JEGYZŐKÖNYV

Adatkezelés XML-ben

Féléves feladat

Repülőtér adatbázis

Készítette: **Szabó Kristóf** Neptunkód: **IDA58U**

Dátum: 2025/12/02

1. Bevezetés:

A féléves feladatom során egy repülőtér adatbázisának egy lehetséges megoldását készítettem el.

Az adatbázisnak képesnek kell lennie a repülőtér működéséhez szükséges adatok átfogó kezelésére, beleértve a légitársaságokat, gépeket, járatokat, utasokat és foglalásokat. A rendszer célja, hogy az információk egyszerűen nyilvántarthatók, szerkeszthetők és lekérdezhetők legyenek, biztosítva ezzel a repülőtér gördülékeny működését.

Fontos szempont volt a járatok és foglalások nyilvántartása, hogy az utasok zökkenőmentesen és pontosan érkezzenek célállomásaikra.

Ezeket a szempontokat figyelembe véve kezdtem el az adatbázis átfogó tervezését, majd később a megvalósítását, biztosítva a kapcsolatok és adatok helyes struktúráját.

2. A feladat leírása:

A feladatom során először egy repülőtér ER modellt készítettem, amelyet egy szerkesztőprogramban felépítettem. Ez az ER modell szolgált alapul az XDM modell elkészítéséhez, ahol figyeltem arra, hogy hű maradjak az eredeti ER struktúrához és a repülőtér adatait pontosan leképezzem.

Miután mindkét modell elkészült, az XDM modell alapján létrehoztam az XML dokumentumot, külön figyelve arra, hogy az XDM modell sajátosságait megfelelően át tudjam vinni az XML struktúrába. Ezt követően elkészítettem a hozzá tartozó XML Schemat, hogy validálni tudjam az XML dokumentumom helyességét.

A harmadik feladatrész a DOM kezelés volt. Elsőként elkészítettem egy Java DOM olvasót, amely képes volt a repülőtér adatait konzolra és fájlba is kiírni. Ezt követően létrehoztam egy adatírásra optimalizált DOM API-t, majd egy lekérdező DOM API-t, amely legalább négy lekérdezést tudott elvégezni az XML dokumentumon, például repülőterek, járatok és utasok lekérdezését. Végül elkészítettem egy DOM módosító API-t, amely legalább négy módosítást tudott végrehajtani a dokumentumon, így biztosítva az adatok szerkeszthetőségét.

3. Feladat: Az adatbázis

3.1 Az ER modell:

A feladatban **hat egyed** található: **Légitársaság, Repülőtér, Gép, Járat, Utas és Foglalás**.
Ezek együtt alkotják a repülőtér-adatbázis teljes struktúráját, amely tartalmazza a légiközlekedési hálózatot felépítő elemeket és az ezeket összekötő kapcsolatokat.

A Légitársaság egyed tulajdonságai

- **LégitársaságID:** A légitársaság egyed elsődleges kulcsa.
- **Alapítás:** A légitársaság alapításának éve.
- **Név:** A légitársaság neve.
- **Székhely:** Összetett tulajdonságként is értelmezhető, a légitársaság székhelyének városa.

A Repülőtér egyed tulajdonságai

- **RepülőtérID:** A repülőtér egyed elsődleges kulcsa.
- **Név:** A repülőtér megnevezése.
- **Cím:** Összetett tulajdonság, tartalmazza az országot és a várost.
- **Főkapacitás:** A repülőtér maximális fogadóképessége (pl. kapuk vagy terminálkapacitás).

A Gép egyed tulajdonságai

- **GépID:** A gép egyed elsődleges kulcsa.
- **LégitársaságID:** Idegen kulcs, a géphez tartozó légitársaság azonosítója.
- **Típus:** A repülőgép típusa (pl. Airbus A320).
- **Gyártási év:** A gép gyártási éve.
- **Üléskapacitás:** A fedélzeten rendelkezésre álló ülések száma.

A Járat egyed tulajdonságai

- **JáratID:** A járat egyed elsődleges kulcsa.
- **GépID:** Idegen kulcs, a járatot teljesítő repülőgép azonosítója.
- **RepülőtérID:** Idegen kulcs, a járat indulási repülőtere.
- **Járatszám:** A járat egyedi azonosító kódja.
- **Menetrend:** Összetett tulajdonság, tartalmazza az indulási és érkezési időpontot.
- **Helyszín:** Többértékű összetett tulajdonság, tartalmazza a kiindulási és cél várost.

Az Utas egyed tulajdonságai

- **UtasID:** Az utas egyed elsődleges kulcsa.
- **Név:** Az utas neve.
- **Születési dátum:** Az utas születési ideje.
- **Útlevélszám:** Az utas útlevele, egyedi azonosítóként funkcionálhat.
- **Telefonszám:** Az utas elérhetősége (többértékű is lehetne).

A Foglalás egyed tulajdonságai

- **FoglalásID:** A foglalás egyed elsődleges kulcsa.
- **JáratID:** Idegen kulcs, a foglaláshoz tartozó járat.
- **UtasID:** Idegen kulcs, a foglalást készítő utas.
- **Dátum:** A foglalás dátuma.
- **Ülésszám:** Az adott utasnak lefoglalt ülés száma.

KAPCSOLATOK

Légitársaság és Gép

A Légitársaság és a Gép egyedek között egy a többhöz kapcsolat van, mivel egy légitársaság több repülőgépet is üzemeltethet, azonban egy gép csak egy légitársasághoz tartozik.

Repülőtér és Járat

A Repülőtér és a Járat egyedek között egy a többhöz kapcsolat van, mivel egy repülőtérhez több induló járat tartozhat, ugyanakkor egy járat egy adott repülőtérről indul.

Gép és Járat

A Gép és a Járat egyedek között egy a többhöz kapcsolat áll fenn. Egy repülőgép több járatot is teljesíthet, de egy konkrét járatot mindig csak egy gép teljesít.

