JEGYZŐKÖNYV

Modern adatbázis rendszerek

Féléves feladatok: Állatkerthálózat + Kupon validator

Készítette: **Martinák Mátyás**
Neptunkód: **KLNSPG**
Dátum: **2025. 03. 22.**

Miskolc, 2025

Tartalomjegyzék

1. fejezet

Állatkerthálózat

A feladat egy vagy több állatkert hálózatát mutatja be, amiben helyet kapnak az egyes állatkertekben dolgozók, azok feladatai, az állatok és élőhelyeik, eledelük, az eledelt gyártó cégek, illetve az állatok örökbefogadói, ha vannak. Mind az ER modell tervezésben és mind az XML megvalósításban angol nyelvet használtam, ugyanis ez a legelterjedtebb nyelv a programozásban
Összesen 6 egyedet hoztam létre, melyek a következők:

- Employee,
- Site,
- Habitat,
- Animal,
- Food,
- User

Legelőször is érdemes pár szót szólni a **Site** egyedről. Innen indul ki minden. Ez az egyed tárolja el az egyes állatkertek legfőbb tulajdonságait, mint pl. név, terület vagy éppen nyitva tartás. Elsődleges kulcsa a `site_id`, ami az állatpark azonosítója.

A Site és az **Employee** egyed között egy `1:N` kapcsolat van, mivel egy állatkerthez több dolgozó is tartozhat, de egy dolgozó, csak egy állatkerthez tartozhat. Az `1:N` kapcsolat neve: **Works**. Egy dolgozónak van azonosítója, vezeték és keresztneve (ami ER modellben egy többágú tulajdonság), neme, születési dátuma és ami a legfontosabb, a dolgozó feladatai, posztjai, amiből lehet egy vagy több, így ez egy többértékű tulajdonság lesz. Ez azért fontos, mivel a relációs modellnél ez a tulajdonság egy külön táblát kap majd, amiben lesz a posztnak egy id-ja, a poszt neve, illetve, hogy kihez tartozik.

Egy állatkerthez több élőhely is tartoztat, de egy élőhely csak egy állatkerthez tartozik. Ezt ábrázolja a **Manage** kapcsolat, ami `1:N` kapcsolattal köti össze a Site és a **Habitat** egyedeket. Az

élőhelynek nincsenek „extra” tulajdonságai, van egy azonosítója, neve, térképen való elhelyezkedése, leírása és kapacitása, hogy mennyi állatot képes egyszerre befogadni.

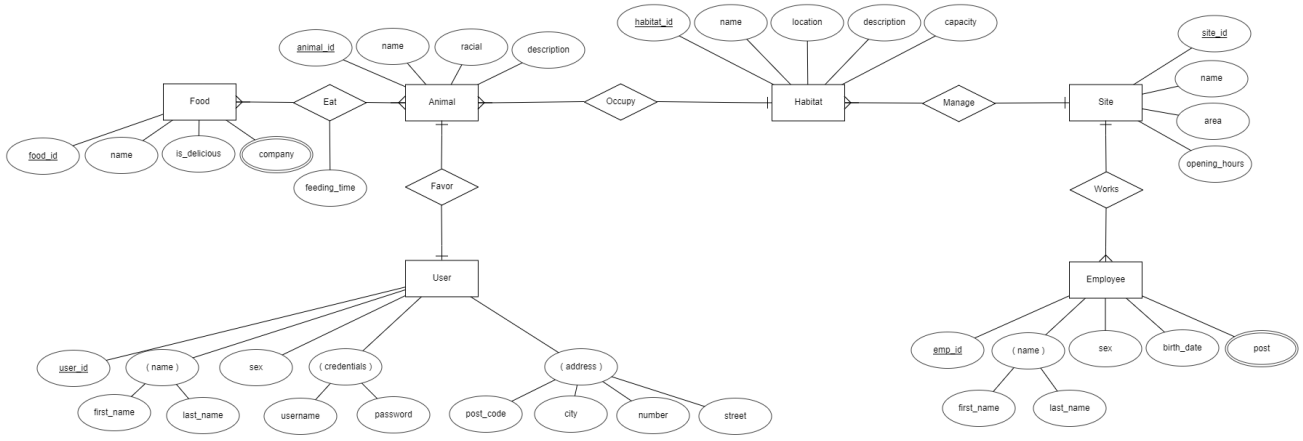
Az **Occupy** kapcsolat szintén `1:N` kapcsolattal köti össze a Habitat-ot az **Animal**-lel. Az állatnak van azonosítója, neve, faja és leírása.

Itt jön a legelső `N:M` kapcsolat, az **Eat**, aminek lesz tulajdonsága, a `feeding_time`, az etetési idő. Fontos, hogy megjegyezzük, az `N:M` kapcsolat külön kapcsolótáblát fog kapni a relációs modellben. Az Eat köti össze az Animalt a **Food**-dal, ami az állat eledelét modellező egyed. Ennek van azonosítója, neve, egy `boolean` (logikai) értéke, ami azt dönti el, hogy finom-e az adott eledel, vagy sem. Ezen kívül van egy többértékű tulajdonsága is, az eledeleket gyártó cégek, amik szintén külön táblát fognak majd kapni a relációs modellben.

Az állatokat örökbe is lehet fogani bizonyos **User**-eknek, ezt a **Favor** `1:1` kapcsolat modellezi. Talán a Usernek van a legtöbb tulajdonsága ebben az adatbázisban. Van természetesen azonosítója, két neve (vezeték és keresztnév), neme, bejelentkezési adatai (felhasználónév, jelszó), mivel online szeretnénk lebonyolítani az állatok örökbefogadását. Ezen kívül címe is van a felhasználónak, ami az irányítószám, város, utca, házszám tulajdonságokból tevődik össze.

1.1. XML/XSD létrehozás

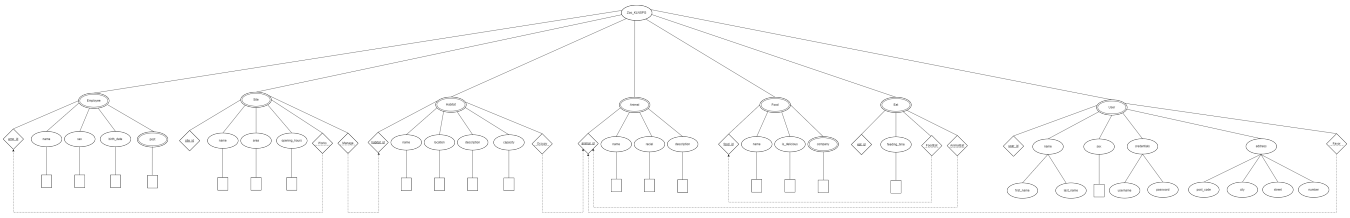
1.1.1. A feladat ER modellje



1.1. ábra. A feladat ER modellje

1.1.2. A feladat XDM modellje

A konvertáláskor figyelembe kell venni az ER modell során definiált kapcsolatokat, azok típusait (`1:1, 1:N, N:M`), illetve az entitások elsődleges kulcsait is. Minden *egy-több* kapcsolat esetében ahhoz az elsődleges kulcshoz kerül a szaggatott nyíl, ahol az ER modellben a „több" szerepel. Az „Eat" és a „Favor" kapcsolatokat kivéve, mindenhol `1:N` kapcsolat szerepel az ER modellben, így az XDM mindenhol majdnem hasonlóan fog kinézni. Az `N:M` kapcsolat esetében egy új modellt veszünk fel, tulajdonsággal és *primary key*-el együtt természetesen, ahonnan a nyilakat a fő entitásokhoz húzzuk. A többágú tulajdonságok itt is több tulajdonsággal rendelkeznek, a többértékű tulajdonságok itt nem kapnak külön modellt. Az XDM modell gyökéreleme: **Zoo_KLNSPG**



1.2. ábra. A feladat XDM modellje

A „Works" kapcsolat `1:N`, ahol a több érték az *Employee*-hoz kerül, így a kapcsolatot is az *emp_id*-hez húzzuk. Szintén ugyan ez a helyzet a „Manage" kapcsolatnál is, ahol a rombuszt a *habitat_id*-hoz húzzuk. Az *Animal* modell egy különleges, ide 3 kapcsolatot is húzunk, melyek:

- „Manage" a *Habitat*-ból
- „AnimalEat" az *Eat*-ből és végül
- „Favor" a *User*-ből

Kettő `1:N` kapcsolat és egy `1:1` kapcsolat húz ide.

1.1.3. Az XDM modell alapján XML dokumentum készítése

Az `XMLKLNSPG.xml` dokumentumot *Visual Studio Code*-ban hoztam létre, és `XML 1.0` szabvány szerint készült el. A dokumentumhoz hozzá kötöttem az `XMLSchemaKLNSPG.xsd` XSD file-t, és definiáltam az egyedeket az XML szabályainak megfelelően. Ahol szükséges volt, gyermek elemeket, valamint attribútumokat használtam a tagok azonosításához.

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

<Zoo_KLNSPG xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="XMLSchemaKLNSPG.xsd">
```

```
<!-- Employee peldanyok-->
<Employee emp_id="1">
  <first_name>Kovacs</first_name>
  <last_name>Janos</last_name>
  <birth_date>1979-11-02</birth_date>
  <sex>M</sex>
  <posts>
    <post>Kisallat gondozo</post>
    <post>Eteto</post>
    <post>Szemetszedo</post>
  </posts>
</Employee>
<Employee emp_id="2">
  <first_name>Jakab</first_name>
  <last_name>Jozsef</last_name>
  <birth_date>1954-12-06</birth_date>
  <sex>M</sex>
  <posts>
    <post>Terrarium takarito</post>
    <post>Gondnok</post>
    <post>Jegyszedo</post>
  </posts>
</Employee>
<Employee emp_id="3">
  <first_name>Balogh</first_name>
  <last_name>Boglarka</last_name>
  <birth_date>2000-11-04</birth_date>
  <sex>F</sex>
  <posts>
    <post>Gondozo</post>
    <post>Eteto</post>
    <post>Jegyszedo</post>
  </posts>
</Employee>

<!-- Site peldanyok-->
<Site site_id="1" Works="3" Manage="1">
  <name>Miskolc Allatkert</name>
  <area>212000</area>
  <opening_hours>09:00 - 17:00</opening_hours>
</Site>
<Site site_id="2" Works="1" Manage="2">
  <name>Fovarosi Allat-es Novenykert</name>
  <area>184001</area>
  <opening_hours>09:00 - 17:30</opening_hours>
</Site>
<Site site_id="3" Works="2" Manage="3">
```

