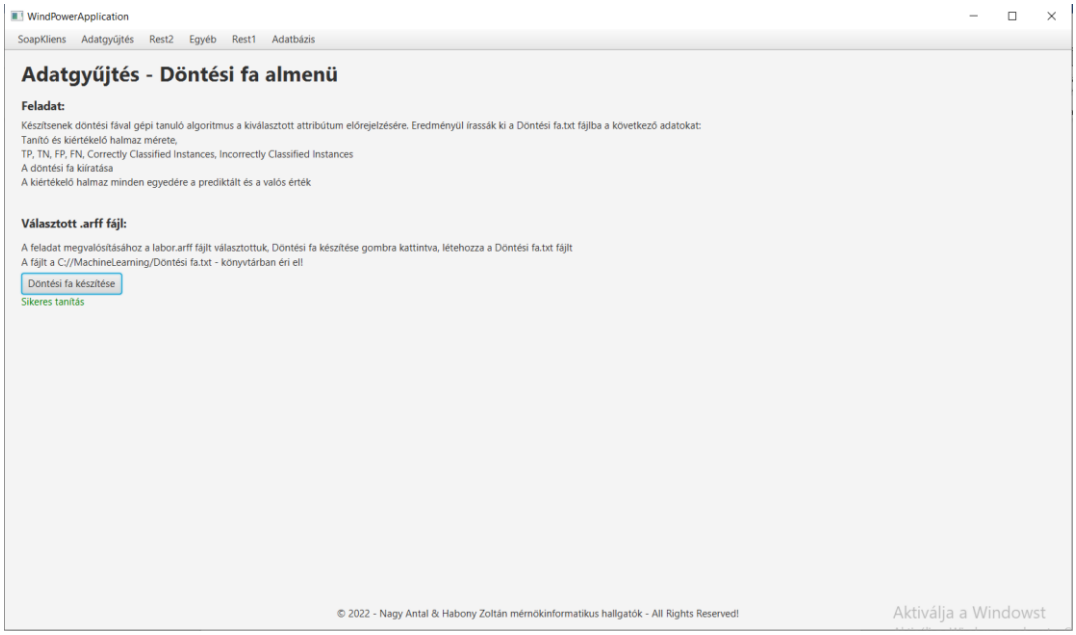
El nem készített oldalak

Az el nem készített oldalaknak is készítettünk egy alapértelmezett megjelenítést, hogy szemléltetni tudjuk, a menüpontoknak a működését.



