



automated
management
of a
zillion
needs

amzn

dokumentáció

Feladatleírás

A félév során egy szimulációs programot kell megvalósítani 3 fős csapatban. A szimulációs programnak egy raktárban dolgozó szállító robotok útvonalát kell megterveznie és a tervet végrehajtania.

Egy raktár tipikus alaprajza:

D ₄		P	P		P	P		P	P		
		P	P 1,3,4		P	P		P	P		
D ₃		P	P		P	P		P	P		
		P	P		P	P		P	P		
D ₂		P	P		P	P		P	P		
		P	P		P	P		P	P		
D ₁		P	P		P	P		P	P		
											R ₃
											R ₂
											R ₁
		S ₁		S ₂		S ₃		S ₄			

A raktár négyzethálóba van szervezve. Vannak benne akkumulátoros robotok (*R*), célállomások (*S*), dokkolónak nevezett töltőállomások (*D*), és podnak nevezett állványok (*P*). A robotok akkumulátorának van egy maximális töltöttségi állapota, ami egy egész szám. Az állványokon termékek vannak, amit a fenti ábrán az állványra írt számok jelölnek. Egy állványon minden termékszámából legfeljebb egy szerepelhet. A termékszámok a célállomás számok közül kerülnek ki. A robotok feladata, hogy a termékeket az azonos számú célállomásokhoz vigyék (1-es termék az *S*₁ célállomáshoz, 2-es az *S*₂-höz, stb.).

Egy robot úgy tud egy terméket elvinni, hogy az állvány alá áll, megemeli az állványt, majd utána az állvánnyal együtt mozog. Egy adott számú termék akkor kerül a célállomásra, ha a robot ráállt az állvánnyal az azonos számú célállomásra, majd a terméket leadja, és ezzel a termék eltűnik az állványról. A termék (vagy termékek) célállomásra szállítása után a robotoknak az állványt vissza kell vinniük az eredeti helyükre.

A robotnak be kell tartania a robotmozgás szabályait:

- A robotok az állványok alatt el tudnak menni.
- Egy négyzetben sohasem lehet egyszerre két robot.
- Egy négyzetben sohasem lehet egyszerre két állvány.
- A robotok állvánnyal nem állhatnak töltőállomásra.
- Mindig gondoskodni kell arról, hogy a robot az akkumulátorának lemerülése előtt el tudjon jutni egy töltőállomásra.

A szimuláció lépésekben történik. Egy központi vezérlő jelöli ki, hogy melyik robot melyik terméket vigye el a célállomására. Tervezéstől függően, a robotok vagy önállóan dolgoznak, vagy egy központi vezérlő mondja meg, hogy melyik lépésben milyen műveletet végezzenek. A központi vezérlő és a robotok is az egész raktár állapotát ismerik. Minden lépésben minden robot végezhet egy műveletet, ami (a termék leadás műveletét leszámítva) eggyel csökkenti az akkumulátor töltöttségi állapotát.

A lehetséges műveletek:

- kilencven fokos fordulat jobbra vagy balra,
- átlépés a menetirány szerinti él-szomszédos négyzetbe (végrehajtható, ha induláskor a cél négyzetben nincs se robot se fal, és betartja a robotmozgás szabályait),
- állvány megemelése (végrehajtható, ha az állvánnyal egy mezőben áll, és innentől együtt mozognak),
- állvány letevése (innentől az állvány nem mozdul, csak a robot mozoghat),
- termék leadása (sikeres, ha megfelelő célállomáson áll, ez a művelet nem csökkenti az akkumulátor töltöttségét),
- akkumulátor töltésének elindítása (végrehajtható, ha töltőállomáson áll, és akkor utána még öt lépésig a robot nem mozdulhat a töltőállomásról, ami alatt teljesen feltöltődik az akkumulátora).

A szimuláció indításához meg kell adni:

- a raktár elrendezését (méretek, állványok, célállomások, dokkolók helye)
- robotok száma és helye, maximális töltöttség (tegyük fel, hogy kezdetben minden robot teljesen feltöltött)
- kiszállítandó termékek helye (melyik állványon vannak)
- (esetleg egyéb konfigurációs adatok, amik segíthetik a megoldást)

Az állítható sebességű szimuláció során szeretnénk látni a raktár alaprajzát, a robotok mozgását, a robotok telítettségi állapotát, a termékeket, az állványokat és mozgásukat.

A szimuláció végén egy napló fájlban meg akarjuk kapni:

- hány lépésig tartott a teljes feladat végrehajtása,
- minden egyes robotra, hogy összesen mennyi energiát fogyasztott,
- összesen mennyi energia kellett a feladat végrehajtásához.

Részfeladatok

- **Alap (elégéshez kell):** szövegfájlban lehet megadni a szimuláció indításához szükséges adatokat, a robotok minimális szinten tudják kezelni a konfliktusokat, a robotok elszállítják a termékeket a célállomásokhoz, statisztika készül.
- **Input adatok szerkesztése (5 pont):** egy külön szerkesztővel lehet létrehozni a szimuláció indításához szükséges adatokat. A szerkesztett adatok elmenthetők és betölthetők.
- **Kifinomult szerkesztő (5 pont):** A raktár szerkesztő lehetővé teszi, hogy kijelöljünk polcokat és együtt áthelyezzük őket. Kijelölt polcokon egyszerre ugyanazokat a termékeket helyezhetjük el.
- **Termékek robotokhoz rendelése kifinomult (5 pont):** optimalizálás történik.
- **Online megrendelések (5 pont):** kiszállítandó termékeket a szimuláció futása közben interaktív módon is ki lehet jelölni a polcokon. Esetleg az input fájlban van megadva, hogy mikor jelenjenek meg új megrendelések.
- **Útvonal keresés (5 pont):** kifinomultabb algoritmust használ, esetleg a konfliktusok elkerülésére is figyel.
- **Hálózati megvalósítás (15 pont):** Legyen lehetőség arra, hogy a szimuláció sebességének állítását, megállítását (és ha van, akkor az online megrendeléseket) egy kliens számítógépen lehessen állítani. A szimuláció egy másik gépen fut.