```
  <name>Debreceni Allatkert es Vidampark</name>
  <area>170000</area>
  <opening_hours>09:00 - 15:30</opening_hours>
</Site>

<!-- Habitat peldanyok-->
<Habitat habitat_id="1" Occupy="3">
  <name>Medve park</name>
  <location>#3</location>
  <description>Az allatkerti medvek elohelye. Jelenleg harom medve talalhato
 itt, Jazmin, Andor es Matyko. Szeretik a latogatokat, mindig erdeklodve
 nezelodnek.</description>
</Habitat>
<Habitat habitat_id="2" Occupy="1">
  <name>Muflonok dombja</name>
  <location>#22</location>
  <description>Vadasparkunk muflonjai itt talalhatoak. Baratsagosak, turista
 kedvelok, szeretik a finom falatokat.</description>
</Habitat>
<Habitat habitat_id="3" Occupy="2">
  <name>Szurikatak szigete</name>
  <location>#18</location>
  <description>Ki ne imadna a kis erdeklodo szurikatakat. Nalunk rogton 4-et
 is orokbe fogadhat, vagy csak latogathat is.</description>
</Habitat>

<!-- Animal peldanyok-->
<Animal animal_id="1">
  <name>Matyko</name>
  <racial>Medve</racial>
  <description>Az allatkert egyik kan medveje</description>
</Animal>
<Animal animal_id="2">
  <name>Kis Hegyes</name>
  <racial>Muflon</racial>
  <description>Az allatkert nosteny muflona</description>
</Animal>
<Animal animal_id="3">
  <name>Mokas</name>
  <racial>Szurikata</racial>
  <description>Az allatkert legfiatalabb szurikataja</description>
</Animal>

<!-- Food peldanyok-->
<Food food_id="1">
  <name>Fagyasztott nyershus</name>
  <is_delicious>false</is_delicious>
  <companies>
```

```
      <company>Family Frost</company>
      <company>Magyar Zoldseg</company>
      <company>Eszkimo</company>
    </companies>
  </Food>
  <Food food_id="2">
    <name>Sargarepa</name>
    <is_delicious>true</is_delicious>
    <companies>
      <company>Allati zoldseg/gyumolcs</company>
      <company>Magyar Zoldseg</company>
      <company>Felix allati eledel</company>
    </companies>
  </Food>
    <Food food_id="3">
    <name>Sult husi</name>
    <is_delicious>true</is_delicious>
    <companies>
      <company>Frosty food</company>
      <company>Eszkimo</company>
      <company>Magyar zoldseg</company>
    </companies>
  </Food>

  <!-- Eat peldanyok-->
  <Eat eat_id="1" FoodEat="1" AnimalEat="3">
    <feeding_time>09:00:00 18:00:00</feeding_time>
  </Eat>
  <Eat eat_id="2" FoodEat="3" AnimalEat="1">
    <feeding_time>07:00:00 20:00:00</feeding_time>
  </Eat>
  <Eat eat_id="3" FoodEat="2" AnimalEat="2">
    <feeding_time>06:00:00 14:00:00 20:00:00</feeding_time>
  </Eat>

  <!-- User peldanyok-->
  <User user_id="1" Favor="3">
    <username>Allatbarat</username>
    <password>allat123</password>
    <sex>M</sex>
    <first_name>Kiss</first_name>
    <last_name>Sandor</last_name>
    <post_code>8200</post_code>
    <city>Veszprem</city>
    <street>Petofi Sandor utca</street>
    <number>3</number>
  </User>
  <User user_id="2" Favor="1">
```

```
    <username>Vadoc</username>
    <password>fegyo02</password>
    <sex>M</sex>
    <first_name>Fegyver</first_name>
    <last_name>Sandor</last_name>
    <post_code>4024</post_code>
    <city>Debrecen</city>
    <street>Kossuth utca</street>
    <number>26</number>
  </User>
    <User user_id="3" Favor="2">
    <username>Possumluvr</username>
    <password>possumlover</password>
    <sex>F</sex>
    <first_name>Kazai</first_name>
    <last_name>Eszter</last_name>
    <post_code>3521</post_code>
    <city>Miskolc</city>
    <street>Uj elet utca</street>
    <number>24</number>
  </User>

  </Zoo_KLNSPG>
```

Programkód 1.1. Az XML dokumentum

1.1.4. Az XML dokumentum alapján XMLSchema készítése

Az `XMLSchemaKLNSPG.xsd` séma file leírja mindazon megkötéseket, amelyeknek az XML dokumentumnak meg kell felelnie. Itt definiálunk minden típust, amit az XML file-ban használni szeretnénk, valamint az adatbázis kapcsolatait `xs:unique` és `xs:keyref` bejegyzésekkel hozom létre.

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

    <!-- Sajat egyszeru tipusok definialasa -->

    <!-- Altalanos sajat tipusok-->
    <xs:element name="name" type="xs:string"/>
    <xs:element name="first_name" type="xs:string"/>
    <xs:element name="last_name" type="xs:string"/>
    <xs:element name="sex" type="sexType"/>
    <xs:element name="description" type="xs:string"/>

    <!-- Employee sajat tipus-->
    <xs:element name="birth_date" type="dateType"/>
    <xs:element name="posts" type="postsType"/>

    <!-- Site sajat tipusok-->
    <xs:element name="area" type="xs:integer"/>
    <xs:element name="opening_hours" type="xs:string"/>

    <!-- Habitat sajat tipus-->
    <xs:element name="location" type="xs:string"/>

    <!-- Animal sajat tipus-->
    <xs:element name="racial" type="xs:string"/>

    <!-- Food sajat tipus-->
    <xs:element name="is_delicious" type="xs:boolean"/>
    <xs:element name="companies" type="companyType"/>

    <!-- Eat sajat tipus-->
    <xs:element name="feeding_time" type="timeListType"/>

    <!-- User sajat tipusok-->
    <xs:element name="username" type="xs:string"/>
    <xs:element name="password" type="xs:string"/>
    <xs:element name="post_code" type="xs:string"/>
    <xs:element name="city" type="xs:string"/>
    <xs:element name="street" type="xs:string"/>
    <xs:element name="number" type="xs:string"/>
```

```
<!-- Simple types-->
<xs:simpleType name="dateType">
  <xs:restriction base="xs:date">
    <xs:minInclusive value="1940-01-01"/>
    <xs:maxInclusive value="2000-12-31"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="sexType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="M"/>
    <xs:enumeration value="F"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="timeListType">
  <xs:list itemType="xs:time"/>
</xs:simpleType>

<xs:simpleType name="postAttributeType">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="companyAttributeType">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>

<!-- Complex types-->
<xs:complexType name="employeeType">
  <xs:sequence>
    <xs:element ref="first_name"/>
    <xs:element ref="last_name"/>
    <xs:element ref="birth_date"/>
    <xs:element ref="sex"/>
    <xs:element ref="posts"/>
  </xs:sequence>
  <xs:attribute name="emp_id" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="postsType">
  <xs:sequence>
    <xs:element name="post" type="xs:string" minOccurs="1" maxOccurs="3"/>
  </xs:sequence>
  <xs:attribute name="post" type="postAttributeType"/>
</xs:complexType>
```

```
<xs:complexType name="siteType">
  <xs:sequence>
    <xs:element ref="name"/>
    <xs:element ref="area"/>
    <xs:element ref="opening_hours"/>
  </xs:sequence>
  <xs:attribute name="site_id" type="xs:integer" use="required"/>
  <xs:attribute name="Works" type="xs:integer" use="required"/>
  <xs:attribute name="Manage" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="habitatType">
  <xs:sequence>
    <xs:element ref="name"/>
    <xs:element ref="location"/>
    <xs:element ref="description"/>
  </xs:sequence>
  <xs:attribute name="habitat_id" type="xs:integer" use="required"/>
  <xs:attribute name="Occupy" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="animalType">
  <xs:sequence>
    <xs:element ref="name"/>
    <xs:element ref="racial"/>
    <xs:element ref="description"/>
  </xs:sequence>
  <xs:attribute name="animal_id" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="foodType">
  <xs:sequence>
    <xs:element ref="name"/>
    <xs:element ref="is_delicious"/>
    <xs:element ref="companies"/>
  </xs:sequence>
  <xs:attribute name="food_id" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="companyType">
  <xs:sequence>
    <xs:element name="company" type="xs:string" minOccurs="1"
maxOccurs="3"/>
  </xs:sequence>
  <xs:attribute name="company" type="companyAttributeType"/>
</xs:complexType>

<xs:complexType name="eatType">
```

```
    <xs:sequence>
      <xs:element ref="feeding_time"/>
    </xs:sequence>
    <xs:attribute name="eat_id" type="xs:integer" use="required"/>
    <xs:attribute name="FoodEat" type="xs:integer" use="required"/>
    <xs:attribute name="AnimalEat" type="xs:integer" use="required"/>
  </xs:complexType>

  <xs:complexType name="userType">
    <xs:sequence>
      <xs:element ref="username"/>
      <xs:element ref="password"/>
      <xs:element ref="sex"/>
      <xs:element ref="first_name"/>
      <xs:element ref="last_name"/>
      <xs:element ref="post_code"/>
      <xs:element ref="city"/>
      <xs:element ref="street"/>
      <xs:element ref="number"/>
    </xs:sequence>
    <xs:attribute name="user_id" type="xs:integer" use="required"/>
    <xs:attribute name="Favor" type="xs:integer" use="required"/>
  </xs:complexType>

  <!-- A gyokerelem osszetett tipusa -->
  <xs:complexType name="zooType">
    <xs:sequence>
      <xs:element name="Employee" type="employeeType" minOccurs="3"
maxOccurs="unbounded"/>
      <xs:element name="Site" type="siteType" minOccurs="3"
maxOccurs="unbounded"/>
      <xs:element name="Habitat" type="habitatType" minOccurs="3"
maxOccurs="unbounded"/>
      <xs:element name="Animal" type="animalType" minOccurs="3"
maxOccurs="unbounded"/>
      <xs:element name="Food" type="foodType" minOccurs="3"
maxOccurs="unbounded"/>
      <xs:element name="Eat" type="eatType" minOccurs="3"
maxOccurs="unbounded"/>
      <xs:element name="User" type="userType" minOccurs="3"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- A gyokerelem definicioja -->
  <xs:element name="Zoo_KLNSPG" type="zooType">

  <!-- Elsodleges kulcsok -->
```

```
<xs:key name="EmployeeKey">
  <xs:selector xpath="Employee"/>
  <xs:field xpath="@emp_id"/>
</xs:key>

<xs:key name="SiteKey">
  <xs:selector xpath="Site"/>
  <xs:field xpath="@site_id"/>
</xs:key>

<xs:key name="HabitatKey">
  <xs:selector xpath="Habitat"/>
  <xs:field xpath="@habitat_id"/>
</xs:key>

<xs:key name="AnimalKey">
  <xs:selector xpath="Animal"/>
  <xs:field xpath="@animal_id"/>
</xs:key>

<xs:key name="FoodKey">
  <xs:selector xpath="Food"/>
  <xs:field xpath="@food_id"/>
</xs:key>

<xs:key name="EatKey">
  <xs:selector xpath="Eat"/>
  <xs:field xpath="@eat_id"/>
</xs:key>

<xs:key name="UserKey">
  <xs:selector xpath="User"/>
  <xs:field xpath="@user_id"/>
</xs:key>

<!-- Idegen kulcsok -->
<xs:keyref name="SiteWork" refer="EmployeeKey">
  <xs:selector xpath="Site"/>
  <xs:field xpath="@Works"/>
</xs:keyref>

<xs:keyref name="SiteManage" refer="HabitatKey">
  <xs:selector xpath="Site"/>
  <xs:field xpath="@Manage"/>
</xs:keyref>

<xs:keyref name="HabitatOccupy" refer="AnimalKey">
  <xs:selector xpath="Habitat"/>
```

```
      <xs:field xpath="@Occupy"/>
    </xs:keyref>

    <xs:keyref name="EatFood" refer="FoodKey">
      <xs:selector xpath="Eat"/>
      <xs:field xpath="@FoodEat"/>
    </xs:keyref>

    <xs:keyref name="EatAnimal" refer="AnimalKey">
      <xs:selector xpath="Eat"/>
      <xs:field xpath="@AnimalEat"/>
    </xs:keyref>

    <xs:keyref name="UserAnimal" refer="AnimalKey">
      <xs:selector xpath="User"/>
      <xs:field xpath="@Favor"/>
    </xs:keyref>

    <!-- Az 1:1 kapcsolat-->
    <xs:unique name="UserAnimalConnect">
      <xs:selector xpath="UserKey"/>
      <xs:field xpath="@Favor"/>
    </xs:unique>
  </xs:element>
</xs:schema>
```

