11 Csapat:	**Dokumentáció**	Tesztvezérelt prog.
Németh Tamás *Maár Dávid* *Riszterer István*	*Feladat: Robotok szimulációja*	*2023.06.05.*

Feladat leírás:

Készítsünk egy olyan rendszert, amellyel raktárban dolgozó robotok működését tudjuk szimulálni. A szállítórobotok az áruházban az áruk pakolásáért felelnek, típustól függően más a pakolási sebességük (áru/óra), illetve a rendelkezésre álló töltésük. Egy töltéssel egy egység áru pakolható. A szállítórobotok töltéséhez töltőrobotok szükségesek, típustól függően más töltési sebességgel (egység/óra). Az áruházba minden nap érkeznek áruk, ezeket kell szétpakolniuk a szállítórobotoknak. Amely árut nem sikerül aznap szétpakolni, a következő napra marad (emiatt halmozódhatnak is az áruk). Az áru lehet romlandó, amelynek a pakolása elsőbbséget élvez a nem romlandóval szemben.

Elvárt programfunkciók:

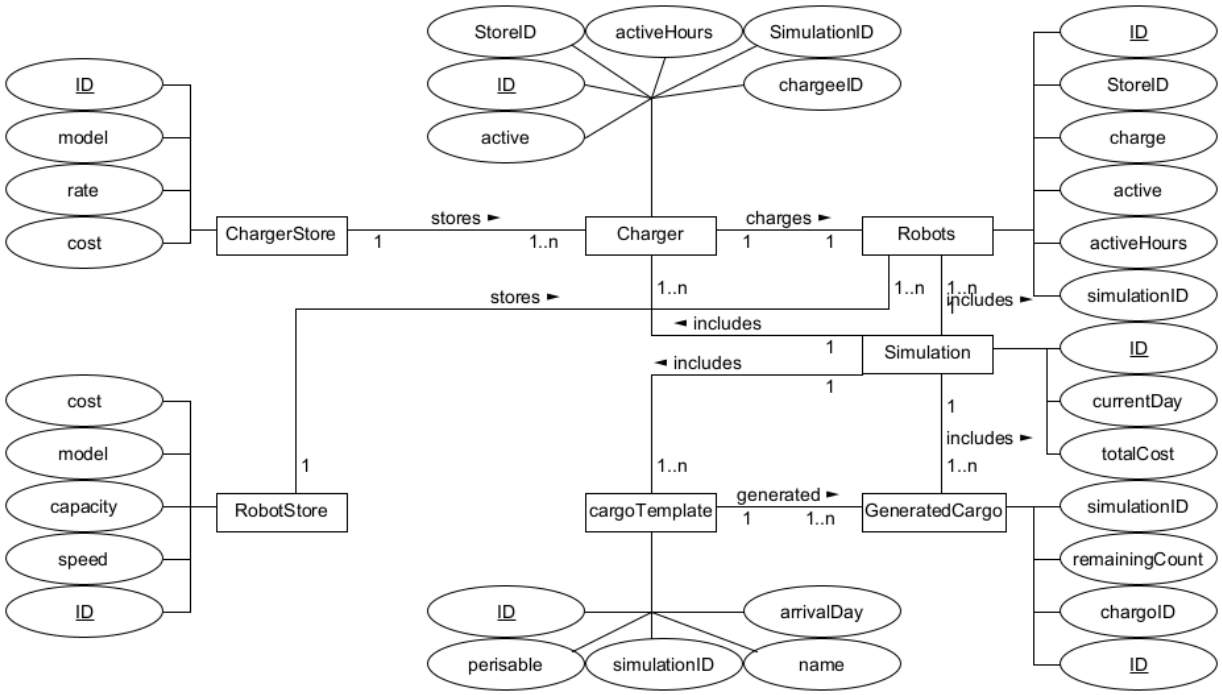
- Szimuláció konfigurálása, amely során megadjuk a kezdő robotokat (egy előre megadott listából kiválasztva, de lehet 0 robottal is kezdeni), a napok számát, valamint naponként az érkezdő árukat (elnevezés, mennyiség, romlandó-e). Az áruk beolvashatóak egy szöveges fájlból is, vagy generálhatóak véletlenszerűen.
- A konfiguráció elmenthető adott névvel, és később bármely korábban elmentett konfiguráció visszatölthető.
