Miskolci Egyetem

Gépészmérnöki és Informatikai Kar

Általános Informatikai Intézeti Tanszék

**Konténerszállító hajók nyílvántartó rendszere**

**Szakdolgozat**

**Készítette:**

**Név:** Pogácsás Benedek

**Neptunkód:** FM4Z3B

**Szak:** Mérnök Informatikus BSc

Korszerű web technológiák szakirány

# 1. A feladat bemutatása

## A feladat leírása vázlatosan, nagy vonalakban:

* Adatbázist kezelő nyílvántartó rendszer a konténert szállító hajók kezelésére
* Java Spring Boot [1] keretrendszerben kerül elkészítésre
* A program csak egy modellje a nyilvántartó rendszereknek, élesben nem lesz használva
* Célom elérni, egyszerűen értelmezhető és könnyen navigálható felületet hozzak létre a projektnek
* Egyszerű kezelhetőségből követkevze szinte bárki képes a rendszert alkalmazni
* Bemenő adat a szükséges információ a résztvevő tényezőkről
* Kimenő adata hajók útjainak adatai
* Az alapvető CRUD műveleteken túl egyéb funkciókat is képes lesz végbe vinni
* Webes alkalmazás formájában lesz megvalósítva

## Módszer:

Egy ilyen rendszer elkészítése nem egyszerű feladat. Sok tényezőre kell odafigyelni, sok helyen elcsúszhat az egész. Ezért legelőször érdemes leülni és tervezni. Ebben a paragrafusban én is ezt fogom tenni.

A rendszer elsősorban adatbázis táblákat fog kezelni, ezekben fogja tárolni, lekérni, valamint alkalmazni a releváns információt. Tehát mindenképpen létre kell hozni egy adatbázist, amelynek táblái reprezentálják a számunkra értékes adatokat. Ilyenre példa lenne egy Hajók, Kikötők, Konténerek táblák lennének.

Ezután meg kell terveznünk a program felépítését, milyen osztályok létrehozására lesz szükségünk, valamint azok hogyan hatnak egymásra. Érdemes ezt grafikus alakban létrehozni, például egy UML diagramban. Ezzel könnyedén átlátjuk majd a rendszerünk felépítését. Itt fel tudjuk tüntetni osztályaink, valamint a hozzájuk tartozó változókat és metódusokat is.

Ezek sikeres elvégzése után elkezdhetünk kódot írni.

## Megvalósítási tanulmány:

|  |  |
| --- | --- |
| Teendők | Leírásuk |
| Adatbázis létrehozása | Érthető a feladatrész, jelenlegi tudással is megoldható. |
| Program felépítésének megtervezése | Érthető a feladatrész, utánajárás szükséges, a program felépítése nem tisztázott. |
| Táblák implementálása kódba | Érthető a feladatrész, jelenlegi tudással is megoldható.  A táblákhoz @Entity annotációval ellátott osztályok létrehozása, odafigyelve a táblák kapcsolataira is. |
| Lekérdezések implementálása | Nem (teljesen) érthető a feladatrész, később tisztázni kell, milyen lekérdezések szükségesek. |

# 2. A szakdolgozat bevezetőjének elkészítése

Szakdolgozatomnak egy webes nyilvántartó rendszer elkészítését választottam. Ennek több oka is van. Egyrészt ez a téma érdekelt, és bár még nem érzem teljesen magabiztosnak magam a témában, így olyan dologgal foglalkozhatok ami iránt van érdeklődésem, és egyszerre fejleszthetem a képességeimet. Egyetemi tanulmányaim során több tantárgyból is tanultunk ilyen rendszerekről, még a készítésükbe is belepillantottunk, így sikeresen elsajátítottam egy alapot, amelyre elkezdhetek építeni. Ha lesz lehetőségem rá, akkor tanulmányaim elvégzése után szívesen dolgoznék ebben a témában.

Ezen belül egy konténer szállító hajók adatait tároló és kezelő rendszert fogok készíteni. Ezt a témát is több indok miatt választottam. Egy egyetemi tantárgyamhoz kellett adatbázist létrehozni, és ugyan ez volt a témám, szóval úgy döntöttem hogy megint felhasználom ezt az ötletet. Abban előnyös ez a téma, hogy kifejezetten ritka, nem sok ember szokott pontosan ebben a témában készíteni projektet, viszont ehhez hasonló témák mégis gyakran előfordulnak. Itt például egy étel kiszállítási applikációra, vagy mondjuk egy posta szolgálat nyilvántartó rendszerére gondolok. Így ilyen szolgáltatásokat vizsgálhatok, megnézhetem hogyan működnek, és átalakíthatom, hogy a saját témámhoz illeszkedjen.

Egy szimulációnak is lehet tekinteni, ahol egy olyan vállalat rendszere kerül elkészítésre, amelyen keresztül rá látnak a saját hajóikra és az azon szállított rakományra, melyik hajójuk szabad, melyik van úton. A cél ehhez egy olyan weboldalas felület fejlesztése, amely képes ehhez tartozó adatok tárolására, valamint ezen adatokhoz hozzáadni, törölni vagy módosítani. Továbbá képes legyen a táblákból lekérdezni információt, és azzokkal több feladatot is végezni.

