

Tic Tac Toe játék elkészítésének dokumentációja

Hogyan kezdte neki a feladatnak?

Átolvastam a leírást, többször, hogy világos legyen a feladat és be tudjam osztani az időt. A játékot ismertem, így a játékszabályok világosak voltak. Mivel más elfoglaltság miatt nem volt időm géphez ülni egyből, így fejben gondoltam végig a frontend és a backend közötti kommunikációk menetét, és hogy milyen adatokat kellene közvetíteni. Majd nekiálltam és megcsináltam.

Mik voltak a tervezés lépései?

- Első körben begyűjtöttem a szükséges adatokat. Backlog, játék lényege, elvárások.
- Lemodelleztem az időbeli, funkcionalitást és layerek közötti kommunikációt
- Megterveztem a megfelelő architekturális felépítést.
- Leprogramoztam, designoltam
- Sokat teszteltem

Első körben kialakítottam a szükséges modelleket. Milyen objektumokra van szükség. Létrehoztam a controller réteget és meghatároztam az endpointokat és az átadni kívánt adatokat. Erre épült a Service réteg, ahol létrehoztam az üzleti logikát a bejövő kérésekhez/adatokhoz.

Mivel tesztelés nem volt feladat, és az idő rövideje miatt sem lett volna gazdaságos, így Swagger segítségével teszteltem manuálisan a végpontokat és az funkcionalitást.

Amikor ez már jól működött, elkezdtem a frontend részét megoldani.

A frontenden ajax kommunikációt választottam, ahol a JavaScripttel kapott adatokat feldolgoztam és el kezdtem designolni a webfelületet. Így már könnyebbé vált a tesztelés is. A megadott listán szereplő feltételeket követve és a játék helyes működéséhez szükséges feltételeket sorra oldottam meg. Szem előtt tartva a későbbi funkcionalitások mint például az adatbázisban tárolást.

Hogyan épül fel az alkalmazás, “architektúra”

A GameController kezeli a REST API hívásokat. Ebből elég kettő: StartGame, ami egy új játék inicializálása, A makeMove pedig a frontenden történő lépést küldi át. Minden esetben egy komplett játékállást kap vissza a frontend, amit megjelenít. Ezek a végpontok a GameService-ben oldják meg az üzleti logikát.

A GameService-ben zajlik a játék felügyelete. A createGame -ben az adatok alapértelmezettre állítása, játék(osok) adatbázisból való beolvasása, ha még nincs adatuk, akkor tárolása kezdő értékekkel.


A makeMove ellenőriz, majd végrehajtja a lépést, ellenőrzi az állást és beállítja a státus értékeit.

A makeAiMove a gép lépését számolja ki.

Nyerés, vagy döntetlen esetén a countWins metódus értékeli ki a játékot és tárolja az eredményeket.

A GameDao(Jdbc) beolvassa / tárolja SQL parancsokkal a játék adatait.

Az adatbázisban a táblát a shema.sql hozza létre.

game	
 id	bigserial
pl1_name	varchar(30)
pl2_name	varchar(30)
pl1_win	int
pl2_win	int
tie	int

A modell rétegen a játék entitásai találhatóak.

A frontend 4 elemből áll. Az index.html, ami a css segítségével a játékmezőt jeleníti meg és behívja a JavaScript parancsokat. Ez 2 file-ban található, az egyik a megjelenítésért felel, a másik az ajax kommunikációért felel REST API-n keresztül.

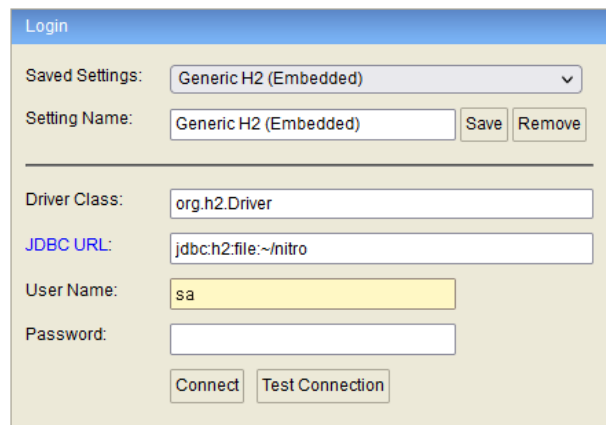
A kommunikáció a frontend és a program között Json formátumban zajlik.