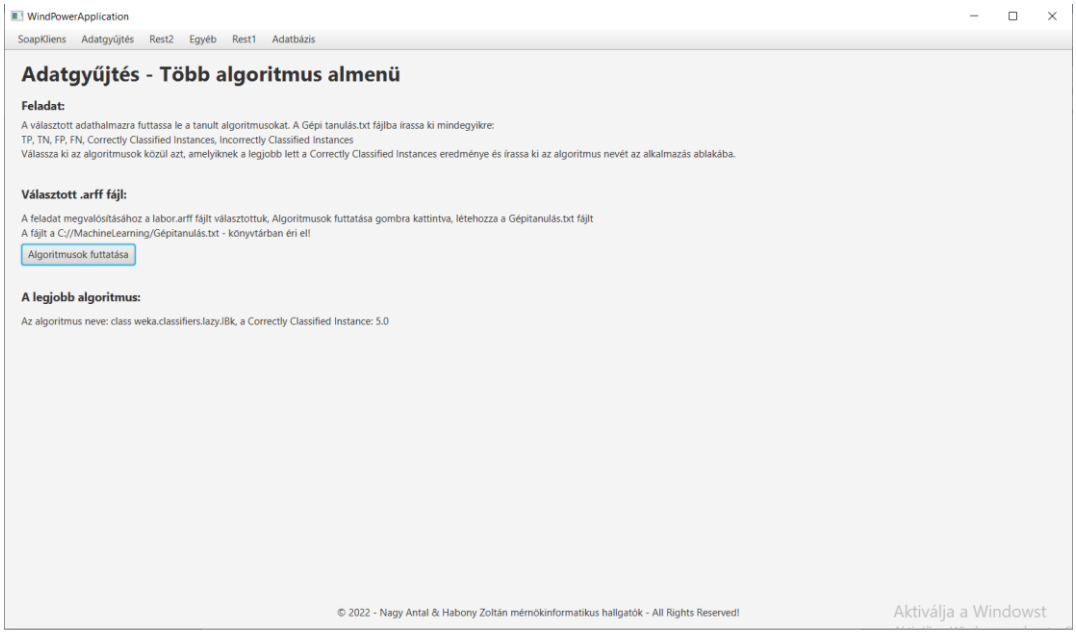
Adatgyűjtés – Döntési fa almenü

Ebben a menüpontban a gom kattintása után a C meghatón létrejön egy MachineLearning mappa, amiben elérheti az alkalmazás által készített gépi tanulás alapján a Döntési fa.txt fájlt.



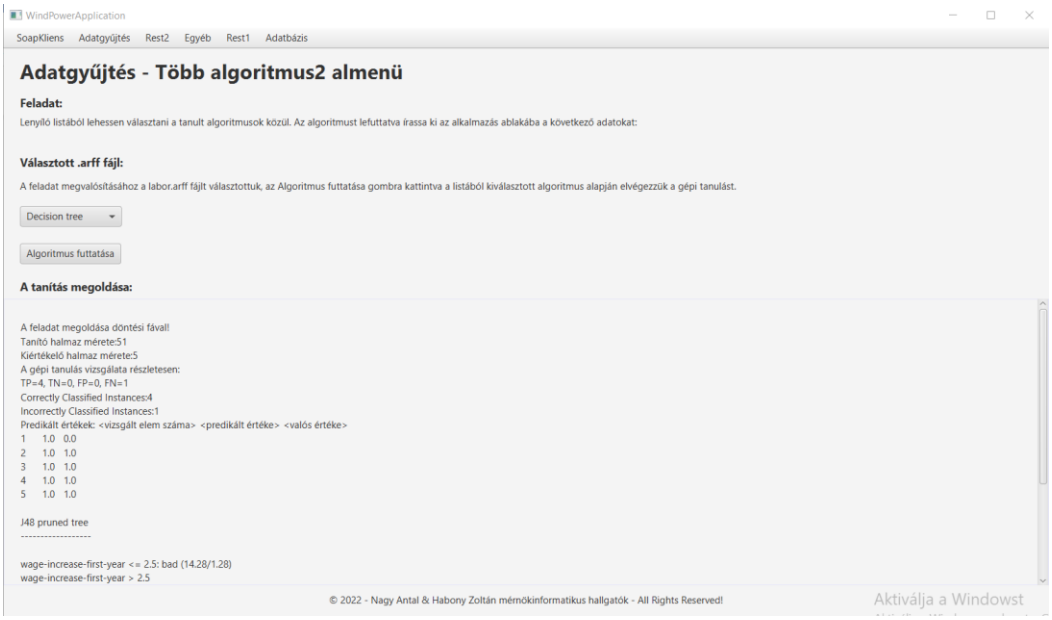
Adatgyűjtés – Több algoritmus almenü

Ebben a menüpontban az órán tanult összes gépi tanulási módszert lefuttatjuk, az eredményeket kiíratjuk egy Gépitanulás.txt fájlba, ami a C meghajtó MachineLearning mappájában található. A tanítás után kiírja a legjobb módszert.



