JEGYZŐKÖNYV

Webes adatkezelő környezetek

Féléves feladat

Egy üzemanyag- és szerviz nyilvántartás adatkezelő rendszere

Készítette: Tóth Péter

Neptunkód: A24E5Z

Dátum: 2025. november

Miskolc, 2025

Tartalomjegyzék

1. Bevezetés

A modern vállalatok és szervezetek működésében kiemelten fontos szerepet játszik a járművek üzemeltetésével kapcsolatos adatok pontos, naprakész és könnyen visszakereshető nyilvántartása. Az üzemanyag-fogyasztás, a szervizelési előzmények, valamint a karbantartási költségek rendszeres követése nemcsak a gazdaságos működés alapja, hanem a flottakezelés átláthatóságát és hatékonyságát is jelentősen javítja. A webes adatkezelő rendszer lehetővé teszi az üzemanyag- és szervizadatok online rögzítését, tárolását és kezelését.

2. Feladat leírása

Feladatként egy webes üzemanyag- és szerviznyilvántartó rendszer adatkezelésének megtervezését és megvalósítását választottam. A rendszer a járművek üzemeltetésével kapcsolatos adatokat – üzemanyag-felhasználás, szervizelések, karbantartások – kezeli, amelyek jól strukturálható információhalmazt alkotnak. A feladat során a legfontosabb adatkezelési szempontokra koncentráltam.

Első lépésként elkészítettem az ER-diagramot és az XDM-modellt, amelyek alapján létrehoztam a nyilvántartás XML dokumentumát, és ehhez egy XMLSchema (XSD) állományt. A dokumentum érvényességét validátorral ellenőriztem.

A második részben a DOM API segítségével dolgoztam fel az XML adatokat: beolvastam, megjelenítettem, módosítottam, majd egy új XML fájlba mentettem. Így a feladat lefedi mind a statikus adatmodell, mind a dinamikus, programozott adatkezelés folyamatát.

3. Alapvető adatszerkezeti modellek és fileok

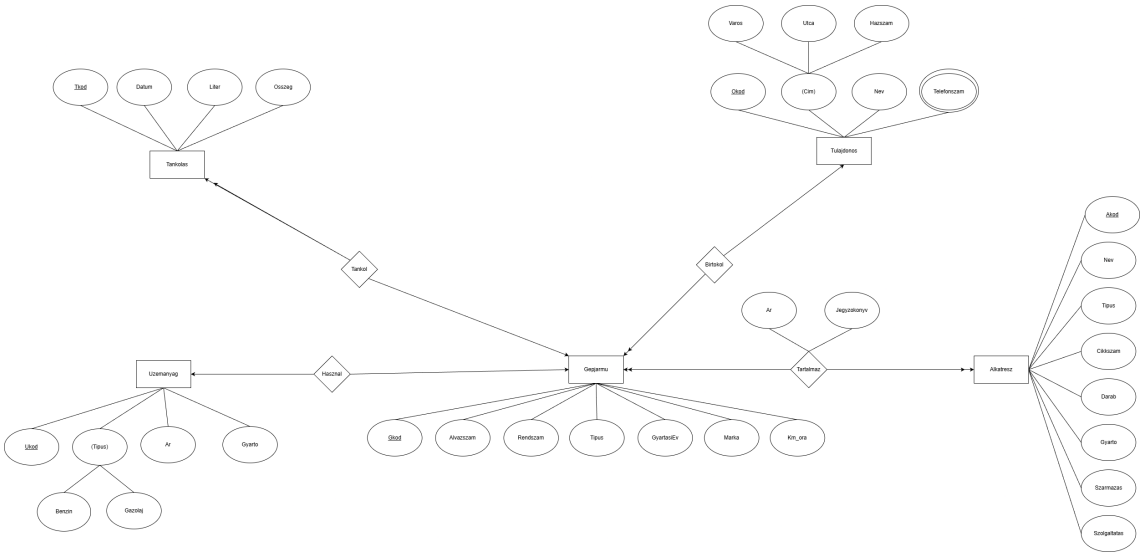
3.1. Az adatbázis ER modell tervezése

Az adatmodellben a Gépjármű (Gepjarmu) lett a központi, fő elem, amelyhez szinte minden más egyed idegen kulccsal csatlakozik, így ez képezi a nyilvántartás gerincét. A gépjárműhöz közvetlenül kapcsolódik a Tulajdonos (1:N kapcsolat, egy tulajdonosnak több autója is lehet), a Tankolás (1:N, egy autóhoz több tankolás tartozhat), az Üzemanyagtípus (1:1, egy autó egyféle üzemanyag típust használ), valamint a Tartalmaz kapcsolati egyed (N:M kapcsolat az Alkatrész és Gépjármű között, mert egy autóba sokféle alkatrész kerülhet be, és egy alkatrész-típus sok autóba beszerelhető).

A feladatban próbáltam bonyolultabb szerkezeteket is létrehozni, mint a Tulajdonos, aminek 3 tulajdonsága is egyszerre többértékű és összetett. Ennek az adatszerkezete egy

egész feladatával is felérhetne, viszont ezért maximalizáltam a tulajdonságok számát 4-ben, így még elég részletes és megfelelően kompakt a feladat teljesítéséhez.

