JEGYZŐKÖNYV

Adatkezelés XML-ben

Féléves feladat

Pizzázó hálózat

Készítette: **Bodnár László** Neptunkód: **D1H8VP**

Dátum: 2025/11/25

Bevezetés:

Az adatkezelés XML-ben című tantárgyra készített féléves feladatom során egy Pizzázó hálózat adatbázisának egy lehetséges megoldását mutatja be. Ez a pizzázó hálózat több ember egyidejű étkeztetésére is alkalmas kell legyen.

A cégnek szüksége van futárokra, beszállítókra és természetesen több ételfajtára is, hogy releváns tudjon maradni. Mindig törekedniük kell a frissességre és az emberek igényeire, legyen ez gyors kiszállítás vagy finom pizza. Ehhez egy jól felépített rendszerre van szükségük amelyben adataik egyszerűen nyilvántarthatóak, szerkeszthetőek és megtekinthetőek, ha ez nem történik meg a cég nem működhet helyesen.

Fontos lesz továbbá futárokat, rendeléseket nyilvántartani, annak érdekében hogy a rendelések minél optimálisabban es helyesen kerüljenek a vásárlók elé.

Ezen szempontok szemelőt tartásával kezdtem hozzá a rendszer átfogó tervezéséhez majd később a megvalósításhoz.

A feladat leírása:

A feladatomnak egy pizzázóhoz terveztem először ER modellt és ezt felepitettem egy szerkesztőprogramban, ezután az ER modell figyelembevételével és alapként használva megkezdtem az átkonvertálást XDM modellre, itt odafigyeltem, hogy hű maradjak az eredeti ER modellemhez.

Miután kész voltam mindkét modellel, következhetett az XDM modell alapján elkészített XML dokumentum, odafigyelve, hogy az XDM modell sajátosságait át tudjam vinni. Amint ezzel is elkészültem, elkészitettem a hozzá tartozó XML Schemat- hogy ellenorizhessem az XML dokumentumom helyességét.

A harmadik feladatrész nem volt más, mint DOM használatával 3 feladatot elvégezni az XML dokumentumomon. Az első egy Java Dom Reader elkészitès volt, amely konzolra és fájlra is kiir. A második egy Adatiirasra kiélezett Dom Api elkészitése volt. A harmadik egy lekérdező DOM api keszitese volt legalaább 4 lekérdezéssel, és az utolsó egy DOM modify api volt ahol legalabb 4 modositast kellett veghezvinni.

1. Feladat: Az adatbázis

1.1 Az ER modell:

A feladatomban 6 egyed található: A bankkártya, vevő,Pizza, Futár, Beszállito és a pizzázó. Ezek együttesen fogják megadni az adatbázis tartalmát. Megtalálhatjuk benne a Pizzazó hálózatot felépitő elemeket és az ezeket összekötő kapcsolatokat.

- **A Bankkártya egyed tulajdonságai**
- Kártyaszám: A Bankkártya egyed elsődleges kulcsa. • Bank: A bank neve, amelyhez a bankkártya tartozik.
- Lejárati dátum: A kártya lejárati dátuma.
- Típus: A bankkártya típusa.

- **A Vevő egyed tulajdonságai** ₀ VevőID: A Vevő egyed elsődleges kulcsa. ₀ Név: A vevő neve. ₀ Telefonszám: A vevő telefonszáma.
  o Cím: Összetett tulajdonság. A vevő címe.

- **A Pizza egyed tulajdonságai** ₀ PizzaID: A Pizza egyed elsődleges kulcsa. ₀ Teljes ár: A rendelt pizza/pizzák teljes ára. Származtatott tulajdonság. ₀ Pizza neve: A pizza neve.
- o Méret: Többértékű tulajdonság. A pizza méretét tárolja.
- o Feltét: Többértékű tulajdonság. A pizzán lévő feltéteket tárolja.

- **A Futár egyed tulajdonságai** o FutárID: A Futár egyed elsődleges kulcsa. o Telefonszám: A futár telefonszáma.

  o Név: A futár neve.

- **A Beszállító egyed tulajdonságai** o BeszállítóID:

A Beszállító egyed elsődleges kulcsa. o Elérhetőség: A beszállító elérhetősége. o Név: A beszállító cég neve. o Cím: Összetett tulajdonság. A beszállító cég címe.

- **A Pizzázó egyed tulajdonságai** o PizzázóID: A Pizzázó egyed elsődleges kulcsa. o Nyitva tartás: A pizzázó nyitva tartási ideje. o Név: A pizzázó neve. o Elérhetőség: Összetett tulajdonság. A pizzázó elérhetőségei.

<u>Egyedek közötti kapcsolat</u>:

- **Pizzázó és Futár:**

A Pizzázó és a Futár egyedek között egy a többhöz kapcsolat van, mivel egy pizzázó alkalmazhat több futárt, de egy futár csak egy pizzázónál dolgozik.

- **Pizzázó és Beszállító:**

A Pizzázó és a Beszállító egyedek között több a többhöz kapcsolat van, mivel egy pizzázó rendelhet több beszállítótól, valamint egy beszállító beszállíthat több pizzázónak is. A kapcsolat paraméterei: a Hozzávalók, amely a beszállító által beszállított hozzávalókat jelenti, valamint a Dátum, azaz a beszállítás dátuma.