Ezek közé tartozik:

* A programnak minden hajó rakományát kezelnie kell
* Lehetnek speciális rakományok is, pl. élőállat, veszélyes anyag, hűtött áru
* Egyes konténerek pontos helyének a lekérése
* Helykihasználtság megoldása, egy hajó nem kaphat a limitjénél több konténert
* Bizonyos cégek konténereinek listázása
* Kikötőkhöz alapinformációk hozzárendelése

Környezetnek a Java Spring Boot keretrendszerét fogom használni. Az adatbázisomnak a MySQL [2] -t választottam.

# 3. Irodalom feldolgozása, háttérinformációk

## Nyílvántartó rendszerek általánosan:

A nyilvántartó rendszerek az idő folyamán rengeteget fejlődtek és változtak, mire elérték a mai napokban használt formájukat. Már az, hogy az emberek az elektronikus verzió létrejötte előtt is használtak ilyen rendszereket sokat mond el fontosságukról.

Az információ rendkívül értékes tud lenni, ezért ha olyan módon tudjuk struktúrálni, amely könnyen átlátható és a következtetések levonása is egyszerű, akkor a saját, vagy akár mások munkáját is lehetősen meg tudjuk könnyíteni. De most éppen az elektronikus, pontosabban webes nyilvántartó rendszerekről lenne szó, amik természetesen ugyanúgy rendelkeznel az előbb említett tulajdonsággal.

Ezek a rendszerek legtöbbször egy, vagy több, adatbázist alkalmaznak az adatok tárolására. Az egyik nagy előnye ennek, hogy egyszerűbbé teszi az információ elérhetőségét, egyszerre több dolgozó és felhasználó egyaránt képes hozzáférni anélkül, hogy kimozdulna a munkahelyéről vagy házából. Előnyt nyújt még, hogy az adatbázisok kezelése is kényelmesebb. SQL parancsok használatával gyorsan tudjuk módosítani, törölni, hozzáadni, illetve le is kérhetünk adatokat. Lehetőség van néhány folyamat automatizálására, valamint védelmet is kaphatunk jogosulatlan hozzáférés ellen is.

## Milyen **előnyei** vannak az ilyen rendszereknek?

A mai világban keletkező adatmennyiség robbanásszerűen nő. Ügyféladatok, terméklisták, szerződések, események, munkafolyamatok mind adatot generálnak, amelyeket már nem lehetne kézzel, papíron vagy egyszerű táblázatokban hatékonyan kezelni. A nyilvántartó rendszerek elsődleges célja, hogy ezeket az adatokat strukturáltan, átlátható módon tárolják és kezeljék.

Ezek az alkalmazások jelentősen növelik a munkafolyamatok hatékonyságát. Az automatizált adatrögzítés, módosítás és lekérdezés révén a felhasználók rengeteg időt takarítanak meg, így gyorsabban tudnak reagálni a változásokra. Egy online adatbázisból néhány kattintással kinyerhető az, amit korábban órákig kellett volna keresni papíralapú dokumentációban.

Minimalizálják az emberi hibákból adódó problémákat. Elkerülhetők a duplikált rögzítések, az elírások és a véletlen adatvesztések, amelyek manuális munkavégzés esetén gyakran előfordulnak. Egy jól megtervezett rendszer szabályokkal, jogosultságokkal és automatikus ellenőrzésekkel segíti a megbízható adatkezelést.

## Milyen **hátrányai** vannak az ilyen rendszereknek?

Bár a nyilvántartó rendszerek számos előnnyel járnak, nem szabad figyelmen kívül hagyni a lehetséges hátrányaikat sem.

Az egyik legnagyobb hátrány a magas kezdeti fejlesztési és bevezetési költség lehet. Egy egyedi fejlesztésű, testreszabott rendszer létrehozása sok időt, pénzt és szaktudást igényelhet, amelyet nem minden szervezet tud vagy akar vállalni.

A rendszerek üzemeltetése és karbantartása szintén folyamatos kiadással jár. Az adatbázisok frissítése, a hibák javítása, az új funkciók fejlesztése mind szakembereket igényel, ráadásul a biztonsági fenyegetések miatt is rendszeres ellenőrzésre és frissítésre van szükség. Ha ezt elhanyagolják, az súlyos adatvesztéshez vagy adatlopáshoz vezethet.

Az adatszivárgás és kibertámadások kockázata szintén komoly veszélyt jelent. Minél több adatot tárolunk digitálisan, annál vonzóbb célponttá válunk a hackerek számára. Egy biztonsági rés kihasználásával érzékeny, akár üzleti titkokat vagy ügyféladatokat is illetéktelenek szerezhetnek meg, ami óriási presztízsveszteséget és jogi következményeket is eredményezhet.

## Hol van helye a világban:

Ezek a rendszerek hasznosnak bizonyulnak akármikor információ tárolás és feldolgozásról van szó, ezért nagyon sok helyen alkalmazzák őket. A legegyértelműbb példa a vállalatok vagy cégek lennének. Ebben az esetben gyakran használnak ilyen rendszereket, hogy mondjuk az alkalmazottakról, ha termékeket árulnak akkor azokról tároljanak adatokat. Az utóbbi manapság egy weboldal segítségével valósul meg, ahol a felhasználók tudnak regisztrálni és vásárolni a termékek közül. Az elsőként említett, vállalat tagjainak adatát tároló rendszereknél kevésbé gyakori a weboldalas megoldás, de természetesen előfordulnak.