Járat és Utas

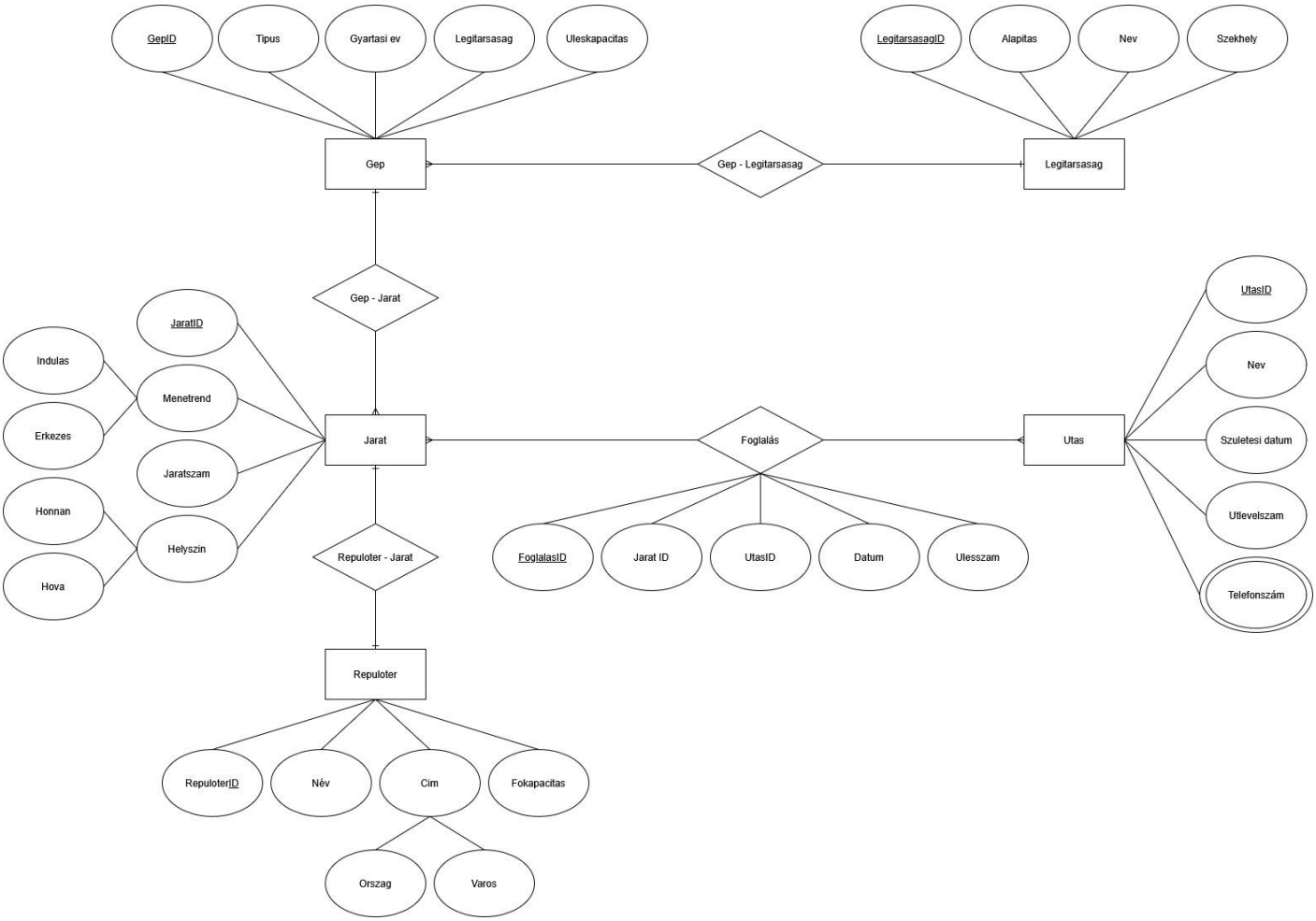
A Járat és az Utas egyedek között több a többhöz kapcsolat van, mivel egy járaton több utas is utazhat, és egy utas több különböző járatra is foglalhat jegyet. Ezt a kapcsolatot a Foglalás egyed valósítja meg.

A kapcsolat paraméterei:

- Dátum: a foglalás dátuma
- Ülésszám: az adott járaton lefoglalt ülés azonosítója

Foglalás – Járat – Utas

A Foglalás egy összekötő egyed, amely a Járat és az Utas közötti több-a-többhöz kapcsolatot oldja fel. Egy foglalás mindig egy adott járathoz és egy konkrét utashoz tartozik.

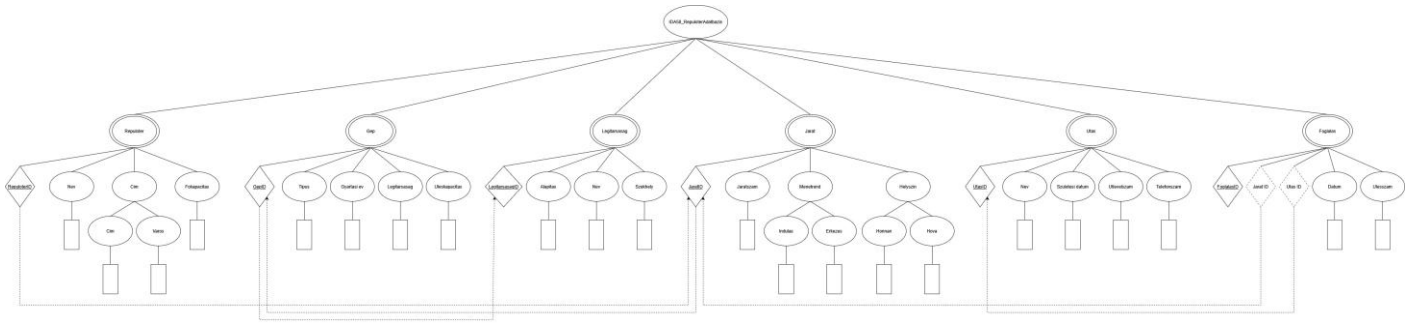


3.2 Adatbázis XDM modellre

Az XDM modellre való átalakítás ennél az adatbázisnál viszonylag egyszerűen megvalósítható volt. A legfontosabb lépést az jelentette, hogy az elsődleges és idegen kulcsokat megfelelően összehangoljuk, majd ezeket attribútumként jelenítsük meg.