- **Pizzázó és Pizza:**

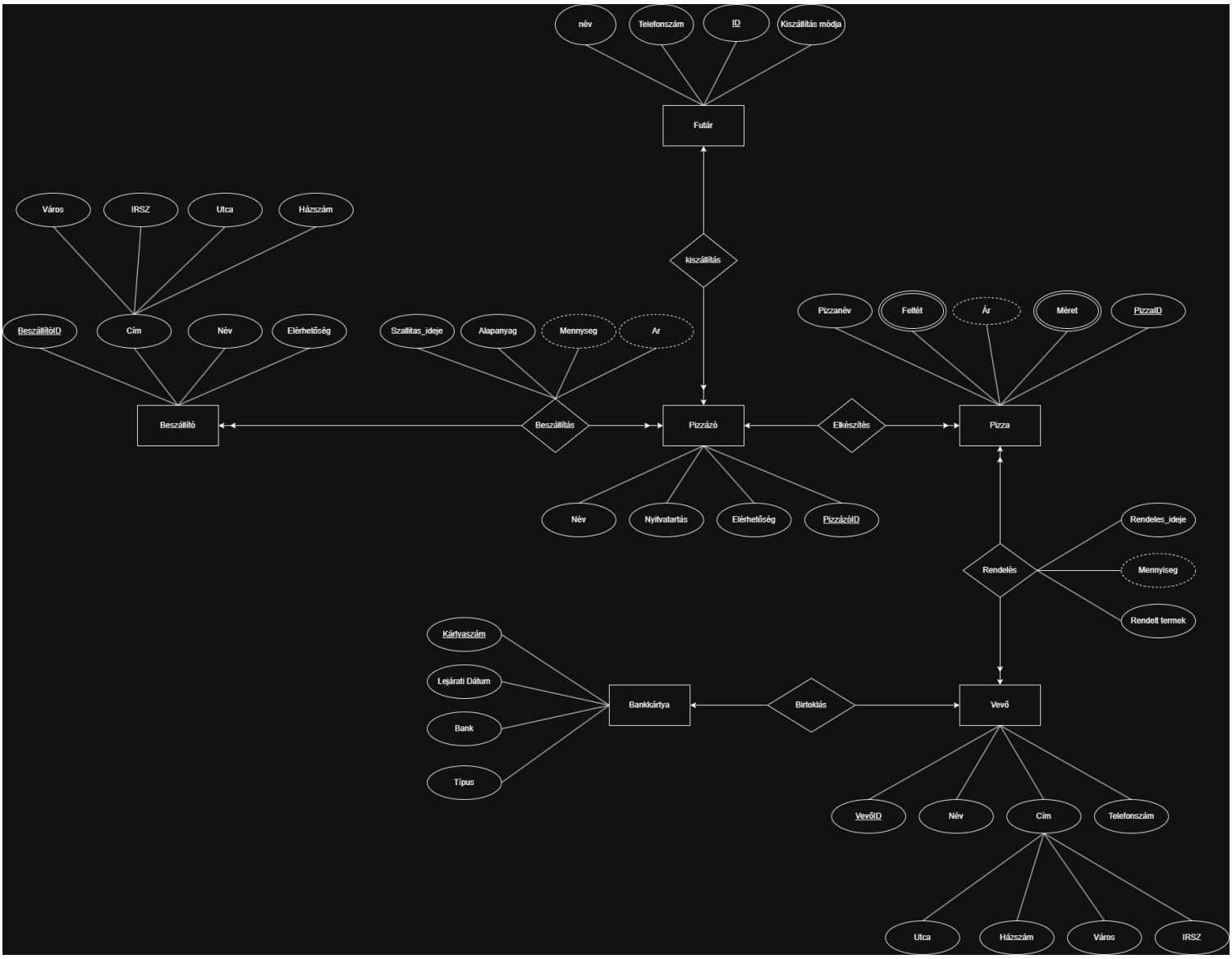
A Pizzázó és a Pizza egyedek között egy a többhöz kapcsolat van, mivel egy pizzázónak lehet több pizzája, de egy pizza csak egy pizzázóhoz tartozhat.

- **Pizza és Vevő:**

A Pizza és a Vevő egyedek között több a többhöz kapcsolat van, mivel egy vevő rendelhet többfajta pizzát, és a pizzából rendelhet több különböző vevő is.

- **Vevő és Bankkártya:**

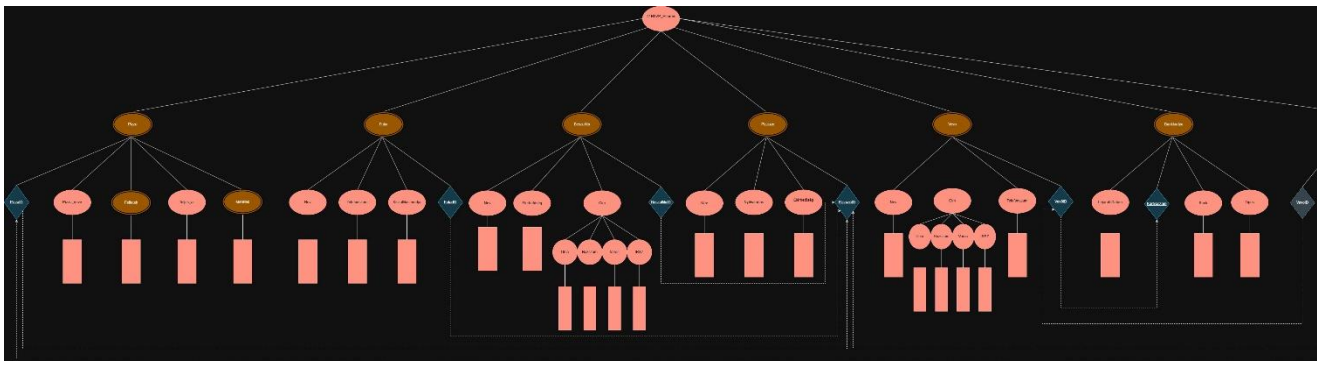
A Vevő és a Bankkártya egyedek között egy az egyhez kapcsolat van, mivel egy vevőnek csak egy bankártyát lehet az oldalhozt hozzárendelnie



1.Ábra ER modell

1.2 Adatbázis XDM modellre

Az XDM modellre történő konvertálás egészen egyszerű volt ez esetben. A legfontosabb elvégzendő feladat az elsődleges és az idegen kulcsok egyeztetése, attribútummá alakítása volt. Az egyese egyedekből többszörösen megjelenő elemek lettek, ezen egyedek tulajdonságai pedig az említett elemek gyerekelemei lettek. Az összetett tulajdonság esetén szintén még két gyerekelem keletkezett .



2. Ábra XDM modell

2.2 Az XML Dokumentum elkészítése:

Az XML dokumentum elkészítése ezek után nagyon egyszerű volt. A megtervezett XDM modell alapján kellett csak implementálni az egyes elemeket. A feladatkiírás szerint minden többször előforduló elemből legalább 3 példányt kellett készíteni. Nem tartottam valóságszerűnek, hogy ugyanazon útvonalon menjenek a járatok, emiatt a helyszínekből, megállásokból, megállókból jóval több lett, így egy valósághű példát láthatunk a dokumentumban.

Az elemeket és az attribútumokat egy az egyben lehet implementálni az XML dokumentumban, ezt tettem meg az 1. programkód esetén.