A szoftver célja, előnyei

- A cég számára kiadásokat spórolunk meg: a robotok karbantartása olcsóbb, mint emberek alkalmazása (munkabér + biztosítás + szabadság stb.).
- A robotok nincsenek munkaidőhöz kötve, akármikor dolgozhatnak, így gyorsul a munkafolyamat.
- Nincs lehetőség emberi hibára, a robotok precíz munkát végeznek.

Mindehhez természetesen elengedhetetlen, hogy a robotok valóban precízen dolgozzanak, megfelelő utat keressenek két pont között, ne ütközzenek össze egymással vagy más polcokkal, ne merüljenek le munka közben. Ennek a biztosítására jött létre ez a szimulációs projekt, hiszen így biztonságos virtuális környezetben tesztelhetjük és tökéletesíthetjük a robotok algoritmusát, valamint a naplózott adatokból statisztikákat készíthetünk a várható megtakarítások becsléséhez.

Elemzés

A feladatot **C# WPF** (Windows Presentation Foundation) **MVVM** architektúrában valósítjuk meg. A modell/nézet/nézetmodell architektúra leválasztja a felhasználói interakció kezelését, valamint az adatmegjelenítést a felülettől.

Mérföldkövek:

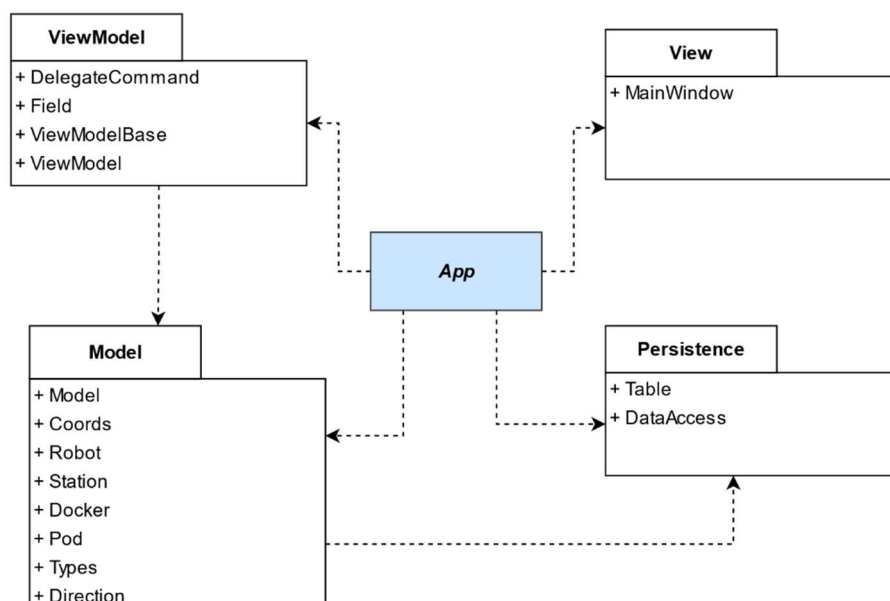
- **4. hét:** specifikáció és tervezés
- **7. hét:** prototípus – alap funkcionalitás, verziókezelő rendszer használata; a felhasználói esetek ~30%-a működik
- **10. hét:** a felhasználói esetek ~90%-a működik; egységtesztek, CI használata
- **13. hét:** release

A munkafolyamat gördülékenysége érdekében az **Atlassian Jira** projektkövető szoftvert használjuk a feladatok rendszerezésére. Verziókezelésre a **GitLab** platformját alkalmazzuk.

A heti scrum meetingeken kívül igyekszünk minél többször egyeztetni a teendőinket egy Microsoft Teams meeting keretei között, illetve ha egyikünk elkészül egy feladattal, a többi csapattagnak ellenőriznie kell a változtatásokat, mielőtt a feladatot befejezettként címkéznénk fel.

Felépítés

Az applikációnk az MVVM architektúra sémáját követve a következőképpen fog felépülni:

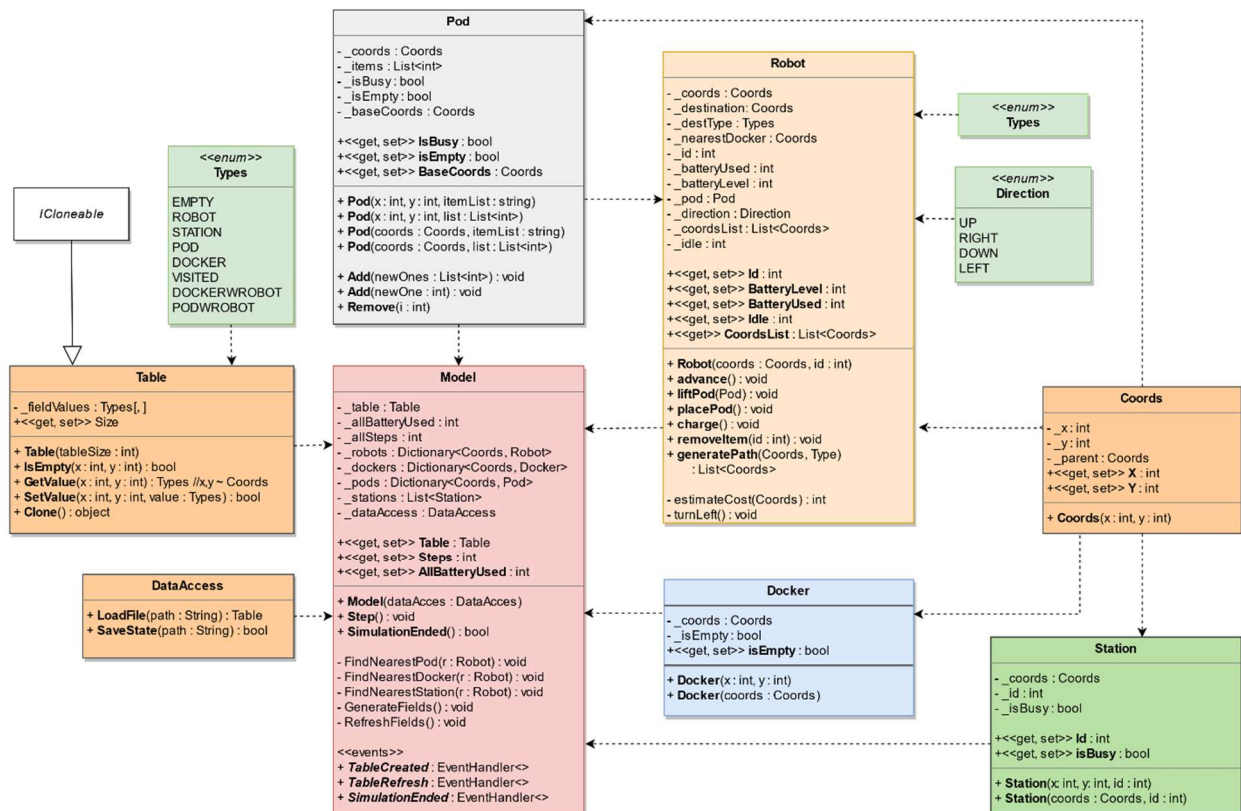


Megvalósítás során az átláthatóság és clean code alapelvek alapján eljárva kisebb részegységekre bontva fogjuk az osztályainkat kialakítani:

Modell

A modell a háttérben a központi vezérlőegység, ami minden szükséges információval rendelkezik, hogy biztosítsa a szimuláció elvárt futását. Tárolja a teljes játéktáblát, naplózza a robotok által elhasznált akkumulátor kapacitást, illetve azok lépéseinek számát is. Tartalmaz továbbá egy-egy listát mindegyik fő szimulációs elemből (Robot, Docker, Station, Pod). A számítógépen található fájlok elérésére szolgáló **DataAccess** osztályból is tartalmaz egy példányt, ezen keresztül képes adott input file-ból felpopulálni egy szimulációs táblát.

Bár központi vezérlőegységként funkcionál, nem itt történik az útvonalkeresés: a modell csak azért felel, hogy megkeresse az adott robothoz legközelebb eső polcot / célállomást, a többbit a robotra bízta. Egyfajta menedzserként gondolhatunk rá, ami gondoskodik róla, hogy minden robot el legyen látva feladattal. Emellett minden lépésnél megkeresi a robothoz legközelebb eső töltőállomást, melynek helye alapján a robot ki tudja számolni az út költségét, így biztosan időben el fog indulni tölteni.



Coords – a koordinátákat szimbolizáló osztály

Az adattagok a matematikában is megszokott x, illetve y elnevezést kapták. Definiálása javítja az áttekinthetőséget és a paraméterekben is könnyebb olvashatóságot biztosít az idegen szemek számára.

Pod – a polcokat szimbolizáló osztály

Mivel a polcokat a termékek leszállítása után szükséges a kezdeti helyükre vinni, így kettő koordináta-adattaggal rendelkeznek: az egyik az aktuális pozíciójukat írja le, míg a másik a polc kiindulási adatait tárolja. Az osztály egy példányában számontartjuk a polcon található termékek listáját; tekintve, hogy ennek a listának a hossza könnyedén lekérhető, egyszerű megállapítani, hogy üres-e egy adott polc, így viszonylag gyorsan kiszűrhetjük, mely polcokkal nincs további teendőnk. Az osztályon belül definiáltunk metódusokat, melyek az árucikkek polcon való elhelyezéseért és onnan való levételéért felelnek.

Docker – a töltőállomásokat szimbolizáló osztály

A töltőállomás is rendelkezik saját pozícióval, azonban ez futás közben nem tud változni, hiszen ezek nem mozgatható pályaelemek. Lehetőséget biztosítunk a modellnek az **isEmpty** property által a töltőállomás foglaltságának lekérdezésére, így nem fordulhat elő olyan, hogy egy robot egy számára elérhetetlen töltőállomást vesz célba.

Station – a célállomásokat szimbolizáló osztály

A célállomást pozícióján kívül az azonosítója (**id**) is egyértelműen meghatározza. Emellett számontartjuk a foglaltsági állapotát is, így el tudjuk kerülni, hogy úgy próbáljon a célállomás mezőjére lépni egy robot, hogy közben egy másik robot már épp használatba veszi azt.

Robot – a robotokat szimbolizáló osztály

A robot is rendelkezik saját koordinátákkal, ami folyamatosan változik annak mozgása során. Mivel a robot kizárólag előre tud mozogni, így el kell tárolnunk az irányt, amely felé pillanatnyilag néz. A robotok akkumulátorral működnek, így rendelkeznek saját töltöttségi szinttel, mely (a termékleadást és a várakozást leszámítva) minden interakció során egységnivel csökken. A robotok is rendelkeznek saját **id** adattaggal a könnyebb azonosíthatóság jegyében.

A robot a modelltől kapja meg a céljának koordinátáit, majd saját maga keresi meg az ideális útvonalat hozzá. Céljának elértekor a modell hívja meg az adott metódusait, melyek olyan funkciókért felelnek, mint például egy polc felemelése vagy egy termék leadása. A robotok közti konfliktusok elkerülésekor figyelembe vesszük azok töltöttségi állapotát, ezzel az összesített energiafelhasználásunk minimalizálására törekszünk.

