***Bevezetés***

cukorbetegség

elhízás

magas vérnyomás

egészséges életmód / egészséges emberek számára is jó

internet/mobil eszközök

***Követelmény-Analízis(Specifikáció)***

Ebben a fejezetben mutatom be a megvalósítandó rendszerrel szemben támasztott követelményeket. Röviden ismertetem a részfeladatok sajátosságait és funkcionalitásait, amelyekre szükség van a specifikált működés érdekében.

Dolgozatom célja egy olyan ETL alkalmazás létrehozása, amely kapcsolatot biztosít két meglévő, strukturálisan jelentős módon eltérő adatbázis között. Az ETL mozaikszó az Extract, Transform, Load kifejezésekből származik, ami annyit tesz, hogy Kinyerés, Átalakítás, Betöltés. Az említett adatbázisok a US Department of Agriculture röviden USDA, Nutrient Database for Standard Reference adatbázis és a Lavinia életmód-tükör mögötti MenuGene adatbázis.

Az USDA adatbázisául szolgáló adatokat az Amerikai Egyesült Államokban található Agricultural Research Service, röviden ARS, az USDA egyik legfőbb kutatócsoportja biztosítja. A minél pontosabb eredmények érdekében több ezer kutató végzi munkáját, hogy megoldást találjanak a mezőgazdasági problémákra, amelyek hatással vannak az amerikai emberek mindennapjaira. Az ARS magas prioritással vezeti a kutatást, hogy kifejlesszék a megoldásokat, amelyek az egész nemzetet érintik. A kutatók egyre több és több élelmiszert vizsgálnak meg és akár ugyanazt a vizsgálatot többször is elvégzik egy adott termék esetében, hogy minél pontosabb adatokkal tudjanak szolgálni. Hozzáférést biztosítanak az ételek és más mezőgazdasági termékek vizsgálatainak eredményeiről. Az információkból felállított adatbázis bárki számára elérhető és felhasználható, természetesen jogtiszta módon. Rendszeres jelleggel frissítik az adatbázist. A legelső verzió 1993-ban készült el SR10-es verzió néven. Azóta számos revízió került ki az USDA kutatóinak köszönhetően, a jelenleg utolsó 2015-ös SR28-as frissítéssel bezárólag. **(forrás USDA)**

A Lavinia életmód-tükör adatbázisa szintén a Pannon Egyetem Műszaki Informatikai Karán működő Egészségügyi Informatikai Kutató-Fejlesztő Központ által fejlesztett MenuGene táplálkozás-tudományi szakértői rendszer szolgáltatásait és adatbázisát használja fel. Az eddigiek során dietetikusok tartották karban az adatbázist. Az USDA adatai szintén forrásul szolgáltak ez idáig is, azonban csak manuális módon volt lehetőség a rendszeres frissítésre, amely jelentős kézi munkát vett igénybe. **(forrás Lavinia)**

A megvalósítandó rendszer feladata, hogy az USDA gyakran frissített adatait felhasználja, és automatizált módon frissítse a Lavinia adatbázisát ezen adatokkal. A kiépített kapcsolat segíti a cél-adatbázist, hogy naprakész legyen, és az aktuális adatokkal szolgáljon a különféle élelmiszerek, tápanyagok, tápanyagtartalmak és mértékegységek tekintetében. Ez azt jelenti, hogy az élelmiszerekhez minél pontosabb értékek társuljanak, amely során új adatok jelenhetnek meg, vagy az eddigi értékek módosulhatnak, vagy épp törlésre kerülhetnek.

A Lavinia jelenleg relációs adatbázist használ a receptek, az ételek, a tápanyagok és a további adatok tárolására, melyet a PostgreSQL relációsadatbázis-kezelő rendszerrel valósít meg. Az USDA adatbázisa viszont szöveges fájlok formájában érhetőek el és tölthetők le. A rendszeres jelleggel publikált frissítések két külön részre bonthatók. Egy új revízió egyrészt tartalmazza az egész adatbázist, az újonnan frissített értékekkel, másrészt pedig tartalmazza az előző verzióval szembeni változtatásokat. Előbbi eset nyilvánvalóan nagyobb adathalmazt takar, hiszen a teljes adatbázist tartalmazza, szemben a verziók közötti változásokat összefoglaló fájlokkal, melyek nagysága töredéke a komplett adatbázisnak.