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

<Etterem_D1H8VP xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
                xs:noNamespaceSchemaLocation="XMLSchemaD1H8VP.xsd">

    <!--Pizzazo-->
```

```
<Pizzazo PizzazoID="P1">
    <Nyitvatartas>10:00-22:00</Nyitvatartas>
    <Nev>Miskolc Pizza</Nev>
    <Telefonszam>06301234567</Telefonszam>
    <Weboldal>miskolcpizza.hu</Weboldal>
</Pizzazo>

<Pizzazo PizzazoID="P2">
    <Nyitvatartas>08:00-20:00</Nyitvatartas>
    <Nev>Kiraly Pizzeria</Nev>
    <Telefonszam>307654321</Telefonszam>
    <Weboldal>kiraly.hu</Weboldal>
</Pizzazo>

<!--Beszallito-->

<Beszallito BeszallitoID="B1">
    <Elerhetoseg>info@beszal1.hu</Elerhetoseg>
    <Nev>Beszallito1</Nev>
    <Cim>
        <Varos>Kazincbarcika</Varos>
        <IRSZ>3700</IRSZ>
        <Utca>Joszerencset ut</Utca>
        <Hazszam>2</Hazszam>
    </Cim>
</Beszallito>
```

```
<Beszallito BeszallitoID="B2">
    <Elerhetoseg>info@beszal2.hu</Elerhetoseg>
    <Nev>Beszallito2</Nev>
    <Cim>
        <Varos>Kazincbarcika</Varos>
        <IRSZ>3700</IRSZ>
        <Utca>Dobo istvan ter</Utca>
        <Hazszam>7</Hazszam>
    </Cim>
</Beszallito>

<!--Futar-->

<Futar FutarID="F1" PizzazoID="P1">
    <Nev>James Bond</Nev>
    <Telefonszam>70115452</Telefonszam>
</Futar>

<Futar FutarID="F2" PizzazoID="P1">
    <Nev>Jack Sparrow</Nev>
    <Telefonszam>7036534444</Telefonszam>
</Futar>

<Futar FutarID="F3" PizzazoID="P2">
    <Nev>Nagy Sándor</Nev>
    <Telefonszam>705543566</Telefonszam>
</Futar>
```

```
<!--Vevo-->

<Vevo VevoID="V1">
    <Nev>Vásár Ló</Nev>
    <Telefonszam>20198772</Telefonszam>
    <Cim>
        <Varos>Kazincbarcika</Varos>
        <IRSZ>3700</IRSZ>
        <Utca>József attila utca</Utca>
        <Hazszam>7</Hazszam>
     </Cim>
</Vevo>

<Vevo VevoID="V2">
    <Nev>K Anna</Nev>
    <Telefonszam>203342344</Telefonszam>
    <Cim>
        <Varos>Miskolc</Varos>
        <IRSZ>3700</IRSZ>
        <Utca>Dobo utca</Utca>
        <Hazszam>3</Hazszam>
    </Cim>
</Vevo>

<!--Bankkartya-->
```

```
<Bankkartya Kartyaszam="17846573498579" VevoID="V1">
    <Bank>OTP Bank</Bank>
    <Lejarati_datum>2026-12-31</Lejarati_datum>
    <Tipus>Mastercard</Tipus>
</Bankkartya>

<Bankkartya Kartyaszam="643724683" VevoID="V2">
    <Bank>KH Bank</Bank>
    <Lejarati_datum>2025-06-30</Lejarati_datum>
    <Tipus>Visa</Tipus>
</Bankkartya>

<!--Pizza-->


<Pizza PizzaID="PIZ1" PizzazoID="P1">
    <Teljes_ar>2800</Teljes_ar>
    <Pizza_neve>Margarita</Pizza_neve>
    <Feltetek>
        <Feltet>Paradicsom</Feltet>
        <Feltet>Sajt</Feltet>
        <Feltet>Oregeno</Feltet>
    </Feltetek>
    <Meretek>
        <Meret>32cm</Meret>
    </Meretek>
</Pizza>
```

```
<Pizza PizzaID="PIZ2" PizzazoID="P1">
    <Teljes_ar>3200</Teljes_ar>
    <Pizza_neve>Son goku</Pizza_neve>
    <Feltetek>
        <Feltet>Paradicsom</Feltet>
        <Feltet>Sajt</Feltet>
        <Feltet>Sonka</Feltet>
        <Feltet>Kukorica</Feltet>
    </Feltetek>
    <Meretek>
        <Meret>32cm</Meret>
        <Meret>45cm</Meret>
    </Meretek>
</Pizza>

<Pizza PizzaID="PIZ3" PizzazoID="P2">
    <Teljes_ar>3500</Teljes_ar>
    <Pizza_neve>Magyaros</Pizza_neve>
    <Feltetek>
        <Feltet>Paradicsom</Feltet>
        <Feltet>Sajt</Feltet>
        <Feltet>Kolbász</Feltet>
        <Feltet>Hagyma</Feltet>
        <Feltet>Erős paprika</Feltet>
    </Feltetek>
    <Meretek>
```

```
            <Meret>30cm</Meret>
        </Meretek>
    </Pizza>

    <!--Rendeles-->

    <Rendeles RendelesID="R1" VevoID="V1" PizzaID="PIZ2"/>
    <Rendeles RendelesID="R2" VevoID="V2" PizzaID="PIZ1"/>
    <Rendeles RendelesID="R3" VevoID="V2" PizzaID="PIZ3"/>