Table – a raktár elrendezését szimbolizáló osztály

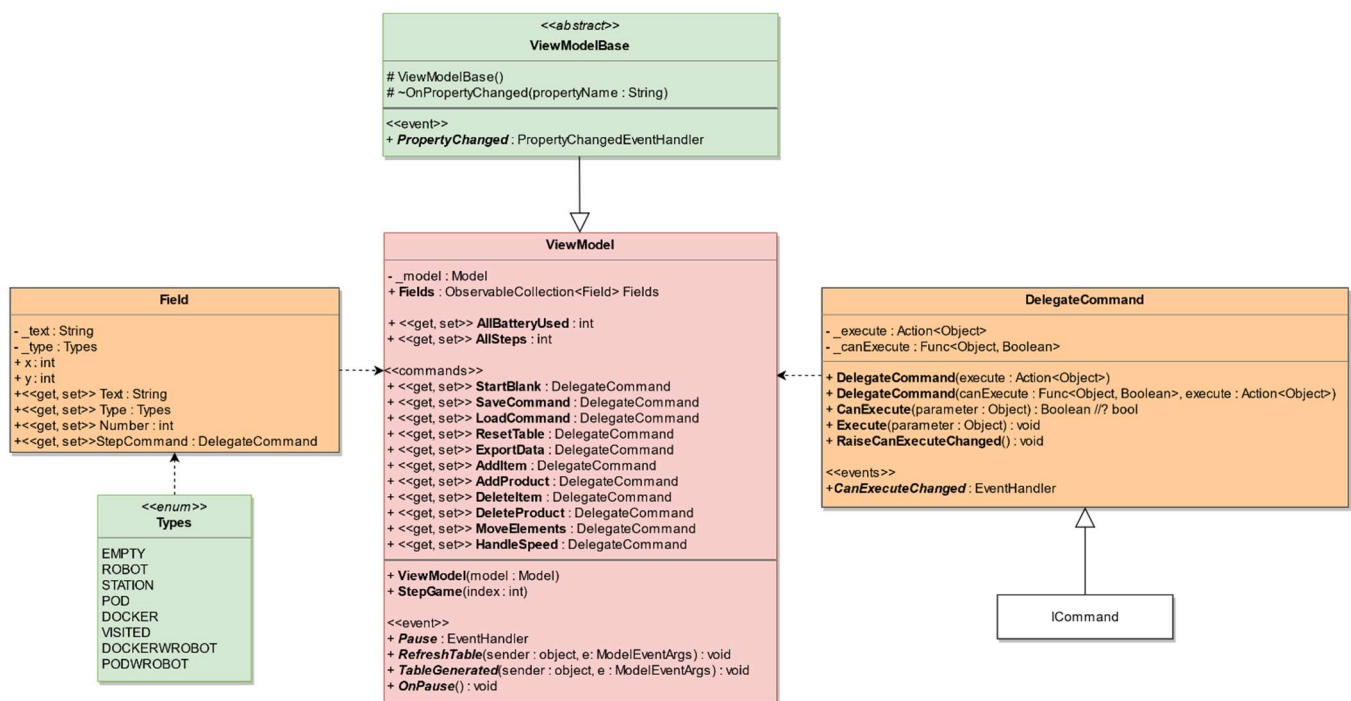
A szimulációs tábla számára önálló osztályt definiáltunk, mely lényegében egy NxM-es mátrix, ahol N a tábla sorait, M a tábla oszlopait leíró szám. Ebben a mátrixban a táblán található objektumok típusai vannak numerikus reprezentálva. Ez a tábla lesz az, ami a felhasználó számára ugyan rejtett marad, de a nézetmodell közvetítésével ezt fogja megjeleníteni a nézet. Ez a tábla a felhasználó számára ugyan rejtett marad, de a nézetmodell ezt fogja felhasználni a nézeten keresztül a szimuláció futásának vizualizációjához.

Felsorolók

- **Direction:** a robotok irányát leíró állapotokat tartalmazza
- **Types:** a **Table** osztály által használt numerikus típusokat tartalmazza

Nézetmodell

A nézetmodellünknek *függőségbefecskendezés* (***dependency injection***) segítségével adjuk át a modellt, így tud mediátorként funkcionálni a nézet és a modell között. A nézet tulajdonságait itt fogjuk kezelni, míg a program logikai működése a modellben zajlik, onnan csak adatokat kérünk le. A modell események kiváltásával tartja a kapcsolatot a nézetmodellel. Az osztályt a **ViewModelBase** absztrakt osztályból származtatjuk.



DelegateCommand











A parancsok definiálása itt történik meg. Ezen keresztül tudunk 'gombnyomásra' továbbítani egy változást a nézet felé, valamint XAML fájlban megtalálható összeköttetések (**binding**) is itt foglalnak helyet.

Field

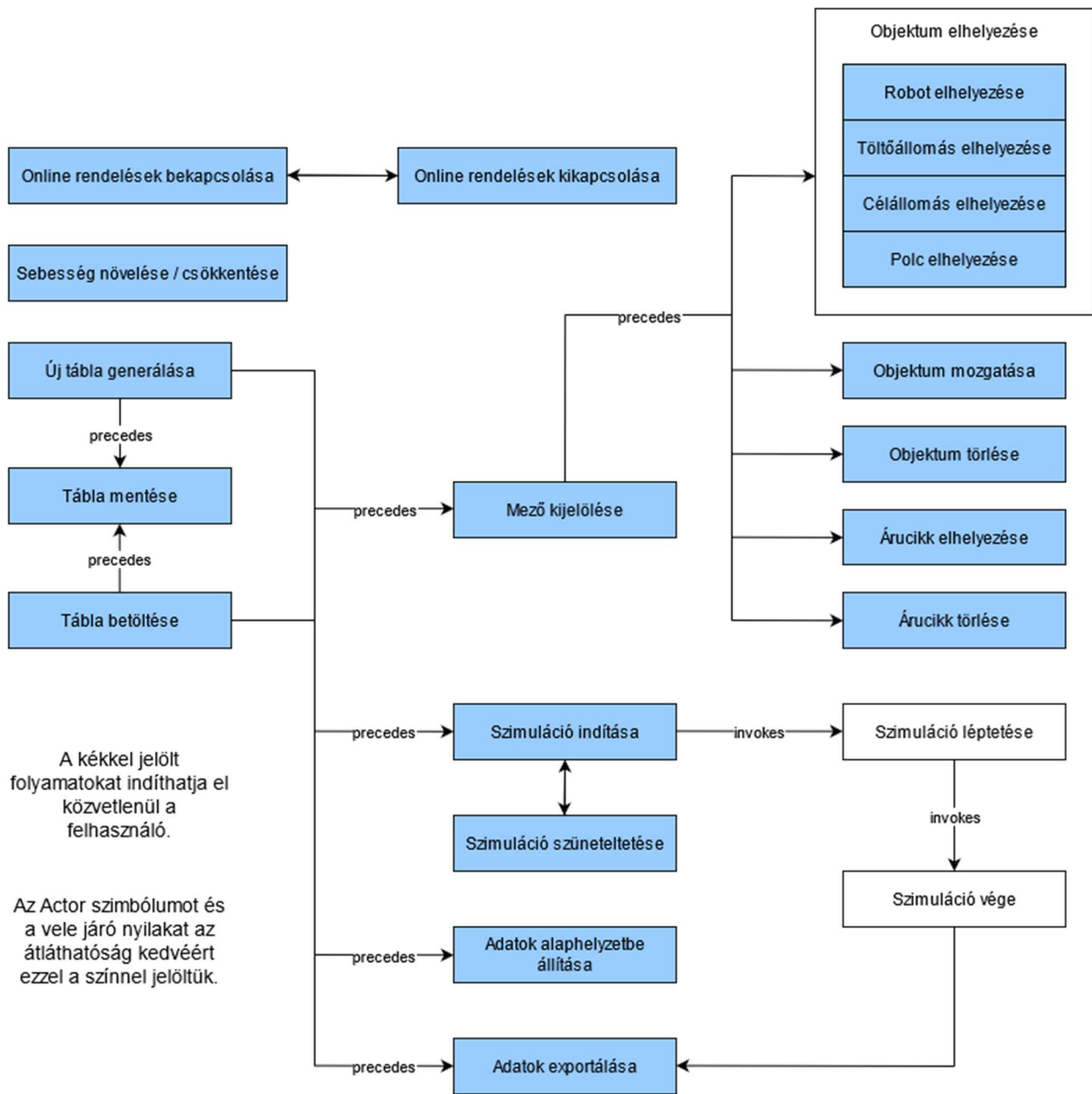
A nézet megjelenítéséhez szükséges adatokat tárolja, eltároljuk egy adott objektum helyét, típusát, pályán elfoglalt sorszámát.