Programkód 1.2. Az XSD dokumentum

1.2. DOM

1.2.1. Adatolvasás

A kód egy Java alapú XML feldolgozó program, amely a DOM (Document Object Model) parserét használja. A DOM parser a teljes XML dokumentumot memóriába tölti, ami gyors hozzáférést biztosít az elemekhez, de nagyobb dokumentumok esetén jelentős memóriaigényt jelenthet. A program beolvassa az XML fájlt, normalizálja azt, és különböző függvények segítségével feldolgozza az XML elemeket, melyek az 'Employees', 'Sites', 'Habitats'. Minden elemcsoport feldolgozása külön függvényben történik, ami javítja a kód olvashatóságát és karbantarthatóságát. Hibakezelés is implementálva van a fájlbeolvasás és parse-lás során.

```
import javax.xml.parsers.*;
import org.xml.sax.SAXException;
import org.w3c.dom.*;
import java.io.*;
```

```
public class DOMReadKLNSPG
{
  // Main metodus
  public static void main(String[] args)
  {
    try
    {
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      DocumentBuilder builder = factory.newDocumentBuilder();
      Document document = builder.parse(new
 File("C:\\projects\\KLNSPG_XMLGyak\\XMLTaskKLNSPG\\XMLKLNSPG.xml"));

      document.getDocumentElement().normalize();
      System.out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
      System.out.println("<Zoo_KLNSPG
 xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
 xsi:noNamespaceSchemaLocation=\"XMLSchemaKLNSPG.xsd\">\n");

      readEmployees(document);
      readSites(document);
      readHabitats(document);
      readAnimals(document);
      readFoods(document);
      readEats(document);
      readUsers(document);

      System.out.println("\n</Zoo_KLNSPG>");
    }
    catch (ParserConfigurationException | IOException | SAXException e)
    {
      e.printStackTrace();
    }
  }

  // Employee Node beolvaso metodus
  private static void readEmployees(Document document)
  {
    NodeList employeeList = document.getElementsByTagName("Employee");
    for (int temp = 0; temp < employeeList.getLength(); temp++)
    {
      Node node = employeeList.item(temp);
      if (node.getNodeType() == Node.ELEMENT_NODE)
      {
        Element eElement = (Element) node;
        String empId = eElement.getAttribute("emp_id");
        String firstName =
 eElement.getElementsByTagName("first_name").item(0).getTextContent();
```

```
        String lastName =
eElement.getElementsByTagName("last_name").item(0).getTextContent();
        String birthDate =
eElement.getElementsByTagName("birth_date").item(0).getTextContent();
        String sex =
eElement.getElementsByTagName("sex").item(0).getTextContent();

        System.out.println("    <Employee emp_id=\"" + empId + "\">");
        printElement("first_name", firstName);
        printElement("last_name", lastName);
        printElement("birth_date", birthDate);
        printElement("sex", sex);

        // Tobberteku tulajdonsag lekezelese
        if (eElement.getElementsByTagName("posts").getLength() > 0) {
          NodeList posts =
eElement.getElementsByTagName("posts").item(0).getChildNodes();
          System.out.println("        <posts>");
          for (int i = 0; i < posts.getLength(); i++) {
            Node postNode = posts.item(i);
            if (postNode.getNodeType() == Node.ELEMENT_NODE) {
              Element postElement = (Element) postNode;
              System.out.println("            <post>" +
postElement.getTextContent() + "</post>");
            }
          }
          System.out.println("        </posts>");
        }

        System.out.println("    </Employee>");
      }
    }
  }

  // Site Node beolvaso metodus
  private static void readSites(Document document)
  {
    NodeList siteList = document.getElementsByTagName("Site");
    for (int temp = 0; temp < siteList.getLength(); temp++)
    {
      Node node = siteList.item(temp);
      if (node.getNodeType() == Node.ELEMENT_NODE)
      {
        Element eElement = (Element) node;
        String siteId = eElement.getAttribute("site_id");
        String works = eElement.getAttribute("Works");
        String manage = eElement.getAttribute("Manage");
```

```
        String name =
eElement.getElementsByTagName("name").item(0).getTextContent();
        String area =
eElement.getElementsByTagName("area").item(0).getTextContent();
        String openingHours =
eElement.getElementsByTagName("opening_hours").item(0).getTextContent();

        System.out.println("    <Site site_id=\"" + siteId + "\" Works=\"" +
works + "\" Manage=\"" + manage + "\">");
        printElement("name", name);
        printElement("area", area);
        printElement("opening_hours", openingHours);
        System.out.println("    </Site>");
      }
    }
  }

  // Habitat Node beolvaso metodus
  private static void readHabitats(Document document)
  {
    NodeList habitatList = document.getElementsByTagName("Habitat");
    for (int temp = 0; temp < habitatList.getLength(); temp++)
    {
      Node node = habitatList.item(temp);
      if (node.getNodeType() == Node.ELEMENT_NODE)
      {
        Element eElement = (Element) node;
        String habitatId = eElement.getAttribute("habitat_id");
        String occupy = eElement.getAttribute("Occupy");
        String name =
eElement.getElementsByTagName("name").item(0).getTextContent();
        String location =
eElement.getElementsByTagName("location").item(0).getTextContent();
        String description =
eElement.getElementsByTagName("description").item(0).getTextContent();

        System.out.println("    <Habitat habitat_id=\"" + habitatId + "\"
Occupy=\"" + occupy + "\">");
        printElement("name", name);
        printElement("location", location);
        printElement("description", description);
        System.out.println("    </Habitat>");
      }
    }
  }

  // Animal Node beolvaso metodus
  private static void readAnimals(Document document)
```

```
 {
   NodeList animalList = document.getElementsByTagName("Animal");
   for (int temp = 0; temp < animalList.getLength(); temp++)
   {
     Node node = animalList.item(temp);
     if (node.getNodeType() == Node.ELEMENT_NODE)
     {
       Element eElement = (Element) node;
       String animalId = eElement.getAttribute("animal_id");
       String name =
eElement.getElementsByTagName("name").item(0).getTextContent();
       String racial =
eElement.getElementsByTagName("racial").item(0).getTextContent();
       String description =
eElement.getElementsByTagName("description").item(0).getTextContent();

       System.out.println("    <Animal animal_id=\"" + animalId + "\">");
       printElement("name", name);
       printElement("racial", racial);
       printElement("description", description);
       System.out.println("    </Animal>");
     }
   }
 }

 // Food Node beolvaso metodus
 private static void readFoods(Document document)
 {
   NodeList foodList = document.getElementsByTagName("Food");
   for (int temp = 0; temp < foodList.getLength(); temp++)
   {
     Node node = foodList.item(temp);
     if (node.getNodeType() == Node.ELEMENT_NODE)
     {
       Element eElement = (Element) node;
       String foodId = eElement.getAttribute("food_id");
       String name =
eElement.getElementsByTagName("name").item(0).getTextContent();
       String isDelicious =
eElement.getElementsByTagName("is_delicious").item(0).getTextContent();

       System.out.println("    <Food food_id=\"" + foodId + "\">");
       printElement("name", name);
       printElement("is_delicious", isDelicious);

       // Tobberteku tulajdonsag lekezelese
       NodeList companiesNodeList =
eElement.getElementsByTagName("companies");
```

```
        if (companiesNodeList.getLength() > 0) {
          Node companiesNode = companiesNodeList.item(0);
          if (companiesNode.getNodeType() == Node.ELEMENT_NODE)
          {
            Element companiesElement = (Element) companiesNode;
            NodeList companyNodeList =
companiesElement.getElementsByTagName("company");
            System.out.println("        <companies>");
            for (int i = 0; i < companyNodeList.getLength(); i++)
            {
              Element companyElement = (Element) companyNodeList.item(i);
              String companyId = companyElement.getAttribute("id");
              System.out.println("            <company id=\"" + companyId +
"\">" + companyElement.getTextContent() + "</company>");
            }

            System.out.println("        </companies>");
          }
        }

        System.out.println("    </Food>");
      }
    }
  }

  // Eat Node beolvaso metodus
  private static void readEats(Document document)
  {
    NodeList eatList = document.getElementsByTagName("Eat");
    for (int temp = 0; temp < eatList.getLength(); temp++)
    {
      Node node = eatList.item(temp);
      if (node.getNodeType() == Node.ELEMENT_NODE)
      {
        Element eElement = (Element) node;
        String eatId = eElement.getAttribute("eat_id");
        String foodEat = eElement.getAttribute("FoodEat");
        String animalEat = eElement.getAttribute("AnimalEat");
        String feedingTime =
eElement.getElementsByTagName("feeding_time").item(0).getTextContent();

        System.out.println("    <Eat eat_id=\"" + eatId + "\" FoodEat=\"" +
foodEat + "\" AnimalEat=\"" + animalEat + "\">");
        printElement("feeding_time", feedingTime);
        System.out.println("    </Eat>");
      }
    }
  }
```

```
// User Node beolvaso metodus
private static void readUsers(Document document)
{
  NodeList userList = document.getElementsByTagName("User");
  for (int temp = 0; temp < userList.getLength(); temp++)
  {
    Node node = userList.item(temp);
    if (node.getNodeType() == Node.ELEMENT_NODE)
    {
      Element eElement = (Element) node;
      String userId = eElement.getAttribute("user_id");
      String favor = eElement.getAttribute("Favor");
      String username =
eElement.getElementsByTagName("username").item(0).getTextContent();
      String password =
eElement.getElementsByTagName("password").item(0).getTextContent();
      String sex =
eElement.getElementsByTagName("sex").item(0).getTextContent();
      String firstName =
eElement.getElementsByTagName("first_name").item(0).getTextContent();
      String lastName =
eElement.getElementsByTagName("last_name").item(0).getTextContent();
      String postCode =
eElement.getElementsByTagName("post_code").item(0).getTextContent();
      String city =
eElement.getElementsByTagName("city").item(0).getTextContent();
      String street =
eElement.getElementsByTagName("street").item(0).getTextContent();
      String number =
eElement.getElementsByTagName("number").item(0).getTextContent();

      System.out.println("    <User user_id=\"" + userId + "\" Favor=\"" +
favor + "\">");
      printElement("username", username);
      printElement("password", password);
      printElement("sex", sex);
      printElement("first_name", firstName);
      printElement("last_name", lastName);
      printElement("post_code", postCode);
      printElement("city", city);
      printElement("street", street);
      printElement("number", number);
      System.out.println("    </User>");
    }
  }
}
```

```
    // Elem kiirato metodus
    private static void printElement(String elementName, String content)
    {
      System.out.println("        <" + elementName + ">" + content + "</" +
   elementName + ">");
    }
}
```