</Etterem_D1H8VP>
```

1.Kódrészlet XML

2.3 XML Schema elkészitése

Az XML Schema készítése során igyekeztem minél pontosabban meghatározni minden szabályt, hogy véletlenül se kerülhessen helytelen adat a dokumentumba.

Egyszerű típusok: Reguláris kifejezésekkel korlátoztam az irányítószámok és telefonszámok formátumát. Az árak esetében minimum értéket határoztam meg, a bankkártyák típusát pedig felsorolással szűkítettem.

Kapcsolatok és kulcsok: Az adatok integritását xs:key és xs:keyref elemekkel biztosítottam, amelyek leképezik a relációs adatbázisok elsődleges és idegen kulcsait

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

    <!--Egyszerű tipusok-->
```

```
    <xs:simpleType name="IrszTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9]{4}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="TelefonszamTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9]{8,15}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="BankkartyaModTipus">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Mastercard"/>
            <xs:enumeration value="Visa"/>
            <xs:enumeration value="Maestro"/>
            <xs:enumeration value="American Express"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ArTipus">
        <xs:restriction base="xs:unsignedInt">
            <xs:minInclusive value="100"/>
        </xs:restriction>
    </xs:simpleType>

    <!--Gyokerlem-->

    <xs:element name="Etterem_D1H8VP">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Pizzazo" type="PizzazoTipus" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Beszallito" type="BeszallitoTipus" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Futar" type="FutarTipus" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Vevo" type="VevoTipus" minOccurs="0"
```

```
maxOccurs="unbounded"/>
                <xs:element name="Bankkartya" type="BankkartyaTipus" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Pizza" type="PizzaTipus" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Rendeles" type="RendelesTipus" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>

        <!--Kulcsok-->

        <xs:key name="PizzazoPK">
            <xs:selector xpath="Pizzazo"/>
            <xs:field xpath="@PizzazoID"/>
        </xs:key>
        <xs:key name="BeszallitoPK">
            <xs:selector xpath="Beszallito"/>
            <xs:field xpath="@BeszallitoID"/>
        </xs:key>
        <xs:key name="FutarPK">
            <xs:selector xpath="Futar"/>
            <xs:field xpath="@FutarID"/>
        </xs:key>
        <xs:key name="VevoPK">
            <xs:selector xpath="Vevo"/>
            <xs:field xpath="@VevoID"/>
        </xs:key>
        <xs:key name="BankkartyaPK">
            <xs:selector xpath="Bankkartya"/>
            <xs:field xpath="@Kartyaszam"/>
        </xs:key>
        <xs:key name="PizzaPK">
            <xs:selector xpath="Pizza"/>
            <xs:field xpath="@PizzaID"/>
        </xs:key>
        <xs:key name="RendelesPK">
            <xs:selector xpath="Rendeles"/>
            <xs:field xpath="@RendelesID"/>
```

```
    </xs:key>

    <xs:keyref name="Futar_Pizzazo_FK" refer="PizzazoPK">
        <xs:selector xpath="Futar"/>
        <xs:field xpath="@PizzazoID"/>
    </xs:keyref>
    <xs:keyref name="Pizza_Pizzazo_FK" refer="PizzazoPK">
        <xs:selector xpath="Pizza"/>
        <xs:field xpath="@PizzazoID"/>
    </xs:keyref>
    <xs:unique name="Bankkartya_Vevo_Unique">
        <xs:selector xpath="Bankkartya"/>
        <xs:field xpath="@VevoID"/>
    </xs:unique>
    <xs:keyref name="Rendeles_Vevo_FK" refer="VevoPK">
        <xs:selector xpath="Rendeles"/>
        <xs:field xpath="@VevoID"/>
    </xs:keyref>
    <xs:keyref name="Rendeles_Pizza_FK" refer="PizzaPK">
        <xs:selector xpath="Rendeles"/>
        <xs:field xpath="@PizzaID"/>
    </xs:keyref>
</xs:element>

<!--Komplex tipusok-->

<xs:complexType name="CimTipus">
    <xs:sequence>
        <xs:element name="Varos" type="xs:string"/>
        <xs:element name="IRSZ" type="IrszTipus"/>
        <xs:element name="Utca" type="xs:string"/>
        <xs:element name="Hazszam" type="xs:string"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="PizzazoTipus">
    <xs:sequence>
        <xs:element name="Nyitvatartas" type="xs:string"/>
        <xs:element name="Nev" type="xs:string"/>
```

```
        <xs:element name="Telefonszam" type="TelefonszamTipus"/>
        <xs:element name="Weboldal" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="PizzazoID" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="BeszallitoTipus">
    <xs:sequence>
        <xs:element name="Elerhetoseg" type="xs:string"/>
        <xs:element name="Nev" type="xs:string"/>
        <xs:element name="Cim" type="CimTipus"/>
    </xs:sequence>
    <xs:attribute name="BeszallitoID" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="FutarTipus">
    <xs:sequence>
        <xs:element name="Nev" type="xs:string"/>
        <xs:element name="Telefonszam" type="TelefonszamTipus"/>
    </xs:sequence>
    <xs:attribute name="FutarID" type="xs:string" use="required"/>
    <xs:attribute name="PizzazoID" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="VevoTipus">
    <xs:sequence>
        <xs:element name="Nev" type="xs:string"/>
        <xs:element name="Telefonszam" type="TelefonszamTipus"/>
        <xs:element name="Cim" type="CimTipus"/>
    </xs:sequence>
    <xs:attribute name="VevoID" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="BankkartyaTipus">
    <xs:sequence>
        <xs:element name="Bank" type="xs:string"/>
        <xs:element name="Lejarati_datum" type="xs:date"/>
        <xs:element name="Tipus" type="BankkartyaModTipus"/>
    </xs:sequence>
```

```
        <xs:attribute name="Kartyaszam" type="xs:string" use="required"/>
        <xs:attribute name="VevoID" type="xs:string" use="required"/>
    </xs:complexType>

    <xs:complexType name="PizzaTipus">
        <xs:sequence>
            <xs:element name="Teljes_ar" type="ArTipus"/>
            <xs:element name="Pizza_neve" type="xs:string"/>
            <xs:element name="Feltetek">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Feltet" type="xs:string"
maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="Meretek">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Meret" type="xs:string" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="PizzaID" type="xs:string" use="required"/>
        <xs:attribute name="PizzazoID" type="xs:string" use="required"/>
    </xs:complexType>

    <xs:complexType name="RendelesTipus">
        <xs:attribute name="RendelesID" type="xs:string" use="required"/>
        <xs:attribute name="VevoID" type="xs:string" use="required"/>
        <xs:attribute name="PizzaID" type="xs:string" use="required"/>
    </xs:complexType>

</xs:schema>
```