Felhasználói felület terv



- | | | |
|--|--|--|
|  Robot |  Station #3 |  Pod alatt elhaladó Robot |
|  Docker |  Pod |  Podot szállító Robot |
|  Podot szállító robot egy Station mezőn | |  Kijelölt mező |
|  Robot egy Docker mezőn | |  Mozgatási cél |

Felhasználói esetek



A felhasználó lehetőségeit a következőképpen kategorizáltuk:

- **Futtatás**
- **Mentés, betöltés, exportálás**
- **Új tábla generálása, alaphelyzetbe állítás**
- **A tábla szerkesztése**
- **Szimuláció indítása, szüneteltetése, egyéb funkciók**

Futtatás

1	Alkalmazás indítása	GIVEN:	Az alkalmazás telepítve van.
		WHEN:	Futtatjuk az alkalmazást.
		THEN:	Az alkalmazás elindul, megjelenik a felhasználói felület.
2	Kilépés	GIVEN:	Az alkalmazás aktív.
		WHEN:	Az alkalmazás felületének lezáró ikonjára kattintunk.
		THEN:	Az alkalmazás bezárul.

Mentés, betöltés, exportálás

3a	Tábla mentése - dialógus megnyitása	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel.
		WHEN:	A File menüben jelezzük mentési szándékunkat a Save gombbal.
		THEN:	Megnyílik egy dialógus a mentéshez.
3b	Tábla mentése - elérési út megadása	GIVEN:	A szimuláció szünetel, a mentés dialógus aktív.
		WHEN:	Kiválasztjuk a könyvtárat, ahol a mentett táblát tárolni szeretnénk, majd megerősítjük mentési szándékunkat az OK gombra kattintva.
		THEN:	A mentés megtörténik, a dialógus bezárul.
3c	Tábla mentése - visszalépés	GIVEN:	A szimuláció szünetel, a mentés dialógus aktív.
		WHEN:	Elállunk mentési szándékunktól a Cancel gombra kattintva.
		THEN:	A dialógus mentés nélkül bezárul.
4a	Tábla betöltése - dialógus megnyitása	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel.
		WHEN:	A File menüben jelezzük betöltési szándékunkat a Load gombbal.
		THEN:	Megnyílik egy dialógus a betöltéshez.
4b	Tábla betöltése - nem létező fájl	GIVEN:	A szimuláció szünetel, a betöltés dialógus aktív.
		WHEN:	Nem létező fájl elérési útját adjuk meg.
		THEN:	Hibaüzenetet kapunk.
4c	Tábla betöltése - invalid fájl	GIVEN:	A szimuláció szünetel, a betöltés dialógus aktív.
		WHEN:	Létező, de betöltésre nem alkalmas fájl elérési útját adjuk meg.
		THEN:	A betöltési dialógus bezárul. A betöltés megghiúsul, hibaüzenetet kapunk.
4d		GIVEN:	A szimuláció szünetel, a betöltés dialógus aktív.

	Tábla betöltése - megfelelő fájl	WHEN:	Létező és betöltésre alkalmas fájl elérési útját adjuk meg.
		THEN:	A betöltési dialógus bezárul, a betöltés megtörténik, megjelenik a korábban elmentett tábla.
4e	Tábla betöltése - visszalépés	GIVEN:	A szimuláció szünetel, a betöltés dialógus aktív.
		WHEN:	Elállunk betöltési szándékunktól a Cancel gombra kattintva.
		THEN:	A dialógus betöltés nélkül bezárul.
5a	Tábla adatainak exportálása - dialógus megnyitása	GIVEN:	Az alkalmazás aktív, a szimuláció szünetel.
		WHEN:	A File menü Export Data gombjára kattintva jelezzük exportálási szándékunkat.
		THEN:	Megnyílik az exportálás dialógus.
5b	Tábla adatainak exportálása - elérési út megadása	GIVEN:	Az exportálás dialógus aktív.
		WHEN:	Kiválasztjuk a könyvtárat, ahol az exportált adatokat tárolni szeretnénk, majd megerősítjük mentési szándékunkat az OK gombra kattintva.
		THEN:	Az adatok mentésre kerülnek, a dialógus bezárul.
5c	Tábla adatainak exportálása - visszalépés	GIVEN:	Az exportálás dialógus aktív.
		WHEN:	A Cancel gombra kattintva elállunk exportálási szándékunktól.
		THEN:	A dialógus az adatok exportálása nélkül bezárul.