Programkód 1.3. DOMReadKLNSPG.java adatolvasó program

1.2.2. Adatmódosítás

Ez a Java program, a DOMModifyKLNSPG, egy XML fájlt olvas be és módosítja azt a DOM (Document Object Model) API segítségével. Az XML fájl, amely egy állatkert adatait tartalmazza, egy előre meghatározott útvonalon található, és a `DocumentBuilder` osztály segítségével parse-oljuk. A program három fő részre oszlik: `modifyEmployees`, `modifySites`, és `modifyAnimals` metódusokra, melyek különböző XML elemeket módosítanak. Az `modifyEmployees` metódus az Employee elemek `emp_id` attribútumát módosítja, minden `emp_id` elé „`EMP_`” előtagot illesztve. A `modifySites` metódus a Site elemek `visitor_capacity` attribútumát állítja be „5000”-re, amely a látogatók maximális számát jelenti. A `modifyAnimals` metódus a *Medve* `racial` értékkel rendelkező Animal elemek `description` elemének szövegét módosítja **„A medve eros es bator”** szövegre. A módosítások után a program egy `Transformer` segítségével visszaalakítja és kiírja a módosított DOM-ot XML formátumban. Az XML kiírás során a `Transformer` beállításai biztosítják a formázott, olvasható kimenetet, az `OutputKeys.INDENT` beállítás segítségével.

```
import javax.xml.parsers.*;
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 org.w3c.dom.*;
import java.io.File;

public class DOMModifyKLNSPG
{
  // Main metodus
  public static void main(String argv[])
  {
    try
    {
      File inputFile = new
 File("C:\\projects\\KLNSPG_XMLGyak\\XMLTaskKLNSPG\\XMLKLNSPG.xml");
```

```
     DocumentBuilderFactory docFactory =
DocumentBuilderFactory.newInstance();
     DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
     Document doc = docBuilder.parse(inputFile);

     modifyEmployees(doc);
     modifySites(doc);
     modifyAnimals(doc);

     TransformerFactory transformerFactory =
TransformerFactory.newInstance();
     Transformer transformer = transformerFactory.newTransformer();
     transformer.setOutputProperty(OutputKeys.INDENT, "yes");
     DOMSource source = new DOMSource(doc);
     StreamResult consoleResult = new StreamResult(System.out);
     transformer.transform(source, consoleResult);
   }
   catch (Exception e)
   {
     e.printStackTrace();
   }
 }

 // Employee peldanyokat modosito metodus
 private static void modifyEmployees(Document doc)
 {
   NodeList employeeList = doc.getElementsByTagName("Employee");
   for (int i = 0; i < employeeList.getLength(); i++)
   {
     Node employee = employeeList.item(i);
     Element eElement = (Element) employee;
     String empId = eElement.getAttribute("emp_id");
     eElement.setAttribute("emp_id", "EMP_" + empId);
   }
 }

 // Site peldanyokat modosito metodus
 private static void modifySites(Document doc)
 {
   NodeList siteList = doc.getElementsByTagName("Site");
   for (int i = 0; i < siteList.getLength(); i++)
   {
     Node site = siteList.item(i);
     Element eElement = (Element) site;
     eElement.setAttribute("visitor_capacity", "5000");
   }
 }
```

```
    // Animal (Medve) peldanyt modosito metodus
    private static void modifyAnimals(Document doc)
    {
      NodeList animalList = doc.getElementsByTagName("Animal");
      for (int i = 0; i < animalList.getLength(); i++)
      {
        Node animal = animalList.item(i);
        Element eElement = (Element) animal;
        if
   ("Medve".equals(eElement.getElementsByTagName("racial").item(0).getTextContent()))
        eElement.getElementsByTagName("description").item(0).setTextContent("A
    medve eros es bator");
      }
     }
  }
```