Az egyes egyedekből többszörösen előforduló XML-elemek jöttek létre, és ezek tulajdonságai az adott elem alá tartozó gyerekelemek formájában jelentek meg.

Az összetett attribútumok esetében további, hierarchikusan alárendelt elemeket kellett létrehozni, amelyek az összetett tulajdonság egyes részeit írják le.



3.3 Az XML Dokumentum elkészitése:

Az XML-dokumentum elkészítése ezt követően már nagyon egyszerű feladat volt. A korábban megtervezett XDM modell alapján csupán implementálni kellett az egyes elemeket. A feladat előírása szerint minden többször előforduló elemnek legalább három példányt kellett szerepelnie.

Az elemek és attribútumok közvetlenül, módosítás nélkül átültethetők voltak az XML-be, és ezt valósítottam meg az első programkódban.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<RepuloterAdatbazis>

    <!-- LÉGITÁRSASÁGOK -->
    <Legitarsasag LegitarsasagID="1">
        <Alapitas>1990</Alapitas>
        <Nev>Hungarian Air</Nev>
        <Szekhely>Budapest</Szekhely>
    </Legitarsasag>

    <Legitarsasag LegitarsasagID="2">
        <Alapitas>1985</Alapitas>
        <Nev>SkyWings Europe</Nev>
        <Szekhely>Vienna</Szekhely>
    </Legitarsasag>

    <Legitarsasag LegitarsasagID="3">
        <Alapitas>2001</Alapitas>
        <Nev>Danube Flights</Nev>
        <Szekhely>Bratislava</Szekhely>
    </Legitarsasag>


    <!-- REPÜLŐTEREK -->
    <Repuloter RepuloterID="1">
        <Nev>Budapest Liszt Ferenc</Nev>
        <Cim>
            <Orszag>Magyarország</Orszag>
            <Varos>Budapest</Varos>
        </Cim>
        <Fokapacitas>120</Fokapacitas>
    </Repuloter>

    <Repuloter RepuloterID="2">
        <Nev>Vienna International Airport</Nev>
        <Cim>
```

```
            <Orszag>Ausztria</Orszag>
            <Varos>Vienna</Varos>
        </Cim>
        <Fokapacitas>150</Fokapacitas>
    </Repuloter>

    <Repuloter RepuloterID="3">
        <Nev>M. R. Štefánik Airport</Nev>
        <Cim>
            <Orszag>Szlovákia</Orszag>
            <Varos>Bratislava</Varos>
        </Cim>
        <Fokapacitas>90</Fokapacitas>
    </Repuloter>


    <!-- GÉPEK -->
    <Gep GepID="1" LegitarsasagID="1">
        <Tipus>Airbus A320</Tipus>
        <GyartasiEv>2012</GyartasiEv>
        <Uleskapacitas>180</Uleskapacitas>
    </Gep>

    <Gep GepID="2" LegitarsasagID="2">
        <Tipus>Boeing 737</Tipus>
        <GyartasiEv>2016</GyartasiEv>
        <Uleskapacitas>160</Uleskapacitas>
    </Gep>

    <Gep GepID="3" LegitarsasagID="3">
        <Tipus>Embraer 195</Tipus>
        <GyartasiEv>2019</GyartasiEv>
        <Uleskapacitas>120</Uleskapacitas>
    </Gep>


    <!-- JÁRATOK -->
    <Jarat JaratID="1" GepID="1" RepuloterID="1">
        <Jaratszam>HA123</Jaratszam>
        <Menetrend>
            <Indulas>2025-03-20T08:30</Indulas>
            <Erkezes>2025-03-20T09:45</Erkezes>
        </Menetrend>
        <Helyszin>
            <Honnan>Budapest</Honnan>
            <Hova>Vienna</Hova>
        </Helyszin>
    </Jarat>
```

```xml
    <Jarat JaratID="2" GepID="2" RepuloterID="2">
        <Jaratszam>SW455</Jaratszam>
        <Menetrend>
            <Indulas>2025-03-22T12:00</Indulas>
            <Erkezes>2025-03-22T14:15</Erkezes>
        </Menetrend>
        <Helyszin>
            <Honnan>Vienna</Honnan>
            <Hova>Budapest</Hova>
        </Helyszin>
    </Jarat>

    <Jarat JaratID="3" GepID="3" RepuloterID="3">
        <Jaratszam>DF210</Jaratszam>
        <Menetrend>
            <Indulas>2025-04-02T06:00</Indulas>
            <Erkezes>2025-04-02T07:20</Erkezes>
        </Menetrend>
        <Helyszin>
            <Honnan>Bratislava</Honnan>
            <Hova>Vienna</Hova>
        </Helyszin>
    </Jarat>


    <!-- UTASOK -->
    <Utas UtasID="1">
        <Nev>Kovács Péter</Nev>
        <SzuletesiDatum>1990-05-12</SzuletesiDatum>
        <Utlevelszam>HX1234567</Utlevelszam>
        <Telefonszam>+36 30 555 1234</Telefonszam>
    </Utas>

    <Utas UtasID="2">
        <Nev>Anna Müller</Nev>
        <SzuletesiDatum>1988-11-03</SzuletesiDatum>
        <Utlevelszam>AT99887766</Utlevelszam>
        <Telefonszam>+43 660 123 4567</Telefonszam>
    </Utas>

    <Utas UtasID="3">
        <Nev>Ján Novak</Nev>
        <SzuletesiDatum>1985-02-17</SzuletesiDatum>
        <Utlevelszam>SK55221133</Utlevelszam>
    </Utas>


    <!-- FOGLALÁSOK -->
    <Foglalas FoglalasID="1" JaratID="1" UtasID="1">
```

```
        <Datum>2025-03-10</Datum>
        <Ulesszam>12A</Ulesszam>
    </Foglalas>

    <Foglalas FoglalasID="2" JaratID="2" UtasID="2">
        <Datum>2025-03-15</Datum>
        <Ulesszam>7C</Ulesszam>
    </Foglalas>