Új tábla generálása, alaphelyzetbe állítás

6a	Új tábla generálása - dialógus megnyitása	GIVEN:	Az alkalmazás aktív, a szimuláció még nem indult el vagy épp szünetel.
		WHEN:	A File menüben a New gombra kattintva jelezzük szándékunkat új tábla generálására.
		THEN:	Megjelenik egy dialógus, ahol megadhatjuk az új tábla méreteit.
6b	Új tábla generálása - méretek megadása	GIVEN:	Az új tábla dialógus aktív.
		WHEN:	Az input mező gombjaival beállítjuk a tábla méreteit, majd az OK gombra kattintva megerősítjük szándékunkat.
		THEN:	Megjelenik az adott méretű üres tábla, az eddigi adatok nullázódnak.
6c	Új tábla generálása - visszalépés	GIVEN:	Az új tábla dialógus aktív.
		WHEN:	A Cancel gombra kattintva visszalépünk szándékunktól.
		THEN:	A dialógus bezárul, a tábla változatlan.

7	Tábla adatainak alaphelyzetbe állítása	GIVEN:	Az alkalmazás aktív, a szimuláció még nem indult el vagy épp szünetel.
		WHEN:	A File menüben a Reset Data gombra kattintva jelezzük szándékunkat az adatok alaphelyzetbe állítására.
		THEN:	Az eddigi adatok (felhasznált energia, lépések száma stb.) nullázódnak, a tábla elemei nem változnak.

A tábla szerkesztése

8a	Mező(k) kijelölése	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel.
		WHEN:	A tábla egyik mezőjére kattintunk.
		THEN:	A kiválasztott mező(k) piros körvonalat kap(nak).
8b	Mező(k) kijelölésének visszavonása	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel, van kijelölt mező.
		WHEN:	A tábla egyik kijelölt mezőjére kattintunk.
		THEN:	A mező eltűnik a kijelölésből, ezzel elveszíti a piros körvonalát.
9a	Robot elhelyezése kijelölt mezőn	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel, van kijelölt mező.
		WHEN:	Az Add menü Item menüpontjának Robot gombjára kattintunk.
		THEN:	Ha a kijelölt mező üres, robotot helyezünk el rajta, ezzel narancssárgára színeződik. Ha a kijelölt mezőn polc van, robotot helyezünk el a polc alá, ezzel a mező narancssárgára színeződik sötét kerettel. Ha a kijelölt mező nem üres vagy nem polc van rajta, nem történik változás.
9b	Töltőállomás (Dock) elhelyezése kijelölt mezőn	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel, van kijelölt mező.
		WHEN:	Az Add menü Item menüpontjának Dock gombjára kattintunk.
		THEN:	A kijelölt üres mezőn töltőállomást helyezünk el, ezek a mezők kékre színeződnek. Ha a kijelölt mező nem üres, nem történik változás.

9c	Célállomás (Station) elhelyezése kijelölt mezőn	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel, van kijelölt mező.
		WHEN:	Az Add menü Item menüpontjának Station gombjára kattintunk.
		THEN:	A kijelölt üres mezőkön célállomást helyezünk el, ezek a mezők zöldre színeződnek. A célállomások kijelölési sorrend alapján sorszámozódnak. Amennyiben a táblán elhelyezett célállomások száma eléri a 4-et, nem jön létre több célállomás. Ha a kijelölt mező nem üres, nem történik változás.
9d	Polc (Pod) elhelyezése kijelölt mezőn	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel, van kijelölt mező.
		WHEN:	Az Add menü Item menüpontjának Pod gombjára kattintunk.
		THEN:	Ha a kijelölt mező üres, polcot helyezünk el rajta, ezzel szürkére színeződik. Ha a kijelölt mezőn robot van, polcot helyezünk el a robot felett, ezzel a mező szürkére színeződik narancssárga kerettel. Ha a kijelölt mező nem üres és a mezőn lévő objektum nem robot, nem történik változás.
9e	Objektum törlése kijelölt mezőről	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel, van kijelölt mező.
		WHEN:	A Del menü Item gombjára kattintunk.
		THEN:	A kijelölt mezőkről törlődnek az elhelyezett objektumok, ezek a mezők fehérre színeződnek. Ha a kijelölt mező üres, nem történik változás.
10a	Kijelölt objektum(ok) mozgatása	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel, van kijelölt mező.
		WHEN:	A Move gombra kattintunk.
		THEN:	Ha a kijelölt mezők között van üres mező, azok eltűnnek a kijelölésből. Ha marad nemüres kijelölt mező, a tábla egyik mezőjére kattintva áthelyezzük az objektumokat. Az áthelyezés a kijelölt mezők közötti első objektumot követik, a többi objektum ehhez viszonyul. Az áthelyezés csak akkor megy végbe, ha az összes kijelölt objektumot át lehet helyezni a megadott helyre.
10b	Visszalépés objektum(ok) mozgatásától	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel, a mozgatási funkció aktív.
		WHEN:	A Move gombra kattintunk.
		THEN:	A mozgatási funkció kikapcsol.

11a	Árucikk elhelyezése kijelölt polcon	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel, van kijelölt mező.
		WHEN:	Az Add menü Product menüpontjának egyik gombjára kattintunk.
		THEN:	A kiválasztott számú árucikk elhelyezzük a kijelölt polcokon, ezeken megjelenik az árucikk száma. Ha a kiválasztott áru eddig is fent volt a polcon, vagy a kijelölt mezőn nincs polc, nem történik változás.
11b	Árucikk törlése kijelölt Pod mezőről	GIVEN:	Az alkalmazás aktív, a szimuláció még nem fut vagy épp szünetel, van kijelölt mező.
		WHEN:	A Del menü Product menüpontjának egyik gombjára kattintunk.
		THEN:	A kiválasztott számú árucikket töröljük a kijelölt polcokról, ezekről eltűnik az árucikk száma. Ha a kiválasztott áru eddig sem volt fent a polcon, vagy a kijelölt mezőn nincs polc, nem történik változás.