## Hasonló rendszerek megvizsgálása közelebbről:

Ehhez a feladat elvégzéséhez megpróbáltam olyan hasonló rendszereket keresni, emelyek segíthetnek a megoldásban. Így ugyanis megnézhetem, hogy ezek a való világi példák milyen módon hozták létre a rendszerüket, és milyen egyéb szolgáltatásokat nyújtanak. Természetesen nem tudom az egész rendszert átvizsgálni, mivel például az oldalak adatbáziaikhoz nincsen hozzáférésem, de ha még ha lenne is, nem hiszem hogy megoszthatnám egy szakdolgozaton belül.

Kutatásom végett több olyan oldalt sikerült találnom, amelyek vagy pontosan abban a témában készültek, amelyben én is készítem feladatomat, vagy nagyban hasonlítanak hozzá.

### INTTRA (by e2open):

Az első ilyen oldal amit találtam, az az INTTRA (by e2open) [4] volt. Ez a vállalat egy felhőalapú, semleges e-kereskedelmi platform, amely lehetővé teszi a felhasználók számára, hogy több hajóstársasággal egyetlen felületen keresztül kezeljék szállítmányaikat.

Szolgáltatásaik közül a legfontosabb természetesen a konténerszállítások elektronikus foglalása. Ezen belül egy platformon keresztül képesek több hajóstársasággal felvenni a kapcsolatot, megbeszélni a szállítás részleteit. Ezt a funkciót én is mindenképpen meg szeretném valósítani a saját rendszerembe, hiszen erre épül szinte minden.

Egy másik kiemelt szolgáltatásuk a valós idejű nyomon követés. Itt a szállítányt kérelmező követheti a weboldal segítségével, hogy a hajó merre jár, mennyire messze van a céljától. Ehhez mind természetesen csak a kezdeményező cég fér hozzá, a hajó biztonsága érdekében. Ez a funkció is nagyon hasznos tud lenni, viszont a feladatomhoz egy kicsit komplikált lehet a megoldása. Ez azért van, mivel az én rendszerem csak egy szimuláció, nem fog használatba kerülni tényleges cégek által, így azt is szimulálnom kéne, ahogyan a hajó az útján halad. Az ötlet viszont tetszik, és nagyon szívesen implementálom, ha találok rá megoldást, de jelen pillanatban nem vagyok magabiztos a képességeimben.

Utolsóként az oldalnak azt a szolgáltatását tekintettem meg, amely lehetővé teszi a vállalati rendszerek (pl. ERP, TMS) közvetlen összekapcsolását az INTTRA platformmal, támogatva az EDIFACT, ANSI és XML szabványokat [5]. Biztos vagyok benne, hogy ez is nagyon hasznos tud lenni az ügyek intézése során, egy kicsit feleslegesnek látom ezt megvalósítani a saját rendszeremben. Ennek az funkciónak az implementálása valószínűleg sokkal több időt venne igénybe, mint amennyit használatba lenne.

### Port of Los Angeles:

A következő vállalat amelynek weboldala nagyban hasonlít az én feladatomhoz, az a Port of Los Angeles [6]. Ez ugyanis a Los Angeles-i Kikötő hivatalos oldala, az Egyesült Államok legforgalmasabb tengeri kikötője, amely kulcsszerepet játszik a nemzetközi kereskedelemben, különösen az ázsiai relációkban. Itt viszont már megfigyelhető egy kis eltérés, hiszen ez csak egy kikötőnek a rakományát követi számon, míg az én rendszerembe ötbb kikötőről is lehet találni információt, beleértve az ott tárolt konténereket is. Viszont nyújtanak olyan szolgáltatásokat, amelyek inspiráltak.

Az INTTRA-hoz hasonlóan itt is lehet elektronikus környezetbe szállítást intézni, valamint a valós idejű áru követés is ugyanúgy megtalálható. Ezen felül nyújtanak olyan szolgáltatást hűtött tároló létesítményekkel [7], amelyek a kikötő közelében találhatók. Ezek a tárolók lehetővé teszik a romlandó áruk, például gyümölcsök, húsok és halak biztonságos és hatékony kezelését. Ez a funkció felkeltette az érdeklődésemet, viszont személy szerint egy kicsit másképpen használnám fel. Ahelyett, hogy ezek a tárolók a kikötőkben legyenek feljegyezve, inkább egyes hajókon lenne meg ez a funkció, de nem mindegyiken. Ezzel az elvégzendő szállításokhoz hozzá lehetne adni egy kis extrát, ha olyan terméket kell fuvarozni, amelynek igénye van hűtött környezetre, akkor legyenek olyan hajók, amelyekben ez megtalálható, és azokra tűzzük ki a feladatot. Ez nem csak a termék hőmérséklet igényeinél jöhet szóba, példáúl élő állatok száálításánál is hasonlóképpen kell eljárnunk, csak ehhez más hajótípusok lennének.

