Népesedési világnap – Java EE esettanulmány

World Population Day – Java EE case study

KACZUR Sándor[1], KALÓ Péter[2]

[1]Eötvös Loránd Tudományegyetem Informatikai Doktori Iskola,
1117 Budapest, Pázmány Péter sétány 1/C, kaczursandor@gmail.com
[2]SZÁMALK-Szalézi Technikum és Szakgimnázium,
1119 Budapest, Mérnök utca 39., kalopeter8@gmail.com

Abstract

In 1987 the United Nations declared July 11 to be World Population Day. The population of our planet reached 5 billion that day. On the worldometer.info website, regularly updated reports are available to the population globally, by region, by country. We start from a Java SE project and develop it into a Java EE project. In the process, we use an object-oriented approach, MVC architectural design pattern. The completed case study is a distributed Java project that displays population in specified regions from 1950 to the current year based on data collected from the web.

The article describes the specification, the development/planning steps, the implementation and presents the result.

Keywords: programming, software development, Java SE, Java EE, further development, OOP, MVC

Kivonat

Az ENSZ 1987-ben július 11-ét a népesedési világnappá nyilvánította. Bolygónk lakossága aznap érte el az 5 milliárdot. A worldometer.info weboldalon folyamatosan frissülő kimutatások érhetők el a népességhez globálisan, régiónként, országonként. Kiindulunk egy Java SE projektből és továbbfejlesztjük Java EE projektté. A folyamat során objektumorientált szemlélettel, MVC architekturális tervezési mintát használunk. Az elkészült esettanulmány egy hálózaton keresztül működő Java projekt, amely webről összegyűjtött adatok alapján jeleníti meg a megadott régiókban a népességszámot 1950-től, egészen az aktuális évig. A cikk ismerteti a specifikációt, a továbbfejlesztés/tervezés lépéseit, a megvalósítást és bemutatja az eredményt.

Kulcsszavak: programozás, szoftverfejlesztés, Java SE, Java EE, továbbfejlesztés, OOP, MVC

1. Bevezetés

Az it-tanfolyam.hu [1] projekt négy fős oktatói csapata háromféle Java szoftverfejlesztő tanfolyamot szervez 2015 óta. A toborzási tevékenység inbound marketingjéhez kötődik egy szakmai blog [2] 2017 óta. A blogon havonta két tartalmas bejegyzés jelenik meg hat kategóriában: Java SE, Java EE, Java adatbázis-kezelő, IT karrier, Rendezvények, Egyéb. Az első három kategóriába a tanfolyamok tematikájához kapcsolódó szakmai tartalom, esettanulmány, újdonság tartozik a programozás, szoftverfejlesztés, objektumorientált tervezés, tervezési minta témakörökben. A blog többféle céllal működik: egyrészt márkaépítő és felkelti a figyelmet, másrészt szakmai orientációt végez és az inbound metodológiának megfelelően követőket/jelentkezőket generál, harmadrészt szakmai közösséget épít.

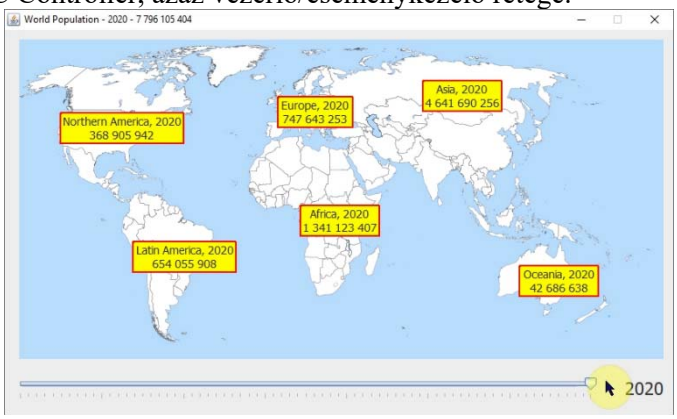
Az ENSZ 1987-ben július 11-ét a népesedési világnappá (World Population Day) nyilvánította [3]. Bolygónk lakossága aznap érte el az 5 milliárdot. További kerek számok voltak: 1999. október 12-én 6 milliárd, 2011. október 30-án 7 milliárd. További kerek számok várhatóak [4]: 2023 – 8 milliárd, 2037 – 9 milliárd, 2057 – 10 milliárd. A KSH elemzése részletes elemzéseket közöl évről-évre a

témában, például: 2019-ben [5], 2018-ban [6]. A worldometer.info weboldalon [7] folyamatosan frissülő kimutatások érhetők el a népességhez globálisan, valamint országonként is: például Magyarország aktuális népesedési adatai [8].