Szimuláció indítása, szüneteltetése, egyéb funkciók

11	Szimuláció indítása	GIVEN:	Az alkalmazás aktív, a szimuláció még nem indult el vagy épp szünetel. A táblán elhelyezésre került mind a négy célállomás és legalább egy robot, illetve töltőállomás.
		WHEN:	A START gombra kattintunk.
		THEN:	A szimuláció elindul, a START gomb felirata STOP -ra vált.
12	Szimuláció szüneteltetése	GIVEN:	A szimuláció fut.
		WHEN:	A STOP gombra kattintunk.
		THEN:	A szimuláció szünetel, a STOP gomb felirata START -ra vált.
13	Szimuláció vége	GIVEN:	A szimuláció fut, az online rendelések funkció nincs bekapcsolva.
		WHEN:	A táblán elhelyezett polcokon nem marad leszállítandó árucikk.
		THEN:	Egy felugró ablak tájékoztatja a felhasználót a szimuláció befejezéséről. Az Export Data gombra kattintva exportálhatjuk a szimuláció naplózott adatait egy fájlba, a Close gombbal pedig bezárhatjuk a felugró ablakot.

14a	Szimuláció gyorsítása	GIVEN:	Az alkalmazás aktív.
		WHEN:	A Speed csúszka karját felfelé mozgatjuk.
		THEN:	A szimuláció gyorsul: a körök közt eltelt idő csökken.
14b	Szimuláció lassítása	GIVEN:	Az alkalmazás aktív.
		WHEN:	A Speed csúszka karját lefelé mozgatjuk.
		THEN:	A szimuláció lassul: a körök közt eltelt idő nő.
15a	Online rendelések funkció bekapcsolása	GIVEN:	Az alkalmazás aktív.
		WHEN:	Az Enable Online Orders gombra kattintunk.
		THEN:	Bekapcsoljuk az online rendelések funkciót: bizonyos időközönként véletlenszerűen felkerül egy termék egy polcra. A gomb felirata Disable Online Orders -re vált.
15b	Online rendelések funkció kikapcsolása	GIVEN:	Az alkalmazás aktív.
		WHEN:	A Disable Online Orders gombra kattintunk.
		THEN:	Kikapcsoljuk az online rendelések funkciót. A gomb felirata Enable Online Orders -re vált.

Funkcionális tesztek

Ahhoz, hogy programunkat a saját szemünk számára biztonságosnak és működőképesnek nyilváníthassuk, szükséges az, hogy összegyűjtsük azon szélsőséges és hétköznapi eseteket, melyek alapján majd tesztelhetjük alkalmazásunk futását.

Adatok betöltése, mentése, szerkesztése

- **Betöltés**
 - rossz input file; kezelés: hibaablak, majd a program futásának folytatása
- **Mentés**
 - csak szüneteltetett szimuláció mellett legyen elérhető
- **Szerkesztés**
 - NxM-es üres pálya generálása
 - Szimulációs elemek felhelyezése, törlése
 - Termékek elhelyezése polcon, termékek eltávolítása
 - Több elem kijelölése és mozgatása
 - Szerkesztés félbehagyása, majd:
 - Új tábla generálása
 - Tábla betöltése
 - Kilépés előtt mentés felajánlása

Szimuláció futása alatti interakciók

- A menü elemeinek letiltása futás közben
- Sebesség változtatása a csúszka segítségével (akár kliens számítógépről)
- Online megrendelések szerkesztése a mezők kijelölésével
- Szimuláció szüneteltetése, folytatása

A program szélsőséges esetei

- A robotok nem alkalmasak szállításra maximális töltéssel sem, mert lemerülnének
 - Hibaüzenet formájában a felhasználó tájékoztatása a problémáról
- Két robot akadályozza egymás útját
 - Az egyik robot megáll és elengedi a másikat
 - Frontális ütközés elkerülése érdekében a nagyobb töltéssel rendelkező robot útjának módosítása