    <Foglalas FoglalasID="3" JaratID="3" UtasID="3">
        <Datum>2025-03-18</Datum>
        <Ulesszam>3B</Ulesszam>
    </Foglalas>

</RepuloterAdatbazis>
```

3.4 XML Schema elkészitése

Az XML séma készítésekor arra törekedtem, hogy minden szabályt a lehető legpontosabban definiáljak, így kizárva a hibás adatok megjelenésének lehetőségét a dokumentumban.

Egyszerű típusok esetén reguláris kifejezésekkel határoztam meg az irányítószámok és telefonszámok formátumát. Az áraknál minimális értéket adtam meg, míg a bankkártyák típusát felsorolással korlátoztam, ezzel is biztosítva a megfelelő adatformátumot.

A kapcsolatok és kulcsok kezelését xs:key és xs:keyref segítségével oldottam meg, amelyek a relációs adatbázisok elsődleges és idegen kulcsait modellezik, és garantálják az adatintegritást a dokumentumon belül.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <!-- Egyszerű típusok -->
    <xs:simpleType name="PositiveInteger">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="DateType">
        <xs:restriction base="xs:date"/>
    </xs:simpleType>

    <xs:simpleType name="DateTimeType">
        <xs:restriction base="xs:dateTime"/>
    </xs:simpleType>
```

```
    <!-- Cím komplex típus -->
    <xs:complexType name="CimType">
        <xs:sequence>
            <xs:element name="Orszag" type="xs:string"/>
            <xs:element name="Varos" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Légitársaság komplex típus -->
    <xs:complexType name="LegitarsasagType">
        <xs:sequence>
            <xs:element name="Alapitas" type="PositiveInteger"/>
            <xs:element name="Nev" type="xs:string"/>
            <xs:element name="Szekhely" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="LegitarsasagID" type="PositiveInteger"
use="required"/>
    </xs:complexType>

    <!-- Repülőtér komplex típus -->
    <xs:complexType name="RepuloterType">
        <xs:sequence>
            <xs:element name="Nev" type="xs:string"/>
            <xs:element name="Cim" type="CimType"/>
            <xs:element name="Fokapacitas" type="PositiveInteger"/>
        </xs:sequence>
        <xs:attribute name="RepuloterID" type="PositiveInteger"
use="required"/>
    </xs:complexType>

    <!-- Gép komplex típus -->
    <xs:complexType name="GepType">
        <xs:sequence>
            <xs:element name="Tipus" type="xs:string"/>
            <xs:element name="GyartasiEv" type="PositiveInteger"/>
            <xs:element name="Uleskapacitas" type="PositiveInteger"/>
        </xs:sequence>
        <xs:attribute name="GepID" type="PositiveInteger" use="required"/>
        <xs:attribute name="LegitarsasagID" type="PositiveInteger"
use="required"/>
    </xs:complexType>

    <!-- Menetrend komplex típus -->
    <xs:complexType name="MenetrendType">
        <xs:sequence>
            <xs:element name="Indulas" type="DateTimeType"/>
            <xs:element name="Erkezes" type="DateTimeType"/>
```

```
        </xs:sequence>
    </xs:complexType>

    <!-- Helyszín komplex típus -->
    <xs:complexType name="HelyszinType">
        <xs:sequence>
            <xs:element name="Honnan" type="xs:string"/>
            <xs:element name="Hova" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Járat komplex típus -->
    <xs:complexType name="JaratType">
        <xs:sequence>
            <xs:element name="Jaratszam" type="xs:string"/>
            <xs:element name="Menetrend" type="MenetrendType"/>
            <xs:element name="Helyszin" type="HelyszinType"/>
        </xs:sequence>
        <xs:attribute name="JaratID" type="PositiveInteger" use="required"/>
        <xs:attribute name="GepID" type="PositiveInteger" use="required"/>
        <xs:attribute name="RepuloterID" type="PositiveInteger"
use="required"/>
    </xs:complexType>

    <!-- Utas komplex típus -->
    <xs:complexType name="UtasType">
        <xs:sequence>
            <xs:element name="Nev" type="xs:string"/>
            <xs:element name="SzuletesiDatum" type="DateType"/>
            <xs:element name="Utlevelszam" type="xs:string"/>
            <xs:element name="Telefonszam" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="UtasID" type="PositiveInteger" use="required"/>
    </xs:complexType>

    <!-- Foglalás komplex típus -->
    <xs:complexType name="FoglalasType">
        <xs:sequence>
            <xs:element name="Datum" type="DateType"/>
            <xs:element name="Ulesszam" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="FoglalasID" type="PositiveInteger"
use="required"/>
        <xs:attribute name="JaratID" type="PositiveInteger" use="required"/>
        <xs:attribute name="UtasID" type="PositiveInteger" use="required"/>
    </xs:complexType>

    <!-- Fő elem -->
```

```
    <xs:element name="RepuloterAdatbazis">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Legitarsasag" type="LegitarsasagType"
minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="Repuloter" type="RepuloterType"
minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="Gep" type="GepType" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Jarat" type="JaratType" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Utas" type="UtasType" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Foglalas" type="FoglalasType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>

        <!-- Kulcsok a fő elemekhez -->
        <xs:key name="LegitarsasagPK">
            <xs:selector xpath="Legitarsasag"/>
            <xs:field xpath="@LegitarsasagID"/>
        </xs:key>

        <xs:key name="RepuloterPK">
            <xs:selector xpath="Repuloter"/>
            <xs:field xpath="@RepuloterID"/>
        </xs:key>

        <xs:key name="GepPK">
            <xs:selector xpath="Gep"/>
            <xs:field xpath="@GepID"/>
        </xs:key>

        <xs:key name="JaratPK">
            <xs:selector xpath="Jarat"/>
            <xs:field xpath="@JaratID"/>
        </xs:key>

        <xs:key name="UtasPK">
            <xs:selector xpath="Utas"/>
            <xs:field xpath="@UtasID"/>
        </xs:key>

        <xs:key name="FoglalasPK">
            <xs:selector xpath="Foglalas"/>
            <xs:field xpath="@FoglalasID"/>
        </xs:key>
```

```
        <!-- Kulcshivatkozások (FK-k) -->
        <xs:keyref name="Gep_Legitarsasag_FK" refer="LegitarsasagPK">
            <xs:selector xpath="Gep"/>
            <xs:field xpath="@LegitarsasagID"/>
        </xs:keyref>

        <xs:keyref name="Jarat_Gep_FK" refer="GepPK">
            <xs:selector xpath="Jarat"/>
            <xs:field xpath="@GepID"/>
        </xs:keyref>

        <xs:keyref name="Jarat_Repuloter_FK" refer="RepuloterPK">
            <xs:selector xpath="Jarat"/>
            <xs:field xpath="@RepuloterID"/>
        </xs:keyref>

        <xs:keyref name="Foglalas_Jarat_FK" refer="JaratPK">
            <xs:selector xpath="Foglalas"/>
            <xs:field xpath="@JaratID"/>
        </xs:keyref>

        <xs:keyref name="Foglalas_Utas_FK" refer="UtasPK">
            <xs:selector xpath="Foglalas"/>
            <xs:field xpath="@UtasID"/>
        </xs:keyref>
    </xs:element>

</xs:schema>
```