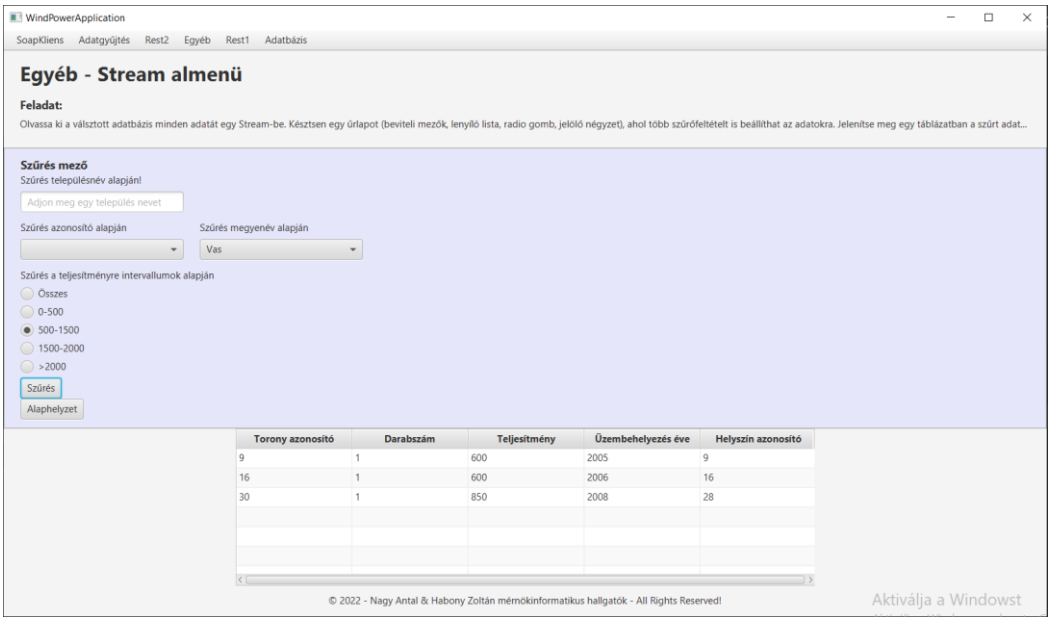
Adatgyűjtés – Több algoritmus2 almenü

Ebben a menüpontban az órán tanult összes gépi tanulási módszer közül lehet választani egyet a lenyiló listából, majd az eredményeket kiíratjuk az alkalmazás ablakába.



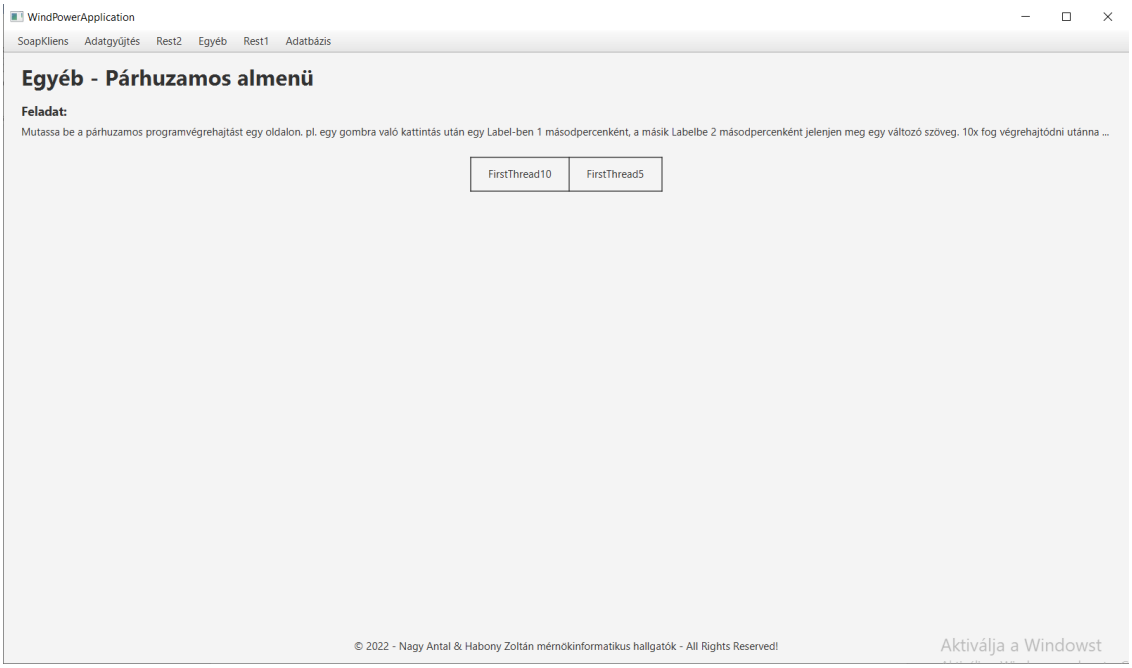
Egyéb – Stream almenü

Ebben a menüpontban a Strem-ekhez tartozó műveleteket tesztelhetjük. Lehetőségünk van szűrni bizonyos tulajdonságokra és kilistázni a tornyokat.