# 4. Saját munka leírása

## 4.1 Előkészületek

A kitűzött feladatom a következő:

* Egy webes nyílvántartó rendszer elkészítése, amely konténerszállító hajók és hozzájuk kapcsolódó adatokat kezel
* Az oldal elkészítésekor arra kell törekednem, hogy minél átláthatóbb, használata pedig egyértelmű és egyszerű legyen
* Szimulációként kell erre tekinteni, ez a program nem kerül tényleges használatra vállalatok által, és a benne feljegyzett cégek is csak a szolgáltatások bemutatásra szolgálnak
* Az adatok egy adatbázisban lesznek tárolva
* Adatok módosítására az oldalon keresztül lesz lehetőség, felhasználó hozzáféréstől függően
* Felhasználói olyan cégek lennének, akik hajó úton szeretnék termékeiket vagy tulajdonukat szállítani a világon
* A cégekből felelős személynek nem lesz szüksége különösebb képzésre, hogy ezt a weboldalas szolgáltatást sikeresen használja, de nem árt olyan embert kitűzni, aki jártas adatok átvitelében és kezelésében
* Regisztrálni felhasználóként önmaguktól nem lehet, e-mail címen vagy telefonon kell felvenni a kapcsolatot, amit követően a rendszer adminja felveszi a céget a rendszerbe, végül visszaküldi a felhasználó nevet és jelszót, amivel beléphet a rendszerbe
* A felhasználónév és jelszó belépés után megváltoztatható
* Egy cég csak a saját rakományát és szállítási kérvényét módosíthatja, addig amíg az el nem indult útjára, utána ezt le kell tiltani
* Az admin hozzáfér az összes táblához és akármilyen műveletet képes rajta végrehajtani
* Ezek a cégek a szállítandó értékeiket különböző, hozzájuk legközelebbi kikötőhöz szállítják, így bekerül a hajókat kezelő vállalat adatbázisába
* Az oldalon keresztül kérvényezhetik, hogy melyik terméket vagy értéket hova szállítsák el
* Ezek után az admin kijelöl egy hajó kapitányt a rakomány célhoz szállítására

Szállításnál figyelembe kell venni a szállított rakomány típusát és súlyát:

* Minden hajónak meg van adva, hogy pontosan mennyit bír szállítani, tehát szükséges megoldani azt a programban, hogy ne lehessen többet bízni egy hajóra, mint amennyit képes szállítani.
* Típus esetén arra kell gondolni, hogy a rakomány karbantartása igényel-e speciális odafigyelést. Ebből kettő fajtát különböztetünk meg:
  + Az áltagos szállítmány, amelynek nincsen különösen fontos igénye. Bármilyen hajó teljesen megfelel a célba szállításhoz.
  + Különleges áru, ami további feltételek teljesítésével szállítható csak. Például olyan termékek szállításához, amelynek hűtött környezetre van szüksége, vagy esetleg állatok szállításához a megfelelő méretű helyet és gondozók felfogadása szükséges. Ezek számára vannak hajók, amelyek képesek az előbb elmített feltételek teljesítésére, de sokkal kevesebb van belőlük.

Ezeken túl a program képes lesz további lekérdezésekre is, ezek a következők:

* Átlalános CRUD műveletek, vagyis Create, Read, Update és Delete, az összes táblára kezeléshez
* Kérvény kezelésénél a rakomány összegzett súlya és a hajó típusa látható lesz, ezzel könnyebb lesz a feladatra alkalmas rakomány szállítót kiválasztani
* Lehetséges lesz egyes konténerek lekérése helyük szerint
* Egyes kikötőkben lévő rakomány számának lekérése
* Melyik kapitány kapta eddig a legtöbb munkát
* Egyéb információ lekérése...

A fent felsorolt feltételek végrehajtásával készülne el a feladatom. Az ezekhez használt környezetek és technológiák:

* Az oldal hátsó kódjához Java nyelven lesz megírba, azon belül pedig a Spring Boot keretrendszert fogok használni
* Az adatok tárolásához egy MySQL adatbázist kerül majd létrehozásra
* A weboldal külsejének kialakításához pedig a népszerű és jól ismert HTML5 és CSS technológiák lesznek használva

## 4.2 Programspecifikáció:

### Adatok kezelése:

Bemenő adatok alatt a programban azokra lehet gondolni, amelyek bekerülnek a rendszerbe valamilyen módon. Ez lehetséges az admin által vagy egyes cégek által. Az egész program ezeken az információkon függ. Innen tudja előállítani a szállítási kérelmeket, melyik kikötőbe történjen a szállítás, melyik kapitány vigye végbe a feladatot, melyik konténereket kerüljenek elvitelre, és azok mit tartalmaznak.

Kimenő adat nem igazán lesz abban az értelemben, hogy a program az adatbázisban tárolt információból egy külön fájlt készít. Erre nem igazán van szükség, mivel a konténer szállító cég és a velük együttműködő cégek információit tárolják.

Minden bevitt adat egy adatbázisban lesz eltárolva, tehát a feladat elvégzéséhez ennek kell elkészíteni a struktúráját olyan módon, hogy az megfeleljen az előírt elvárásoknak. Viszont mielőtt létrehoznánk ezt az adatbázist, érdemes először leülni és megtervezni a szerkezetét. Erre többféle módszer is létezik, de én első sorban egy Entity-Relationship Modellt (ER Model) készítettem, ami a következőképpen néz ki:

A black background with white squares

AI-generated content may be incorrect.

. ábra - Az adatbázisom ER Modellje

Ez a modellt az erdplus.com[3] online szolgáltatás segítségével hoztam létre.