Nem szabad figyelmen kívül hagyni, hogy a cél-adatbázison való frissítés végrehajtásának függetlennek kell lennie a folyamat időpontjától és számától. Elengedhetetlen, hogy bármikor és bármennyiszer futtatható állapotban legyen, továbbá támogassa az egymás utáni adatbázis-frissítéseket.

Elengedhetetlen a megfelelő hibatűrés a szoftver üzemeltetése során. Az adatbázis visszagörgetése magas prioritást kap bármilyen fellépő hiba esetén a frissítés során. A rendellenesség előtti műveleteket vissza kell állítani a korábbi állapotra. Ez azt jelenti, hogy az egész frissítési folyamatnak problémamentesen kell lefutnia, ellenkező esetben az adatbázis visszaáll az előző állapotára.

A szoftver nélkülözhetetlen részét képezi az adatbázis-frissítés folyamatának naplózása valamilyen formában. A jelentésnek szöveges formátumban kell készülnie úgy, hogy könnyen értelmezhető legyen a felhasználó számára. Tartalmaznia kell egyaránt a sikeres és sikertelen műveletek kimenetelét, miszerint az esetleges későbbi hibakeresés alkalmával könnyű dolga legyen a felhasználónak a fennálló problémák kijavítására. Legegyszerűbb megoldás, ha egy közismert karakterkódolással ellátott szöveges fájlokba mentjük a naplófolyamatokat. A naplófájl felépítésének a lehető legegyszerűbbnek kell lennie a könnyű olvashatóság és értelmezhetőség kedvéért.

A felhasználóbarát működés érdekében szükség van egy könnyen kezelhető és átlátható grafikus felhasználói felületre, amely arra szolgál, hogy tájékoztassa a felhasználót a műveletek kimeneteléről. A grafikus interfész fő feladata, hogy a felhasználóval valós időben közvetítse a szöveges fájlok feldolgozásának folyamatát, illetve az adatbázis-frissítés eredményét, mindezt jól látható és értelmezhető megjelenési formában.

***Felhasznált technológiák***

Az alábbi fejezetben ismertetem a dolgozatom kivitelezéséhez felhasznált technológiákat. Fontos szempont volt számomra, hogy széles körben használható legyen az alkalmazás, de legfőképpen Windows-os környezetben. Ebből az okból kifolyólag választottam a Java programozási nyelvet és a hozzákapcsolódó keretrendszereket, interfészeket. A felsorolt technológiák egy részét tanulmányaim során volt lehetőségem elsajátítani, de számos új tapasztalattal gazdagodtam az újonnan megismert technológiákkal kapcsolatban.

***PostgresSQL***

A PostgreSQL, más néven Postgres egy relációsadatbázis-kezelő rendszer, amelyet a Lavinia rendszere is használ. Szabad szoftver, melynek fejlesztését önkéntesek végzik közösségi alapon. A munka elsődleges koordináló oldala a postgresql.org. Kezdetben a Berkeley Egyetemen indult meg a fejlesztése a nyolcvanas években, majd a kilencvenes évek közepére elhagyta az egyetem falait és nyílt forráskódúvá vált.

A relációsadatbázis-kezelő rendszer (RDBMS) egy olyan adatbázis-kezelő rendszer, amelynek logikai adatbázisát szoftverkomponensei kizárólag a relációs adatmodellek elvén épülnek fel, illetve kérdezhetőek le. Kizárólag a relációs adatmodell alapú megközelítést támogatja. A relációsadatbázis-kezelő rendszerek szabványos adat hozzáférési nyelve az SQL (Structured Query Language). Az SQL segítségével könnyen és érthetően leírhatók akár az összetettebb CRUD (Create, Read, Update, Delete) funkciók is.