2.Kódrészlet XML Schema

3. Feladat: DOM API-k használata

Project name: XML_FÉLÉVES_D1H8VP

Package: D1H8VP.domparse.hu

Class names: D1H8VPDOMRead, D1H8VPDOMWrite. D1H8VPDOMQuery, D1H8VPDOMModify

2.1 Adatolvasás

A D1H8VPDomRead osztály feladata az elkészített XML állomány beolvasása és tartalmának strukturált megjelenítése. A megoldáshoz a **DOM** parsert alkalmaztam, amely a teljes XML dokumentumot fa-struktúraként tölti be a memóriába Ez a megközelítés lehetővé teszi az elemek közötti hierarchikus navigációt és az adatok kényelmes elérését.

```
package D1H8VP.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;

public class D1H8VPDomRead {

    public static void main(String[] args) {
        try {
             //DOM Parser
            File xmlFile = new File("XML_FÉLÉVES_D1H8VP/Etterem_D1H8VP.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);


            doc.getDocumentElement().normalize();

            //  StringBuilder
```

```
            StringBuilder output = new StringBuilder();

            output.append("Étterem Feldolgozás D1H8VP \n");

            //  Elemek feldolgozása
            processElements(doc, output, "Pizzazo");
            processElements(doc, output, "Beszallito");
            processElements(doc, output, "Futar");
            processElements(doc, output, "Vevo");
            processElements(doc, output, "Bankkartya");
            processElements(doc, output, "Pizza");
            processElements(doc, output, "Rendeles");

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

            //Kiírás fájlba
            try (PrintWriter writer = new PrintWriter(new
FileWriter("etterem_read_output_D1H8VP.xml"))) {
                writer.print(output.toString());
            }
            System.out.println("\nAdatok sikeresen kiírva az 'etterem_read_output_D1H8VP.'
fájlba.");

        } 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("
(").append(nodeList.getLength()).append(" db) ===\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 "Pizzazo":    formatPizzazo(element, sb);    break;
                    case "Beszallito": formatBeszallito(element, sb); break;
                    case "Futar":      formatFutar(element, sb);      break;
                    case "Vevo":       formatVevo(element, sb);       break;
                    case "Bankkartya": formatBankkartya(element, sb); break;
                    case "Pizza":      formatPizza(element, sb);      break;
                    case "Rendeles":   formatRendeles(element, sb);   break;
                }
            }
        }
    }
```

```
    // Segedfuggvenyek szebb formazashoz

    private static void formatPizzazo(Element element, StringBuilder sb) {
        sb.append("Pizzázó (ID: ").append(element.getAttribute("PizzazoID")).append(")\n");
        sb.append("  Név: ").append(getElementText(element, "Nev")).append("\n");
        sb.append("  Nyitvatartás: ").append(getElementText(element,
"Nyitvatartas")).append("\n");
        sb.append("  Telefon: ").append(getElementText(element,
"Telefonszam")).append("\n");
        sb.append("  Web: ").append(getElementText(element, "Weboldal")).append("\n");
    }

    private static void formatBeszallito(Element element, StringBuilder sb) {
        sb.append("Beszállító (ID:
").append(element.getAttribute("BeszallitoID")).append(")\n");
        sb.append("  Név: ").append(getElementText(element, "Nev")).append("\n");
        sb.append("  Elérhetőség: ").append(getElementText(element,
"Elerhetoseg")).append("\n");
        sb.append("  Cím: ").append(formatCim(element)).append("\n");
    }

    private static void formatFutar(Element element, StringBuilder sb) {
        sb.append("Futár (ID: ").append(element.getAttribute("FutarID")).append(")\n");
        sb.append("  Pizzázó ID: ").append(element.getAttribute("PizzazoID")).append("\n");
        sb.append("  Név: ").append(getElementText(element, "Nev")).append("\n");
        sb.append("  Telefon: ").append(getElementText(element,
"Telefonszam")).append("\n");
    }

    private static void formatVevo(Element element, StringBuilder sb) {
        sb.append("Vevő (ID: ").append(element.getAttribute("VevoID")).append(")\n");
        sb.append("  Név: ").append(getElementText(element, "Nev")).append("\n");
        sb.append("  Telefon: ").append(getElementText(element,
"Telefonszam")).append("\n");
        sb.append("  Cím: ").append(formatCim(element)).append("\n");
    }

    private static void formatBankkartya(Element element, StringBuilder sb) {
        sb.append("Bankkártya (Szám:
").append(element.getAttribute("Kartyaszam")).append(")\n");
        sb.append("  Vevő ID: ").append(element.getAttribute("VevoID")).append("\n");
        sb.append("  Bank: ").append(getElementText(element, "Bank")).append("\n");
        sb.append("  Lejárat: ").append(getElementText(element,
"Lejarati_datum")).append("\n");
        sb.append("  Típus: ").append(getElementText(element, "Tipus")).append("\n");
    }

    private static void formatPizza(Element element, StringBuilder sb) {
        sb.append("Pizza (ID: ").append(element.getAttribute("PizzaID")).append(")\n");
        sb.append("  Pizzázó ID: ").append(element.getAttribute("PizzazoID")).append("\n");
        sb.append("  Név: ").append(getElementText(element, "Pizza_neve")).append("\n");
        sb.append("  Ár: ").append(getElementText(element, "Teljes_ar")).append(" Ft\n");
        sb.append("  Feltétek: ").append(formatList(element, "Feltetek",
"Feltet")).append("\n");
```

```
        sb.append("  Méretek: ").append(formatList(element, "Meretek",
"Meret")).append("\n");
    }

    private static void formatRendeles(Element element, StringBuilder sb) {
        sb.append("Rendelés (ID:
").append(element.getAttribute("RendelesID")).append(")\n");
        sb.append("  Vevő ID: ").append(element.getAttribute("VevoID")).append("\n");
        sb.append("  Pizza ID: ").append(element.getAttribute("PizzaID")).append("\n");
    }


    //  segédfüggvények

    //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 "";
    }

    //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 irsz = getElementText(cimElement, "IRSZ");
            String varos = getElementText(cimElement, "Varos");
            String utca = getElementText(cimElement, "Utca");
            String hazszam = getElementText(cimElement, "Hazszam");
            // Strukturáltan összerakja, felesleges szóközök nélkül
            return irsz + " " + varos + ", " + utca + " " + hazszam;
        }
        return "N/A";
    }

    //Lista adatait gyujti
    private static String formatList(Element parent, String listTagName, String itemTagName)
{
        NodeList listNodes = parent.getElementsByTagName(listTagName);
        if (listNodes.getLength() > 0) {
            Element listElement = (Element) listNodes.item(0);
            NodeList itemNodes = listElement.getElementsByTagName(itemTagName);
            StringBuilder items = new StringBuilder();
            for (int i = 0; i < itemNodes.getLength(); i++) {
                items.append(itemNodes.item(i).getTextContent().trim());
                if (i < itemNodes.getLength() - 1) {
                    items.append(", ");
                }
            }
```

```
            return items.toString();
        }
        return "N/A";
    }
}
```

3.Kodreszlet DOMRead

2.2 Adatiras

A D1H8VPDomWrite osztály feladata egy új, valid XML dokumentum programozott előállítása a memóriában, majd annak fájlrendszerbe történő mentése. A program a korábban definiált adatmodellnek megfelelő struktúrát építi fel, demonstrálva a DOM létrehozási és manipulációs képességeit.