- A szimuláció elindítható a megadott paraméterekkel, majd naponként léptethető. A szimulációból bármely nap után kiléphetünk, illetve elmenthetjük az aktuális állást, így később bármikor folytathatjuk.
- Minden nap végén láthatóak az adott napon megérkezett, és az esetlegesen a következő napra hátra maradt áruk.
- Bármely nap vásárolható új robot (listából választva), illetve lehetőség van a robot átmeneti kivonására és visszaállítására (a kivont robotot nem kell tölteni, így neki nincs költsége).
- A munkában lévő szállítórobotok sorban szállítják helyükre az árukat (de a romlandók elsőbbséget élveznek), és minden áruszállítással csökken a töltésük. Amikor már nem tudnak tovább szállítani, tölteni kell őket (ez automatikusan bekövetkezik), amihez szabad töltőrobotra van szükség. Amennyiben nincs ilyen, a robot várakozó állapotba kerül, amíg fel nem szabadul egy töltőrobot. Egy töltőrobot egyszerre csak egy szállító robotot tölthet (a töltőrobotot már nem kell külön tölteni), és mindig maximumig tölti.
- A szimuláció az eredményeket napokra vetítve számítsa, de a tevékenységeket órában mérje, azaz például, ha egy áru elhelyezése, vagy egy robot töltése átcsúszik a következő napra, azt vegye figyelembe.
- A szimuláció végén a program kilistázza, hogy mely robotok vettek részt a szimulációban (típus, aktív napok száma, inaktív napok száma), milyen áruk maradtak meg (elnevezés, mennyiség, romlandó-e), illetve mennyi volt az összköltség.