A blogon 2020. július 11-én jelent meg Java SE esettanulmány Népesedési világnap címmel [9] Balogh Péter és Kaczur Sándor közös munkájaként, amit az évforduló inspirált. Mindketten az it-tanfolyam.hu projekt oktatói. A szerzők megterveztek és elkészítettek egy swing GUI-s Java programot, amely megjeleníti a worldometer.info weboldalról kinyerhető adatok alapján régiónként (kontinensenként) az elérhető adatokat 1950-től 2020-ig egy világtérképen. A blog bejegyzés tartalmát, az elkészült program rövid dokumentációját, valamint a publikált forráskód-részletét tekintjük kiinduló állapotnak. Ezt továbbfejlesztette Kaczur Sándor és Kaló Péter közös munkaként. Kaló Péter 2020-ban szerzett szoftverfejlesztő OKJ végzettséget. A cikk a Java SE esettanulmányból Java EE esettanulmánnyá való továbbfejlesztésének lépéseit, folyamatát ismerteti.

2. Specifikáció

A kiinduló állapotnak tekintett Java SE projekt objektumorientált szemlélettel készült el, MVC architekturális tervezési mintát [10] követve, angol nyelvű interfész, osztály, változó, objektum, metódus nevekkel. A projekt neve: `WorldPopulation`, a csomag neve: `worldpopulation`. A program képernyőképe az 1. ábrán látható. A felület `JFrame` utód, rajta szabványos komponensekkel: `JPanel`, `JLabel`, `JSlider`. Ez az MVC View, azaz nézet/megjelenítési rétege. A régiónként rendelkezésre álló népesedési adatok interfészbe „beégetett” konstans adatszerkezetként állnak rendelkezésre. Ez az MVC Model, azaz modell/adattároló rétege. Az eseménykezelés a „csúszka” beállításából áll, amely frissíti az aktuális évszám alapján a modelltől elkért adatokkal frissíti a felületen a régiónként megjelenő adatokat és a program címsorát, valamint egérrel áthelyezhetőek a régiók adatait tartalmazó téglalapok. Ez az MVC Controller, azaz vezérlő/eseménykezelő rétege.



1. ábra: *A kiinduló állapotú Java SE projekt képernyőképe*

Az MVC szeparáció nem teljes, de egységes szemlélettel készült el az objektumok lokális felelősségi körét szem előtt tartva. A Java SE-ből EE irányba való továbbfejlesztési szempontokat az MVC rétegekhez kötődően külön-külön gyűjtöttük össze.

A modell réteghez kötődő továbbfejlesztés: le kell cserélni a konstans adatszerkezeteket!

- Első lépésben valósuljon meg a Java EE hálózati kommunikáció és a program az összes adatot a weboldalról gyűjtse össze! Mindezt mindig induláskor, automatikusan. Ehhez tárolni átmeneti adatszerkezetekben elegendő a memóriában az adatokat. Példa: 6 régió 1950-2020-ig tárolt 6*71 db adata. Ehhez jól használható az `org.jsoup` csomag [11], amely támogatja weboldal HTML szöveges tartalmának indexes scrape/crawler-szerű feldolgozását/parszolását.
- Második lépésben csökkenthetjük az adatforgalmat azzal, hogy egy adott évfordulóig/dátumig helyi állományrendszerben tárolt adatfájlban tároljuk az adatokat (például az előző év végi vagy a kiadáskori adatokat) és csupán a hiányzó vagy frissítendő adatokat töltjük le a weboldalról és ezekkel dinamikusan kiegészítjük a fájlból betöltött adatokat a memóriában. Ehhez tudni kell, hogy a weboldalon az adatok az aktuális évben az aktuális évre vonatkozóan folyamatosan frissülnek. Amíg ezt a mondatot valaki elolvassa, addig Ázsiában születik legalább 3-4 ember. A