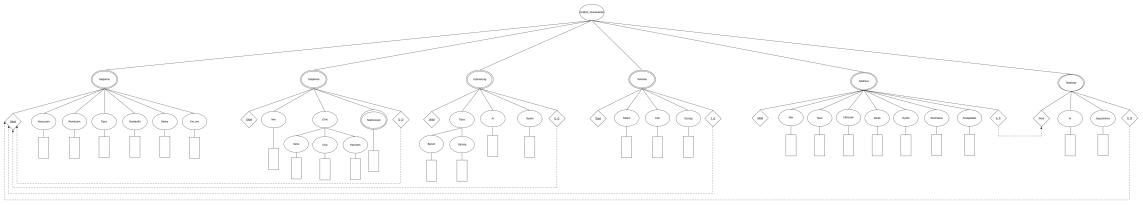
Az Alkatrész egyedet egy külön kapcsolati tábla (Tartalmaz) köti össze a gépjárművekkel, és ezen keresztül jelennek meg a kapcsolat tulajdonságai is: Ár (integer) és Jegyzőkönyv (string)



1. ábra. A nyilvántartás ER modellje

3.2. Az adatbázis konvertálása XDM modellre

Ebben a feladatban a már meglévő ER modellt kellett átalakítani egy XDM modellre. Az átalakításhoz létrehoztam egy mesterséges gyökeret "A24E5Z_Nyilvantartas" néven, és ennek a gyerek egyedei lettek a "Gepjarmu", a "Tulajdonos", a "Uzemanyag", a "Tankolas", a "Alkatresz" és a "Tartalmaz". Fontos megjegyezni, hogy itt a kapcsolatokat máshogy kell jelezni, mint az ER modellnél, mivel ott entitások között van kapcsolat az XDM modellnél pedig PK-FK kapcsolat van. Az XDM-nél minden tulajdonságból gyerek elem, a kulcs tulajdonságból pedig attribútum lesz. A modell alapvető felépítése a séma helyett hierarchia.



2. ábra. A nyilvántartás XMD modellje

3.3. Az XDM modell alapján XML dokumentum készítése

Az XML struktúrája az XDM szerkezete alapján készült. A gyökérelem "<A24E5Z_>", ez tartalmazza gyerekelemként az összes többi entitást (pl.: <Tulajdonos>, <Gepjarmu>, <Alkatresz>), ezeknek saját egyedi azonosító attribútumok van (okod, gkod, akod), amik itt már tényleges kulcs értéket kaptak. A kódban található többértékű tulajdonság amik több értékkel is rendelkeznek

Listing 1. A nyilvántartás adatszerkezet részlete

```
<Gepjarmu gkod="G001" alvazszam="ALV123456789" rendszam="ABC-123
   " O_G="O001">
    <Tipus>Személyautó</Tipus>
    <GyartasiEv>2015</GyartasiEv>
    <Marka>Opel</Marka>
    <Km_ora>120000</Km_ora>
</Gepjarmu>
```

Ebben a részletben egy adott gépjármű adatai tárolódik.

3.4. Az XML dokumentum alapján XMLSchema készítése

Az XMLSchema az XML dokumentum alapján készült. Az XSD 3 fő elemre bontható kulcsok, Idegen kulcsok és koplex típusok. A kulcsok részben minden kulcs ami megtalálható az XML fileban fel lett véve. Az idegen kulcs részben felvetettem az idegen kulcsokat és azokat össze kötöttem az elsődleges kulcsokkal. A harmadik részben pedig komplex típusokat hoztam létre. Minden fő elem kapott magának egy külön típust, valamint az összetett elemek is kaptak külön típusokat. A komplex típusok elemekből és ha rendelkeznek vele, akkor attribútumokból állnak. Az elemek a komplex típus gyerek elemit foglalják magukba, amiknek szintén van külön típusaik, akár még több komplex típus.

Listing 2. Az XSD modell részlete

```
          <xs:complexType name="GepjarmuType">
            <xs:sequence>
              <xs:element name="Tipus" type="xs:string"/>
              <xs:element name="GyartasiEv" type="xs:gYear"/>
              <xs:element name="Marka" type="xs:string"/>
              <xs:element name="Km_ora" type="xs:integer"/>
            </xs:sequence>
            <xs:attribute name="gkod" type="xs:ID" use="required
               "/>
            <xs:attribute name="alvazszam" type="xs:string" use=
               "optional"/>
            <xs:attribute name="rendszam" type="xs:string" use="
               required"/>
            <xs:attribute name="O_G" type="xs:IDREF" use="
               required"/>
          </xs:complexType>
```

4. Java adatszerkezet megvalósítása DOM API segítségével

4.1. Adatolvasás

A program célja, hogy XML dokumentumból Java objektum-szerűen kinyerje és megjelenítse az adatokat. A feldolgozás az org.w3c.dom könyvtár DOM API-ján keresztül történik, amely az XML fájlokat fa-szerkezetként kezeli, így minden elem és attribútum könnyen elérhető.