4. Feladat: DOM API-k használata

Project neve: IDA58UDOMParse

Package neve: ida58u.domparse.hu

Class nevek: IDA58UDomRead, IDA58UDomQuery, IDA58UDomModify

4.1 Adatolvasás

A **IDA58UDomRead** osztály feladata az elkészített XML fájl beolvasása és tartalmának rendezett, áttekinthető megjelenítése. Ehhez DOM alapú feldolgozást alkalmaztam, amely az egész XML dokumentumot egy memóriában tárolt fa-struktúrává alakítja. Ez a módszer lehetővé teszi az elemek közötti könnyű bejárást, valamint a hierarchikus felépítésű adatok hatékony elérését.

```
package ida58u.domparse.hu;
```

```java
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class IDA58UDomRead {
      public static void main(String[] args) {
        try {
             //DOM Parser
            File xmlFile = new File("IDA58U_XML.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);

            doc.getDocumentElement().normalize();

            //  StringBuilder
            StringBuilder output = new StringBuilder();

            output.append("Repülőtér adatbázis betöltése \n");

            //  Elemek feldolgozása
            processElements(doc, output, "Legitarsasag");
            processElements(doc, output, "Repuloter");
            processElements(doc, output, "Gep");
            processElements(doc, output, "Jarat");
            processElements(doc, output, "Utas");
            processElements(doc, output, "Foglalas");


            // Kiírás a konzolra
            System.out.println(output.toString());

            //Kiírás fájlba
            try (PrintWriter writer = new PrintWriter(new
FileWriter("IDA58U_read_output.txt"))) {
                writer.print(output.toString());
            }
            System.out.println("\nAdatok fájlba írása sikeres.");

        } catch (ParserConfigurationException | SAXException | IOException
e) {
            e.printStackTrace();
        }
    }

    // Feldolgozo metodus
    private static void processElements(Document doc, StringBuilder sb,
String tagName) {
        NodeList nodeList = doc.getElementsByTagName(tagName);
        if (nodeList.getLength() > 0) {
```

```
            sb.append("\n ||| ").append(tagName.toUpperCase()).append("
|||\n");
        }

        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;
                sb.append("---------------------------------------\n");

                // Speciális formazas
                switch (tagName) {
                    case "Legitarsasag":
                        sb.append("Légitarsasag ID:
").append(element.getAttribute("LegitarsasagID")).append("\n");
                        sb.append("Név: ").append(getElementText(element,
"Nev")).append("\n");
                        sb.append("Alapítás:
").append(getElementText(element, "Alapitas")).append("\n");
                        sb.append("Székhely:
").append(getElementText(element, "Szekhely")).append("\n");
                        break;
                    case "Repuloter":
                        sb.append("Repülőtér ID:
").append(element.getAttribute("RepuloterID")).append("\n");
                        sb.append("Név: ").append(getElementText(element,
"Nev")).append("\n");
                        sb.append("Cím:
").append(formatCim(element)).append("\n");
                        sb.append("Főkapacitás:
").append(getElementText(element, "Fokapacitas")).append("\n");
                        break;
                    case "Gep":
                        sb.append("Gép ID:
").append(element.getAttribute("GepID")).append("\n");
                        sb.append("Típus: ").append(getElementText(element,
"Tipus")).append("\n");
                        sb.append("Gyártási év:
").append(getElementText(element, "GyartasiEv")).append("\n");
                        sb.append("üléskapacitás:
").append(getElementText(element, "Uleskapacitas")).append("\n");
                        break;
                    case "Jarat":
                        sb.append("Járat ID:
").append(element.getAttribute("JaratID")).append("\n");
                        sb.append("Járatszám:
").append(getElementText(element, "Jaratszam")).append("\n");
                        sb.append("Menetrend:
").append(formatMenetrend(element)).append("\n");
                        sb.append("Helyszin:
").append(formatHelyszin(element)).append("\n");
                        break;
                    case "Utas":
                        sb.append("Utas ID:
").append(element.getAttribute("UtasID")).append("\n");
                        sb.append("Név: ").append(getElementText(element,
"Nev")).append("\n");
                        sb.append("Születési dátum:
").append(getElementText(element, "SzuletesiDatum")).append("\n");
                        sb.append("Útlevélszám:
").append(getElementText(element, "Utlevelszam")).append("\n");
```

```
                        sb.append("Telefonszám:
").append(getElementText(element, "Telefonszam")).append("\n");
                        break;
                    case "Foglalas":
                        sb.append("Foglalás ID:
").append(element.getAttribute("FoglalasID")).append("\n");
                        sb.append("Dátum: ").append(getElementText(element,
"Datum")).append("\n");
                        sb.append("Ülésszám:
").append(getElementText(element, "Ulesszam")).append("\n");
                        break;
                }
            }
        }
    }

    //Cim blokk formazasa
    private static String formatCim(Element parent) {
        NodeList cimList = parent.getElementsByTagName("Cim");
        if (cimList.getLength() > 0) {
            Element cimElement = (Element) cimList.item(0);
            String orszag = getElementText(cimElement, "Orszag");
            String varos = getElementText(cimElement, "Varos");
            return orszag + " - " + varos;
        }
        return "N/A";
    }

    //Cim blokk formazasa
    private static String formatMenetrend(Element parent) {
      NodeList menetrendList = parent.getElementsByTagName("Menetrend");
        if (menetrendList.getLength() > 0) {
            Element menetrendElement = (Element) menetrendList.item(0);
            String indulas = getElementText(menetrendElement, "Indulas");
            String erkezes = getElementText(menetrendElement, "Erkezes");
            return indulas + " -> " + erkezes;
        }
        return "";
    }

    //Helyszín blokk formazasa
    private static String formatHelyszin(Element parent) {
        NodeList helyszinList = parent.getElementsByTagName("Helyszin");
        if (helyszinList.getLength() > 0) {
            Element helyszinElement = (Element) helyszinList.item(0);
            String honnan = getElementText(helyszinElement, "Honnan");
            String hova = getElementText(helyszinElement, "Hova");
            return honnan + " -> " + hova;
        }
        return "";
    }

    //gyokerelem szoveges tartalmanak visszaadasa
    private static String getElementText(Element parent, String tagName) {
        NodeList nodeList = parent.getElementsByTagName(tagName);
        if (nodeList.getLength() > 0) {
            Node node = nodeList.item(0);
            if (node.getFirstChild() != null &&
node.getFirstChild().getNodeType() == Node.TEXT_NODE) {
                return node.getFirstChild().getNodeValue().trim();
            }
```

```
        }
        return "";
    }
}
```