Az adatbázis adataihoz való hozzáférést, manipulációt, valamint az adatszerkezet tanulmányozásához a pgAdmin 4 programot használtam. Ez egy ingyenesen elérthető szoftver a PostgreSQL fejlesztőitől. E program melletti választásomat indokolta az, hogy biztosítja az egyszerű kezelőfelület és a szükséges funkciókat a feladatom során.

/\*\*\*\*\*\*\*

ÁBRA: PG Admin képernyőkép

\*\*\*\*\*\*\*/

***Java***

A letölthető fájlok kezelésére, az adatbázis elérésére és manipulálására olyan szoftver kell, amely vezérli az adatátvitelt és közben erről tájékoztatást nyújt a felhasználó számára a folyamatról. Ezért döntöttem a Java nyelven történő implementálásról a feladat során.

A Java egy általános célú, objektumorientált programozási nyelv, amelyet a Sun Microsystems fejlesztett a kilencvenes évek elejétől. Jelenleg az Oracle gondozásában áll. A Java alkalmazásokat jellemzően bájtkód formátumra alakítják. A bájtkód futtatását a Java virtuális gép (Java Virtual Machine) végzi, ami vagy interpretálja a bájtkódot, vagy natív gépi kódot készít belőle, és azt futtatja.

A program fejlesztéséhez a NetBeans IDE fejlesztőkörnyezetet választottam. A Java 8-as környezet szükséges a program futtatásához.

A feladat implementálásához a Java nyelvet választottam. A specifikáció során említett funkciókat könnyű vele megvalósítani. A fájlkezelést, az adatbázis kapcsolatot, a grafikus megjelenítést és az ezek közötti adatátvitelt egyaránt egyszerű felépíteni. Emellett fontos szempont volt, hogy széles körben használható legyen a megvalósított szoftver. A Java erőssége, hogy platformfüggetlen, csupán a Java Runtime Environment szükséges a programok futtatásához. Fejlesztési szempontból viszont elengedhetetlen a Java Development Kit, amely tartalmazza az előbb említett környezetet. Habár a Java Virtual Machine gyorsasága nem éri el a hardware közeli nyelvekét, ez nem jelent számottevő hátrányt a működésben.

***JDBC***

A Java Database Connectivity, röviden JDBC egy API a Java programozási nyelvhez, amely az adatbázishozzáférést támogatja. A JDBC definiálja az adatbázisok lekérdezéséhez és módosításához szükséges osztályokat és metódusokat, miközben igazodik a relációs adatmodellhez. Egyik fajtája a PostgreSQL JDBC interfész, amelyet a PostgreSQL fejlesztői adtak ki.

/\*\*\*\*\*\*\*

java példakód a tervezéshez, prepared/callable statement

ÁBRA: JDBC működési elve

\*\*\*\*\*\*\*/

***JavaFX***

A JavaFX egy szoftver platform az asztali alkalmazások létrehozásához, amely a Swing mellett a Java Standard Edition alapértelmezett GUI könyvtára. A JavaFX applikációkat bármilyen asztali, mobil eszközön vagy böngészőben lehet futtatni. A grafikus felületet egy XML fájl definiálja, amelyet FXML fájlban tárolunk. Az XML fájl segítségével pontosan leírhatók az objektumok pozíciója és egyéb tulajdonságaik. Minden objektumhoz megadható egy egyedi azonosító, amellyel hivatkozhatunk rá a forráskódban. A keretrendszer támogatja az audió, a videó és az animáció implementálását is. A grafikus felület a Scene Builder program segítségével egyszerűen összeállítható és nem kell aggódnunk az XML fájl összeállításában, mert automatikusan legenerálja azt.