Terv

Típus definíciók:

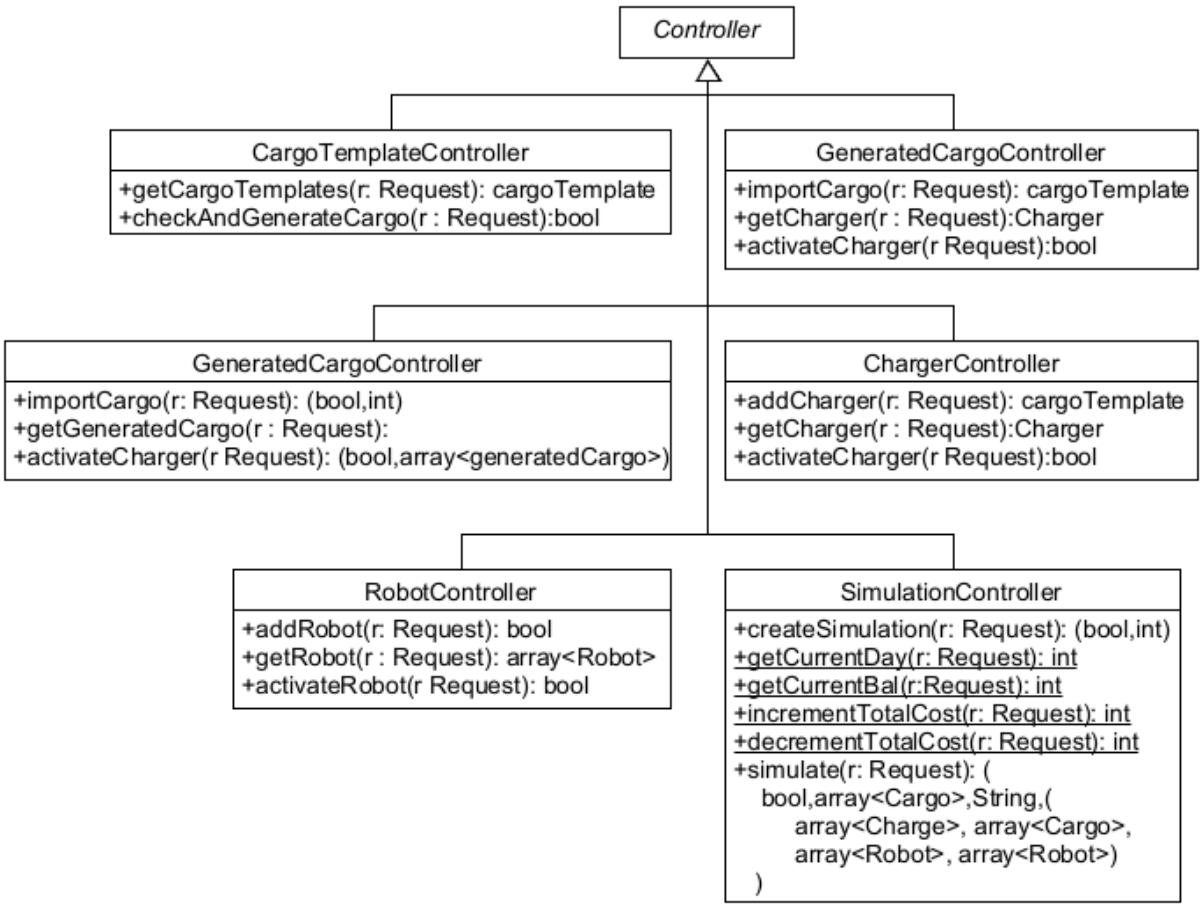
Simulation	=	(totalDays×totalCost)	totalCost,tolalDays∈ℕ
GeneratedCargo	=	(remainingCount)	remainingCount∈ℕ
CargoTemplate	=	(arrivalDay×name×perisable)	arrivalDay∈Dátum name∈Szöveg perisable∈Logikai érték
RobotStore	=	(cost×modelName×capacity×speed)	cost∈ℕ capacity,speed∈R modelName∈Szöveg

Charger	=	(activeHours×active)	activeHours∈ℕ active∈Logikai érték
ChargerStore	=	(model×rate×cost)	model∈Szöveg cost∈ℕ rate∈R
Robot	=	(charge×active×activeHours)	active∈Logikai érték activeHours∈ℕ charge∈R
Típus megvalósítás:			
Simulation	=	Record(Integer×Integer)	
GeneratedCargo	=	Integer	
CargoTemplate	=	Record(Integer×String×Boolean)	
RobotStore	=	Record(Integer×String×Float×Float)	
Charger	=	Record(Integer×Boolean)	
ChargerStore	=	Record(String×Float×Integer)	
Robot	=	Record(Boolean×Integer×Float)	
E-K diagram			