4.2 Adatlekérdezés:

A **D1H8VPDomQuery** osztály feladata, hogy a betöltött XML adatbázison célzott, üzleti logikához kapcsolódó kérdésekre adjon választ. A program a DOM fa felépítését használja az információk keresésére, szűrésére és összekapcsolására, ezzel bemutatva, milyen lehetőségek rejlenek az XML-alapú adatok feldolgozásában.

A dokumentum beolvasása és normalizálása után az osztály öt különféle lekérdezést hajt végre. A megoldások során NodeList objektumokon történő iterációt és feltételes vizsgálatokat alkalmaztam a szükséges információk kinyerésére.

1. **Egyszerű listázás:** Listázza ki az összes repülőtér nevét
2. **Szűrés beágyazott elem alapján:** Listázza az összes repülőteret ahol a főkapacitás nagyobb mint 100
3. **Szűrés attribútum alapján:** Listázza ki az összes foglalást március 15. Utáni indulással
4. **Szűrés:** Listázza ki az összes járatot ahol a Gép típusa Airbus
5. **Relációs összekapcsolás:** Listázza ki azokat az utasokat akik Londonba utaznak

```
package ida58u.domparse.hu;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashSet;
import java.util.Set;

public class IDA58UDomQuery {
      public static void main(String[] args) {
        try {
            //DOM Parser
            File xmlFile = new File("IDA58U_XML.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);
            doc.getDocumentElement().normalize();

            // StringBuilder
            StringBuilder output = new StringBuilder();
```

```
            output.append("Repülőtér adatbázis lekérdezések \n");

            //Lekérdezések
            lekerdezes1_MindenRepuloterNeve(doc, output);
            lekerdezes2_RepterekSzazFoFelett(doc, output);
            lekerdezes3_MarciusTizenotUtaniFoglalasok(doc, output);
            lekerdezes4_AirbusJaratok(doc, output);
            lekerdezes5_KiUtazikLondonba(doc, output);


            System.out.println(output.toString());

            // Kiírás fájlba
            try (PrintWriter writer = new PrintWriter(new
FileWriter("IDA58U_query_output.xml"))) {
                writer.print(output.toString());
            }
            System.out.println("\nLekérdezések fájlba írása sikeres");

        } catch (ParserConfigurationException | SAXException | IOException
e) {
            e.printStackTrace();
        }
    }

    //LEKÉRDEZÉSEK

      //Listázza ki az összes repülőtér nevét.
      private static void lekerdezes1_MindenRepuloterNeve(Document doc,
StringBuilder sb) {
          sb.append("\n1. LEKÉRDEZÉS: Minden repülőtér neve \n");

          NodeList repterNodeList = doc.getElementsByTagName("Repuloter");
          for (int i = 0; i < repterNodeList.getLength(); i++) {
              Node node = repterNodeList.item(i);
              if (node.getNodeType() == Node.ELEMENT_NODE) {
                  Element repter = (Element) node;
                  String nev = getElementText(repter, "Nev");
                  sb.append(" - ").append(nev).append("\n");
              }
          }
      }

      //Listázza ki azokat a repülőtereket, amelyeknek a főkapacitása 100
fölött van.
      private static void lekerdezes2_RepterekSzazFoFelett(Document doc,
StringBuilder sb) {
            sb.append("\n2. LEKÉRDEZÉS: Repülőterek 100 fő felett\n");

          NodeList repterNodeList = doc.getElementsByTagName("Repuloter");
          for (int i = 0; i < repterNodeList.getLength(); i++) {
              Node node = repterNodeList.item(i);
              if (node.getNodeType() == Node.ELEMENT_NODE) {
                  Element repter = (Element) node;
                  String kapacitasStr = getElementText(repter,
"Fokapacitas");
                  try {
                      int kapacitas = Integer.parseInt(kapacitasStr);
                      if (kapacitas > 100) {
                          sb.append(" - ").append(getElementText(repter,
"Nev"))
```

```
                                .append(" (Főkapacitás:
").append(kapacitas).append(")\n");
                        }
                    } catch (NumberFormatException e) {
                        // Hibás adat, kihagyjuk
                    }
                }
            }
        }

        //Listázza ki azoknak az utasoknak a nevét, akik március 15. után
foglaltak.
        private static void
lekerdezes3_MarciusTizenotUtaniFoglalasok(Document doc, StringBuilder sb) {
            sb.append("\n3. LEKÉRDEZÉS: Március 15. utáni foglalások\n");

            Set<String> utasIDk = new HashSet<>();
            NodeList foglalasList = doc.getElementsByTagName("Foglalas");
            for (int i = 0; i < foglalasList.getLength(); i++) {
                Element foglalas = (Element) foglalasList.item(i);
                String datumStr = getElementText(foglalas, "Datum");
                if (datumStr != null && !datumStr.isEmpty()) {
                    String[] parts = datumStr.split("-");
                    if (parts.length == 3) {
                        int honap = Integer.parseInt(parts[1]);
                        int nap = Integer.parseInt(parts[2]);
                        if (honap > 3 || (honap == 3 && nap > 15)) {
                            utasIDk.add(foglalas.getAttribute("UtasID"));
                        }
                    }
                }
            }

            if (utasIDk.isEmpty()) {