múltbeli évekhez kötődő historikus adatok nem változnak. Példa: 6 régió 1950-2019-ig fájlban tárolt 6*70 db adatához hozzáfűzzük a hiányzó 2020-as évi 6 db adatot.
- Harmadik lépésben már a hiányzó adatokkal a helyi adatfájl is bővíthető (inkrementálisan), amivel még tovább redukálható a webes adatforgalom.
- Meg kell tervezni a helyi adatfájl formátumát. Mivel a webről szöveges adat érkezik, célszerű ezt megfelelő karakterkódolással szintén szöveges formátumban tárolni. Egyetlen adatfájl elegendő, mert az adatok mennyisége nem indokolja a régiónkénti szeparációt külön adatfájlokba. Európai példa adat: `"{c:[{v:new Date(2020,6,1)},{v:747636026}]}"`. Saját egyéni fájlformátumot specifikálni nem érdemes. Elegendő a webről érkező tartalmat – kiválogatást/kihagyást követően – közvetlenül áttölteni helyi adatfájlba.

A nézet réteghez kötődő továbbfejlesztés:
- A régiók adatait megjelenítő komponensek vonszolásának alapja az általuk elfoglalt téglalap alakú terület közepén alapul, emiatt igen „darabosan” mozog. Finomítsuk pixel pontossá a drag & drop technikával való vonszolásukat, bárhol fogjuk is meg ezeket az egérrel!
- Fontos, hogy a régiók adatait megjelenítő komponensek ne legyenek egérrel kivonszolhatók a térkép területéről.
- Szükséges egy univerzális hibaüzenetet megjeleníteni képes felbukkanó párbeszédpanel.

A vezérlő réteghez kötődő továbbfejlesztés:
- A fájlkezeléshez és a hálózati adatforgalomhoz kötődő problémákra fel kell készülni. Például sikertelen lehet a világtérkép betöltése helyi képfájlból (1), az olvasás a helyi adatfájlból (2), a hálózati kapcsolat felépítése (3), a hálózatról érkező adatok feldolgozása (4), az írás a helyi adatfájlba (5). Kétfelé kell csoportosítani a felmerülő problémákat. Egyrészt kritikus esetben előfordulhat, hogy a GUI-t sem tudjuk összeépíteni (1 és/vagy 2). Ilyenkor az önálló – hibaüzenetet tartalmazó – párbeszédablak alkotja a felületet. Másrészt „nem világvége” esetben (3 és 4, vagy külön 5) esetleg hiányzik az utolsó év, de a korábbi 70 év adatai böngészhetők. Ilyenkor a főablak megjelenik, és fókuszvesztést követően felbukkan a nem önálló – hibaüzenetet tartalmazó – párbeszédablak. Át kell gondolni azt is, hogy a hibaüzenet után kilépés történjen, vagy valamilyen ésszerű kompromisszumként a program futhat még tovább.

Globális szempontok, több MVC réteghez kötődően:
- Egységes szemléletű kivételkezelés szükséges. Minden hiba „legyen kivezetve a GUI-ra”, azaz értesüljön róla a felhasználó. Tehát bárhol és bármi miatt keletkezik kivétel – többnyire a modell rétegben –, arról tudjon a nézet réteg is.
- A Java SE témakörbe tartozik egy eseményvezérelt, grafikus felhasználói felülettel rendelkező swing-es program, amely OO szemlélettel, MVC architekturális tervezési mintával valósul meg és helyi fájlrendszerben kép- és adatfájlokat képes használni kivételkezeléssel. A Java EE témakörbe tartozik a hálózati kapcsolat megvalósítása, weboldalak feldolgozása, azokból adatok kinyerése, mindez szintén kivételkezeléssel. A továbbfejlesztés során is betartandók az OO és MVC elvek.

3. Tervezés

A tervezést interfészekre és osztályokra bontva külön végezzük el, tekintettel az OO és MVC elvekre. A cikk nem tartalmazza az elkészült UML osztálydiagramot, a terjedelmi korlát miatt.