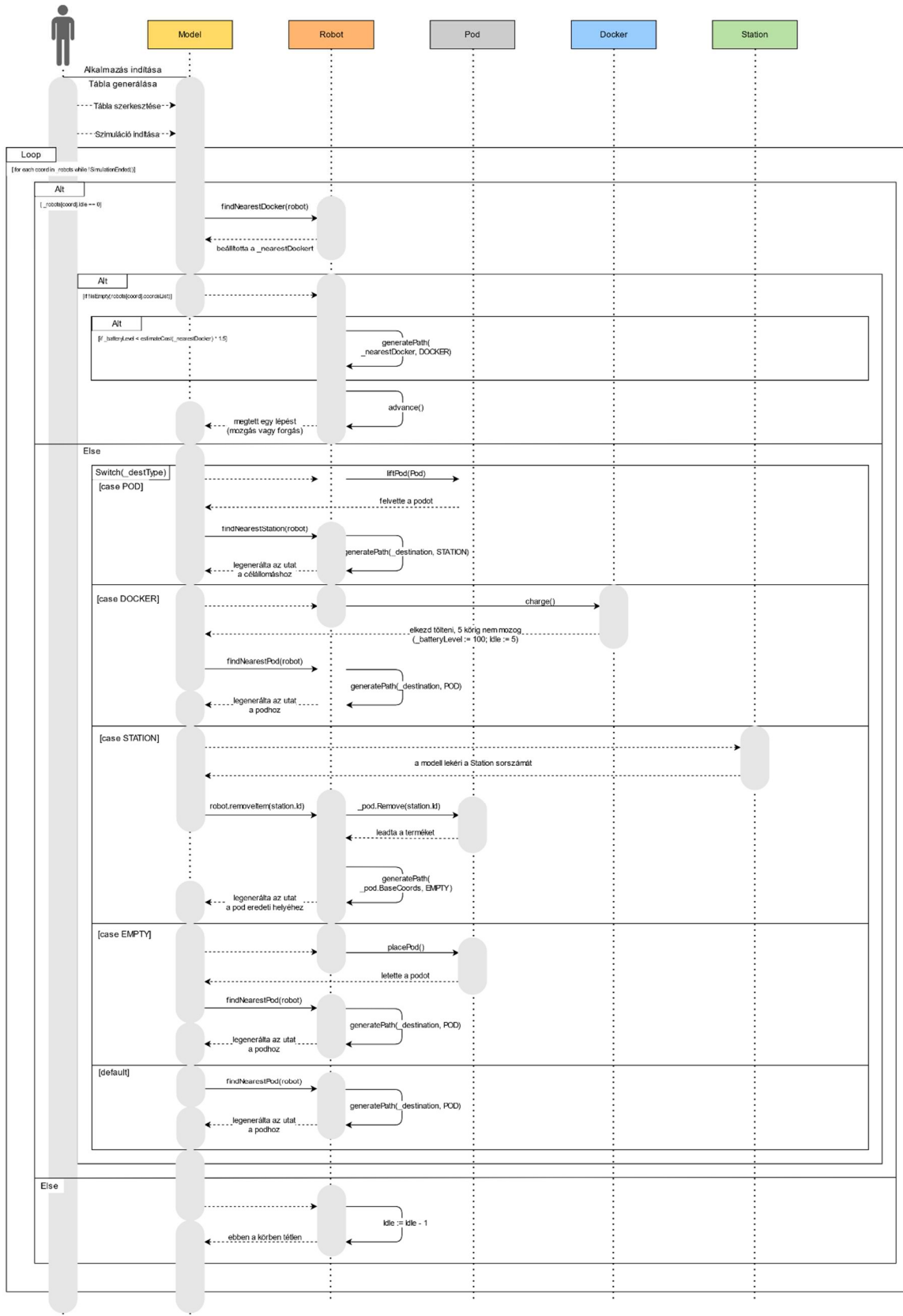
A program hétköznapi esetei

- A robotok képesek termékeket leszállítani, az elemek megfelelő távolságra vannak
 - Szimuláció futtatása, majd a végén az adatok összegzése
 - A felhasználó tájékoztatása a szimuláció végéről
 - Az adatok naplózása

Nem funkcionális követelmények

- Intuitív felhasználói felület
- A program „bolondbiztos”: tudja kezelni a hibás emberi tevékenységet hibaüzenetekkel és korlátozásokkal
- A program ellenőrzi a betöltéshez használt fájl adatainak helyességét
- Hálózati kommunikáció
- Minimális műveletidő és tárhelyigény
- Agilis munkafolyamat
- Objektumorientált fejlesztés
- Clean code alapelvek

A program működése



Az applikáció elindításakor új tábla generálódik, amit a felhasználónak lehetősége van szerkeszteni, vagy betölthet másik táblát egy fájlból is. A szimuláció indításakor a modell végigiterál a robotokon, és megvizsgálja, hogy épp várakoznak-e; ha igen, csökkenti eggyel a várakozási időt, és a következő robotra lép.

Ellenkező esetben a modell először megkeresi és átadja a robotnak a hozzá legközelebb eső töltőállomás koordinátáit, majd megvizsgálja, hogy a robotnak pillanatnyilag van-e legenerált útja. Ha van, akkor lépteti a robotot: ekkor a robot először megvizsgálja, hogy kritikus-e a töltöttségi szintje. Ha igen, akkor a korábbi úticélját felülírja a legközelebbi töltőállomás. Ezután a robot az útjának következő mezőjére lép (ha felé néz, akkor lép, egyébként csak irányba forog).

Ha a robotnak nincs útja, az azt jelenti, hogy elérte a célját. A `_destType` adattag azt tartja számon, hogy a célja milyen típusú mező, így ez alapján egyszerű elágazást írunk:

- **POD**
A robot elérte a polcot; a modell arra utasítja, hogy vegye fel a polcot, majd a modell a polc termékei alapján megkeresi a legközelebbi (megfelelő) célállomást, amihez a robot legenerálja az utat.
- **DOCKER**
A robot töltőállomáshoz ért, ami azt jelenti, hogy töltenie kell: a modell meghívhatja vele a `charge()` metódust és 5 körre lezárja a töltőállomást és a robotot egyaránt.
- **STATION**
A robot elérte a célállomást. A modell lekéri az állomás számát, ezt továbbítja a robotnak, ami a szállított polcra leveszi az azonos számú terméket. Ezután vagy a következő célállomáshoz kell mennie a robotnak, vagy a polc eredeti helyére.
- **EMPTY**
Ha a cél üres mező volt, az csak azt jelentheti, hogy a robotnak ide kell visszahoznia a szállított polcot. A modell a polc leadására utasítja a robotot, majd megkeresi a legközelebbi polcot, amin termékek vannak, és a robot legenerálja hozzá az utat.
- **default**
Ha a cél típusa nem az előző négy opció egyike, az azt jelenti, hogy a szimuláció most indul, ez a robotok első köre. A modell megkeresi a hozzájuk legközelebb eső polcot, a robot legenerálja hozzá az utat.

Fejlesztési adatok

Felhasznált technológiák:

- [Atlassian Jira](#)
- [Git](#)
- [GitLab](#)
- [Microsoft Teams](#)
- [Visual Studio 2019 Community Edition](#)

Fejlesztők:

- **Amtmann Kristóf** (alias Puszedli)
- **Borsy Máté** (alias Sziporka)
- **Bur Bence** (alias Csuporka)

© Pindur Pandurak 2021

Tartalom

Feladatleírás	1
Részfeladatok	3
A szoftver célja, előnyei.....	3
Elemzés.....	4
Mérőföldkövek:	4
Felépítés	4
Modell	5
<i>Coords</i> – a koordinátákat szimbolizáló osztály	5
<i>Pod</i> – a polcokat szimbolizáló osztály	6
<i>Docker</i> – a töltőállomásokat szimbolizáló osztály.....	6
<i>Station</i> – a célállomásokat szimbolizáló osztály	6
<i>Robot</i> – a robotokat szimbolizáló osztály	6
<i>Table</i> – a raktár elrendezését szimbolizáló osztály.....	6
Felsorolók	7
Nézetmodell	7
<i>DelegateCommand</i>	7
<i>Field</i>	7
Felhasználói felület terv	8
Felhasználói esetek	9
Futtatás.....	10
Mentés, betöltés, exportálás	10
Új tábla generálása, alaphelyzetbe állítás.....	11
A tábla szerkesztése	12
Szimuláció indítása, szüneteltetése, egyéb funkciók	14
Funkcionális tesztek	15
Adatok betöltése, mentése, szerkesztése.....	15
Szimuláció futása alatti interakciók.....	16
A program szélsőséges esetei	16
A program hétköznapi esetei	16
Nem funkcionális követelmények	16
A program működése.....	17