***Apache Commons IO***

Az Apache Commons IO egy Java könyvtár fájlműveletekhez, melyet az Apache Foundation felügyelete alatt fejlesztettek. Osztályok sokaságát biztosítja a fejlesztők számára, hogy egyszerűbb, rövidebb és érthetőbb kód íródhasson a fájlok kezelésére. A feladatom során a fájlműveletek, különösképpen az olvasás rendkívüli szerepet játszanak a rendszer részeként, hiszen fájlok által beolvasott tartalommal kell frissíteni az adott adatbázist. A megvalósított szoftverben használhattam volna az alapértelmezett Java osztályokat a fájlműveletekhez, de ez a könyvtár jelentős mértékben megkönnyíti a munkát implementálás közben.

***JUnit***

A JUnit egy alapvető nyílt forráskódú keretrendszer a Java programozási nyelvhez, unit tesztelés céljából. Számos osztályt, annotációt és funkciót biztosít, hogy megfelelő teszteket írjon fejlesztő. A unit tesztelés támogatja a Test Driven Development (Teszt Vezérelt Fejlesztés) módszertan metodikáját, miszerint egy új funkció implementálása előtt megírjuk az ahhoz tartozó unit tesztet. Ezáltal a produkciós kód megírása nélkül is ismerhető a funkció elvárt működése. Ilyen formában a tesztek egyfajta specifikációként, dokumentációként is szolgálhatnak.

***Git***

A Git egy nyílt forráskódú, elosztott verziókezelő szoftver. Feladata, hogy a projekt fájljainak különböző verzióit tárolja és megossza a projekt felhasználói között. Azért döntöttem a Git használata mellett, mert implementálás közben nagyon hasznos, ha egy fájl korábbi verziójához szeretnék hozzáférni és használni. Véleményem szerint kevés olyan rendszer van, amely felveheti a versenyt a Gittel mind a hatékonyságban és mind az egyszerű kezelésben.

***Tervezés***

Az ETL mozaikszó nem egy ismeretlen fogalom az informatika világában. Olyan cégek és vállalatok számára létfontosságú lehet ezen alkalmazások használata, amelyek többek közt az üzleti intelligenciával (angolul Business Intelligence, röviden BI) is foglalkoznak, mint például bankok vagy nagyvállalatok. Ezek az alkalmazások lehetővé teszik, hogy információt nyerjünk ki és felhasználásukkal, olyan üzleti döntések születhetnek, amelyek növelik az üzleti teljesítményt. Belső üzleti folyamatok elemzése és optimalizálása után jelentős profit érhető el és előnyhöz juttathat a piaci versengésben. Az üzleti intelligencia jövője nem kétséges, hiszen múltbeli és jelenlegi elemzések is végezhetőek, valamint a jövőre tekintve is készülhetnek előrejelzések. Számos alkalmazási területe van. Ezek közé tartozik a riport és dashboard készítés, statisztikai elemzések létrehozása, valamint az adatbányászat.

***Rendszer terve***

A tervezés folyamata rendkívüli szerepet játszik a rendszer megvalósítása előtt. A rendszer tervezési fázisához hozzátartozik a kliens oldal és a szerver oldal megtervezése mellett a két oldal viszonyának koncepciója is. Továbbá nem elhanyagolható a grafikus felhasználói felület terve, hiszen a felhasználó ezen keresztül kommunikál az alkalmazással, különféle interakciókkal.

A specifikációból pontosan kiderül, hogy az alkalmazás egy adminisztrátori szerepkört betöltő felhasználónak készül, akinek hozzáférési joga van a Lavinia adatbázisához. Egyéb jogosultságú aktorok nem kapnak szerepet.

/\*\*\*\*\*\*\*\*\*\*

ÁBRA: USE-CASE DIAGRAM

hozzá egy kis szöveg

\*\*\*\*\*\*\*\*\*\*/

A szoftver több modulból fog felépülni. Ezek a modulok a későbbi implementációban reprezentálják a specifikációban említett szegmenseket és funkcionalitást. A rendszer főbb funkcióit megvalósító alkotórészei a következők:

* adatbázis interfész
* grafikus felhasználói interfész
* vezérlő osztályok
* modell osztályok
* naplózás vezérlő

Java nyelven nem okoz gondot elkülöníteni a különböző feladatokat ellátó modulokat. Az azonos feladatokért felelős osztályokat egy könyvtárba, úgy nevezett Java package-be lehet szervezni. Ily módon könnyebben átláthatók az implementált részegységek.