                sb.append(" (Nincs március 15. utáni foglalás.)\n");
                return;
            }

            NodeList utasList = doc.getElementsByTagName("Utas");
            for (int i = 0; i < utasList.getLength(); i++) {
                Element utas = (Element) utasList.item(i);
                if (utasIDk.contains(utas.getAttribute("UtasID"))) {
                    sb.append(" - ").append(getElementText(utas,
"Nev")).append("\n");
                }
            }
        }

        //Listázza ki az összes Airbus típusú géppel közlekedő járatot.
        private static void lekerdezes4_AirbusJaratok(Document doc,
StringBuilder sb) {
            sb.append("\n4. LEKÉRDEZÉS: Airbus típusú géppel közlekedő
járatok \n");

            NodeList jaratList = doc.getElementsByTagName("Jarat");
            NodeList gepList = doc.getElementsByTagName("Gep");

            for (int i = 0; i < jaratList.getLength(); i++) {
                Element jarat = (Element) jaratList.item(i);

                // A Jarat elemnek van GepID attribútuma (pl. GepID="1")
```

```
            String gepID = jarat.getAttribute("GepID");
            if (gepID == null || gepID.isEmpty()) continue;

            // Megkeressük a Gep elemet a GepID alapján
            Element gepTalalt = null;
            for (int j = 0; j < gepList.getLength(); j++) {
                Element gep = (Element) gepList.item(j);
                String thisGepID = gep.getAttribute("GepID");
                if (gepID.equals(thisGepID)) {
                    gepTalalt = gep;
                    break;
                }
            }

            if (gepTalalt == null) {
                // nincs ilyen gép bejegyezve – kihagyjuk
                continue;
            }

            String tipus = getElementText(gepTalalt, "Tipus");
            // Ellenőrizzük, hogy a típus tartalmazza-e az "Airbus" szót
(pl. "Airbus A320")
            if (tipus != null && tipus.contains("Airbus")) {
                // Járatszám (a séma szerint <Jaratszam> a tag)
                String jaratszam = getElementText(jarat, "Jaratszam");
                sb.append(" - ").append(jaratszam).append(" (Gép:
").append(tipus).append(", GepID: ").append(gepID).append(")\n");
            }
        }
    }

    //Listázza ki a Londonba utazó utasokat.
    private static void lekerdezes5_KiUtazikLondonba(Document doc,
StringBuilder sb) {
        sb.append("\n5. LEKÉRDEZÉS: Ki utazik Londonba?\n");

        Set<String> utasIDk = new HashSet<>();

        // Megkeressük a londoni járatok ID-ját
        NodeList jaratList = doc.getElementsByTagName("Jarat");
        Set<String> londonJaratok = new HashSet<>();

        for (int i = 0; i < jaratList.getLength(); i++) {
            Element jarat = (Element) jaratList.item(i);

            NodeList helyszinList =
jarat.getElementsByTagName("Helyszin");
            if (helyszinList.getLength() > 0) {
                Element helyszinElement = (Element) helyszinList.item(0);
                String honnan = getElementText(helyszinElement,
"Honnan");
                String hova = getElementText(helyszinElement, "Hova");
                  if ("London".equals(hova)) {
                      londonJaratok.add(jarat.getAttribute("JaratID"));
                  }
            }
        }

        if (londonJaratok.isEmpty()) {
            sb.append(" (Nincs Londonba tartó járat.)\n");
            return;
```

```
            }

            // Utasok a londoni járatokon
            NodeList foglalasList = doc.getElementsByTagName("Foglalas");
            for (int i = 0; i < foglalasList.getLength(); i++) {
                Element foglalas = (Element) foglalasList.item(i);

                if (londonJaratok.contains(foglalas.getAttribute("JaratID")))
{
                    utasIDk.add(foglalas.getAttribute("UtasID"));
                }
            }

            if (utasIDk.isEmpty()) {
                sb.append(" (Senki nem foglalt Londonba.)\n");
                return;
            }

            // Utasok nevei
            NodeList utasList = doc.getElementsByTagName("Utas");
            for (int i = 0; i < utasList.getLength(); i++) {
                Element utas = (Element) utasList.item(i);
                if (utasIDk.contains(utas.getAttribute("UtasID"))) {
                    sb.append(" - ").append(getElementText(utas,
"Nev")).append("\n");
                }
            }
        }


    private static String getElementText(Element parent, String tagName) {
        NodeList nodeList = parent.getElementsByTagName(tagName);
        if (nodeList.getLength() > 0) {
            Node node = nodeList.item(0);
            if (node.getFirstChild() != null &&
node.getFirstChild().getNodeType() == Node.TEXT_NODE) {
                return node.getFirstChild().getNodeValue().trim();
            }
        }
        return "";
    }

}
```

4.3 Adat módosítás:

A **IDA58UDomModify** osztály feladata a betöltött XML adatbázis tartalmának programozott módosítása. A kód szemlélteti, hogyan lehet a memóriában található DOM-fa elemein változtatásokat végezni, majd az így módosított dokumentumot tartósan elmenteni.

A megvalósítás során a program öt különféle manipulációs műveletet hajt végre, amelyek lefedik a DOM alapú módosítás legfontosabb lehetőségeit.