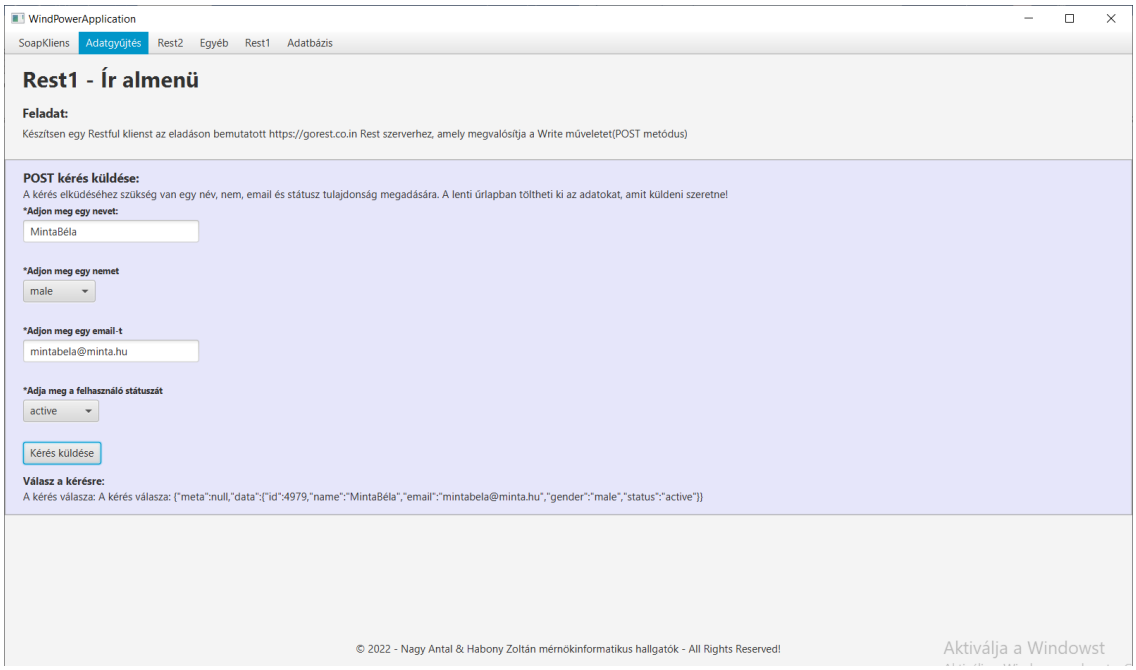
Egyéb – Párhuzamos almenü

Ebben a menüpontban lehetőségünk van tesztelni a párhuzamos végrehajtást a grafikus alkalmazásokban. A baloldali mezőben 1mp-ként jelenik meg egy szöveg, a jobboldaliban pedig 2mp-ként. Ebben az esetben 10 futás után és a feladatok elvégzésével törlődnek a felhasznált szálak.