Programkód 1.4. DOMQModifyKLNSPG.java adatmódosító program

1.2.3. Adatlekérdezés

A program a Java `DOM Parser`-t használja XML fájlunk feldolgozására, ami lehetővé teszi a XML elemek olvasását és manipulálását egy objektumorientált módon. A kód, az XML fájlt, a `File` objektumon keresztül tölti be, biztosítva ezzel a fájl elérését és kezelését. A `DocumentBuilderFactory` és `DocumentBuilder` osztályok használata a fájl DOM reprezentációjának létrehozásához szükséges, ami egy strukturált, fa-szerű modellt biztosít az XML adatok számára. A program minden egyes lekérdezést egy for ciklus segítségével hajt végre, ahol a `getElementsByTagName` metódus segítségével specifikus XML elemeket keres. Az elemek feldolgozása során a `Node` és `Element` interfészeket használja, amelyek lehetővé teszik az egyes elemek attribútumainak és tartalmának elérését. A lekérdezések eredményét egy `StringBuilder` objektumba gyűjti, amely hatékonyan kezeli a nagy mennyiségű stringek összefűzését. A kód, az összegyűjtött adatokat XML-szerű formátumban állítja elő.

```
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.SAXException;

public class DOMQueryKLNSPG
{
  // Main metodus
  public static void main(String argv[]) throws SAXException, IOException,
   ParserConfigurationException
  {
```

```
 File xmlFile = new
File("C:\\projects\\KLNSPG_XMLGyak\\XMLTaskKLNSPG\\XMLKLNSPG.xml");

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

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

 StringBuilder outputBuilder = new StringBuilder();

 // Lekerdezes a ferfi Employee-kre
 NodeList employeeList = doc.getElementsByTagName("Employee");
 outputBuilder.append("<Employees>\n");
 for (int i = 0; i < employeeList.getLength(); i++)
 {
   Node node = employeeList.item(i);
   if (node.getNodeType() == Node.ELEMENT_NODE)
   {
     Element element = (Element) node;
     String sex =
element.getElementsByTagName("sex").item(0).getTextContent();
     if (sex.equals("M"))
     {
       String empId = element.getAttribute("emp_id");
       String firstName =
element.getElementsByTagName("first_name").item(0).getTextContent();
       String lastName =
element.getElementsByTagName("last_name").item(0).getTextContent();
       String birthDate =
element.getElementsByTagName("birth_date").item(0).getTextContent();
       outputBuilder.append(String.format("  <Employee emp_id=\"%s\">\n",
empId));
       outputBuilder.append(String.format("
<first_name>%s</first_name>\n", firstName));
       outputBuilder.append(String.format("
<last_name>%s</last_name>\n", lastName));
       outputBuilder.append(String.format("
<birth_date>%s</birth_date>\n", birthDate));
       outputBuilder.append(String.format("    <sex>%s</sex>\n", sex));

       // Posts es azok elemeinek kezelese
       NodeList postsList = element.getElementsByTagName("posts");
       if (postsList.getLength() > 0) {
         Node postsNode = postsList.item(0);
         if (postsNode.getNodeType() == Node.ELEMENT_NODE)
         {
           outputBuilder.append("    <posts>\n");
```

```
            NodeList postList =
((Element)postsNode).getElementsByTagName("post");
            for (int j = 0; j < postList.getLength(); j++)
            {
              Node postNode = postList.item(j);
              if (postNode.getNodeType() == Node.ELEMENT_NODE)
              {
                String post = postNode.getTextContent();
                outputBuilder.append(String.format("
<post>%s</post>\n", post));
              }
            }
            outputBuilder.append("    </posts>\n");
          }
        }

        outputBuilder.append("  </Employee>\n");
      }
    }
  }
  outputBuilder.append("</Employees>\n");

  // Lekerdezes a legnagyobb m^3-ru Site-ra
  NodeList siteList = doc.getElementsByTagName("Site");
  int maxArea = 0;
  Element maxAreaElement = null;
  for (int i = 0; i < siteList.getLength(); i++)
  {
    Node node = siteList.item(i);
    if (node.getNodeType() == Node.ELEMENT_NODE)
    {
      Element element = (Element) node;
      int currentArea =
Integer.parseInt(element.getElementsByTagName("area").item(0).getTextContent());
      if (currentArea > maxArea)
      {
        maxArea = currentArea;
        maxAreaElement = element;
      }
    }
  }
  if (maxAreaElement != null)
  {
    String siteId = maxAreaElement.getAttribute("site_id");
    String name =
maxAreaElement.getElementsByTagName("name").item(0).getTextContent();
    outputBuilder.append("\n<LargestSite>\n");
    outputBuilder.append(String.format("  <Site ID=\"%s\">\n", siteId));
```

```
    outputBuilder.append(String.format("    <Name>%s</Name>\n", name));
    outputBuilder.append(String.format("    <Area>%d</Area>\n", maxArea));
    outputBuilder.append("  </Site>\n");
    outputBuilder.append("</LargestSite>\n");
  }

  // Lekerdezes az 1999 utan szuletett Employee-kre
  outputBuilder.append("\n<EmployeesBornAfter1999>\n");
  for (int i = 0; i < employeeList.getLength(); i++)
  {
    Node node = employeeList.item(i);
    if (node.getNodeType() == Node.ELEMENT_NODE)
    {
      Element element = (Element) node;
      String birthDate =
element.getElementsByTagName("birth_date").item(0).getTextContent();
      int birthYear = Integer.parseInt(birthDate.substring(0, 4));
      if (birthYear > 1999) {
        String empId = element.getAttribute("emp_id");
        String firstName =
element.getElementsByTagName("first_name").item(0).getTextContent();
        String lastName =
element.getElementsByTagName("last_name").item(0).getTextContent();
        String sex =
element.getElementsByTagName("sex").item(0).getTextContent();

        outputBuilder.append(String.format("  <Employee emp_id=\"%s\">\n",
empId));
        outputBuilder.append(String.format("
<first_name>%s</first_name>\n", firstName));
        outputBuilder.append(String.format("
<last_name>%s</last_name>\n", lastName));
        outputBuilder.append(String.format("
<birth_date>%s</birth_date>\n", birthDate));
        outputBuilder.append(String.format("    <sex>%s</sex>\n", sex));

        // Posts es azok elemeinek kezelese
        NodeList postsList = element.getElementsByTagName("posts");
        if (postsList.getLength() > 0) {
          Node postsNode = postsList.item(0);
          if (postsNode.getNodeType() == Node.ELEMENT_NODE)
          {
            outputBuilder.append("    <posts>\n");
            NodeList postList =
((Element)postsNode).getElementsByTagName("post");
            for (int j = 0; j < postList.getLength(); j++) {
              Node postNode = postList.item(j);
              if (postNode.getNodeType() == Node.ELEMENT_NODE)
```

```
            {
              String post = postNode.getTextContent();
              outputBuilder.append(String.format("
<post>%s</post>\n", post));
            }
          }
          outputBuilder.append("    </posts>\n");
        }
      }

      outputBuilder.append("  </Employee>\n");
    }
  }
 }
 outputBuilder.append("</EmployeesBornAfter1999>\n");

 // Lekerdezes a "Medve park" nevu Habitat description-jere
 NodeList habitatList = doc.getElementsByTagName("Habitat");
 outputBuilder.append("\n<MedveParkDescription>\n");
 for (int i = 0; i < habitatList.getLength(); i++)
 {
   Node node = habitatList.item(i);
   if (node.getNodeType() == Node.ELEMENT_NODE)
   {
     Element element = (Element) node;
     String name =
element.getElementsByTagName("name").item(0).getTextContent();
     if (name.equals("Medve park"))
     {
       String description =
element.getElementsByTagName("description").item(0).getTextContent();
       outputBuilder.append(String.format("
<Description>%s</Description>\n", description));
     }
   }
 }
 outputBuilder.append("</MedveParkDescription>\n");

 // Lekerdezes a 3. id-ju User lakcimere
 NodeList userList = doc.getElementsByTagName("User");
 outputBuilder.append("\n<UserAddress id=\"3\">\n");
 for (int i = 0; i < userList.getLength(); i++)
 {
   Node node = userList.item(i);
   if (node.getNodeType() == Node.ELEMENT_NODE)
   {
     Element element = (Element) node;
     if (element.getAttribute("user_id").equals("3"))
```

```
        {
          String postCode =
   element.getElementsByTagName("post_code").item(0).getTextContent();
          String city =
   element.getElementsByTagName("city").item(0).getTextContent();
          String street =
   element.getElementsByTagName("street").item(0).getTextContent();
          String number =
   element.getElementsByTagName("number").item(0).getTextContent();
          outputBuilder.append(String.format("  <Address>\n"));
          outputBuilder.append(String.format("    <PostCode>%s</PostCode>\n",
   postCode));
          outputBuilder.append(String.format("    <City>%s</City>\n", city));
          outputBuilder.append(String.format("    <Street>%s</Street>\n",
   street));
          outputBuilder.append(String.format("    <Number>%s</Number>\n",
   number));
          outputBuilder.append("  </Address>\n");
        }
      }
    }
    outputBuilder.append("</UserAddress>\n");

    System.out.println(outputBuilder.toString());
  }
}
```

Programkód 1.5. DOMQueryKLNSPG.java adatlekérdező program

1.2.4. Adatírás

A kód fő funkciója, hogy beolvassa és kiírja az XML tartalmakat a konzolra és fájlba is. A `main` metódusban az *XMLKLNSPG* fájlt olvassa be a megadott útvonalról, használva a `DocumentBuilderFactory` és `DocumentBuilder` osztályokat az XML struktúra elemzéséhez. Ezután normalizálja a dokumentumot a `normalize` metódussal, ami segít a DOM fa struktúrájának rendezésében.

A kiíratást a `TransformerFactory` és a `Transformer` osztályok segítségével végezzük el, pont ahogy az adatmódosító kódban.

A `writeDocumentToFile` metódus is egy `Transformer` objektumot használ az XML dokumentum fájlba írásához, tiszteletben tartva az XML formázási szabályokat. Ez a metódus lehetővé teszi egy új XML dokumentumok mentését. A kiírási folyamat során a kód biztosítja az XML szabványoknak megfelelő indentálást és formázást, ami olvashatóbbá és könnyebben értelmezhetővé teszi a kimenetet.

A kivételkezelés a kódban biztosítja, hogy az XML olvasás vagy írás közben fellépő hibák megfelelően kezelve legyenek, megelőzve ezzel a program összeomlását.

```
import org.w3c.dom.*;
import org.xml.sax.SAXException;

import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.*;

public class DOMWriteKLNSPG
{
  public static void main(String[] args)
  {
    try
    {
      File inputFile = new
   File("C:\\projects\\KLNSPG_XMLGyak\\XMLTaskKLNSPG\\XMLKLNSPG.xml");
      DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
      DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
      Document doc = dBuilder.parse(inputFile);
      doc.getDocumentElement().normalize();
      System.out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
      TransformerFactory transformerFactory = TransformerFactory.newInstance();
      Transformer transformer = transformerFactory.newTransformer();
      transformer.setOutputProperty(OutputKeys.INDENT, "yes");
      DOMSource source = new DOMSource(doc);
      StreamResult consoleResult = new StreamResult(System.out);
      transformer.transform(source, consoleResult);
      writeDocumentToFile(doc, "XMLKLNSPG1.xml");

      System.out.println("The content has been written to the output file
   successfully.");
    }
    catch (SAXException | IOException | ParserConfigurationException |
   TransformerException e)
    {
      e.printStackTrace();
    }
  }
  private static void writeDocumentToFile(Document doc, String filename)
   throws TransformerException
  {
    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");
    DOMSource source = new DOMSource(doc);
    StreamResult result = new StreamResult(new File(filename));
```

```
    transformer.transform(source, result);
  }
}
```

Programkód 1.6. DOMWriteKLNSPG.java adatíró program

2. fejezet

Kupon validátor

A mobilalkalmazások rohamos terjedése és az okostelefonok szinte állandó jelenléte az életünkben óriási lehetőségeket teremt a különféle üzleti modellek és szolgáltatások számára. Ebben a digitális korban, amikor a felhasználók egyre inkább kényelmi szempontok alapján döntenek, az ajándékutalványok és kuponok kezelésére szolgáló alkalmazások jelentősége kiemelkedően nőtt. Az ilyen típusú alkalmazások nem csupán a vásárlói élmény javítását szolgálják, hanem lehetővé teszik a vállalkozások számára is, hogy pontosabb képet kapjanak ügyfeleik viselkedéséről és preferenciáiról.

A „Tattoo Pont Client" alkalmazás fejlesztése során az Expo keretrendszer alkalmazásával egy olyan mobilalkalmazást hoztam létre, amely képes a felhasználók igényeit magas szinten kiszolgálni, miközben a fejlesztési folyamatot jelentősen leegyszerűsíti és felgyorsítja. Az Expo platform előnye, hogy egy közös kódbázison keresztül lehet natív alkalmazásokat fejleszteni iOS és Android platformokra egyaránt, ezáltal jelentősen csökkentve a fejlesztési és karbantartási költségeket.

Az alkalmazás ajándékkupon generáló részének fejlesztése során a C# nyelvet és a .NET keretrendszert választottam, ami lehetővé tette számomra, hogy biztonságos, skálázható és megbízható backend szolgáltatást hozzak létre. A C# programom az ajándékutalványok generálását és azok adatbázisban való tárolását végzi. A MongoDB-t választottam adatbázis-kezelő rendszernek, mivel annak dokumentum-orientált jellege kiválóan alkalmas a dinamikus és strukturálatlan adatok kezelésére, mint amilyenek az ajándékutalványok adatai is.

A generált ajándékutalványok QR kódokat tartalmaznak, amelyek egyedi azonosítót (ObjectId) hordoznak, így biztosítva az utalványok egyediségét és könnyen nyomon követhetőségét. Az *IsValid* és *CreationDate* mezők további funkciókat látnak el az utalványok kezelésében, például a validáció során az *IsValid* mező segítségével könnyedén megállapítható, hogy az utalvány érvényes-e még, míg a `CreationDate` információkat szolgáltat az utalvány kibocsátásának idejéről, és ezt felhasználva azonosíthatja a felhasználó az ajándékkupon érvényességét is.

Az alkalmazás különlegessége abban rejlik, hogy a modern technológiák és a felhasználóbarát interfész ötvözésével egy olyan platformot hoz létre, amely nemcsak a végfelhasználók számára kínál kényelmet és egyszerűséget, hanem a vállalkozások számára is értékes adatokat szolgáltat az

ügyfelek viselkedéséről. Az alkalmazás így nem csak egy technológiai megoldás, hanem egy üzleti eszköz is, amely hozzájárul a vásárlói lojalitás növeléséhez és az üzleti döntések megalapozásához.