3.1. Interfészek

Amit lehet, konstansként interfészbe (szeparálva) teszünk és ezeket az MVC rétegekhez kötődő osztályok implementálják. Az interfészek között öröklődési kapcsolat is megjelenik. Az ősinterfész a `WorldPopulationConstants`, benne az évszám intervallum `MIN_YEAR` és `MAX_YEAR` határaival, valamint a megjeleníthető régiók neveivel tömbben: `REGION_NAME_ARRAY`. Két utódinterfész épül az ősre: `ModelConstants` és `ViewConstants` (2. ábra).

Előbbi interfész az adatforráshoz kapcsolódik: `URL_COMMON` az URL eleje, `URL_ARRAY` az URL végei régiónként tömbben. Európai példa URL: `"https://www.worldometers.info/world`

-population/europe-population". Ide került még a DATAFILE és TEMPFILE, amelyek a helyi fájlok útvonalait tárolják.

Utóbbi interfész a megjelenítéshez kapcsolódik: WORLD_MAP_IMAGE_FILE a háttérkép annak WORLD_MAP_RECT méretével együtt, valamint a régiónkénti REGION_RECT_ARRAY téglalapok tömbje a kezdeti pozíciókkal/méretekkel, TITLE a sablon a program címsorához (frissítendő az évszámmal és az összesített népességgel). A megfelelő utódinterfészt mindig implementálja az MVC szerint hozzá illeszkedő osztály.

3.2. Osztályok

A megtervezett osztályok között is előfordul öröklési kapcsolat.

A belépési pont a WorldPopulation.java fájlban található (3. ábra).

Három összetartozó elemi adatot fog össze a RegionData POJO, ezek name, year, population nevű, rendre String, int, long típusú adatok. Például: "Európa, 2020, 747636026". Tartalmaz három függvényt: getName(), getPopulation(), valamint toString(). Utóbbi HTML formátumban, két sorba tördelve, a népességet ezres szeparátorokkal adja vissza a megjelenítendő adatokat. Például: "<html> Europe, 2020
747 636 026 </html>".

A webről adatokat szerez és tárolja a Model osztály, a java.io, java.nio, java.net [12, 13] és az org.jsoup csomagokra építve (4. ábra). A program kliensként hat régióra vonatkozó adatot gyűjt össze, alkalmazkodva a szerver adatforráshoz. A címsorban lévő összesített adat is elérhető közvetlenül a weboldalon, de a kisebb adatforgalom érdekében long típusú globalPopulation nevű példányváltozóban hasznos inkább a kliensben összesíteni. A Model osztály metódusai a következők:

- readDataFromWeb(): a továbbfejlesztés 1. lépését valósítja meg,
- readDataFromFile(): a továbbfejlesztés 2. lépését valósítja meg,
- processData(): a belső adatábrázolást megvalósító metódus,
- getter metódusok: getRegionName(), getRegionPopulation(), getRegionData(), getRegionList(), mindegyik generikus listát ad vissza, rendre String, Long, String és RegionData típusút/osztályút.

A JLabel-ből származik az igényekhez alakított RegionLabel osztály. Ennek van előre megadott pozíciója, mérete, betűtípusa, betűmérete, sárga háttérszíne, piros kerete. Ezenkívül a téglalap átlátszó, valamint a benne megjelenő HTML tartalom vízszintesen középre igazított. A téglalap egérrel megfogva – drag and drop – áthelyezhető, ami a MouseMotionAdapter egérmozgást figyelő absztrakt osztály MouseMotionListener figyelő interfésztől implementált mouseDragged() metódusának felülírásával válik lehetővé. A téglalap térkép területéről való kivonszolását korlátozzák a MouseAdapter absztrakt osztály MouseListener figyelőinterfésztől implementált mousePressed() és mouseReleased() metódusai. A mozgathatóságáért és az eredeti pozícióba való „visszaugrásáért” az osztályból példányosított objektum saját maga felel (4. ábra).