***Adatbázisok ismertetése***

A specifikációban említett adatbázisok meghatározottak voltak számomra, így új adatbázis létrehozására nem volt szükség. Tervezés során csak a meglévő adatbázisok felépítését és struktúráját kellett megismernem, valamint az USDA adatbázis frissítésének koncepcióját. Mindkét adatbázis a relációs adatmodell elve alapján épül fel.

***Relációs adatbázis***

A relációs adatmodell 1970-es megjelenése E. F. Codd nevéhez fűződik. Azóta meghatározó szerepet játszik az informatikában, többek közt az adattároló alkalmazásokban. A relációs adatszerkezet előnye, hogy könnyen értelmezhető mind a fejlesztők és mind a felhasználók számára is. Így akár tekinthetjük a kommunikáció eszközének is. További előnye, hogy egy adatmodell mögötti logikai relációit, bármilyen módosítás nélkül importálhatunk egy másik adatbáziskezelő rendszerbe. Ezen felül, az adatbázis tervezése meghatározott módon kivitelezhető a normál formák bevezetésével.

A relációt egy táblázattal lehet szemléltetni, amely sorról sorra tárolja a kívánt adatokat. A relációs adatbázis pedig ilyen relációk halmaza. A relációk egyedi nevet kell, hogy viseljenek ugyanúgy, mint az oszlopok egy reláción belül. A reláció architektúrája azonos struktúrájú rekordokat fejez ki, ezen sorok sorrendje nem fontos. Ezekben a sorokban tároljuk a logikailag összefüggő adatokat, viszont kettő vagy több megegyező sor nem fordulhat elő, ezzel elkerülve a redundanciát. Oszloponként csak meghatározott típusú mennyiségek tárolhatóak, például numerikus, szöveges vagy dátum formátumban. Egy relációnak megadhatunk kényszereket is, amelyek korlátozzák az adott adathalmazban előforduló értékeket.

A relációs adatbázis-kezelő rendszerek (angolul Relational Database Management System) kizárólag a relációs adatmodell alapján vannak megvalósítva. Ezen rendszerek segítségével építhetjük fel, kezelhetjük adatbázisainkat. Egy adatbázis-kezelő rendszer három rétegből épül fel. Legalsó réteg a fizikai réteg. A fizikai réteg definiálja az adat fizikai tárolását az adott hardver-egységen. Felette helyezkedik el a logikai réteg, amely a logikai kapcsolatokat határozza meg a tárolt információk között. Végül a legfelső szint a fogalmi vagy alkalmazási réteg. Ez biztosítja az adatokhoz való hozzáférési felületet a felhasználók részére. A relációs adatbázis-kezelő rendszerek nem valósulhatnak meg az ACID tulajdonságok nélkül. Ezen tulajdonságok az atomicitás (atomicity), a konzisztencia (consistency), az izoláció (isolation) és a tartósság (durability). Ezáltal rendszer garantálja, hogy több művelet egy műveletként hajtódjon végre, nem valósulhatnak meg részlegesen és ezek után is konzisztens állapotban marad. Tranzakciók egymástól függetlenül, elszeparáltan is futtathatóak és sikeres mentés után, hardver vagy egyéb hiba esetén is tartós marad a változás.