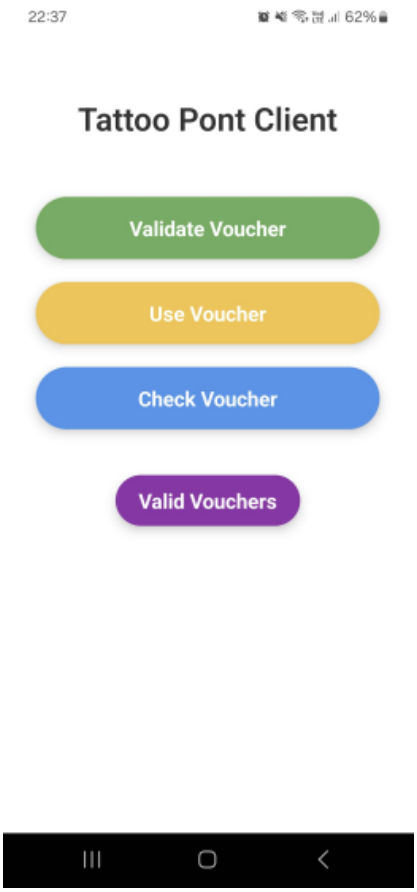
2.1. ábra. Egy kigenerált ajándékkártya

2.1. React Native struktúra és tartalmi felépítés

A *Tattoo Pont Client* alkalmazás kezdőlapja egy React Native alapú mobilalkalmazás felhasználói interfészét valósítja meg. Az alkalmazás a modern mobilfejlesztési keretrendszer előnyeit kihasználva biztosít cross-platform kompatibilitást és felhasználóbarát dizájnt. A kezdőlap felépítése a következő elemekből áll:

- **Fő konténer**: Egy `View` komponens, amely az alkalmazás kezdőképernyőjének alapját képezi, és központosítja a felhasználói felület elemeit.
- **Címsor**: Egy `Text` komponens, amely kiemeli az alkalmazás nevét, segítve ezzel a felhasználók azonosítási folyamatát.
- **Gombok**: `TouchableOpacity` komponensek, amelyek a különböző funkciók elérését teszik lehetővé, mint például a kuponok érvényesítése, felhasználása, ellenőrzése és az érvényes kuponok listájának megtekintése.
- **Navigációs logika**: A `useNavigation` hook segítségével implementált logika, amely lehetővé teszi a felhasználók számára, hogy az alkalmazás különböző képernyői között navigáljanak.

A kezdőlap vizuális stílusát a `StyleSheet.create` metódussal definiált stílusok határozzák meg, amelyek az alkalmazás vizuális megjelenésének minden aspektusát lefedik, beleértve a színsémát, a gombok formáját, a szöveg méretét és súlyát, valamint az árnyékolást és kiemelést.



2.2. ábra. Kezdőlap

```
import React from 'react';
import { View, Text, TouchableOpacity, StyleSheet } from 'react-native';
import { useNavigation } from '@react-navigation/native';

export default function Index() {
    const navigation = useNavigation();

    return (
        <View style={styles.container}>
            <Text style={styles.headerText}>Tattoo Pont Client</Text>
            <View style={styles.buttonGroup}>
                <TouchableOpacity
                    style={[styles.button, styles.validateButton]}
                    onPress={() => navigation.navigate('Validation')}>
                    <Text style={styles.buttonText}>Validate Voucher</Text>
                </TouchableOpacity>
```

```
                <TouchableOpacity
                    style={[styles.button, styles.useButton]}
                    onPress={() => navigation.navigate('Use')}>
                    <Text style={styles.buttonText}>Use Voucher</Text>
                </TouchableOpacity>
                <TouchableOpacity
                    style={[styles.button, styles.checkButton]}
                    onPress={() => navigation.navigate('Check')}>
                    <Text style={styles.buttonText}>Check Voucher</Text>
                </TouchableOpacity>
            </View>
            <TouchableOpacity
                style={styles.specialButton}
                onPress={() => navigation.navigate('voucherlist')}>
                <Text style={styles.specialButtonText}>Valid Vouchers</Text>
            </TouchableOpacity>
        </View>
    );
}

const styles = StyleSheet.create({
    container: {
        flex: 1,
        backgroundColor: '#FFFFFF',
        alignItems: 'center',
        justifyContent: 'flex-start',
        paddingTop: 60,
    },
    headerText: {
        fontSize: 28,
        color: '#333',
        fontWeight: '600',
        marginBottom: 50,
    },
    buttonGroup: {
        width: '100%',
        paddingHorizontal: 30,
    },
    button: {
        backgroundColor: '#E8E8E8',
        paddingVertical: 15,
        paddingHorizontal: 20,
        borderRadius: 30,
        alignItems: 'center',
        justifyContent: 'center',
        marginBottom: 20,
        shadowColor: "#000",
        shadowOffset: {
```

```
            width: 0,
            height: 2,
        },
        shadowOpacity: 0.25,
        shadowRadius: 3.84,
        elevation: 5,
    },
    buttonText: {
        fontSize: 18,
        fontWeight: 'bold',
        color: '#FFFFFF',
    },
    validateButton: {
        backgroundColor: '#4CAF50',
    },
    useButton: {
        backgroundColor: '#FFC107',
    },
    checkButton: {
        backgroundColor: '#2196F3',
    },
    specialButton: {
        backgroundColor: '#9C27B0',
        paddingHorizontal: 20,
        paddingVertical: 10,
        borderRadius: 30,
        shadowColor: "#000",
        shadowOffset: {
            width: 0,
            height: 2,
        },
        shadowOpacity: 0.25,
        shadowRadius: 3.84,
        elevation: 5,
        marginTop: 20,
    },
    specialButtonText: {
        color: '#FFFFFF',
        fontSize: 18,
        fontWeight: 'bold',
    },
});
```

Programkód 2.1. index.js fő felhasználói felület

2.1.1. Validációs lap

A szoftver kezdőlapján található első gomb segítségével érhető el a `Validation` nevű osztály, amely az ajándékutalványok érvényesítésére szolgál. Ez a funkció kulcsfontosságú eleme az alkalmazásnak, lehetővé téve a felhasználók számára, hogy QR kódokat szkenneljenek és ellenőrizzék azokat egy központi adatbázissal való kommunikáció során.

A `Validation` osztály fő jellemzői és működése a következőképpen írható le:

- A kezdőlapon található első gombra kattintva a felhasználókat átirányítjuk a `Validation` osztályra.
- A vonalkód-szkennelési folyamatot az `expo-barcode-scanner` könyvtár `BarCodeScanner` komponense végzi, amely integrálva van az Expo keretrendszerbe. Ez a komponens hozzáférést biztosít a készülék kamerájához, lehetővé téve a QR kódok gyors és hatékony beolvasását.
- Miután a QR kód sikeresen beolvasásra került, a `Validation` osztály egy API-n keresztül kommunikál egy adatbázissal, hogy ellenőrizze, a szkennelt kupon szerepel-e az adatbázisban. Ez a lépés biztosítja, hogy csak érvényes és az adatbázisban regisztrált kuponok kerüljenek érvényesítésre.
- Amennyiben a kupon megtalálható az adatbázisban, a rendszer ellen- őrzi annak érvényességét. Ha a kupon még nem került felhasználásra, az osztály végrehajtja az érvényesítési folyamatot. Ellenkező esetben a felhasználót tájékoztatja a kupon érvénytelenségéről vagy korábbi felhasználásáról.
- Az érvényesítési folyamat eredményéről a felhasználók visszajelzést kapnak egy felugró üzenet formájában, amely tájékoztatja őket a folyamat sikerességéről vagy esetleges hibáiról. Az üzenet színe (zöld sikeres érvényesítés, piros hiba esetén) vizuális visszajelzést is nyújt.

2.3. ábra. Sikeres és sikertelen érvényesítés

```
import React, { useState, useEffect } from 'react';
import { Text, View, StyleSheet } from 'react-native';
import { BarCodeScanner } from 'expo-barcode-scanner';

export default function Validation() {
    const [hasPermission, setHasPermission] = useState(null);
    const [scanned, setScanned] = useState(false);
    const [message, setMessage] = useState('');
    const [messageColor, setMessageColor] = useState('green');

    useEffect(() => {
        (async () => {
            const { status } = await BarCodeScanner.requestPermissionsAsync();
            setHasPermission(status === 'granted');
        })();
    }, []);

    const handleBarCodeScanned = async ({ data }) => {
        setScanned(true);

        try {
            const response = await
   fetch(`https://europe-west1.gcp.data.mongodb-api.com/app/tattoopontbackend-xwtxj/endp
   {
                method: 'GET'
            });
```

```
            if (response.ok) {
                const voucherData = await response.json();
                if (!voucherData || Object.keys(voucherData).length === 0)
                    throw new Error('Voucher is not in the database!');

                if (voucherData.IsValid)
                    throw new Error('Voucher is already valid!');

                await
    fetch(`https://europe-west1.gcp.data.mongodb-api.com/app/tattoopontbackend-xwtxj/endp
    {
                    method: 'PUT'
                });

                setMessage('Validation went successfully!');
                setMessageColor('green');
            } else
                throw new Error('Voucher is not found in the database!');
        } catch (error) {
            setMessage(error.message);
            setMessageColor('red');
        }
    };

    if (hasPermission === null)
        return <Text>Requesting for Camera Permission!</Text>;

    if (hasPermission === false)
        return <Text>No access to open camera!</Text>;

    return (
        <View style={styles.container}>
            {scanned ? (
                <Text style={[styles.message, {color:
    messageColor}]}>{message}</Text>
            ) : (
                <BarCodeScanner
                    onBarCodeScanned={handleBarCodeScanned}
                    style={StyleSheet.absoluteFillObject}
                />
            )}
        </View>
    );
}

const styles = StyleSheet.create({
    container: {
        flex: 1,
```

