Game of life

# Rendszerterv

## Feladat

A Conway féle életjáték egy implementációjának elkészítése, úgy ahogy soha nem jutna eszembe: külön szerver oldali üzleti logika, REST API interface és böngésző oldali megjelenítés.

## Architektúra

Az alkalmazás user interface egy böngészőből futtatható SPA oldal. Az üzleti logika egy távoli szerveren fut, amellyel a kliens REST hívásokon keresztül kommunikál. A feladat megoldásához választott környezet és programozási nyelv:

### Backend

* Php 5.x
* Phalcon keretrendszer
* Clab2 keretrendszer
* Mongo adatbázis
* Doctrine-odm objektumkezelő

### Frontend

* Html 5
* angular js
* typescript

## Megvalósítás

A játék adatait tároló és a szabályokat vezérlő logikát két külön objektumra kell bontani. Így a játéktér adatkezelése és a szabályok külön kezelhetőek lesznek.

A megvalósítás során létre kell hozni egy clab2 backend modult, egy clab2 angular frontend modult és egy ezeket és a futtatási környezetet integráló központi repositoryt

* clab2/goalapi: Üzleti logika és REST interface és azok megvalósításai, swagger dokumentáció
* angular/clab2-goalapi-module: angular js komponens, vezérlés és html template
* site/cgol: integrált keretrendszer és alkalmazásfüggő beállítások

A clab2 keretrendszer lehetőségeit kihasználva biztosítható, hogy mind az adattárolásnak, mind a szabályrendszernek több egymástól független implementációja megvalósuljon. Az implementációktól függően az adatok menthetően mongo vagy sql adatbázisba, doctrine vagy más tetszőleges adatkezelő segítségével. A játékszabályokból tetszés szerinti változat készíthető. Az alkalmazásban az aktuális implementációk a konkrét konfigurációban jelölhetőek ki.

Üzleti logika Interfészek:

* ILivingSpace: az élettér paramétereit és állapotát leíró és tároló objektumok interfésze
* ILivingSpaceFilter: az elmentet életterek közt kereső objektumok interfésze
* ILivingSpaceAdapter: Az élettér objektumok adatkezelését végző objektum interfésze
* IGameControllerAdapter: A játékszabályokat tartalmazó és az élettér állípotát manipuláló objektum interfésze

REST API interfészek

* IGameService: a játék állapotait kezelő interfész
* ILivingSpaceService: a játék életterének tárolásával, előkeresésével kapcsolatos interfész

Frontend modul részei

* A REST API swagger dokumentációja alapján generált model és Rest kliensek
* GameController: a user interface-t vezérlő logika
* living-space template: a megjelenítésért felelős angular template

## Interfész leírások

### ILivingSpace

A játék életterének paramétereit és a cellák aktuális állapotát tároló objektum.

Adatmezők:

* id, integer: az élettér adatbázis azonosítója
* name, string: az élettér elnevezése
* width, integer: az élettér vízszintes mérete
* height, integer: az élettér függőleges mérete
* cells, array<array>int>>: az élettér állapot mátrixa
* step, integer: az élettér generációjának kora ( a lefutott iterációk száma)
* savedOn: integer: a mentés időpontja (UNIX timestamp)

### ILivingSpaceFilter

A játék élettereinek keresési paramétereit tároló objektum.

Keresési paraméterek: id

Metódusok: getResults

### ILivingSpaceAdapter

A játék életereit mentő és elökereső adatbázis adapter

Metódusok: get, search, save, fetch, parse, create, createSpace

### IGameControllerAdapter

Az élettér állapotát a játékszabályoknak megfelelően manipuláló adapter

Metódusok:

* next: Az élettér következő állapotát előállító fv. Bemeneti paraméter: az élettér (aktuális állapotban), visszatérési érték: az élettér (a következő iteráció után)

Működés: A cellák állapotát meg kell változtatni az aktuális játékszabály implementációjának megfelelően.

### IGameService

A szabályrendszer üzleti logikáját kiajánló REST interfésze

Metódusok: initAction, nextAction

### ILivingSpaceService

Az élettér adatmanipulációs REST interfésze

Metódusok: samplesAction, loadAction, savegAction, getAction, saveAction

## Implementációk

Ha még lenne egy nap kifejteném.