A kód a javax.xml.parsers csomag osztályait használja egy üres dokumentum példányosítására.

```
package D1H8VP.domparse.hu;

import org.w3c.dom.Document;
import org.w3c.dom.Element;

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

public class D1H8VPDomWrite {

    public static void main(String[] args) {
        try {

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.newDocument();

            //  yökérelem létrehozása
            Element root = doc.createElement("Etterem_D1H8VP");
            root.setAttribute("xmlns:xs", "http://www.w3.org/2001/XMLSchema-instance");
            root.setAttribute("xs:noNamespaceSchemaLocation", "XMLSchemaD1H8VP.xsd");
```

```
            doc.appendChild(root);

            //ADATOK HOZZÁADÁSA


            addPizzazo(doc, root, "P1", "10:00-22:00", "Miskolc Pizza", "06301234567",
"miskolcpizza.hu");


            addBeszallito(doc, root, "B1", "info@beszal1.hu", "Beszallito1",
"Kazincbarcika", "3700", "Joszerencset ut", "2");


            addFutar(doc, root, "F1", "P1", "James Bond", "70115452");


            addVevo(doc, root, "V1", "Vásár Ló", "20198772", "Kazincbarcika", "3700",
"József attila utca", "7");


            addBankkartya(doc, root, "17846573498579", "V1", "OTP Bank", "2026-12-31",
"Mastercard");


            addPizza(doc, root, "PIZ2", "P1", "3200", "Son goku",
                    new String[]{"Paradicsom", "Sajt", "Sonka", "Kukorica"},
                    new String[]{"32cm", "45cm"});


            addRendeles(doc, root, "R1", "V1", "PIZ2");


            // MENTÉS FÁJLBA ÉS KIÍRÁS

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

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

            DOMSource source = new DOMSource(doc);

            // Kiírás konzolra
            StreamResult consoleResult = new StreamResult(System.out);
            transformer.transform(source, consoleResult);

            // Kiírás fájlba
            StreamResult fileResult = new StreamResult(new File("etterem_output.xml"));
            transformer.transform(source, fileResult);

            System.out.println("\n\nSikeres mentés: etterem_output.xml");

        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }

    //SEGÉDFÜGGVÉNYEK A CSOMÓPONTOK LÉTREHOZÁSÁHOZ

    private static void addPizzazo(Document doc, Element root, String id, String nyitva,
String nev, String tel, String web) {
        Element pizzazo = doc.createElement("Pizzazo");
        pizzazo.setAttribute("PizzazoID", id);

        pizzazo.appendChild(createElement(doc, "Nyitvatartas", nyitva));
        pizzazo.appendChild(createElement(doc, "Nev", nev));
        pizzazo.appendChild(createElement(doc, "Telefonszam", tel));
        pizzazo.appendChild(createElement(doc, "Weboldal", web));

        root.appendChild(pizzazo);
    }

    private static void addBeszallito(Document doc, Element root, String id, String email,
String nev, String varos, String irsz, String utca, String hsz) {
        Element beszallito = doc.createElement("Beszallito");
        beszallito.setAttribute("BeszallitoID", id);

        beszallito.appendChild(createElement(doc, "Elerhetoseg", email));
        beszallito.appendChild(createElement(doc, "Nev", nev));

        // Cím struktúra
        Element cim = doc.createElement("Cim");
        cim.appendChild(createElement(doc, "Varos", varos));
        cim.appendChild(createElement(doc, "IRSZ", irsz));
        cim.appendChild(createElement(doc, "Utca", utca));
        cim.appendChild(createElement(doc, "Hazszam", hsz));
        beszallito.appendChild(cim);

        root.appendChild(beszallito);
    }

    private static void addFutar(Document doc, Element root, String id, String pid, String
nev, String tel) {
        Element futar = doc.createElement("Futar");
        futar.setAttribute("FutarID", id);
        futar.setAttribute("PizzazoID", pid);

        futar.appendChild(createElement(doc, "Nev", nev));
        futar.appendChild(createElement(doc, "Telefonszam", tel));

        root.appendChild(futar);
    }

    private static void addVevo(Document doc, Element root, String id, String nev, String
tel, String varos, String irsz, String utca, String hsz) {
        Element vevo = doc.createElement("Vevo");
        vevo.setAttribute("VevoID", id);

        vevo.appendChild(createElement(doc, "Nev", nev));
        vevo.appendChild(createElement(doc, "Telefonszam", tel));
```

```
        Element cim = doc.createElement("Cim");
        cim.appendChild(createElement(doc, "Varos", varos));
        cim.appendChild(createElement(doc, "IRSZ", irsz));
        cim.appendChild(createElement(doc, "Utca", utca));
        cim.appendChild(createElement(doc, "Hazszam", hsz));
        vevo.appendChild(cim);

        root.appendChild(vevo);
    }

    private static void addBankkartya(Document doc, Element root, String szam, String vid,
String bank, String lejarat, String tipus) {
        Element kartya = doc.createElement("Bankkartya");
        kartya.setAttribute("Kartyaszam", szam);
        kartya.setAttribute("VevoID", vid);

        kartya.appendChild(createElement(doc, "Bank", bank));
        kartya.appendChild(createElement(doc, "Lejarati_datum", lejarat));
        kartya.appendChild(createElement(doc, "Tipus", tipus));

        root.appendChild(kartya);
    }

    private static void addPizza(Document doc, Element root, String id, String pid, String
ar, String nev, String[] feltetek, String[] meretek) {
        Element pizza = doc.createElement("Pizza");
        pizza.setAttribute("PizzaID", id);
        pizza.setAttribute("PizzazoID", pid);

        pizza.appendChild(createElement(doc, "Teljes_ar", ar));
        pizza.appendChild(createElement(doc, "Pizza_neve", nev));

        // Feltétek lista
        Element feltetekNode = doc.createElement("Feltetek");
        for (String f : feltetek) {
            feltetekNode.appendChild(createElement(doc, "Feltet", f));
        }
        pizza.appendChild(feltetekNode);

        // Méretek lista
        Element meretekNode = doc.createElement("Meretek");
        for (String m : meretek) {
            meretekNode.appendChild(createElement(doc, "Meret", m));
        }
        pizza.appendChild(meretekNode);

        root.appendChild(pizza);
    }

    private static void addRendeles(Document doc, Element root, String rid, String vid,
String pid) {
        Element rendeles = doc.createElement("Rendeles");
        rendeles.setAttribute("RendelesID", rid);
        rendeles.setAttribute("VevoID", vid);
        rendeles.setAttribute("PizzaID", pid);
        root.appendChild(rendeles);
```

```
    }

    // segédfüggvény szöveges elemek létrehozásához
    private static Element createElement(Document doc, String name, String value) {
        Element node = doc.createElement(name);
        node.appendChild(doc.createTextNode(value));
        return node;
    }
}
```

4.Kodreszlet DOMWeite

2.3 Adatlekérdezés

A D1H8VPDomQuery osztály célja, hogy a betöltött XML adatbázison specifikus, üzleti jellegű kérdéseket válaszoljon meg. A program a DOM fa-struktúráját használja fel az adatok keresésére, szűrésére és összekapcsolására, demonstrálva az XML-ben rejlő információkinyerési lehetőségeket.

A dokumentum beolvasását és normalizálását követően a program öt különböző lekérdezést futtat le. A megoldások során a NodeList objektumokon történő iterációt, valamint feltételes elágazásokat alkalmaztam a kívánt adatok kinyeréséhez.

A program különböző nehézségi szintű műveleteket hajt végre:

1. **Egyszerű listázás:** Minden pizzázó nevének kilistázása
2. **Szűrés beágyazott elem alapján:** Azoknak a beszállítóknak a keresése, akiknek a telephelye "Kazincbarcika".
3. **Szűrés attribútum alapján:** A "P1" azonosítójú pizzázóhoz tartozó futárok listázása idegen kulcs ellenőrzésével.
4. **Keresés beágyazott listában:** Olyan pizzák keresése, amelyek feltétlistájában (Feltetek/Feltet) szerepel a Sajt.