Néhány relációs adatbázis-kezelő rendszer, amelyek közt van kereskedelmi, illetve nyílt forráskódú is:

* Oracle
* MS SQL Server
* DB2
* PostgreSQL
* MySQL
* SQLite

***USDA***

Az USDA National Nutrient Database for Standard Reference adatbázisa a legfőbb adatforrás az élelmiszer összetételek tekintetében az egész Egyesült Államokban. A 90-es évek elejétől elektronikus formában is elérhető adatbázisuk. Ez az adatbázis minden egyes revíziónál letölthető fájlok alakjában találhatóak meg az USDA honlapján. Egy revízió publikálásánál egyrészt elérhető az újonnan frissült teljes adatbázis, amely tartalmazza a régi és az új adatokat egyaránt. Másrészt letölthetőek csak az előző verzióhoz képesti változások, amely magába foglalja az újonnan hozzáadott, megváltoztatott vagy törölt adatokat. Ezek mellett letölthető egy rövidített adatbázis szerkezet is szöveges illetve Microsoft Excel fájl formátumban. Viszont ez nem tartalmazza az összes tápanyagot és ily módon nem felhasználható a rendszer számára. A számomra felhasználható fájlok egyaránt rendelkezésre állnak ASCII (ISO/IEC 8859-1 szabvány) formátumban kódolva és Microsoft Access adatbázis formájában is.

Az USDA által használt relációs adatbázis főbb tábláit a következőkben mutatom be.

**Food Description, Nutrient Data, Weight**

|  |  |  |
| --- | --- | --- |
| **Mezőnév** | **Típus** | **Leírás** |
| NDB\_No | alfanumerikus |  |
| FdGrp\_Cd | alfanumerikus |  |
| Long\_Desc | alfanumerikus |  |
| Shrt\_desc | alfanumerikus |  |
| ComName | alfanumerikus |  |
| ManufacName | alfanumerikus |  |
| Survey | alfanumerikus |  |
| Ref\_desc | alfanumerikus |  |
| Refuse | numerikus |  |
| SciName | alfanumerikus |  |
| N\_Factor | numerikus |  |
| Pro\_Factor | numerikus |  |
| Fat\_Factor | numerikus |  |
| CHO\_Factor | numerikus |  |

|  |  |  |
| --- | --- | --- |
| **Mezőnév** | **Típus** | **Leírás** |
| NDB\_No | alfanumerikus |  |
| Nutr\_No | alfanumerikus |  |
| Nutr\_Val | numerikus |  |
| Num\_Data\_Pts | numerikus |  |
| Std\_Error | numerikus |  |
| Src\_Cd | alfanumerikus |  |
| Deriv\_Cd | alfanumerikus |  |
| Ref\_NDB\_No | alfanumerikus |  |
| Add\_Nutr\_Mark | alfanumerikus |  |
| Num\_Studies | numerikus |  |
| Min | numerikus |  |
| Max | numerikus |  |
| DF | numerikus |  |
| Low\_EB | numerikus |  |
| Up\_EB |  |  |
| Stat\_cmt | alfanumerikus |  |
| AddMod\_Date | alfanumerikus |  |
| CC | alfanumerikus |  |

|  |  |  |
| --- | --- | --- |
| **Mezőnév** | **Típus** | **Leírás** |
| NDB\_No | alfanumerikus |  |
| Seq | alfanumerikus |  |
| Amount | numerikus |  |
| Msre\_Desc | alfanumerikus |  |
| Gm\_Wgt | numerikus |  |
| Num\_Data\_Pts | numerikus |  |
| Std\_Dev | numerikus |  |

SR28 darabszám

|  |  |
| --- | --- |
| **Fájl** | **Darabszám** |
| ADD\_FOOD | 370 |
| ADD\_NUTR | 35 899 |
| ADD\_WGT | 538 |
| CHG\_FOOD | 753 |
| CHG\_NUTR | 14 271 |
| CHG\_WGT | 341 |
| DEL\_FOOD | 199 |
| DEL\_NUTR | 11 426 |
| DEL\_WGT | 328 |

***Lavinia***

/\*\*\*\*\*\*\*\*\*\*

kapott teszt adatok, postgres kezelőfelet, tanulmányozás

\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*

ÁBRA: RENDSZERTERV

\*\*\*\*\*\*\*\*\*\*/

***Megvalósítás***

***Tesztelés***

* verifikáció, validáció
* futásidő 10 / 100 elemre fájlonként