Fejlesztési Dokumentáció

Nitro Solutions Zrt |1097 Budapest, Könyves K. krt. 12-14

Tic tac Toe játék

Klug Attila

2021

Tartalom

[Fejlesztés témájának meghatározása 2](#_Toc64464303)

[Megvalósíthatósági követelmények 2](#_Toc64464304)

[Architektúra és tervezés 2](#_Toc64464305)

[Backend 2](#_Toc64464306)

[Követelmények: 2](#_Toc64464307)

[Tervezési fázisok 3](#_Toc64464308)

[Frontend 4](#_Toc64464309)

[Követelmények 4](#_Toc64464310)

[Tervezési fázisok 4](#_Toc64464311)

[Megvalósítás 4](#_Toc64464312)

[Az üzleti logika funkcionalitása: 4](#_Toc64464313)

[Egy játékos módban való működés 5](#_Toc64464314)

[Két játékos módban való működés 5](#_Toc64464315)

# Fejlesztés témájának meghatározása

Készíteni kell egy amőba (Tic Tac Toe) játékot a felhasználók számára. Az játék lényege, hogy egy 3x3 -as négyzetrácson 2 felhasználó (vagy 1 felhasználó és komputer) egymást követően jelölnek a négyzet rács egy részébe. Az egyik felhasználó csak a X-et, ellenfele pedig egy O-t tesz. Akinek egy sorban vagy oszlopban sikerül 3 ugyanolyan jelet tennie, akkor az a felhasználó nyer.

## Megvalósíthatósági követelmények

Elindul egy új játék üres táblával. Megjegyzi, hogy ki nyert, és mutatja a képernyőn az aktuális soron következő játékost, valamint a játék eredményeit. Ha vége a játéknak, akkor elindul egy új játék üres táblával.

Az alábbi eredményeket kell megjeleníteni:

* Játékos 1 győzelmek számát
* Játékos 2 győzelmek számát
* Döntetlenek számát
* Computer győzelmek számát

Adatbázisba kell menteni, hogy ki hányszor nyert eddig, mennyi a döntetlen, így, ha újra elindítom a játékot, akkor mutatja az eddigi állást. Az 1 felhasználó mód esetén a gép ellen játszunk. Ehhez kell megvalósítani egy kis mesterséges logikát. A képernyőre szükség van egy gombra is, amellyel a játék módját tudjuk váltani miszerint 1 vagy több felhasználós módban játszunk e.

# Architektúra és tervezés

Az alkalmazás architektúraiban 2 fő részből kell, hogy álljon.

* Backend
* Frontend

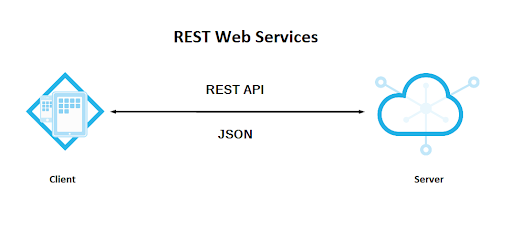
A két fő rész fogja együttesen adni a megoldást. A fejlesztést ebben a sorrendben is kell végig vinni.

## Backend

A backend (szerver) megvalósítása Java Spring keretrendszer környezet felhasználásával történjen.

### Követelmények:

* A backend a frontenddel JSON alapú REST API végpontokon keresztül kommunikáljon.



* A szerver kapcsolódjon egy Microsoft SQL adatbázis szerverhez Hibernate komponens (technológia) felhasználásával.
* Minden logika, kiértékelés a játék során a szerver oldalon kell, hogy történjen így minimális funkciók elegendőek a kliens oldalra majd. Ezzel egy böngésző teljesítménye helyett a szerver teljesítményét használjuk ki inkább.

### Tervezési fázisok

1. Első sorban az alapkörnyezetet kell létrehoznunk a rendszer számára. Létre kell hoznunk a projectet, melyben az alap könyvtárakat és package-et létrehozzuk az alábbiak számára:

|  |  |
| --- | --- |
| Komponensek | Package |
| Spring | hu.mydomain.springcore |
| objektumok | hu.mydomain.domain |
| interfészek | hu.mydomain.intf |
| szolgáltatások | hu.mydomain.services |
| adatbázis – adat kezelés | hu.mydomain.repositories |

1. implementáljuk be a projecthez a szükséges bővítmények behúzásához szükséges Tools-ot. Jelen esetünkben a Gradle lesz számunkra a megfelelő segédeszköz.
   1. Húzzuk be a szükséges bővítményeket:
      1. Spring boot
      2. Spring web
      3. Spring jdbc
      4. Hibernate
      5. Microsoft SQL jdbc
2. Szerver működése és üzleti logikájának elkészítése.
   1. Első a Spring környezet működésének megvalósítása
   2. Aztán a játék (Game) objektumunk elkészítése
      1. Paraméterei:
         1. 3x3 rács
         2. Játékos 1 eredmény
         3. Játékos 2 eredmény
         4. Computer eredmény
         5. Döntetlen
         6. És egyéb változók a játék vezérléshez
   3. Adatbázis kapcsolódás megvalósítása a Spring-hez.
   4. Game (játék) mentésének és betöltésének megvalósítása a repositories package-ben.
   5. Game interfészének elkészítése a frontend számára.
      1. JSON alapú objektum átadás.
   6. Üzleti logika megvalósítása a játékhoz.
      1. Itt szükséges egy „mesterséges intelligencia”, mely a gép elleni játék során a gép ez által fogja tenni a következő lépést.