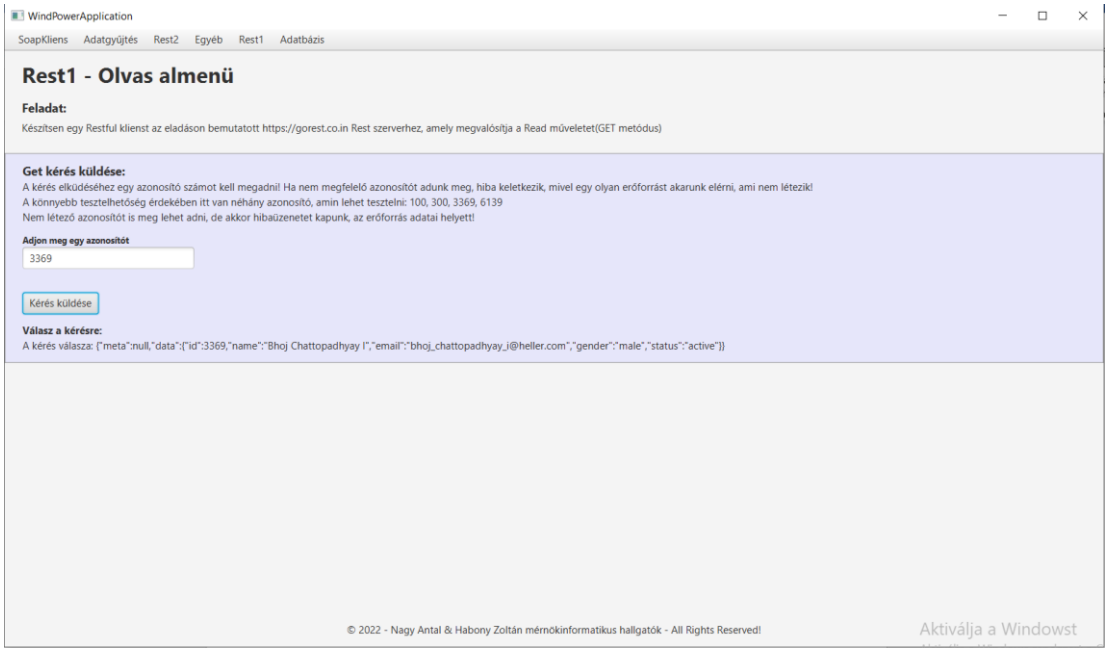
Rest1 – Ír almenü

Ebben a menüpontban tesztelhetjük a REST kliensünk POST metódusát. Az adatok kitöltése kötelező és ellenőrizzük a hitellességét a kérés elküldése előtt. A kérés válaszáról értesítjük a felhasználót.



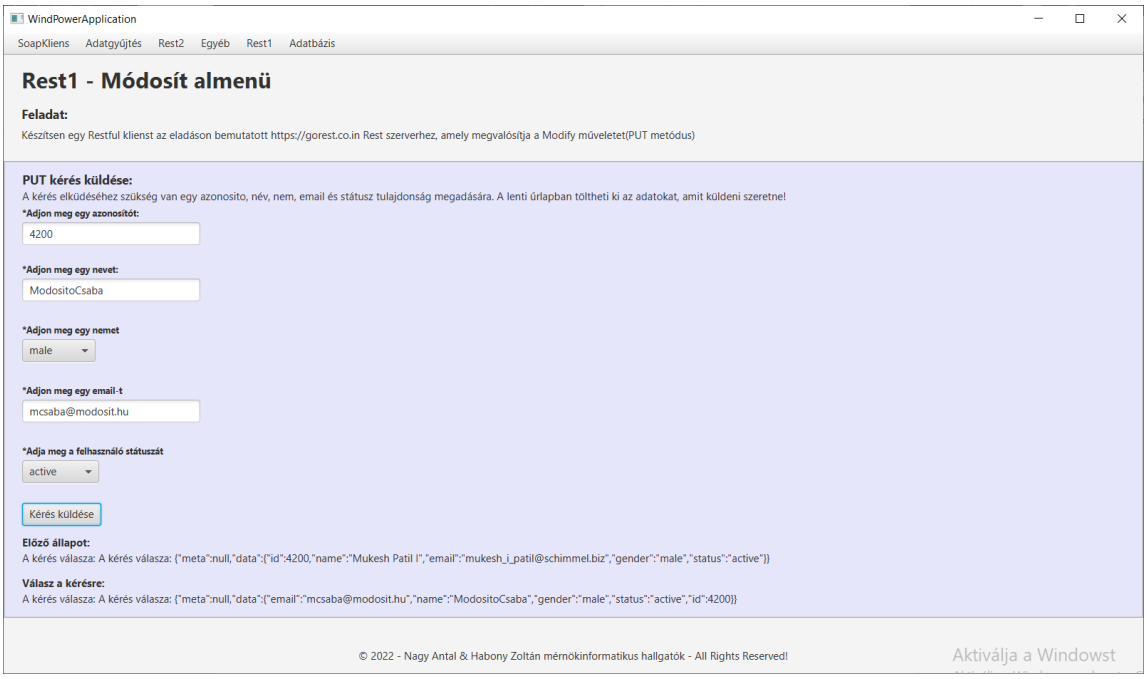
Rest1 – Olvas almenü

Ebben a menüpontban tesztelhetjük a REST kliensünk GET metódusát. Egy azonosító megadásával kérhetünk el egy erőforrásról adatot. Hibás azonosító megadását követően tájékoztatjuk a felhasználót a helyes kitöltésről. Az alábbi esetben egy helyes kitöltés, és egy kérés válasza található.