Ebből a modellből szépen látható, hogy milyen táblák lesznek szükségesek, azok hogyan kapcsolódnak a többi táblákhoz, és mind ezeknek milyen attribútumokkal rendelkezniük. A modell elég terjedelmes lett, ezért ebből a képből lehet, hogy nem teljesen kilátható minden. Ezért ehhez az ER modellhez készítettem egy sémát is, ami pontosabban megmutatja nekünk az adatbázis szerkezetét. Ez a következő féleképpen néz ki:

A screenshot of a computer screen

AI-generated content may be incorrect.Hasonlóan az ER modellhez, ez a séma is az erdplus.com online szolgáltatás segítségével készült el.

2. ábra - Az adatbázisom sémája

A táblák és attribútumaik leírása:

* **Ship:** A hajók információit tároló tábla.
  + ship\_id (PK): A hajó egyedi azonosítója
  + ship\_brand: A hajó márkája
  + ship\_weigth\_limit: A hajó által elbírt tehet maximuma
  + ship\_type: A hajó típusa, az alapján hogy milyen árut képes szállítani
* **Captain:** A kapitányok információit tároló tábla. Ez a tábla Egy-Egy kapcsolatba van a **Ship** táblával, mivel azt feltételezzük, hogy minden kapitánynak saját hajója van, ezt használják a konténerek szállításához.
  + captain\_id (PK): A kapitány egyedi azonosítója
  + captain\_name: A kapitány neve
  + captain\_nationality: A kapitány nemzetisége
* **Port:** A kikötők adatait tartalamzó tábla. Ez a tábla Egy-Több kapcsolatba van a **Cargo** és a **Request** táblákkal.
  + port\_id (PK): A kikötő egyedi azonosítója
  + port\_country: A kikötő országa
  + port\_city: A város amelyben a kikötő megtalálható
* **Company:** A cégek adatait tartalmazó tábla.
  + company\_id (PK): A cég egyedi azonosítója
  + company\_name: A cég neve
  + company\_industry: Az ipar amelyben a cég üzemel
* **Cargo:** A konténerek információit tároló tábla. Ez a tábla Egy-Több kapcsolatba van a **Company** táblával, valamint Több-Több kapcsolatba a **Request** táblával.
  + cargo\_id (PK): A konténer egyedi azonosítója
  + cargo\_weight: A konténer súlya
  + cargo\_type: A konténer típusa, átlagos vagy különleges
  + cargo\_content: A konténer tartalma
  + company\_id (FK): A cég amelyhez tartozik a konténer
  + port\_id (FK): A kikötő ahol elhelyezkedik a konténer
* **Request:** A kérvények információit tároló tábla. Ez a tábla Egy-Egy kapcsolatba van az **Order** táblával.
  + request\_id (PK): A kérvény egyedi azonosítója
  + request\_date: A kérvény beadásának dátuma
  + request\_delivery\_date: A kérvényezett áru megérkezésénem időpontja
  + company\_id (FK): A cég amelyik beadta a kérvényt
  + port\_id (FK): A kikötő ahova szállítani kell a rakományt
* **request\_cargo:** A tábla amely tartalmazza, hogy egyes kérvényekhez melyik konténerek vannak hozzárendelve. Ez a tábla felelős a **Cargo** és a **Request** táblák közötti Több-Több kapcsolat nyomon követéséért.
  + request\_id (FK): A kérvény azonosítója
  + company\_id (FK): A cég amelyhez tartozik a konténer
* **Order:** A rendelések adatait tartalmazó tábla. Itt a kérvények kiegészülnek az árut szállító megfelelő hajóval.
  + order\_id (PK): A rendelés egyedi azonosítója
  + ship\_id (FK): Az a hajó azonosítója, amelyik szállítani fogja a konténereket
* **Order\_Status:** A rendelés státuszának nyilvántartó táblája. Egy-Egy kapcsolatba van az **Order** táblával.
  + os\_id (PK): A rendelés egyedi azonosítója
  + os\_value: A rendelés státuszának értéke, úton vagy végbevitt

### Osztálydiagram:

Az adatbázis megtervezésén kívül fontos a backend osztályok kapcsolatait és felépítésüket is előre megfontolni. Ennek szemléltetésének egyik módja egy osztálydiagramm készítése. Ehhez az alábbi ábrát hoztam létre:

A diagram of a company

AI-generated content may be incorrect.

3. ábra - Vázlatos osztálydiagramm

Ez a kép a Visual Paradigm Online [8] webes alkalmazás segítségével hoztam létre.

A fenti ábra nagyon vázlatos, kezdetleges, de jól feltünteti azokat az osztályokat, amelyek létrehozására szükségem lesz, valamint azok kapcsolatait. Ez az ábra viszont még nagyon messze áll a tényleges rendszer bemutatásától, nem csak az osztályok változóik és függvényeik hiánya miatt.

A kapcsolatok leírása:

* Az **Order** tábla a **Request** leszármazottja. Ezzel örökli azokat az adatokat, amelyeket kérvényeztek a konténerek szállításához, és kiegészül egy státus értékkel és az árut szállító hajóval.
* Az **Order\_Status** kompozíciós kapcsolatba van az **Order** táblával. Önmagában nem létezhet, kapcsolódnia kell egy **Order**-hez.
* A **request\_cargo** kompozíciós kapcsolatba van a **Request** és a **Cargo** táblákkal. Ez tarlamazza azokat azt adatokat, hogy egy kérvénybe melyik konténerek kerüljene szállításra. Mindkét táblára szüksége van a létezéshez.