Endpoint tesztelésekhez a Swagget lehet használni: <http://localhost:8080/swagger-ui.html>

Endpoint doc: <http://localhost:8080/v2/api-docs>

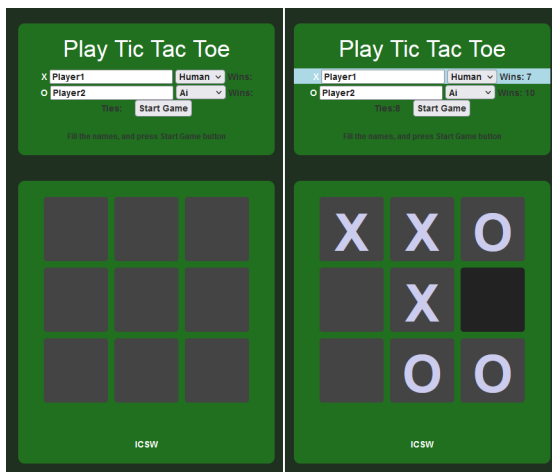
Adatbázis piszkáláshoz:

<http://localhost:8080/h2-console/>



Docker image létrehozását a docker.build.bat segíti, a docker.run.bat futtatja is.

A működését, és hogy mit miért úgy csináltál?



A program kezdőképernyőjén 2 mező látható.

A felső mezőben lehet megadni a játékosok neveit, majd a Start Game gomb megnyomásával lehet elindítani a játékot.

Ez a gomb mindig aktív, így a játék végeztével, vagy játék közben új menetet lehet indítani.

Az alsó a játékmező, ahova klikkelve el lehet helyezni az X / O jeleket (automatikusan) ha a felső mezőben elindítottuk a játékot.

Amelyik játékosnak előbb sikerül 3 saját jelét egy sorban / oszlopban / átlóban kirakni, az nyer. Ezt a program alulra kiírja, ki nyert. Ha a tábla betelik, úgy, hogy senki sem nyert, akkor az döntetlen. Ezt is kiírja.

A program játékos névpárok alapján tárolja az elért eredményeket (ki hányszor nyert és hány döntetlen volt). Ezt H2 fájl adatbázisban tárolja Jdbc-n keresztül. A Start Game megnyomására, onnan a megfelelő eredményeket betölti és kiírja a képernyőre, amit a játék eseményei alapján rendszeresen ment.

Ha az egyik játékost Ai módba választjuk, akkor az emberi játékos után a gép kiszámol egy lépést, ami meg is jelenít. A lépés folyamata: winnerMove (van-e olyan lépése, amivel nyerhet?) defenseMove (van-e olyan lépése, amivel az ellenfél azonnali nyeresét meg tudja akadályozni?), bestMove (ez egy taktikai lépés lenne, akár az első is, ezt nem csináltam meg időhiány miatt), és ha nincs semmi ötlete a gépnek, akkor randomMove (keres egy véletlenszerű szabad helyet)

A táblán a keresést Regex mintaillesztéssel oldja meg.

Ismert hibák:

Az programban van 2 ismert hiba. Ezek a játék élvezetét nem rontják. Nem bonyolult megoldani őket, de idő limit miatt nem lettek megoldva.

1, Bár a játékos nevek megváltoztathatóak, nem tud több kliensről egyszerre futni a program.

2, A program felváltva adja a kezdési lehetőséget. Ha az Ai kezd, akkor ez csak egy játékos interakcióval tudja indítani. Ahova klikkel a játékos, ott kezd az Ai.