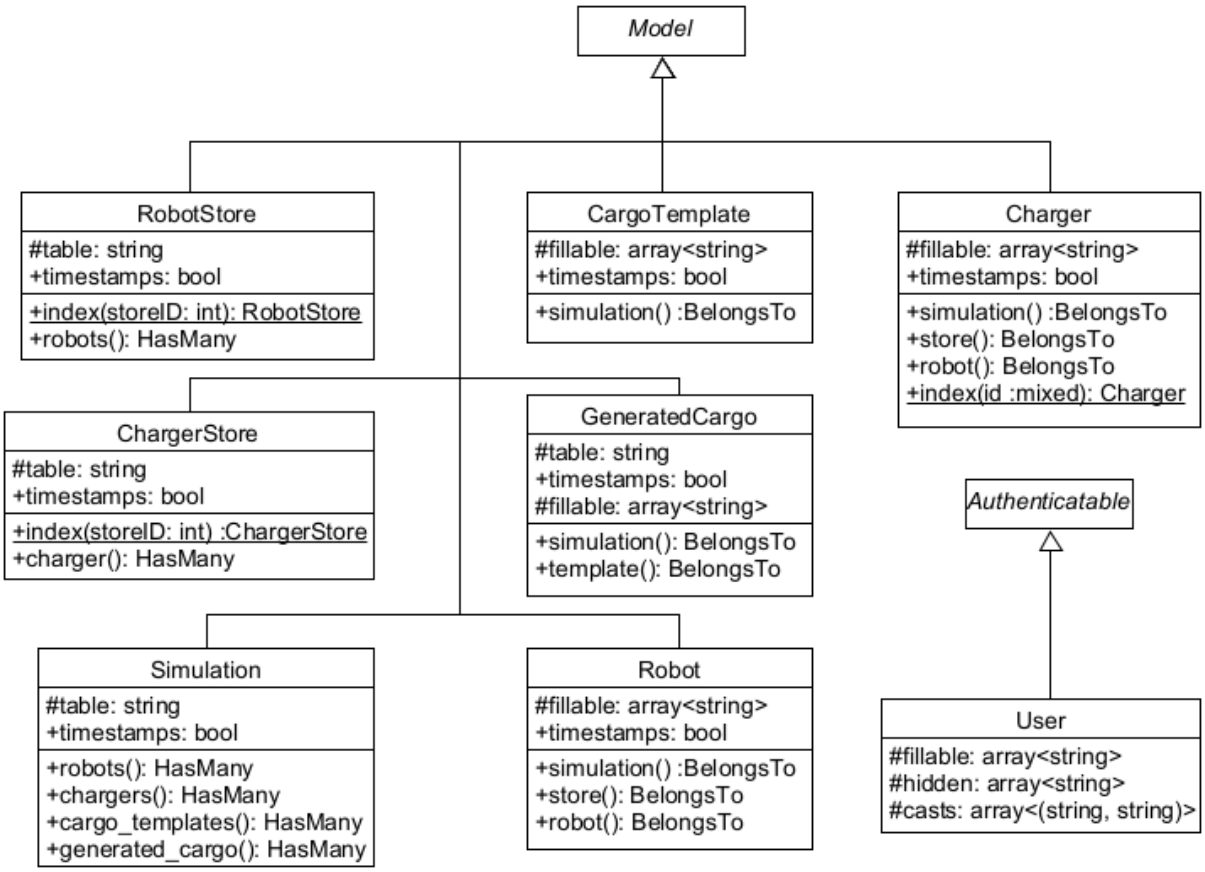


Osztálydiagram

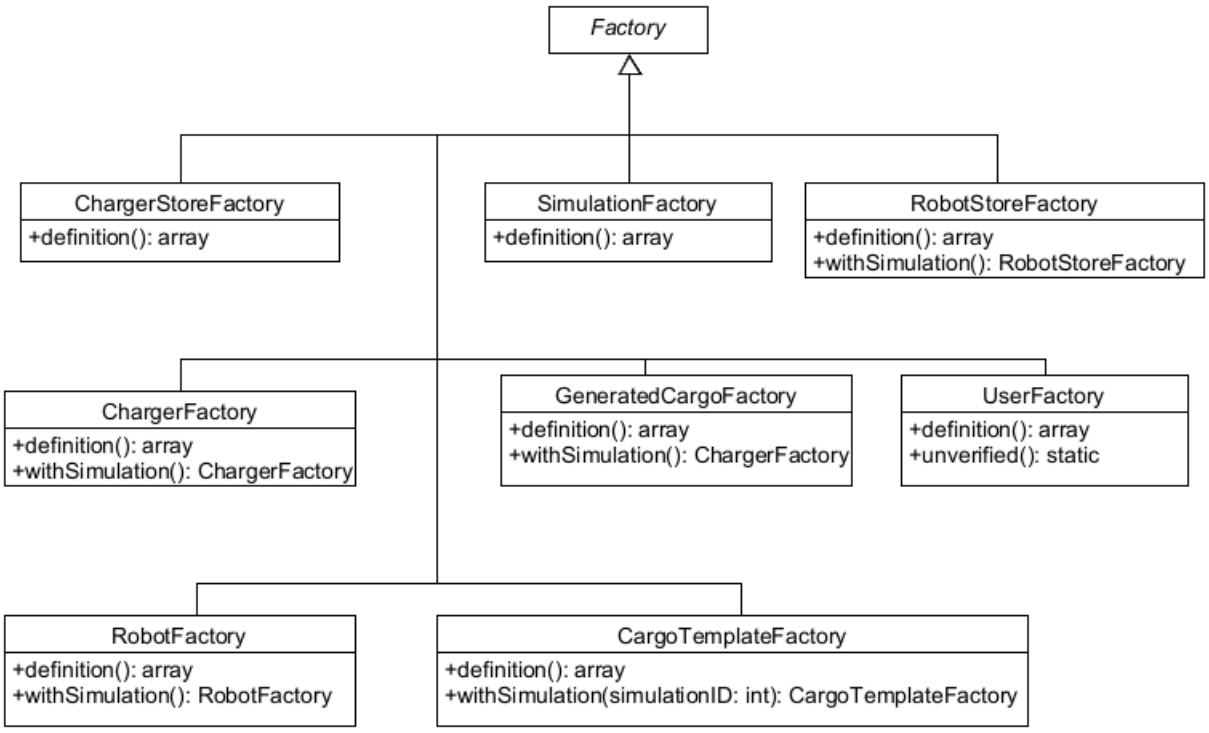