Listing 3. Javaban dokumentum létrehozás

```
        File xmlFile = new File("A24E5Z_XML.xml");

        DocumentBuilderFactory factory = DocumentBuilderFactory.
           newInstance();
        DocumentBuilder dBuilder = factory.newDocumentBuilder();

        Document doc = dBuilder.parse(xmlFile);
        doc.getDocumentElement().normalize();
```

Ez a rész hozza létre magát a DOM objektumot

Listing 4. Javaban nyilvántartás objektum létrehozás

```
NodeList gepList = doc.getElementsByTagName("Gepjarmu");

        for (int i = 0; i < gepList.getLength(); i++) {
            Node nNode = gepList.item(i);

            System.out.println("\nAktuális elem: " + nNode.
               getNodeName());

            if (nNode.getNodeType() == Node.ELEMENT_NODE) {
                Element elem = (Element) nNode;

                System.out.println("Gkód: " + elem.getAttribute(
                   "gkod"));
                System.out.println("Alvázszám: " + elem.
                   getAttribute("alvazszam"));
                System.out.println("Rendszám: " + elem.
                   getAttribute("rendszam"));
                System.out.println("Tulaj (O_G): " + elem.
                   getAttribute("O_G"));

                System.out.println("Típus: " + elem.
                   getElementsByTagName("Tipus").item(0).
                   getTextContent());
                System.out.println("Gyártási év: " + elem.
                   getElementsByTagName("GyartasiEv").item(0).
                   getTextContent());
                System.out.println("Márka: " + elem.
                   getElementsByTagName("Marka").item(0).
                   getTextContent());
                System.out.println("Km óra: " + elem.
                   getElementsByTagName("Km_ora").item(0).
                   getTextContent());
            }
```

Ez a kódrészlet egy adott ág kezelését írja le. A program feldogozza a mozi elemeket és attribútumokat, majd külön kiírja azokat a konzolra.

4.2. Adat-lekérdezés

Ez a program a DomRead-el szemben csak szimplán kiírja az adatokat, hanem emberileg olvasható és feldolgozható formában és mennyiségben írja ki azt.

Listing 5. Javaban rendezett adatkiírás

```
            System.out.println("3. Minden tankolás dátuma:");
            NodeList tankList = doc.getElementsByTagName("
               Tankolas");
            for (int i = 0; i < tankList.getLength(); i++) {
                Element t = (Element) tankList.item(i);
                String datum = t.getElementsByTagName("Datum").
                   item(0).getTextContent();
                System.out.println("  - " + datum);
            }
            System.out.println();
```

Ebben a kódrészletben először kinyer a program minden hasznos elementet és attribútumot, majd azt egy kompakt módon olvashatóan kiírja.

4.3. Adatmódosítás

Egy DomModify program alapvető feladata az adott filerendszerben valamilyen változtatást végrehajtani és egy új fileba elmenteni azokat a változtatásokat.

Listing 6. Javaban adat változtatás

```
NodeList carList = doc.getElementsByTagName("Gepjarmu");

            for (int i = 0; i < carList.getLength(); i++) {
                Node car = carList.item(i);

                NamedNodeMap attr = car.getAttributes();
                Node gkod = attr.getNamedItem("gkod");

                if ("G002".equals(gkod.getTextContent())) {

                    NodeList list = car.getChildNodes();

                    for (int j = 0; j < list.getLength(); j++) {
                        Node node = list.item(j);

```

```
16                          if (node.getNodeType() == Node.
                               ELEMENT_NODE) {
17                              Element eElement = (Element) node;
18
19                              if ("Marka".equals(eElement.
                                   getNodeName())) {
20                                  eElement.setTextContent("Toyota"
                                       );
21                              }
22
23                              if ("Km_ora".equals(eElement.
                                   getNodeName())) {
24                                  eElement.setTextContent("75000")
                                       ;
25                              }
26
27                              if ("GyartasiEv".equals(eElement.
                                   getNodeName())) {
28                                  eElement.setTextContent("2020");
29                              }
30                          }
31                      }
32                  }
33              }
```

Ebben a kódrészletben a második gépjárműre szűr rá, hogy csak azt változtassa meg majd a példány módosul "Toyota(Márka), 2020(Gyártási év) 75000(Km/ora)"-ra.

Listing 7. Javaban file-ba mentés

```
TransformerFactory transformerFactory = TransformerFactory.
   newInstance();
            Transformer transformer = transformerFactory.
               newTransformer();
            transformer.setOutputProperty(OutputKeys.ENCODING, "
               UTF-8");
            transformer.setOutputProperty(OutputKeys.INDENT, "
               yes");
            transformer.setOutputProperty("{http://xml.apache.
               org/xslt}indent-amount", "2");

            DOMSource source = new DOMSource(doc);

            System.out.println("--- Módosított XML fájl ---");
            StreamResult result = new StreamResult(new File("
               A24E5Z_XMLA24E5Z.xml"));
            transformer.transform(source, result);
```

A kód egy "transformer" nevezetű objektum segítségével menti el a megváltoztatott adatokat. A programban a "setOutputProperty()" segítségével állítjuk be a kimeneti file formátumát.