A grafikus felhasználói felületet adja a JFrame utód View osztály. Három GUI komponensből áll: pnWorldMap – háttérkép JPanel, lbYear – kiválasztott/aktuális év JLabel, slYear – kiválasztható/görgethető aktuális év JSlider. Izgalmas megoldani egymásra/egymáson elhelyezni a komponenseket [14]. Egy JLayeredPane komponens DEFAULT_LAYER rétegére kerül a térképet tartalmazó háttérkép, majd a PALETTE_LAYER rétegére kerül dinamikusan a hat RegionLabel osztályú/típusú objektum (6. ábra). A csúszka komponens slYearStateChanged() eseménykezelő metódusa (7. ábra) vezérlőként megszólítja a modell réteget és a visszakapott adatokkal frissíti a nézet réteget (a címsorban lévő összesítéssel együtt, ezres szeparátorokkal).

4. Megvalósítás

4.1. Interfészek

```java
package worldpopulation;

public interface WorldPopulationConstants {
  int MIN_YEAR=1950, MAX_YEAR=2020;
  String[] REGION_NAME_ARRAY={
    "Europe", "Asia", "Africa",
    "Northern America", "Latin America", "Oceania"};
}

public interface ModelConstants extends WorldPopulationConstants {
  String URL_COMMON="https://www.worldometers.info/world-population/";
  String[] URL_ARRAY={
    URL_COMMON + "europe-population/",
    URL_COMMON + "asia-population/",
    URL_COMMON + "africa-population/",
    URL_COMMON + "northern-america-population/",
    URL_COMMON + "latin-america-and-the-caribbean-population/",
    URL_COMMON + "oceania-population/"
  };
  File DATAFILE=new File("./files/DataFrom1950ToCurrentYearMinusOne.txt");
  File TEMPFILE=new File("./files/datas_20200717.txt");
}

public interface ViewConstants extends WorldPopulationConstants {
  File WORLD_MAP_IMAGE_FILE=new File("./images/world-map.png");
  Rectangle WORLD_MAP_RECT=new Rectangle(0, 0, 800, 397);
  Rectangle[] REGION_RECT_ARRAY={
    new Rectangle(320, 70, 95, 40), //Europe
    new Rectangle(500, 50, 100, 40), //Asia
    new Rectangle(310, 160, 100, 40), //Africa
    new Rectangle(50, 90, 155, 40), //Northern America
    new Rectangle(140, 250, 130, 40), //Latin America
    new Rectangle(620, 280, 100, 40) //Oceania
  };
  String TITLE="World Population - #YEAR# - #POPULATION#";
}
```

2. ábra: *A Java EE projekt interfészeinek forráskódja*

4.2. Osztályok

```java
public WorldPopulation() {
  model=new Model();
  try {
    model.readDataFromFile();
    model.processData(model.getDataArray());
    view=new View(model);
  }
  catch(IOException e1) {
    JOptionPane.showMessageDialog(null,
      "Hiba! Az adatok lekérése/feldolgozása során probléma merült fel!\n"+
      "A program 2020.07.17-es adatokkal kerül megjelenítésre!",
      "Lekérés/Feldolgozási hiba", JOptionPane.ERROR_MESSAGE);
    try {
      model.readDataFromFile(model.getDataArray());
      view=new View(model);
    }
    catch(IOException e2) {
      JOptionPane.showMessageDialog(null,
        "Hiba! Az adatok beolvasása során probléma merült fel!\n"+
        "Az alkalmazás bezárásra kerül!",
        "Olvasási hiba", JOptionPane.ERROR_MESSAGE);
    }
  }
}
```

3. ábra: *A Java EE projekt belépési pontjának kivételkezelése*

```
    addMouseMotionListener(new MouseMotionAdapter() {
31    @Override
      public void mouseDragged(MouseEvent e) {
33      regionLocation.x=getX()+e.getX()-pressedLocation.x;
34      regionLocation.y=getY()+e.getY()-pressedLocation.y;
35      setLocation(regionLocation);
36    }
37  });
    addMouseListener(new MouseAdapter() {
39    @Override
      public void mousePressed(MouseEvent e) {
41      pressedLocation=e.getPoint();
42    }
43    @Override
      public void mouseReleased(MouseEvent e) {
45      if(!parentBounds.contains(regionLocation.x+bounds.getWidth()/2,
46                                regionLocation.y+bounds.getHeight()/2))
47        setLocation(bounds.getLocation());
48    }
49  });
```