5. **Relációs összekapcsolás** Egy összetett lekérdezés, amely megkeresi a "Magyaros" pizza rendelőit.

```
package D1H8VP.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 D1H8VPDomQuery {

    public static void main(String[] args) {
        try {
            //DOM Parser
            File xmlFile = new File("XML_FÉLÉVES_D1H8VP/Etterem_D1H8VP.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);
            doc.getDocumentElement().normalize();

            // StringBuilder
            StringBuilder output = new StringBuilder();
            output.append("DOM LEKÉRDEZÉSEK D1H8VP \n");

            //Lekérdezések
            query1_MindenPizzazoNeve(doc, output);
            query2_KazincbarcikaiBeszallitok(doc, output);
            query3_P1Futarai(doc, output);
            query4_PizzakSajtFeltettel(doc, output);
            query5_KiRendeltMagyarosPizzat(doc, output);


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

            // Kiírás fájlba
            try (PrintWriter writer = new PrintWriter(new
FileWriter("etterem_lekerdezesek_D1H8VP.xml"))) {
                writer.print(output.toString());
            }
```

```
            System.out.println("\nLekérdezések sikeresen kiírva
etterem_lekerdezesek_D1H8VP.xml ");

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

    //LEKÉRDEZÉSEK


      //LEKÉRDEZÉS1: Listázza ki minden pizzázó nevét.

    private static void query1_MindenPizzazoNeve(Document doc, StringBuilder sb) {
        sb.append("\n1. LEKÉRDEZÉS: Minden pizzázó neve \n");
        NodeList pizzazoNodeList = doc.getElementsByTagName("Pizzazo");

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


      //LEKÉRDEZÉS : Listázza ki Kazincbarcika városban beszállítók nevét.

    private static void query2_KazincbarcikaiBeszallitok(Document doc, StringBuilder sb) {
        sb.append("\n 2. LEKÉRDEZÉS: Kazincbarcikai beszállítók \n");
        NodeList beszallitoNodeList = doc.getElementsByTagName("Beszallito");

        for (int i = 0; i < beszallitoNodeList.getLength(); i++) {
            Node node = beszallitoNodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element beszallito = (Element) node;

                // Cim vasros elem
                Element cim = (Element) beszallito.getElementsByTagName("Cim").item(0);
                String varos = getElementText(cim, "Varos");

                if ("Kazincbarcika".equals(varos)) {
                    String nev = getElementText(beszallito, "Nev");
                    sb.append("  - ").append(nev).append(" (").append(varos).append(")\n");
                }
            }
        }
    }


     // LEKÉRDEZÉS : Listázd ki a P1 PizzazoIDhez pizzázóhoz tartozó futárok nevét.

    private static void query3_P1Futarai(Document doc, StringBuilder sb) {
        sb.append("\n 3. LEKÉRDEZÉS: A P1 pizzázó futárai \n");
```

```
        NodeList futarNodeList = doc.getElementsByTagName("Futar");

        for (int i = 0; i < futarNodeList.getLength(); i++) {
            Node node = futarNodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element futar = (Element) node;


                String pizzazoID = futar.getAttribute("PizzazoID");

                if ("P1".equals(pizzazoID)) {
                    String nev = getElementText(futar, "Nev");
                    sb.append("  - ").append(nev).append(" (ID:
").append(futar.getAttribute("FutarID")).append(")\n");
                }
            }
        }
    }


     // LEKÉRDEZÉS: Listázza ki az összes olyan pizza nevét amelyiken van sajt feltét.

    private static void query4_PizzakSajtFeltettel(Document doc, StringBuilder sb) {
        sb.append("\n 4. LEKÉRDEZÉS: Pizzák sajt feltéttel \n");
        NodeList pizzaNodeList = doc.getElementsByTagName("Pizza");

        for (int i = 0; i < pizzaNodeList.getLength(); i++) {
            Node pizzaNode = pizzaNodeList.item(i);
            if (pizzaNode.getNodeType() == Node.ELEMENT_NODE) {
                Element pizza = (Element) pizzaNode;
                String pizzaNev = getElementText(pizza, "Pizza_neve");

                // Feltetek ellenorzese
                Element feltetekElement = (Element)
pizza.getElementsByTagName("Feltetek").item(0);
                NodeList feltetList = feltetekElement.getElementsByTagName("Feltet");

                for (int j = 0; j < feltetList.getLength(); j++) {
                    String feltet = feltetList.item(j).getTextContent();
                    if ("Sajt".equals(feltet.trim())) {
                        sb.append("  - ").append(pizzaNev).append("\n");
                        break;
                    }
                }
            }
        }
    }


     // LEKÉRDEZÉS : Listázd ki a vevő nevét, aki magyaros pizzát rendel.

    private static void query5_KiRendeltMagyarosPizzat(Document doc, StringBuilder sb) {
        sb.append("\n 5. LEKÉRDEZÉS: Ki rendelt magyaros pizzát?\n");

        // MAgyaros pizzaID
        String magyarosPizzaID = null;
```

```
        NodeList pizzaList = doc.getElementsByTagName("Pizza");
        for (int i = 0; i < pizzaList.getLength(); i++) {
            Element pizza = (Element) pizzaList.item(i);
            if ("Magyaros".equals(getElementText(pizza, "Pizza_neve"))) {
                magyarosPizzaID = pizza.getAttribute("PizzaID");
                break;
            }
        }

        if (magyarosPizzaID == null) {
            sb.append("  (Nem található magyaros pizza az adatbázisban.)\n");
            return;
        }
        sb.append("  (A magyaros pizza ID-ja: ").append(magyarosPizzaID).append(")\n");

        // VEvoID aki rendelt pizazat
        Set<String> vevoIDk = new HashSet<>();
        NodeList rendelesList = doc.getElementsByTagName("Rendeles");
        for (int i = 0; i < rendelesList.getLength(); i++) {
            Element rendeles = (Element) rendelesList.item(i);
            if (magyarosPizzaID.equals(rendeles.getAttribute("PizzaID"))) {
                vevoIDk.add(rendeles.getAttribute("VevoID"));
            }
        }

        if (vevoIDk.isEmpty()) {
            sb.append("  (Senki nem rendelt magyaros pizzát.)\n");
            return;
        }

        //VevoIDhez tartozo nev
        NodeList vevoList = doc.getElementsByTagName("Vevo");
        for (int i = 0; i < vevoList.getLength(); i++) {
            Element vevo = (Element) vevoList.item(i);
            if (vevoIDk.contains(vevo.getAttribute("VevoID"))) {
                sb.append("  - ").append(getElementText(vevo, "Nev")).append("\n");
            }
        }
    }


    // SEGÉDFÜGGVÉNY

    //Adott nevu gyerekelem tartalma
    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 "";
    }
}
```

2.4 Adat módosítás:

A D1H8VPDomModify osztály feladata a betöltött XML adatbázis tartalmának programozott módosítása. A kód demonstrálja, hogyan lehet a memóriában lévő DOM fán változtatásokat végezni majd az eredményt perzisztens módon elmenteni.