### Képernyőtervek és Használati mód:

A program nyitó oldala egy bemutatkozással indítana a vállalatról, mivel foglalkozik és milyen szolgáltatásokat tud nyújtani. Az oldal alján megtalálható a vállalat e-mail címe és telefon száma, ezen keresztül veszik fel a kapcsolatot a regisztrációhoz.

Az oldalak színkészleténél nélkülözhetetlenek az olyan alapszínek mint a fekete, fehér és szürke, ezek gyakran lesznek használva. Ezen túl figyelembe véve a weboldal témáját, a két is sokszor elő fog fordulni, mindenféle árnyalatban. Bár kevésbé gyakran, de a zöld és sárga színek is megtalálhatóak lesznek. Betűméretek szempontjából nem lesz semmilyen különleges alkalmazás. Fontosabb címek nagyobb mérettel jelennek meg, a leíró szöveg hagyományos 12-es nagyságú. Esetleges kiemelések, aláhuzások, betűdöntések és betű beszínezése is előfordul, de ritka.

Egy felső navigációs léc minden oldalon meg lesz található. A nyitó oldal léce csak egy ikont fog tartalmazni a bal felső sarokban, valamint egy Bejelentkezés gombot a jobb felső sarokban. Erre kattintva továbbjutunk a Bejelentkezés képernyőjére. Itt továbbléphetünk egy cég felhasználójával vagy az admin felhasználóval.

Cégként belépve a navigációs lécen több opció is megjelenik. Ezek különféle oldalakhoz visznek, amelyek a céghez tartozó információkat sorolja fel, vagy új adat felviteléhez ad lehetőséget. Ilyen a céghez tartozó rakományok adatait, azok módosítása, törlése vagy akár új felvétele, leadott kérvények adatai, új kérvény leadása, valamint a felhasználó adatainak változtatása, tehát név és jelszó módosításához opciók.

Adminként belépve mégtöbb opció jelenik meg a navigációs sávunkon. Itt hozzáférünk az összes táblához, és akármilyen műveletet képesek vagyunk rajtuk végrehajtani. Ezen túl itt is megjelenik a felhasználói adat módosításának opciója, a kényelem kedvéért. Lesz egy külön opció a kérvények kezeléséhez és elindításához. Ekkor kell a még nem elindított kérvényekhez egy olyan hajót rendelni, amely megfelel az előírt követelményeknek. Ezen túl több specifikus információt tartalmazó gombhoz is hozzá fogunk tudni férni.

Az adatokat lekérdező oldalak javarészt a táblák feltüntetéséből fognak állni, minden adat egy törlés és frissítés gombbal kiegészítve. Az oldal tetején, de a navigációs léc alatt, olvasható lesz egy leírás a tábla tartalmáról. Ettől nem messze egy „új” gombot lehet majd találni, aminem segítségével új adatot tudunk felvinni az adatbázisunkba.

Bejelentkezés után a jobb felső sarokban lévő gomb lecserélődik egy Kijelentkezés gombra. Ezt megnyomva kilépünk a belépett felhasználóból, és visszakerülünk a nyitó oldalra.

### Hardver és Szoftver környezet bemutatása:

Hardver terén szerencsére nem szükséges semmilyen bővítmény vagy speciális kiegészítő. Mivel egy webes alkalmazást készítek, ezért nagy valószínűséggel akárki képes lenne elérni az oldalt, hardvertől függetlenül.

Szofter környezetének terén is nagy szerencsém van, mivel az általam használt alkalmazások vagy teljesen ingyenesek, vagy az egyetemi felhasználóm által nyertem ingyenes hozzáférést. Ezért ebből a szempontból sincs olyan sok teendőm.

## 4.3 A felhasználható fejleszőeszközök kiválasztása

### Adatbázis kiválasztása:

A webes alkalmazásom adatbázisának kiválasztásához megvizsgáltam több kezelő rendszert, hogy a feladatom elvégzéséhez a legjobban illőt válasszam. A következő adatbáziknak néztem utána részletesen:

#### *PostgreSQL*:

A PostgreSQL [9] egy nyílt forráskódú, objektum-relációs adatbázis-kezelő rendszer, amelyet eredetileg az UC Berkeley fejlesztett ki (POSTGRES projekt néven). Ma az egyik legfejlettebb és legmegbízhatóbb adatbázis-kezelő rendszerként tartják számon.

Főbb jellemzői:

* Elsődleges kulcsok, idegen kulcsok, CHECK és NOT NULL megszorítások mind megtalálhatóak
* A PostgreSQL a klasszikus relációs modellt (táblák, sorok, kulcsok) ötvözi **objektum-orientált funkciókkal**:
  + Öröklődés táblák között
  + Felhasználói adattípusok
  + Komplex adatszerkezetek tárolása
* Képes garantálni az ACID elv jellemzőit
* Definiálhatunk saját triggereket
* Többféle adatstruktúrát is támogat, például: JSON, XML
* Felhasználó is készíthet függvényeket és eljárásokat a PL/pgSQL segítségével
* Teljes Spring Boot támogatás

#### MySQL:

A MySQL [10] egy nyílt forráskódú relációs adatbázis-kezelő rendszer, amelyet eredetileg a MySQL AB fejlesztett, majd a Sun Microsystems, később pedig az Oracle Corporation tulajdonába került. A világ egyik legismertebb és legszélesebb körben használt adatbázisa, különösen népszerű webes fejlesztésben.