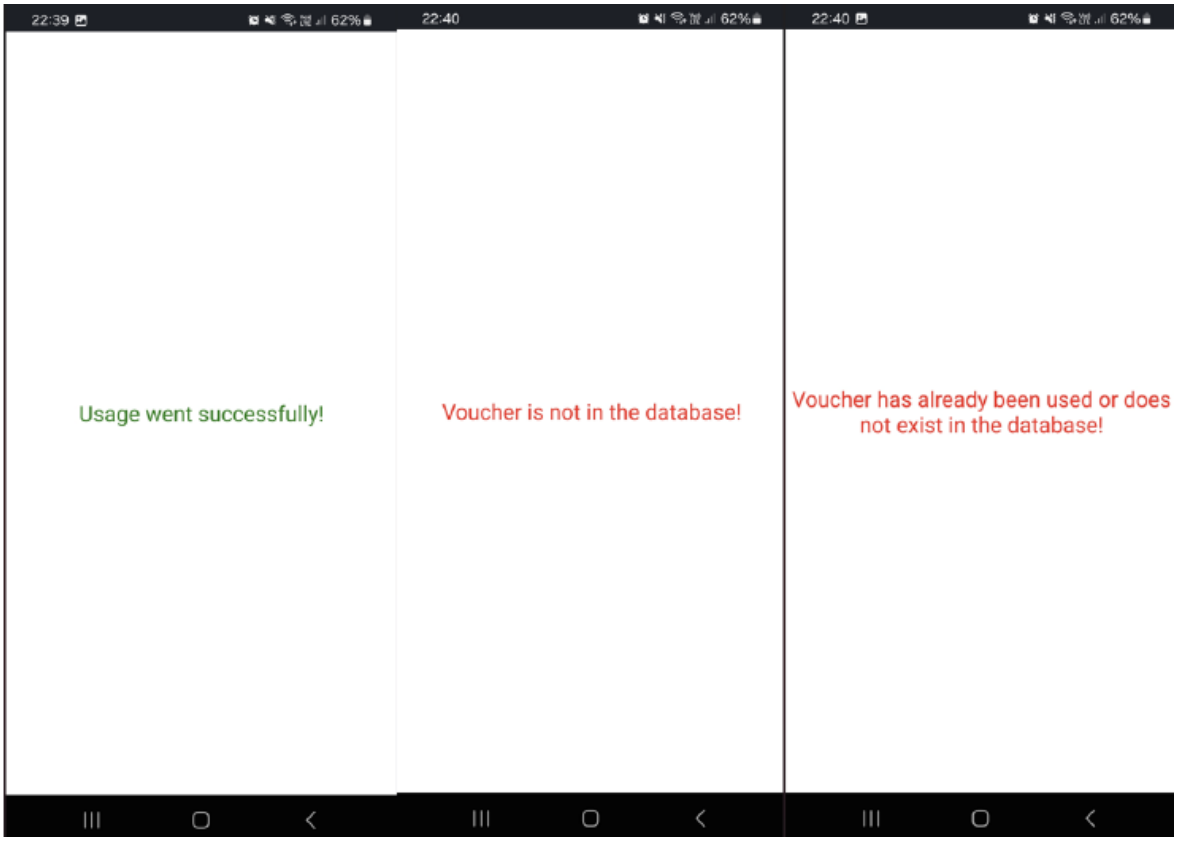
```
        flexDirection: 'column',
        justifyContent: 'center',
    },
    message: {
        fontSize: 20,
        textAlign: 'center',
    },
});
```

Programkód 2.2. validation.js validációs felület

2.1.2. Felhasználási lap

A `Use` osztály egy kulcsfontosságú komponens a mobilalkalmazásban, amely a kuponok felhasználásának logikáját valósítja meg. Az osztály felelős a QR kódok beolvasásáért és azok ellenőrzéséért, hogy a kuponok érvényeseke, és nem kerültek-e már felhasználásra. A főbb jellemzők és működés a következők:

- **Kamera használati engedély**: A komponens a `useEffect` hook segítségével kéri és ellenőrzi a kamera használati engedélyét. Amennyiben az engedély megadásra kerül, lehetővé teszi a QR kódok beolvasását. Ellenkező esetben tájékoztatja a felhasználót az engedély hiányáról.
- **QR kód beolvasása**: A `BarCodeScanner` komponens felhasználásával az osztály folyamatosan várja a QR kódok beolvasását. Amikor egy QR kód beolvasásra kerül, a `handleBarCodeScanned` függvény aktiválódik.
- **Adatbázis ellenőrzés**: A beolvasott QR kód alapján az osztály egy API híváson keresztül ellenőrzi a kupon érvényességét egy adatbázisban. Amennyiben a kupon érvénytelen, már felhasznált, vagy az adatbázisban nem található, a rendszer hibaüzenetet jelenít meg.
- **Kupon felhasználása**: Ha a kupon érvényes és nem került még felhasználásra, az osztály elvégzi a szükséges lépéseket annak felhasználására. Ez magában foglalhatja a kupon adatbázisból történő törlését, ezzel jelezve, hogy a kupon már felhasznált.
- **Felhasználói visszajelzés**: A folyamat eredményétől függően a felhasználók visszajelzést kapnak egy szöveges üzenet formájában, amely tájékoztatja őket a művelet sikerességéről vagy esetleges hibáiról. Az üzenet színe (zöld sikeres felhasználás, piros hiba esetén) segít a felhasználóknak az eredmény azonnali értelmezésében.



2.4. ábra. Sikeres és sikertelen felhasználási kísérletek

```
import React, { useState, useEffect } from 'react';
import { Text, View, StyleSheet } from 'react-native';
import { BarCodeScanner } from 'expo-barcode-scanner';

export default function Use() {
    const [hasPermission, setHasPermission] = useState(null);
    const [scanned, setScanned] = useState(false);
    const [message, setMessage] = useState('');
    const [messageColor, setMessageColor] = useState('green');

    useEffect(() => {
        (async () => {
            const { status } = await BarCodeScanner.requestPermissionsAsync();
            setHasPermission(status === 'granted');
        })();
    }, []);

    const handleBarCodeScanned = async ({ data }) => {
        setScanned(true);
```

```
     try {
         const response = await
fetch(`https://europe-west1.gcp.data.mongodb-api.com/app/tattoopontbackend-xwtxj/endp
{
             method: 'GET'
         });
         const voucher = await response.json();

         if (response.ok && voucher) {
             if (!voucher.IsValid)
                 throw new Error('Voucher is invalid!');

             const creationDate = new Date(voucher.CreationDate);
             const oneYearAgo = new Date();
             oneYearAgo.setFullYear(oneYearAgo.getFullYear() - 1);

             if (creationDate < oneYearAgo) {
                 await
fetch(`https://europe-west1.gcp.data.mongodb-api.com/app/tattoopontbackend-xwtxj/endp
{
                     method: 'DELETE'
                 });
                 throw new Error(`Voucher is older than one
year!\nExpiration Date: ${creationDate.toDateString()}`);
             }

             await
fetch(`https://europe-west1.gcp.data.mongodb-api.com/app/tattoopontbackend-xwtxj/endp
{
                 method: 'DELETE'
             });

             setMessage('Usage went successfully!');
             setMessageColor('green');
         } else {
             throw new Error('Voucher has already been used or does not
exist in the database!');
         }
     } catch (error) {
         setMessage(error.message);
         setMessageColor('red');
     }
 };

 if (hasPermission === null)
     return <Text>Requesting for Camera Permission!</Text>;
```

```
    if (hasPermission === false)
        return <Text>No access to open camera!</Text>;

    return (
        <View style={styles.container}>
            {scanned ? (
                <Text style={[styles.message, {color:
   messageColor}]}>{message}</Text>
            ) : (
                <BarCodeScanner
                    onBarCodeScanned={handleBarCodeScanned}
                    style={StyleSheet.absoluteFillObject}
                />
            )}
        </View>
    );
}

const styles = StyleSheet.create({
    container: {
        flex: 1,
        flexDirection: 'column',
        justifyContent: 'center',
    },
    message: {
        fontSize: 20,
        textAlign: 'center',
    },
});
```

Programkód 2.3. use.js felhasználási felület

Összefoglalva, a `Use` osztály egy esszenciális része az alkalmazásnak, amely lehetővé teszi a felhasználók számára, hogy hatékonyan használják fel a QR kód alapú kuponjaikat. Az osztály biztosítja a kuponok érvényességének ellenőrzését, a felhasználás folyamatának kezelését, valamint a felhasználók tájékoztatását az eredményről.

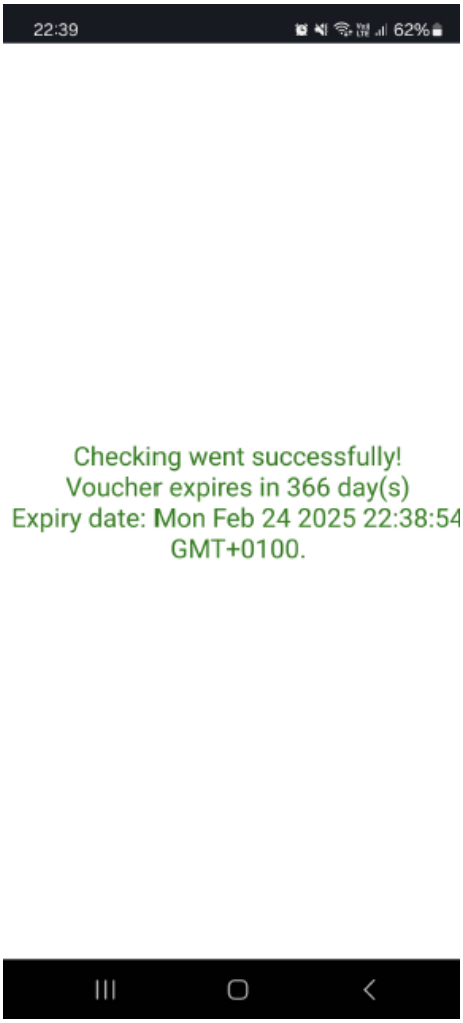
2.1.3. Ellenőrző gomb

A `Check` osztály egy fontos komponens az alkalmazásban, amelynek feladata a QR kódok alapján történő kuponok ellenőrzése. Ez a folyamat magában foglalja a kupon adatbázisban való létezésének és az érvényességi időszakának ellenőrzését.
A főbb jellemzők és működés a következők:

- **Kamera használati engedély**: Az osztály a `useEffect` hook segítségével kéri a kamera használati engedélyét a felhasználótól. Amennyiben az engedélyt megadják, az alkalmazás képes lesz a QR kódok beolvasására.