Controllers



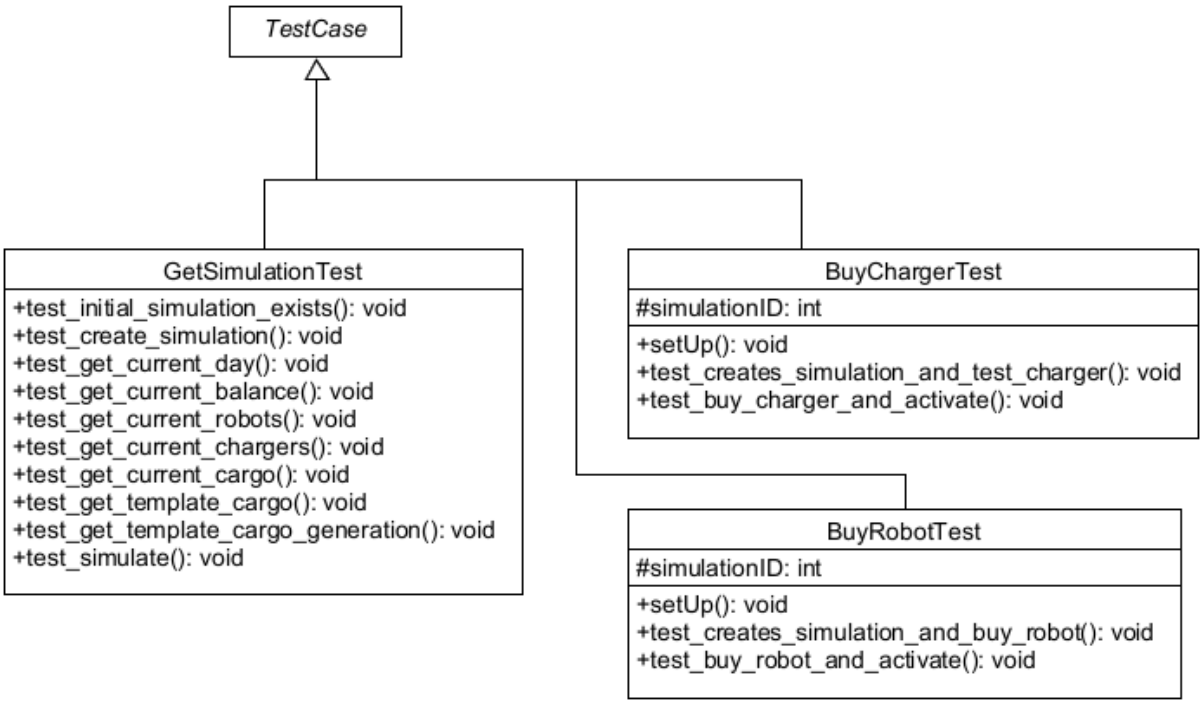
Models



Factories



TestCase



Struktogram:

SimulationController::simulate()

```
$simulationID = $request->route('id');
$simulation = Simulation::find($simulationID);
$cargoList = $simulation->generated_cargo;
$cargoList = $cargoList->sortBy(function ($cargo) {
  return ($cargo->template->perishable) ? 0 : 1;})->values();
```

```
$finished = [];
$robots = $simulation->robots;
$chargers = $simulation->chargers;
$log = "DAILY LOG \n\n";
$chargingRobots = [];
$chargedRobots = [];
$data = array();
```

i= 0...23

```
$log .= "Hour: $i \n";
$data[$i] = new \stdClass();
```

Charger in Chargers

Charger→ active == True

```
$charger->active_hours++;
$timeLeft = $charger->store->rate;
```

$timeLeft > 0 && isset($charger->robot)

```
$log .= " - Charger" . $charger->id . " is charging robot" .
$charger->robot->id . "\n";
$data[$i]->charges[] = [
    chargerID' => $charger->id,
   'chargerModel' => $charger->store->model,
   'robotID' => $charger->robot->id,
   robotModel' => $charger->robot->store->model,
];
$charger->robot->charge++;
```

$charger->robot->charge >= $charger->robot->store->capacity)

```
$charger->robot->charge =(
    $charger->robot->store->capacity);
unset($chargingRobots[$charger->robot->id]);
$chargedRobots[] = $charger->robot;
$charger->robot()->disassociate();
```

```
$timeLeft--;
```

```
$log .= "-- Starting Packing\n"
```

Robot in Robots

($robot->active)==true

```
$robot->active_hours++;
$timeLeft = $robot->store->speed;
```

($timeLeft > 0 && $robot->charge > 0 && $cargoList->count() > 0)

```
$targetCargo = $cargoList->first();
$data[$i]->cargo[] = [
   'cargoID' => $targetCargo->id,
   'cargoName' => $targetCargo->template->name,
   'robotID' => $robot->id,
   'robotModel' => $robot->store->model,
 ];
 $log .= " - Robot" . $robot->id . " is packing cargo" .
$targetCargo->id . "\n";
```

A STRUKTOGRAM

A STRUKTOGRAM

```
$targetCargo->remaining_count--;
```

($targetCargo->remaining_count == 0)

```
$finished[] = $targetCargo;
$cargoList->shift();
```

```
$timeLeft--;
$robot->charge--;
```

($robot->charge == 0)

```
$robot->active = false;
$data[$i]->depletedRobots[] = [
  "robotID" => $robot->id,
  "robotModel" => $robot->store->model,
];
 $log .= " - Robot" . $robot->id . " is depleted\n";
```

Charger in Chargers

!isset($charger->robot))

```
$charger->robot()->associate($robot);
chargingRobots[] = $robot;
 break;
```

```
$log .= "-- Finished Packing\n"
```

ChargedRobot in ChargedRobots

```
$chargedRobot->active = true;
$data[$i]->chargedRobots[] = [
  "robotID" => $chargedRobot->id,
  "robotModel" => $chargedRobot->store->model,
];
$log .= " - Robot" . $chargedRobot->id . " is fully charged\n";
$chargedRobot->save();
```

```
$chargedRobots = [];
```

```
$log .= "\nEND OF DAY\n"
```

Cargo in CargoList

```
$cargo->save()
```

Cargo in Finished

```
$cargo->delete()
```

```
$simulation->currentDay++
$simulation->save()
```

Robot in Robots

```
$robot->save()
```

Charger in Chargers

```
$charger->save();
```

RETURN *response()→json([*

```
  'success' => true,
  'remainingCargo' => $cargoList,
  'log' => $log, 'data' => $data
]);
```

- **A struktogram**