## Frontend

A Frontend adja a kliens oldalt, amelyet a felhasználók kezelnek majd.

### Követelmények

* A Frontend (kliens) oldal HTML alapon kell megvalósítani.
* A Frontend REST API-n keresztül kell kommunikálnia a backend-del.
* Az oldal megjelenítését 3 részre kell bontani (rendezve).

|  |  |
| --- | --- |
| Cím | |
| 3x3 mező | Eredmények megjelenítése |

### Tervezési fázisok

1. Első a weboldal megvalósítása és elrendezés komponenseinek elkészítése.
   1. HTML
   2. DIV tagok
2. Megjelenítéshez szükséges stílusok elkészítése
   1. 3x3 mező stílus
   2. Eredmények megjelenítéséhez szükséges stílus
3. A REST kommunikációhoz szükséges komponensek behúzása és használatának elkészítése. Gyors egyszerű megoldás: JQuery.
4. Esemény kezelés megvalósítása:
   1. Négyzetrács egér kattintás.

# Megvalósítás

## Az üzleti logika funkcionalitása:

Amikor a rendszer fút, és megjelenik a weboldal, akkor a weboldal betöltése után hozzáfordul a szerverez és lekérdezi a Játékot (Game) a rendszertől. A szerver az adatbázisból kiolvassa az utoljára elmentet játékot az adatbázisból. Ha ilyen nem volt, akkor létrehoz egyet, és azt elmenti az adatbázisba, majd ezt visszaküldi válaszként a weboldalnak. Az oldal frissíti az adatokat az oldalon és a játék ezután kezdődik.

## Egy játékos módban való működés

A játékos rákattint egy üres mezőre. A Game objektumunk 3x3-as rácsában a megfelelő mezőt kitölti és beküldi a szerverre. Ezután a szerver fogadja az objektumot és megvizsgálja, hogy milyen módban vagyunk.

Ha egy játékos módban, akkor az üzleti logikában lévő Computer algoritmus kitölti a következő mezőt.

* Elindítja a kiértékelést a mezőkre és ha még nincs azonos 3 mező egy sorban vagy oszlopban és még van szabad mező akkor visszaküldi a kliens számára az objektumot.
* Amennyiben vannak azonos mezők, úgy a logika megvizsgálja, hogy kinek a tétjei voltak ezek és annak függvényében változtatja az objektumot és visszaküldi a kliensnek.
* Ha elfogytak a mezők, úgy, hogy egyik félnek sem volt 3 azonos mezője, akkor döntetlen állapottal küldi vissza a kliensnek az objektumot.

## Két játékos módban való működés

A játékos rákattint egy üres mezőre. A Game objektumunk 3x3-as rácsában a megfelelő mezőt kitölti és beküldi a szerverre. Ezután a szerver fogadja az objektumot és megvizsgálja, hogy milyen módban vagyunk.

Ha egy játékos módban, akkor az alábbiak alapján megyünk tovább:

* Elindítja a kiértékelést a mezőkre és ha még nincs azonos 3 mező egy sorban vagy oszlopban és még van szabad mező akkor visszaküldi a kliens számára az objektumot.
* Amennyiben vannak azonos mezők, úgy a logika megvizsgálja, hogy kinek a tétjei voltak ezek és annak függvényében változtatja az objektumot és visszaküldi a kliensnek.
* Ha elfogytak a mezők, úgy, hogy egyik félnek sem volt 3 azonos mezője, akkor döntetlen állapottal küldi vissza a kliensnek az objektumot.

Az oldal frissíti az adatokat az oldalon minden szervertől kapott válasz esetén.

# Objektumok kezelés

A fő objektumunk a Game (Játék) osztály (entitásunk) lesz.

változói:

|  |  |  |  |
| --- | --- | --- | --- |
| Változó | Típus | Leírás | Alapértelmezett érték |
| id | String | Egyedi azonosító. Minden entitás új létrehozásnál generálódik automatikusan. | UUID |
| playerOne | Integer | Az 1-es játékos eredményeit tároljuk ebben a változóban. | 0 |
| playerTwo | Integer | Az 2-es játékos eredményeit tároljuk ebben a változóban. | 0 |
| computer | Integer | Az computer játékos eredményeit tároljuk ebben a változóban. | 0 |
| draw | Integer | Döntetlenek számát tároljuk ebben a változóban | 0 |
| multiplayer | boolean | Egy vagy több játékos mód változója | false |
| theEnd | boolean | Vége van-e a játéknak | false |
| numberOfMatches | Integer | Összes játékkör száma | 0 |
| fields | Boolean[][] | Tábla mezők 3x3 méretben. |  |
| selectedPlayer | String | A soron következő játékos megjelenítéséhez használt változó | „Player 1” |
| clickMe | boolean | Ki az aktuális játékos | true |
| winner | String | A nyertes kiíratásához használt változó |  |
| restart | boolean | Játék újraindításához szükséges változó | false |

Move Objektum: Segéd tábla a mesterséges inteligenciához.