- **QR kód beolvasása**: A `BarCodeScanner` komponens segítségével az osztály folyamatosan várja a QR kódok beolvasását. Beolvasás után a `handleBarCodeScanned` függvény kerül aktiválásra.
- **Adatbázis ellenőrzés**: A QR kód tartalmának felhasználásával az osztály egy API hívás segítségével lekéri a kupon adatait az adatbázisból. Ellenőrzi a kupon létezését, valamint annak létrehozási és lejárati dátumát.
- **Lejárati idő számítása**: Amennyiben a kupon létezik az adatbázisban, az osztály kiszámítja a lejárati dátumot a létrehozási idő alapján, majd összehasonlítja azt a jelenlegi dátummal, hogy megállapítsa a kupon érvényességét.
- **Felhasználói visszajelzés**: Az ellenőrzési folyamat eredményétől függően a felhasználók egy üzenetet kapnak, amely tájékoztatja őket a kupon érvényességéről vagy annak hiányáról. Az üzenet színe (zöld, ha a kupon érvényes; piros, ha a kupon érvénytelen vagy nem található az adatbázisban) az eredménytől függ.

A `Check` osztály tehát lehetővé teszi a felhasználók számára, hogy gyorsan és hatékonyan ellenőrizzék a kuponjaik érvényességét. Ez a folyamat növeli az alkalmazás használati értékét, biztosítva a felhasználók számára egy megbízható eszközt a kuponok kezelésére.



2.5. ábra. Ellenőrzési folyamat visszajelzése

```
import React, { useState, useEffect } from 'react';
import { Text, View, StyleSheet } from 'react-native';
import { BarCodeScanner } from 'expo-barcode-scanner';

export default function Check() {
    const [hasPermission, setHasPermission] = useState(null);
    const [scanned, setScanned] = useState(false);
    const [message, setMessage] = useState('');
    const [messageColor, setMessageColor] = useState('green');

    useEffect(() => {
        (async () => {
            const { status } = await BarCodeScanner.requestPermissionsAsync();
```

```
            setHasPermission(status === 'granted');
        })();
    }, []);

    const handleBarCodeScanned = async ({ data }) => {
        setScanned(true);

        try {
            const response = await
fetch(`https://europe-west1.gcp.data.mongodb-api.com/app/tattoopontbackend-xwtxj/endp
{
                method: 'GET'
            });

            if (response.ok) {
                const voucherData = await response.json();
                if (!voucherData || Object.keys(voucherData).length === 0 ||
!voucherData.CreationDate)
                    throw new Error('Voucher is not in the database or missing
creation date!');

                const creationDate = new Date(voucherData.CreationDate);
                const expirationDate = new Date(creationDate);
                expirationDate.setFullYear(expirationDate.getFullYear() + 1);

                const currentDate = new Date();
                const timeDiff = expirationDate.getTime() -
currentDate.getTime();
                const daysRemaining = Math.ceil(timeDiff / (1000 * 3600 * 24));

                if (daysRemaining > 0) {
                    setMessage(`Checking went successfully!\nVoucher expires
in ${daysRemaining} day(s)\nExpiry date: ${expirationDate}.`);
                    setMessageColor('green');
                } else {
                    setMessage('Voucher is not found in the database!');
                    setMessageColor('red');
                }
            } else
                throw new Error('Voucher is not found in the database!');
        } catch (error) {
            setMessage(error.message);
            setMessageColor('red');
        }
    };

    if (hasPermission === null)
        return <Text>Requesting for Camera Permission!</Text>;
```

```
    if (hasPermission === false)
        return <Text>No access to open camera!</Text>;

    return (
        <View style={styles.container}>
            {scanned ? (
                <Text style={[styles.message, {color:
   messageColor}]}>{message}</Text>
            ) : (
                <BarCodeScanner
                    onBarCodeScanned={handleBarCodeScanned}
                    style={StyleSheet.absoluteFillObject}
                />
            )}
        </View>
    );
}

const styles = StyleSheet.create({
    container: {
        flex: 1,
        flexDirection: 'column',
        justifyContent: 'center',
    },
    message: {
        fontSize: 20,
        textAlign: 'center',
    },
});
```

Programkód 2.4. check.js ellenőrzési felület

2.1.4. Érvényes kuponok kilistázása

A `VoucherList` osztály egy fontos komponens az alkalmazásban, amely a felhasználóknak lehetővé teszi az érvényes kuponok aktuális számának megtekintését. Az osztály dinamikusan frissíti ezt az információt, biztosítva a legfrissebb adatok elérhetőségét.
A főbb jellemzők és működés a következők:

- **Állapotkezelés**: A komponens a `useState` hook segítségével kezeli az érvényes kuponok számát (*couponCount*) és a frissítés állapotát (*refreshing*).
- **Adatlekérés**: A `useEffect` hook használatával, a komponens betöltésekor automatikusan meghívódik a `fetchCouponCount` függvény, amely lekéri az érvényes kuponok számát egy adatbázisból egy API hívás segítségével.

- **Frissítési logika**: A `ScrollView` komponens `RefreshControl` eleme lehetővé teszi a felhasználók számára, hogy lehúzással frissítsék az érvényes kuponok számát. Ez a folyamat újraindítja az `fetchCouponCount` függvényt.
- **Felhasználói felület**: A komponens egy gördíthető nézetet (*ScrollView*) használ, amely középen jeleníti meg az érvényes kuponok számát egy szöveges elem (*Text*) segítségével. A dizájn egyszerű, de informatív, biztosítva a felhasználók számára a könnyű érthetőséget és hozzáférést.

2.6. ábra. Az érvényes kuponok száma

```
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet, RefreshControl, ScrollView } from
   'react-native';

export default function VoucherList() {
    const [couponCount, setCouponCount] = useState(0);
    const [refreshing, setRefreshing] = useState(false);

    useEffect(() => {
```

```
        fetchCouponCount();
    }, []);

    const fetchCouponCount = async () => {
        setRefreshing(true);
        try {
            const response = await
   fetch(`https://europe-west1.gcp.data.mongodb-api.com/app/tattoopontbackend-xwtxj/endp
   {
                method: 'GET'
            });
            const data = await response.json();
            if (data && typeof data.count === 'number') {
                setCouponCount(data.count);
            }
        } catch (error) {
            console.error(error);
        } finally {
            setRefreshing(false);
        }
    };

    return (
        <ScrollView
            style={styles.container}
            contentContainerStyle={styles.contentContainer}
            refreshControl={
                <RefreshControl refreshing={refreshing}
   onRefresh={fetchCouponCount} />
            }>
            <View style={styles.countContainer}>
                <Text style={styles.countText}>Number of valid vouchers:
   {couponCount}</Text>
            </View>
        </ScrollView>
    );
}

const styles = StyleSheet.create({
    container: {
        flex: 1,
    },
    contentContainer: {
        flexGrow: 1,
        justifyContent: 'center',
        alignItems: 'center',
    },
    countContainer: {
```

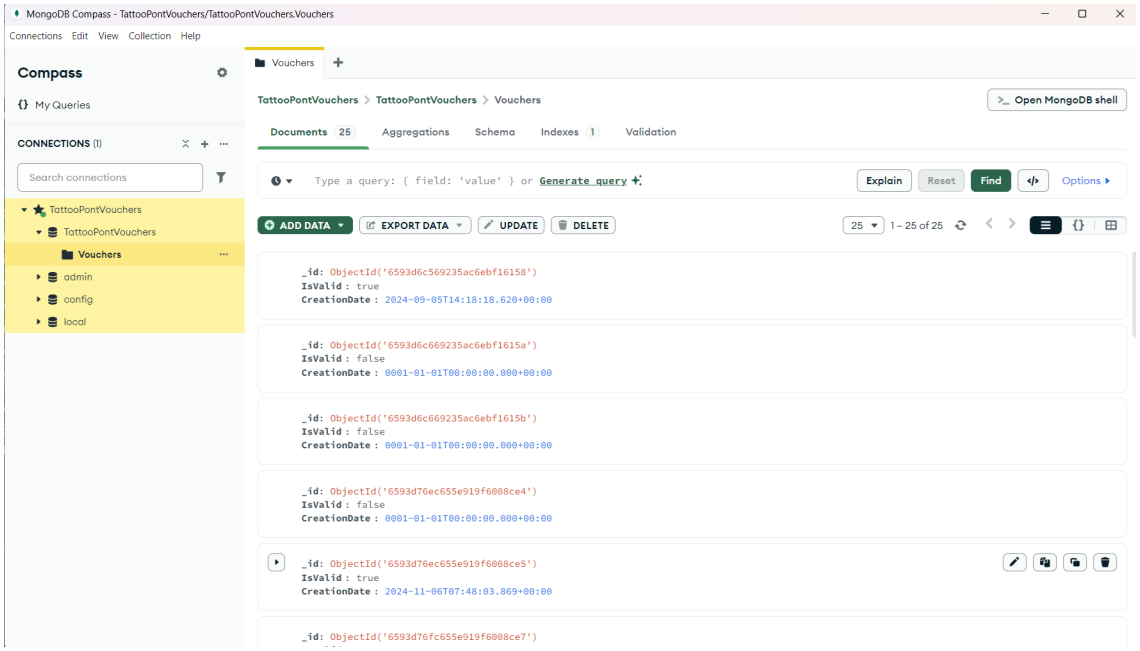
```
        backgroundColor: '#f9f9f9',
        borderWidth: 1,
        borderColor: '#ddd',
        borderRadius: 5,
        padding: 20,
    },
    countText: {
        fontSize: 18,
    },
});
```

Programkód 2.5. voucherlist.js listázási felület

Az osztály tehát egy interaktív és felhasználóbarát módot biztosít az érv ényes kuponok nyomon követésére, lehetővé téve a felhasználók számára, hogy naprakészek maradjanak az aktuális kuponkínálattal kapcsolatban. A frissítési funkció növeli az alkalmazás használhatóságát, ösztönözve a felhasználókat arra, hogy aktívan használják az alkalmazást a kuponok kezelésére.

2.2. MongoDB kliens alkalmazás

A MongoDB konnekció, illetve validációra a *MongoDB Compass* alkalmazást használtam. Ez az alkalmazás lehetővé teszi, hogy létrehozzunk benne kollekciókat, illetve monitorozzuk azokat, ha esetleg egy külső program férne hozzá a MongoDB adatbázisunkhoz, azaz jelen esetben.



2.7. ábra. MongoDB Compass alkalmazása

Az ábrán látható, ahogyan a MongoDB kliens alkalmazás kilistázza a meglévő dokumentumokat, a hozzájuk tartozó propertykkel együtt. A még nem érvényesített kuponok az adatbázisban `IsValid : false`-ként jelennek még, illetve a `CreationDate` változó is egy alapértelmezett értékkel inicializálódik.

A létrehozott és érvényesített kupon `IsValid` propertyje `true` értéket fog felvenni az érvényesítés után, majd a `CreationDate` is meg kapja az érvényesítéskor felvett értéket, ezek alapján tudja majd a backend kiszámolni az egzakt lejárati időt.