1. **Tartalom frissítése:** Módosítsa a “3”-as azonosítójú reptér főkapacitását 300 főre

2. **Attribútum módosítása:** A “SW455” azonosítójú járat gépét módosítsa egy “Airbus” típusú gépre.
3. **Új elem hozzáadása:** Adjon hozzá egy telefonszámot a “3”-as azonosítójú utashoz.
4. **Elem törlése:** Törölje a “2”-es azonosítójú utas foglalását.

```
package ida58u.domparse.hu;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.File;
import java.io.IOException;

public class IDA58UDomModify {
      public static void main(String[] args) {
        File inputFile = new File("IDA58U_XML.xml");
        File outputFile = new File("IDA58U_XML_modositasok.xml");

        try {
            // XML beolvasása
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();


            StringBuilder report = new StringBuilder();
            report.append("Repülőtér adatbázis módosítások \n");

            modositas1_FokapacitasModositas(doc, report);
            modositas2_GepModositas(doc, report);
            modositas3_UjElemHozzaadasa(doc, report);
            modositas4_ElemTorlese(doc, report);

             // kiírása konzolra
            System.out.println(report.toString());
            //modositas kiirasa
            saveXml(doc, outputFile);

            System.out.println("\nMódosítások sikeresen mentve a
IDA58U_XML.xml_modositasok.xml fájlba");

        } catch (ParserConfigurationException | SAXException | IOException
| TransformerException e) {
            e.printStackTrace();
```

```
        }
    }

     // Módosítja a 3-as repülőtér főkapacitását
    private static void modositas1_FokapacitasModositas(Document doc,
StringBuilder report) {
        report.append("\n 1. MÓDOSÍTÁS: A 3-as ID-jú repülőtér
kapacitásának módosítása 300-ra\n");
        NodeList repuloterList = doc.getElementsByTagName("Repuloter");
        for (int i = 0; i < repuloterList.getLength(); i++) {
            Element repuloter = (Element) repuloterList.item(i);
            if ("3".equals(repuloter.getAttribute("RepuloterID"))) {
                Node fokapacitasNode =
repuloter.getElementsByTagName("Fokapacitas").item(0);
                String regiFokapacitas = fokapacitasNode.getTextContent();
                fokapacitasNode.setTextContent("300");
                report.append("3-as repülőtér módosítva.\n");
                report.append("Régi
főkapacitás:").append(regiFokapacitas).append(", Új főkapacitás: 300\n");
                return;
            }
        }
        report.append("HIBA: 3-as ID-jú repülőtér nem található.\n");
    }


    //A SW455-as számú járat gépét módosítjuk egy Airbus gepre
    private static void modositas2_GepModositas(Document doc, StringBuilder
report) {
        report.append("\n2. MÓDOSÍTÁS: Járat gép típusának módosítása \n");

        NodeList gepList = doc.getElementsByTagName("Gep");
        String airbusGep = "";

        for (int i = 0; i < gepList.getLength(); i++) {
            Element gep = (Element) gepList.item(i);

            String tipus = getElementText(gep, "Tipus");
            if (tipus != null && tipus.contains("Airbus")) {
                  //Találjuk meg az első airbus gépet
                  airbusGep = gep.getAttribute("GepID");
              }
        }

        NodeList jaratList = doc.getElementsByTagName("Jarat");
        for (int i = 0; i < jaratList.getLength(); i++) {
            Element jarat = (Element) jaratList.item(i);
            if ("SW455".equals(getElementText(jarat,"Jaratszam"))) {
                String regiGep = jarat.getAttribute("GepID");
                jarat.setAttribute("GepID", airbusGep);
                report.append("SW455 járat gépe módosítva.\n");
                report.append("Régi GepID: ").append(regiGep).append(", Új
GepID: ").append(airbusGep).append("\n");
                return;
            }
        }
        report.append("HIBA: SW455 járat nem található.\n");
    }


    //Telefonszám hozzáadása a 3-as utashoz
```

```
    private static void modositas3_UjElemHozzaadasa(Document doc,
StringBuilder report) {
        report.append("\n3. MÓDOSÍTÁS: Telefonszám hozzáadása 3-as utashoz
\n");
        NodeList utasList = doc.getElementsByTagName("Utas");
        for (int i = 0; i < utasList.getLength(); i++) {
            Element utas = (Element) utasList.item(i);
            if ("3".equals(utas.getAttribute("UtasID"))) {
                Element telefonszam = doc.createElement("Telefonszam");
                telefonszam.setTextContent("+52 616 222 9876");
                utas.appendChild(telefonszam);
                report.append("SIKER: +52 616 222 9876 telefonszám
hozzáadva a 3-as utashoz.\n");
                return;
            }
        }
        report.append(" HIBA: 3-as utas nem található.\n");
    }


    //2-es utas foglalásának törlése
    private static void modositas4_ElemTorlese(Document doc, StringBuilder
report) {
        report.append("\n 5. MÓDOSÍTÁS: 2-es utas foglalásának törlése
\n");
        NodeList foglalasList = doc.getElementsByTagName("Foglalas");


        for (int i = foglalasList.getLength() - 1; i >= 0; i--) {
            Element foglalas = (Element) foglalasList.item(i);
            if ("2".equals(foglalas.getAttribute("UtasID"))) {
                String toroltFoglalasID =
foglalas.getAttribute("FoglalasID");
                Node szulo = foglalas.getParentNode();
                szulo.removeChild(foglalas);

                report.append("SIKER: 2 utas (FoglalasID:
").append(toroltFoglalasID).append(") foglalása törölve \n");
                return;
            }
        }
        report.append(" HIBA: 2 utashoz tartozó foglalás nem
található.\n");
    }

    private static String getElementText(Element parent, String tagName) {
        NodeList nodeList = parent.getElementsByTagName(tagName);
        if (nodeList.getLength() > 0) {
            Node node = nodeList.item(0);
            if (node.getFirstChild() != null &&
node.getFirstChild().getNodeType() == Node.TEXT_NODE) {
                return node.getFirstChild().getNodeValue().trim();
            }
        }
        return "";
    }

    //Segedfuggveny a menteshez
    private static void saveXml(Document doc, File file) throws
TransformerException {
```

```
        TransformerFactory transformerFactory =
TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();

        //kimenet beallitasa
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-
amount", "4");

        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(file);

        transformer.transform(source, result);
    }

}
```

Tartalom