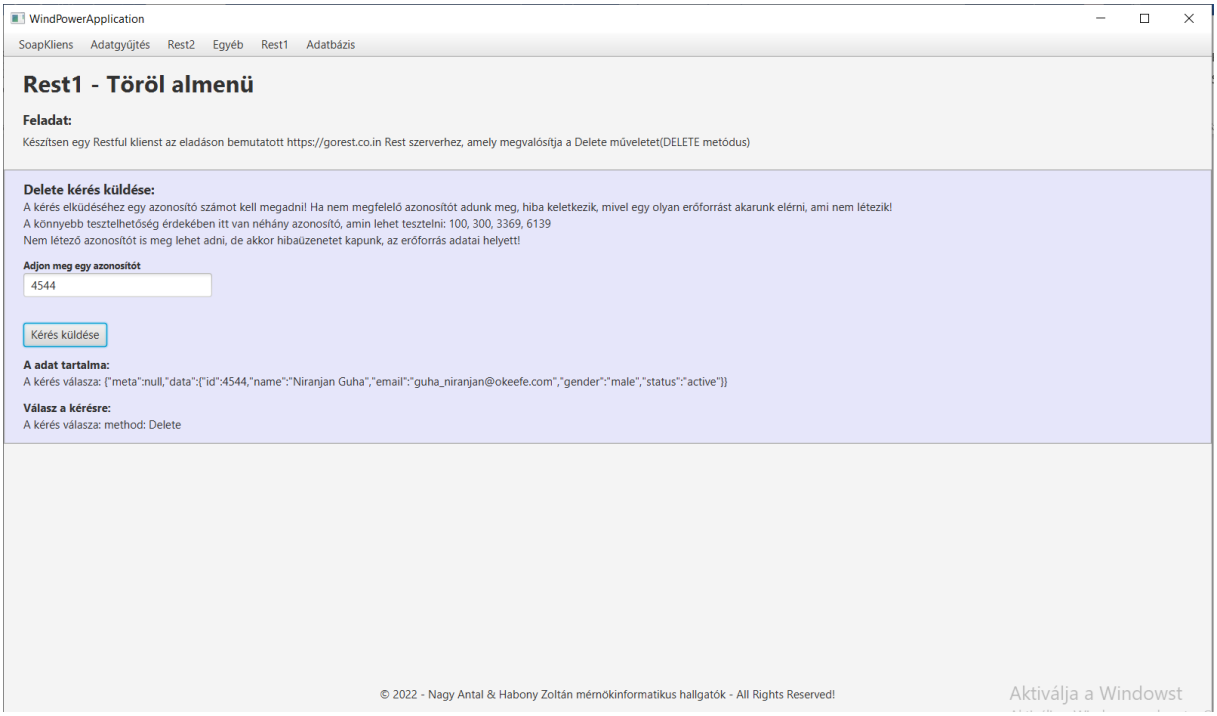
Rest1 – Módosít almenü

Ebben a menüpontban tesztelhetjük a REST kliensünk PUT metódusát. A megadott azonosítóhoz tartozó adatokat vagyunk képesek módosítani. A módosítás előtti és utáni értékeit is láthatjuk.



Rest1 – Töröl almenü

Ebben a menüpontban tesztelhetjük a REST kliensünk DELETE metódusát. Egy azonosító megadása után elvégezhető a törlés. A törlés után kiíratjuk a törölt elem adatait.



Github

Készítők

NagyAntal (C44IK7) - NagyAntal-dev

Habony Zoltán (RYEEHN) – zoltanhabony

Elérés

https://github.com/NagyAntal-dev/szeleromu_java