Főbb jellemzői:

* Az adatokat strukturált táblákban tárolja, ahol sorok és oszlopok határozzák meg a szerkezetet, kapcsolatok definiálhatók idegen kulcsokkal (FOREIGN KEY)
* Alapértelmezett tárolómotorja az InnoDB, amely biztosít záráskezelést, és támogatja az ACID-tranzakciók, valamint -evlek jelenlétét
* Támogatott az SSL, IP-cím alapú korlátozás
* Felhasználható hozzá a MySQL Workbench [11], egy grafikus bővítmény, amelynek segítségével könnyebben átláthatjuk az adatbázisunkat
* Ad indexelési lehetőségeket, amelyek a következők: PRIMARY KEY, UNIQUE, FULLTEXT, INDEX

#### Apache CouchDB:

Az Apache CouchDB [12] egy nyílt forráskódú, dokumentum-orientált NoSQL adatbázis-kezelő rendszer, amelyet az Apache Software Foundation fejleszt. A CouchDB nem relációs adatbázis, hanem JSON-alapú dokumentumokat tárol és RESTful HTTP API-n keresztül kommunikál. Az egyik fő erőssége a replikáció és offline szinkronizáció.

Főbb jellemzői:

* Az adatokat JSON dokumentumokban tárolja. Minden dokumentumnak van egy egyedi „\_id” és verziót követő „\_rev” mezője. A dokumentumok bármilyen sémával rendelkezhetnek, de van lehetőség séma nélküli tárolásra is.
* A CouchDB minden CRUD műveletet HTTP-kéréseken keresztül végez a GET, PUT, POST, DELETE parancsok segítségével.
* Nincsenek SQL-lekérdezések. MapReduce függvényekkel hozhatsz létre „nézeteket” (views) JavaScript-ben:
  + map: kulcs-érték párokat generál a dokumentumokból
  + reduce: aggregálja az eredményeket (összegzés, számlálás stb.)
* A CouchDB egyik legerősebb funkciója a kétirányú replikáció: offline módon is működik, majd szinkronizál. Ez ideális mobilalkalmazásokhoz vagy edge computinghoz.
* Képes konfliktuskezelésre, dokumentumszintű verziókövetés.

#### H2:

A H2 [13] egy nyílt forráskódú, tisztán Java nyelven írt relációs adatbázis-kezelő rendszer. Beágyazottan futtatható, vagyis az alkalmazással egy folyamaton belül működik. A H2 adatbázist gyakran használják tesztelési célokra, könnyű fejlesztési környezetekben, valamint kis alkalmazásokhoz.

Főbb Jellemzői:

* Működése nem függ semmilyen platformtól, bármely JVM környezetben fut.
* Könnyű integrálni Java/Spring projektekbe, nincs szükség külső telepítésre.
* Akár kétféle módban is futtatható:
* Embedded mode: a H2 a Java alkalmazás részeként fut, fájl- vagy memóriamódban.
* Server mode: más alkalmazások is csatlakozhatnak JDBC-n keresztül.
* Spring Boot és Hibernate automatikusan felismeri és inicializálja a H2 adatbázisokat

#### Döntésem:

Ennek a döntésnek a meghozatala felettébb nehéz volt. Mind a négy adatbázis kezelő-rendszerben volt valami, ami hasznos lett volna a számomra, így akármelyik segítségével sikeresen eltudnám készíteni a projektemet. De a legszimpatikusabb nekem a PostgreSQL és a MySQL volt, ezek egy picivel több előnyt nyújtottak.

Végül a döntés inkább azon múlt, hogy mennyi tapaszalatom volt a két kezelő rendszerrel. Mivel eddig egyetemi tanulmányaim során akármikor egy adabázist kellett létrehoznom, szinte mindig MySQL mellett döntöttem, és ez most sem fog változni.

### Weboldal felületének elkészítése:

A weboldal kinézetét a már nagyon jól ismert HTML és CSS segítségével fogom kialakítani.

### Backend technológiájának kiválasztása:

Ennek a kiválasztásánál is több programozási nyelv állt rendelkezésemre. Szempontok szerint olyan nyelveket próbáltam keresni, amelyek nagy segítséget nyújtottak egy webes alkalmazás elkészítésénel. Ezért következőket vettem megfontolás alá:

* **Java [14]**: Platformfüggetlen, objektum-orientált programozási nyelv, amelyet elsősorban nagy megbízhatóságú, hordozható és biztonságos alkalmazások fejlesztésére használnak. Világszerte népszerű választás vállalati és webes szoftverfejlesztéshez.
* **C# [15]**: Egy modern, objektum-orientált programozási nyelv, amelyet a Microsoft fejlesztett ki a .NET platform részeként. Főként asztali, webes és mobilalkalmazások fejlesztésére használják, erőteljes nyelvi eszközökkel és kiváló integrációval a Windows-ökoszisztémában.
* **C++ [16]**: Nagy teljesítményű, általános célú programozási nyelv, amely támogatja az objektum-orientált és a procedurális programozást is. Gyakran használják rendszerszintű szoftverek, játékok, beágyazott rendszerek és nagy teljesítményű alkalmazások fejlesztésére.
* **Javascript [17]**: Egy dinamikus, magas szintű programozási nyelv, amelyet elsősorban weboldalak interaktív elemeinek fejlesztésére használnak. A modern JavaScript lehetővé teszi a kliens- és szerveroldali alkalmazások létrehozását is, például Node.js segítségével.