4. ábra: *Részlet a `RegionLabel` osztály forráskódjából (egéreseményekből)*

```
75  private void readDataFromWeb() throws IOException {
76    GregorianCalendar gc=new GregorianCalendar();
77    int currentYear=gc.get(Calendar.YEAR);
78    int currentYearMinusOne=currentYear-1;
      for(int i=0; i<URL_ARRAY.length; i++) {
80      Document doc=Jsoup.connect(URL_ARRAY[i]).get();
81      Elements scriptElements=doc.getElementsByTag("script");
82      String cypd="";
83      int numberOfScript=0;
84      for(Element element : scriptElements)
85        for(DataNode node : element.dataNodes()) {
86          if(numberOfScript==2)
87            cypd=node.getWholeData();
88          numberOfScript++;
89        }
90      if (isWriteDataFromWeb) {
91        String cymo=cypd.substring(
92          cypd.indexOf("rows:")+8,
93          cypd.indexOf("{c:[{v:new Date(  " + currentYearMinusOne)+47).trim();
94        String cy=cypd.substring(
95          cypd.indexOf("{c:[{v:new Date(  " + currentYear),
96          cypd.indexOf("]});")-1);
97        writedOutList.add(cymo+"\n$");
98        dataArray.add(cymo+"\n"+cy);
99      }
100     else
101       lastDates.add(cypd.substring(cypd.indexOf("{c:[{v:new Date( " +
102         currentYear), cypd.indexOf("]});") - 1));
103   }
104 }
```

5. ábra: *Részlet a `Model` osztály forráskódjából (olvasás az adatforrás weboldalról)*

```
28          JLayeredPane lpLayer=new JLayeredPane();
29          BufferedImage image=ImageIO.read(WORLD_MAP_IMAGE_FILE);
30          JLabel lbWorldMap=new JLabel(new ImageIcon(image));
31          lbWorldMap.setBounds(WORLD_MAP_RECT);
32          lpLayer.add(lbWorldMap, JLayeredPane.DEFAULT_LAYER);
33          for(int i=0; i<REGION_NAME_ARRAY.length; i++) {
34            RegionLabel rl=new RegionLabel(WORLD_MAP_RECT, REGION_RECT_ARRAY[i]);
35            regionLabelList.add(rl);
36            lpLayer.add(rl, JLayeredPane.PALETTE_LAYER);
37          }
```

6. ábra: *Részlet a `View` osztály forráskódjából (többrétegű GUI építése)*

```
146   private void slYearStateChanged(javax.swing.event.ChangeEvent evt) {
147     int year=MIN_YEAR+slYear.getValue();
148     slYear.setToolTipText(""+year);
149     lbYear.setText(""+year);
150     ArrayList<String> regionDataList=model.getRegionData(year);
151     for(int i=0; i<regionLabelList.size(); i++)
152       regionLabelList.get(i).setText(regionDataList.get(i));
153     setTitle(TITLE.replace("#YEAR#", ""+year).replace("#POPULATION#", ""+
154       String.format("%,d", model.getPopulation())));
155   }
```

7. ábra: *Részlet a `View` osztály forráskódjából (csúszka komponens eseménykezelés)*

5. További fejlesztési lehetőségek

Bár ez maga is egy továbbfejlesztési projekt, adódnak még további tartalékok, illetve lehetőségek a továbbfejlesztésére. Egyrészt cserélhető lenne a modell rétegbeli helyi fájlban megvalósuló adattárolás más formátumra (JSON, XML), illetve adatbázisra is. Előbbi kettő pusztán gyakorlási céllal, utóbbi nagyobb léptékű változtatást jelentene, bár nem feltétlenül lenne szükséges minden CRUD-beli művelet. Másrészt mindig gyenge pontot jelenthet a weboldalról történő közvetlen olvasás, adatfeldolgozás, konvertálás műveletsorozata, így beépíthető lenne többféle biztonsági kapu/lépcső is a folyamatba. Abból kiindulva, hogy a kiolvasott adataink nem kötődnek szervesen a formázott látványhoz – hiszen a weboldalon megjelenő grafikon adatforrására építünk – ez rövidtávon nem jelent veszélyt, de azért sosem árt hosszútávon gondolkodni, több mindenre számítani/felkészülni. Kisebb változtatást a weboldalon túlél a program adatfeldolgozó része, de nagyobb, strukturális jellegűt nem biztos. Harmadrészt a mozgatható saját címkekomponens esetén hasznos lehet reagálni a fókuszeseményre. Például azzal, hogy megváltozik az egérkurzor, jelezve ezzel azt, hogy „megfogott valamit az egér”.