Megvalósított műveletek: A program öt különböző típusú manipulációt hajt végre az adatállományon, lefedve a DOM kezelés legfontosabb funkcióit:

1. **Tartalom frissítése** A "PIZ1" azonosítójú pizza árát módosítja egy új értékre (2990).
2. **Attribútum módosítása** Az "F1" azonosítójú futár PizzazoID attribútumát írja át, ezzel szimulálva a futár áthelyezését egy másik pizzázóba.
3. **Új elem beszúrása** A "V1" azonosítójú vevőhöz egy teljesen új, a sémában nem definiált <Email> gyerekelemet ad hozzá, bővítve a vevő adatait.
4. **Lista bővítése:** A "PIZ1" pizza <Feltetek> listájához ad hozzá egy új <Feltet> elemet ("Gomba"), demonstrálva a meglévő struktúra bővítését.
5. **Elem törlése** A "V2" vevőhöz tartozó Bankkartya elemet távolítja el a fából a szülő csomóponton keresztül.

```
package D1H8VP.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 D1H8VPDomModify {

    public static void main(String[] args) {
        File inputFile = new File("XML_FÉLÉVES_D1H8VP/Etterem_D1H8VP.xml");
        File outputFile = new File("Etterem_D1H8VP_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("DOM MÓDOSÍTÁSOK D1H8VP\n");



            modify1_ArModositas(doc, report);
            modify2_FutarAthelyezes(doc, report);
            modify3_UjElemHozzaadasa(doc, report);
            modify4_UjListaelem(doc, report);
            modify5_ElemTorlese(doc, report);

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

            System.out.println("\nA módosított XML mentve ide: " + outputFile.getName() +
"'");

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


     // Átírja a PIZ1 pizza árát 2990-re.

    private static void modify1_ArModositas(Document doc, StringBuilder report) {
        report.append("\n 1. MÓDOSÍTÁS: PIZ1 pizza árának frissítése \n");
        NodeList pizzaList = doc.getElementsByTagName("Pizza");
        for (int i = 0; i < pizzaList.getLength(); i++) {
            Element pizza = (Element) pizzaList.item(i);
            if ("PIZ1".equals(pizza.getAttribute("PizzaID"))) {
                Node arNode = pizza.getElementsByTagName("Teljes_ar").item(0);
                String regiAr = arNode.getTextContent();
                arNode.setTextContent("2990");
                report.append(" PIZ1 (Margarita) ára módosítva.\n");
                report.append("Régi ár:").append(regiAr).append(", Új ár: 2990\n");
```

```
                return;
            }
        }
        report.append("  - HIBA: PIZ1 pizza nem található.\n");
    }


     // Áthelyezi F1" futárt a P1 pizzázótól a P2-be

    private static void modify2_FutarAthelyezes(Document doc, StringBuilder report) {
        report.append("\n2. MÓDOSÍTÁS: F1 futár masik pizzazohoz \n");
        NodeList futarList = doc.getElementsByTagName("Futar");
        for (int i = 0; i < futarList.getLength(); i++) {
            Element futar = (Element) futarList.item(i);
            if ("F1".equals(futar.getAttribute("FutarID"))) {
                String regiPizzazo = futar.getAttribute("PizzazoID");
                futar.setAttribute("PizzazoID", "P2");
                report.append(" SIKER: F1 (James Bond) futár áthelyezve.\n");
                report.append("Régi PizzazoID: ").append(regiPizzazo).append(", Új
PizzazoID: P2\n");
                return;
            }
        }
        report.append(" HIBA: F1 futár nem található.\n");
    }


     // Hozzáadunk egy <Email> elemet V1 vevőhöz.

    private static void modify3_UjElemHozzaadasa(Document doc, StringBuilder report) {
        report.append("\n3. MÓDOSÍTÁS: Email cím hozzáadása V1 vevőhöz \n");
        NodeList vevoList = doc.getElementsByTagName("Vevo");
        for (int i = 0; i < vevoList.getLength(); i++) {
            Element vevo = (Element) vevoList.item(i);
            if ("V1".equals(vevo.getAttribute("VevoID"))) {
                Element email = doc.createElement("Email");
                email.setTextContent("vasar.lo@email.com");
                vevo.appendChild(email);
                report.append("SIKER: '<Email>vasar.lo@email.com</Email>' hozzáadva V1
vevőhöz.\n");
                return;
            }
        }
        report.append(" HIBA: V1 vevő nem található.\n");
    }

    //gomba feletet hozzaadasa a pizzahoz
    private static void modify4_UjListaelem(Document doc, StringBuilder report) {
        report.append("\n4. MÓDOSÍTÁS: Új feltét  hozzáadása a PIZ1 pizzához \n");
        NodeList pizzaList = doc.getElementsByTagName("Pizza");
        for (int i = 0; i < pizzaList.getLength(); i++) {
            Element pizza = (Element) pizzaList.item(i);
            if ("PIZ1".equals(pizza.getAttribute("PizzaID"))) {

                Node feltetekNode = pizza.getElementsByTagName("Feltetek").item(0);
```

```
                Element ujFeltet = doc.createElement("Feltet");
                ujFeltet.setTextContent("Gomba");

                feltetekNode.appendChild(ujFeltet);
                report.append("SIKER: '<Feltet>Gomba</Feltet>' hozzáadva PIZ1
pizzához.\n");
                return;
            }
        }
        report.append(" HIBA: PIZ1 pizza nem található.\n");
    }

    //v2 kartyaszam torlese
    private static void modify5_ElemTorlese(Document doc, StringBuilder report) {
        report.append("\n 5. MÓDOSÍTÁS: 'V2' vevő bankkártyájának törlése \n");
        NodeList kartyaList = doc.getElementsByTagName("Bankkartya");


        for (int i = kartyaList.getLength() - 1; i >= 0; i--) {
            Element kartya = (Element) kartyaList.item(i);
            if ("V2".equals(kartya.getAttribute("VevoID"))) {
                String toroltKartyaSzam = kartya.getAttribute("Kartyaszam");
                Node szulo = kartya.getParentNode();
                szulo.removeChild(kartya);

                report.append("SIKER: V2 vevő (Kartyaszam:
...").append(toroltKartyaSzam.substring(toroltKartyaSzam.length() - 4)).append(")
bankkártyája törölve.\n");
                return;
            }
        }
        report.append(" HIBA: V2 vevőhöz tartozó bankkártya nem található.\n");
    }

    //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

Ábra tartalomjegyzék:

Kódrész Tartalomjegyzék: