JEGYZŐKÖNYV

Webes adatkezelő környezetek

Féléves feladat

Laptop/telefon szervíz

Készítette: **Bodó Tamás**
Neptunkód: **HDK6NX**

Dátum: **2025. december**

Miskolc, 2025

Tartalomjegyzék

Bevezetés

A feladat célja az XML technológiák alkalmazásának gyakorlati bemutatása egy valós, jól strukturált adatmodellen keresztül. A beadandó témája egy laptop- és telefon szerviz adatainak leírása, modellezése és feldolgozása. A projekt két nagy részre tagolódik: az első részben az adatok modellezése és az XML, illetve XSD állományok elkészítése történik, míg a második részben ezeknek a fájloknak a feldolgozása valósul meg Java nyelven, DOM technológiával. A feladat célja, hogy a hallgató elsajátítsa az XML dokumentumok szerkezetének megtervezését, validálását, és ezek programozott feldolgozását. A projekt során létrejött fájlok (ER modell, XDM modell, XML dokumentum, XSD séma, DOM feldolgozó Java osztályok) együttesen mutatják be az XML technológia teljes életciklusát, az adattervezéstől egészen a feldolgozásig.

A feladat leírása:

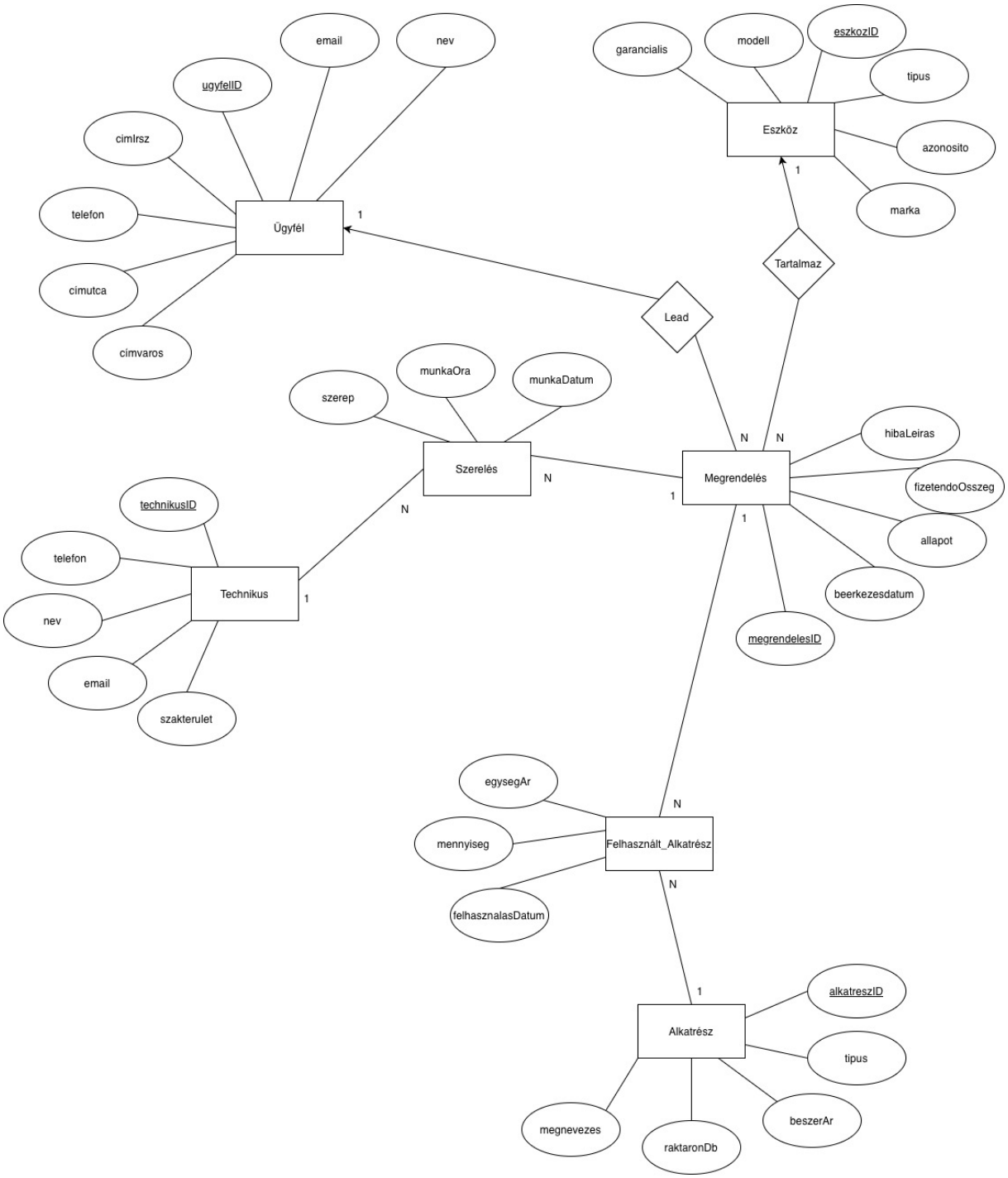
A projekt egy laptop- és telefon szerviz működésének adatkezelését modellezi. A szerviz ügyfelei különböző eszközöket (telefonokat, laptopokat) hoznak be javításra, amelyekhez megrendelések tartoznak. Egy megrendelés több szerelést és több felhasznált alkatrészt is tartalmazhat. A szerelést technikus végzi, a felhasznált alkatrészek pedig raktári adatokhoz kapcsolódnak.
A feladat célja, hogy ezt az adatstruktúrát először adatmodell szinten (ER és XDM diagram), majd XML és XSD formában, végül pedig Java program segítségével is megvalósítsuk.
Az első feladat során létrejött:
– az ER modell, amely az adatok kapcsolatát ábrázolja,
– az XDM modell, amely az XML hierarchiát mutatja,
– az XML dokumentum, amely konkrét adatokat tartalmaz,
– az XSD séma, amely validálja az XML felépítését.
A második feladat során a Java DOM feldolgozással három program készült:
– HDK6NXDomRead.java: adatbeolvasás és kiíratás,
– HDK6NXDomQuery.java: adatlekérdezések,
– HDK6NXDomModify.java: adatmódosítás és új elem beszúrása.

1. Laptop/telefon szervíz

XML és XSD dokumentum készítése laptop-telefon szerviz adatai alapján.

1.1 Az adatbázis ER modell tervezése

A tervezés során a szerviz működését modelleztem, ahol az ügyfél több eszközt hozhat, az eszközöknek több megrendelésük lehet, és egy megrendeléshez több szerelés vagy alkatrész is kapcsolódhat.
Az ER modell legalább öt entitást tartalmaz: Ügyfél, Eszköz, Megrendelés, Technikus és Alkatrész.
A kapcsolatok típusa vegyes:
– Ügyfél – Eszköz között 1:N kapcsolat,
– Eszköz – Megrendelés között 1:N kapcsolat,
– Megrendelés – Szerelés között 1:N kapcsolat,
– Megrendelés – Alkatrész között M:N kapcsolat,
– Szerelés – Technikus között 1:1 kapcsolat.
A modell tartalmaz kulcs, összetett és többértékű tulajdonságokat is.
A modell draw.io programban készült, és HDK6NX_ER.jpg néven lett mentve.



1. ábra: A laptop/telefon szervíz ER modellje.

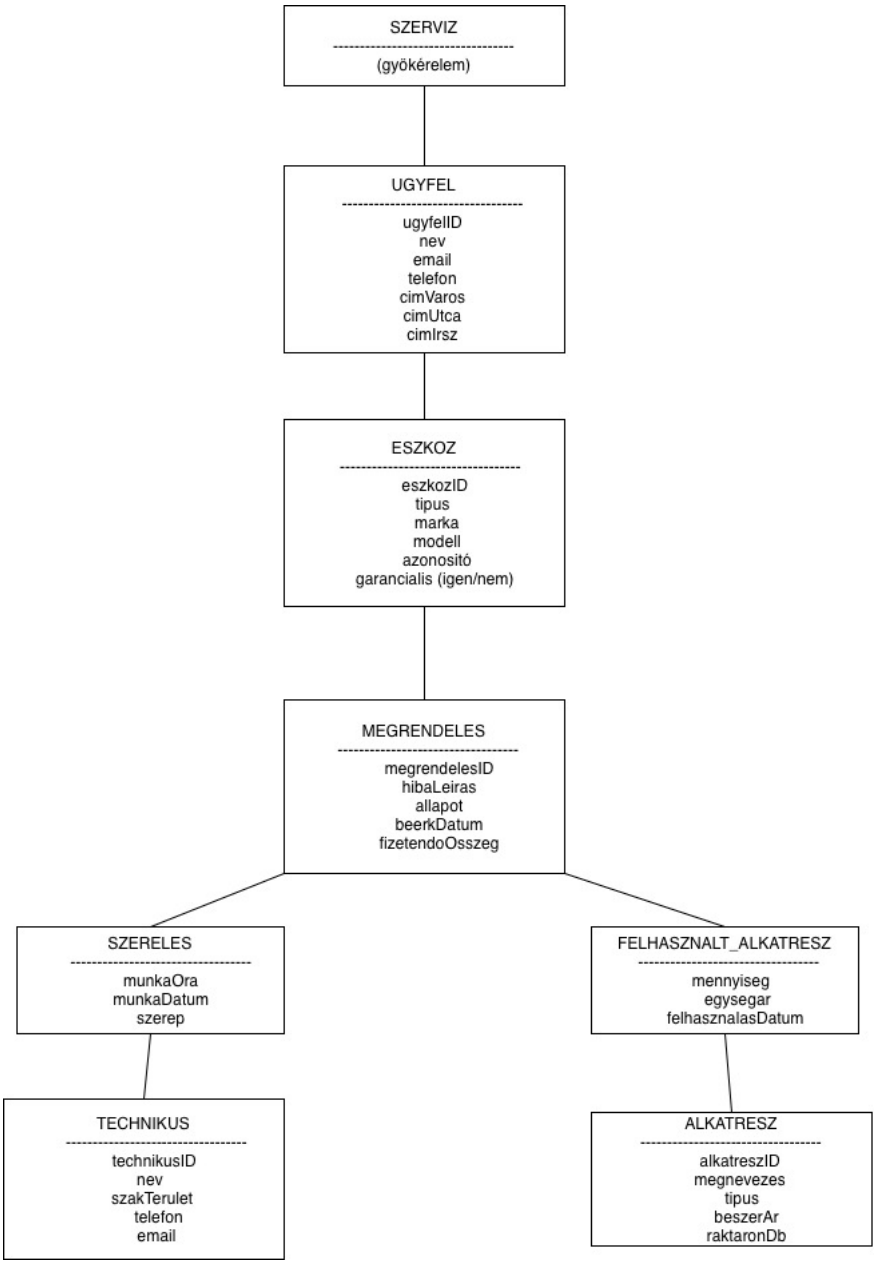
1.2 Az adatbázis konvertálása XDM modellre

Az XDM modell a korábban elkészített, az **1. ábrán** látható ER modell XML alapú leképezése, amely hierarchikusan mutatja be az adatokat.
A gyökérelem a *szerviz*, ezen belül helyezkednek el az ügyfelek, minden ügyfélhez több eszköz tartozhat, az eszközökhöz megrendelések, a megrendelésekhez pedig szerelések és felhasznált

alkatrészek kapcsolódnak.

Az XDM modell az ER modellben megtervezett kapcsolatok hierarchikus formában való megjelenítésére szolgál, az elemek közötti összefüggések és az attribútumok megőrzésével. A modell egyértelműen ábrázolja az elemek közötti kapcsolatokat és az adatöröklődési viszonyokat is, ezzel biztosítva az XML dokumentum szerkezeti alapját.

Az XDM modell draw.io programmal készült, és **HDK6NX_XDM.jpg** néven került elmentésre.



2. ábra: A laptop/telefon szervíz XDM modellje.

1.3 Az XDM modell alapján XML dokumentum készítése

A **2. ábrán** bemutatott XDM modell struktúrája alapján készült el az XML dokumentum, amely a szerviz adatait tartalmazza.
A dokumentum gyökéreleme a *szerviz*, amely magában foglalja az ügyfeleket, az eszközöket, a megrendeléseket, valamint a szerelésekhez és alkatrészekhez kapcsolódó adatokat.
Az XML dokumentumban két ügyfél (Kiss Péter és Nagy Anna) szerepel, mindkettőhöz több eszköz és megrendelés tartozik, az XDM modellen megjelenített hierarchiának megfelelően.
A dokumentum az XDM modellhez igazodva tartalmazza az attribútumokat (ugyfelID, eszkozID, megrendelesID, technikusID, alkatreszID), amelyek az egyes elemek azonosítására szolgálnak.
Az XML-ben minden többszörös előfordulású elem (például eszköz, megrendelés, alkatrész) több példányban is szerepel, biztosítva ezzel a modellben meghatározott adatmennyiség és kapcsolatok megjelenítését.
A fájl neve: **HDK6NX_XML.xml**.

1.4 Az XML dokumentum alapján XMLSchema készítése

Az XSD séma biztosítja az XML dokumentum formai helyességét és szerkezeti szabályait.
A sémában több saját típus is létrejött, például GarancialisTipus, TelefonTipus, DatumTipus.
A komplex típusok (CimTipus, UgyfelTipus, EszkozTipus, MegrendelesTipus, TechnikusTipus, AlkatreszTipus stb.) az egyes XML elemek struktúráját írják le.
A séma tartalmaz ref, key és keyref elemeket is, amelyek az idegen kulcs kapcsolatokat jelölik.
A séma validálás sikeres volt, az XML dokumentum megfelel a megadott szabályoknak.
A fájl neve: HDK6NX_XSD.xsd.

2. A MÁSODIK feladat címe

DOM feldolgozás laptop-telefon szerviz XML dokumentumhoz
Project name: HDK6NXDOMParse
Package: hdk6nx.domparse.hu
Class names: HDK6NXDomRead, HDK6NXDomQuery, HDK6NXDomModify

2.1 Adatolvasás

A HDK6NXDomRead.java fájl beolvassa az XML dokumentumot és a teljes tartalmát blokk formában kiírja a konzolra.
A program a DocumentBuilderFactory és DocumentBuilder osztályokat használja, a feldolgozás DOM alapú.
A kód bejárja az összes ügyfelet, eszközt, megrendelést, szerelést és alkatrészt, és minden adatot kiír hierarchikusan, tagolva.
A beolvasás során a getTextContent segédfüggvény használatával történik a szöveges adatok biztonságos kinyerése.
A futás eredményeként a konzolon a teljes XML tartalma megjelenik strukturált formában.

Az alábbi kódpéldában látható az ügyfél elemek kigyűjtése:

```
// 5. összes <ugyfel> elem kigyűjtése
NodeList ugyfelLista = doc.getElementsByTagName("ugyfel");

for (int i = 0; i < ugyfelLista.getLength(); i++)
{
    Node ugyfelNode = ugyfelLista.item(i);

    if (ugyfelNode.getNodeType() == Node.ELEMENT_NODE)
    {
        Element ugyfelElem = (Element) ugyfelNode;

        // ügyfél attribútum
        String ugyfelID = ugyfelElem.getAttribute("ugyfelID");
        System.out.println("ÜGYFÉL ID: " + ugyfelID);

        // sima gyermek-elemek
        String nev = getTextContent(ugyfelElem, "nev");
        String email = getTextContent(ugyfelElem, "email");
        String telefon = getTextContent(ugyfelElem, "telefon");

        System.out.println("  Név: " + nev);
        System.out.println("  Email: " + email);
        System.out.println("  Telefon: " + telefon);

        // cím kiolvasása (al-elemek)
        Element cimElem = (Element)
ugyfelElem.getElementsByTagName("cim").item(0);
        if (cimElem != null)
        {
            String varos = getTextContent(cimElem, "cimVaros");
            String utca = getTextContent(cimElem, "cimUtca");
            String irsz = getTextContent(cimElem, "cimIrsz");
            System.out.println("  Cím: " + varos + ", " + utca + " (" + irsz +
")");
        }
```

2.2. Adat-lekérdezés

A HDK6NXDomQuery.java fájl feladata az XML dokumentumban lévő adatok lekérdezése feltételek alapján.
A program három lekérdezést hajt végre:
– az összes ügyfél neve és azonosítója,
– a „javítás alatt” és „befogadva” állapotú megrendelések listázása,
– az összes technikus adatainak megjelenítése.
A kód DOM metódusokkal dolgozik, xPath kifejezéseket nem használ.
A futás eredményeként a konzolon megjelennek a kiválasztott adatok rendezett formában.

2.3. Adatmódosítás

A HDK6NXDomModify.java fájl példát mutat a DOM dokumentum módosítására és bővítésére.
A program két műveletet végez:
– módosítja a megrendelésID=M002 elem állapotát „javítás alatt”-ra,
– új megrendelést szúr be az U002 ügyfél E003 eszközéhez M004 azonosítóval.
Az új elem tartalmazza a hiba leírását, az állapotot, a dátumot és a fizetendő összeget.
A módosított DOM fájl HDK6NX_XML_updated.xml néven kerül mentésre, UTF-8 formátumban, behúzásokkal formázva.
A program sikeresen lefutott, és a létrejött XML fájlban megjelenik a módosítás és az új megrendelés is.

Az alábbi kód a dokumentumot egy új XML fájlba írja ki:

```
/**
 * Dokumentum kiírása fájlba
 */
private static void kiirXmlFajlba(Document doc, String fajlNev) throws
TransformerException
{
    TransformerFactory tf = TransformerFactory.newInstance();
    Transformer transformer = tf.newTransformer();

    // szebb, behúzott xml
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");
    transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount",
"2");
    transformer.setOutputProperty(OutputKeys.METHOD, "xml");
    transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
```

```
    DOMSource source = new DOMSource(doc);
    StreamResult result = new StreamResult(new File(fajlNev));
    transformer.transform(source, result);
}
```

Gyakorlatvezető: Dr. Bednarik László