Web aplikacija logičke zagonetkeb brodovlje

Dumančić, Filip

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:969356

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij računarstva

WEB APLIKACIJA LOGIČKE ZAGONETKE BRODOVLJE

Diplomski rad

Filip Dumančić

Osijek, 2018.

SADRŽAJ

1. UVOD 1
1.1. Zadatak diplomskog rada 1
2. PROGRAMSKO OKRUŽENJE 2
2.1. Uvod u programske jezike 2
2.2. PHP 3
2.2.1. Općenito o PHP-u 3
2.2.2. Primjeri sintakse PHP-a 3
2.3. LARAVEL 5
2.3.1. Pristup bazama podataka 6
2.3.2. *Command line alat* 7
2.3.3. Rutiranje 7
2.3.4. Predložak pogleda (*View template*) 7
2.3.5. Raspored zadataka 8
2.3.6. Sustav datoteka 8
2.3.7. Autentikacija 8
2.4. JAVASCRIPT 8
3. BATTLESHIP (POTAPANJE BRODOVA) 10
3.1. Uvod u Battleship (potapanje brodova) 10
4. RJEŠAVANJE DIPLOMSKOG ZADATKA 12
4.1. Programski dio rada 12
4.2. Rad programskog dijela 13
4.3. Analiza programskog dijela 18
5. ZAKLJUČAK 22
LITERATURA 23
SAŽETAK 24
ABSTRACT 25
ŽIVOTOPIS 26

1. UVOD

Matematičko logička zagonetka *"Potapanje brodova"*, izvornog imena *"Battleships"*, je jednostavna igra pogađanja u kojoj sudjeluju dva igrača. Širom svijeta poznata je još od vremena Prvog svjetskog rata. Još 1930-ih godina, različite tvrtke su izdavale svoje inačice igre koje su se igrale pomoću papira s poljima i olovke. Milton Bradley, 1967. godine, izdaje prvu inačicu igre u obliku plastične kutije. Deset godina kasnije, 1977. godine, izlazi prva računalna verzija ove igre pod imenom *Electronic Battleship.*

Prije početka igre, oba igrača na polja jednake veličine postavljaju svoju kombinaciju brodova. Cilj igre je prvi pogoditi postavljenu kombinaciju protivnikovih brodova. Oba igrača na raspolaganju imaju jednak broj brodova različite veličine. Svaki igrač postavlja svoju kombinaciju brodova koju smatra najtežom protivniku za pogoditi. Zatim igrači igraju naizmjenično birajući koordinate polja na kojima smatraju da se nalazi dio protivnikovog broda. Rezultat pogađanja svaki igrač upisuje u svoje polje za praćenje uspješnosti pogađanja. Polja istih koordinata mogu se gađati samo jednom. Pobjednik je igrač koji prvi uspije pogoditi cijelu protivnikovu kombinaciju brodova.

1.1. Zadatak diplomskog rada

Zadatak ovog diplomskog rada je napraviti programsku realizaciju matematičko logičke zagonetke "*Potapanje brodova*" u obliku web aplikacije. Ideja je napraviti rješenje u kojem će igrač i računalo međusobno igrati jedno protiv drugoga te pogađati prethodno postavljene kombinacije brodova.

2. PROGRAMSKO OKRUŽENJE

2.1. Uvod u programske jezike

Programski jezik kojeg koriste mikroprocesor i ostali logički sklopovi naziva se strojni jezik. Kao instrukcije, strojni jezik koristi nizove binarnih riječi koje logičkim sklopovima opisuju kako i na koji način trebaju raditi i obrađivati podatke.

Pomoću programa za uređivanje teksta, programer piše kod u obliku mnemoničkih oznaka. Kada je kod napisan, poziva se program za prevođenje mnemoničkih oznaka u binarne instrukcije strojnog jezika. Program koji izvodi prevođenje naziva se asembler (engl. *assembler*), a programski jezik zove se asemblerski jezik. Program koji je napisan u asemblerskom jeziku zove se izvorni program (engl. *source code*). Ovakav način programiranja sastoji se od dva koraka. U prvom koraku programer piše izvorni program, a zatim, u drugom koraku napisani program se prevodi iz izvornog u izvršni program. Na ovaj način, programer je oslobođen pisanja binarnih instrukcija i do neke mjere postignuta je strojna nezavisnost izvornog programa.