6. Összegzés

Egy projekt továbbfejlesztése mindig kihívásokkal teli. A folyamat része alternatívák keresése, egyes részek cseréje (öröklődéssel, képességek felüldefiniálásával, MVC rétegbeli ekvivalenciával), akár több lépésben is. Lehet rangsorolni, fontossági sorrendbe állítani egyszer szempontokat, például a hatékonyság klasszikus dimenziója szerint: használjunk kevesebb memóriát, generáljunk kisebb hálózati adatforgalmat, legyen az algoritmus, a forráskód könnyebben dokumentálható, módosítható, továbbfejleszthető. Oktatási és tanfolyami környezetben is lényeges, hogy álljanak rendelkezésre evolúciós projektek. Ezekkel jól érzékeltethetőek a fenti szempontok. Mindig érdemes megmutatni, hogy a szakmai tudásunk más-más részhalmazából kiindulva, ezek egymásra építésével, a fokozatosság elvét szem előtt tartva, hogyan jutunk/juthatunk el A-ból B-be. Az is ki kell mindig hangsúlyozni, hogy többféle megoldás/megvalósítás is alkalmas/használható lehet, és az is fontos képesség, hogy ezek közül kiválasszuk a megfelelőt és indokolni is tudjuk a választásunkat. Az esettanulmányt felhasználjuk az it-tanfolyam.hu projekt Java SE szoftverfejlesztő és Java EE szoftverfejlesztő tanfolyamain [15, 16] is, bekerült az eLearning tananyagba.

Irodalmi hivatkozások

[1] it-tanfolyam.hu projekt kezdőlap, https://it-tanfolyam.hu, 2020.08.20.
[2] it-tanfolyam.hu projekt szakmai blog, https://it-tanfolyam.hu/blog, 2020.08.20.
[3] World Population Day, https://en.wikipedia.org/wiki/World_Population_Day, 2020.07.11.
[4] World Population Milestones, https://www.worldometers.info/world-population/#milestones, 2020.07.11.
[5] KSH STATISZTIKAI TÜKÖR, Népesedési világnap, 2019. július 11., https://www.ksh.hu/docs/hun/xftp/ stattukor/nepesedesi19.pdf, 2020.07.20.
[6] KSH STATISZTIKAI TÜKÖR, Népesedési világnap, 2018. július 11., https://www.ksh.hu/docs/hun/xftp/ stattukor/nepesedesi18.pdf, 2020.07.20.
[7] Current World Population, https://www.worldometers.info/world-population/, 2020.07.20.
[8] Hungary Population, https://www.worldometers.info/world-population/hungary-population/, 2020.07.20.
[9] Balogh P.: Népesedési világnap, https://it-tanfolyam.hu/nepesedesi-vilagnap, blog bejegyzés, 2020.07.11.
[10] Eckstein, R.: Java SE Application Design With MVC, https://www.oracle.com/technical-resources/articles/ javase/mvc.html, 2020.08.02.
[11] org.jsoup csomag: Java HTML Parser, https://jsoup.org/, 2020.08.02.
[12] java.nio csomag: Java NIO Package, https://www.javatpoint.com/java-nio-package, 2020.08.02.
[13] java.net csomag: Java NIO vs IO, https://www.javatpoint.com/java-nio-vs-input-output, 2020.08.02.
[14] How to Use Layered Panes (The Java Tutorials > Creating a GUI With JFC/Swing > Using Swing Components), https://docs.oracle.com/javase/tutorial/uiswing/components/layeredpane.html, 2020.08.02.
[15] Java SE szoftverfejlesztő tanfolyam, https://it-tanfolyam.hu/java-se-szoftverfejleszto-tanfolyam/, eLearning tananyag, 2020.09.01.
[16] Java EE szoftverfejlesztő tanfolyam, https://it-tanfolyam.hu/java-ee-szoftverfejleszto-tanfolyam/, eLearning tananyag, 2020.09.01.