Alapos átgondolás után a **Java** nyelv mellett döntöttem. Ennek kettő indoka is volt:

Eddigi informatikus tanulmányaim során ezt a nyelvet használtam a legtöbbet, ezért ebben éreztem magam a legmagabiztosabbnak.

Ez a nyelv rendelkezik egy Spring Boot nevezetű keretrendszerrel, amely leegyszerűsíti a Spring-alapú alkalmazások fejlesztését az automatikus konfiguráció, beépített szerverek és minimális boilerplate kód révén. Segítségével gyorsan, kevesebb konfigurációval lehet robusztus, önálló webes alkalmazásokat készíteni.

### Fejlesztői környezet kiválasztása:

### Projektmenedzsment eszköz kiválasztása:

A Spring Boot projektek legtöbbször Maven-t vagy Gradle-t használnak build és projektmenedszment eszközeként.

#### Maven:

A Maven [18] egy build automatizáló és projektmenedzsment eszköz, amelyet elsősorban Java alapú projektekhez használnak. A legfontosabb célja a projekt függőségeinek (könyvtárak, eszközök) kezelése, valamint a fordítás, tesztelés, csomagolás és telepítés folyamatainak automatizálása. Apache Maven a hivatalos neve.

A Maven elterjedt választás, mert:

* Egyszerű konfiguráció (pom.xml)
* Hatékony függőségkezelés a Spring Boot starter csomagokkal
* Jó integráció IDE-kkel (pl. IntelliJ, Eclipse)

XML alapú konfigurációval dolgozik, a legfontosabb, központi konfigurációs fájlja: pom.xml (Project Object Model). Itt az alkalmazásunk fontosabb adatait találhatjuk meg, amelyek közé tartoznak a Kényszerek (Dependencies).

|  |  |
| --- | --- |
| Előnyök: | Hátrányok: |
| * Központi függőségkezelés * Verziókezelés egyszerűsége * Transzitív függőségek automatikus kezelése * Közösségi támogatás és bővítmények | * Verzióütközések a transzitív függőségek miatt * Túl sok plugin lassíthatja a buildet * IDE cache problémák |

#### Gradle

A Gradle [19] egy modern, nagy teljesítményű build automatizációs eszköz, amelyet elsősorban Java-alapú projektekhez használnak, de támogat más nyelveket is (pl. Kotlin, Groovy, Scala, C++, stb.). A Gradle lehetővé teszi, hogy összetett projekteket hatékonyan lehessen fordítani, tesztelni, csomagolni, dokumentálni és telepíteni.

|  |  |
| --- | --- |
| Előnyök: | Hátrányok: |
| * Nagyon gyors build idő * Könnyen bővíthető * IDE barát (IntelliJ, Eclipse támogatás) * Nagy és aktív közösség | * Összetett konfiguráció (főleg nagy projekteknél) * DSL nyelv elsajátítása szükséges (Groovy/Kotlin) |

#### Maven vs Gradle:

Egymás mellett a két build automatizációs eszközt egyszerű összevetni, ezzel döntésemet is jobban át tudom gondolni. Nyilván mindkettőnek megvannak az előnyei, de én a **Maven** mellett döntöttem.

Ennek az oka a konfigurációjának az egyszerűségéből ered, minden megtalálható egy fájlban. Ennek egy másik következménye a függőségek kezelésének tisztasága is.

## 4.4 A terv ismertetése

Források:

[1] Spring Boot: <https://spring.io/projects/spring-boot> (2025)

[2] MySQL: <https://www.mysql.com/> (2025)

[3] erdplus: <https://erdplus.com/> (2025.05.17)

[4] INTTRA (by e2open): <https://www.inttra.com/> (2025.05.21)

[5] INTTRA (by e2open): Integration Services: <https://www.inttra.com/services/integration/> (2025.05.21)

[6] Port of Los Angeles: <https://www.portoflosangeles.org/> (2025.05.21)

[7] Port of Los Angeles: Cold Storage Facilities: <https://www.portoflosangeles.org/business/supply-chain/warehousing-and-distribution> (2025.05.21)

[8] Visual Paradigm Online: <https://online.visual-paradigm.com/diagrams/solutions/free-class-diagram-tool/> (2025.05.22)

[9] PostgreSQL: <https://www.postgresql.org/about/> (2025.05.22)

[10] MySQL: <https://www.mysql.com/> (2025.05.22)

[11] MySQL Workbench: <https://www.mysql.com/products/workbench/> (2025.05.22)

[12] Apache CouchDB: <https://couchdb.apache.org/> (2025.05.22)

[13] H2: <https://www.h2database.com/html/main.html> (2025.05.22)

[14] Java: <https://www.java.com/en/download/help/whatis_java.html> (2025.05.22)

[15] C#: <https://dotnet.microsoft.com/en-us/languages/csharp> (2025.05.22)

[16] C++: <https://isocpp.org/> (2025.05.22)

[17] Javascript: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (2025.05.22)

[18] Maven: <https://docs.spring.io/spring-boot/maven-plugin/index.html> (2025.05.22)

[19] Gradle: <https://docs.gradle.org/current/userguide/userguide.html> (2025.05.22)