Pisanje programa pomoću asemblerskog jezika, odnosno korištenje asemblera, programeru omogućava potpunu kontrolu nad svim dijelovima računala. Iz tog razloga, programer bi trebao što je moguće više biti upoznat s arhitekturom računala za koje piše program. Sve operacije koje računalo izvršava su pod kontrolom programera, stoga programiranje u asembleru postaje opsežno čak i za najjednostavnije naredbe. Zbog toga, programiranje u asembleru koristi se samo za zadatke vezane uz upravljanje *hardwareom*. Izvorni program pisan u asemblerskom jeziku prilagođen je za arhitekturu svakog tipa računala [1]. Za programiranje većine zadataka koriste se složeniji programski jezici.

Prethodno navedeni nedostatci asemblerskih jezika razlog su zašto su razvijeni viši programski jezici. Neki viši programski jezici su: *Pascal, C, C++, Python, PHP, Java, JavaScript*. Viši programski jezici oslobađaju programera potrebe poznavanja arhitekture svakog računala za koje se piše program te omogućavaju izvođenje istog programa na računalima različite arhitekture. Iz tog razloga programiranje je brže i jednostavnije. Programi napisani u jezicima više razine moraju prije izvođenja biti prevedeni u izvršni kod. Za to se brine prevoditelj (engl. *compiler*). Naredbe programskih jezika više razine prilagođene su operacijama koje će se izvršavati i tipovima podataka s kojima program radi. To je osnovna razlika između višeg programskog jezika i asemblerskog. Prevoditelj prevodi program napisan u nekom višem

programskom jeziku u kod koji je moguće izvršiti na zadanom računalu. Na ovaj način, program napisan u programskom jeziku više razine moguće je izvršavati na svakom računalu ukoliko ima prevoditelj za taj jezik.

Razvoj viših programskih jezika započeo je 60-ih godina 20. stoljeća. Viši programski jezici koji su koristili prevoditelje popularno su nazvani autokodovi (*Cobol, Fortran*). Nešto kasnije, 1995. godine, pojavljuju se prve verzije programskih jezika *PHP* i *JavaScript*.

2.2. PHP

2.2.1. Općenito o PHP-u

PHP (rekurzivna skraćenica za PHP: *Hypertext Preprocessor*) je široko korišten skriptni jezik opće uporabe koji je posebno pogodan za razvoj web aplikacija jer se može lako ugraditi u HTML. PHP jezik izvodi se na poslužitelju, odnosno, izvršavanje skripti obavlja se na poslužitelju na kojem je smještena web stranica.

Skriptni jezik je oblik programskog jezika koji je interpretiran, a ne *kompajliran*. U programskim jezicima kao što su *C* ili *C++*, program se sprema u izvršnu datoteku, dok se program koji je napisan u skriptnom jeziku interpretira naredbu po naredbu. Neki drugi primjeri skriptnih jezika su *Java, Ruby, Python*.

PHP je besplatan i otvorenog je tipa. Također, PHP podržava mnoge baze podataka (*MySQL, Sybase, Oracle*...) te je kompatibilan s različitim platformama kao što su *Linux, Windows, Unix*. Ono što je karakteristično za PHP je da se izvršava na poslužitelju, generirajući HTML koji se šalje klijentu. Klijent prima rezultate pokretanja skripte, ali nema uvid u izvorni kod. Iako je PHP iznimno jednostavan za nove korisnike, s druge strane nudi mnoge napredne značajke za profesionalne programere.

2.2.2. Primjeri sintakse PHP-a

Umjesto puno dodatnih naredbi, PHP stranice sadrže HTML s ugrađenim kodom koji radi 'nešto' (u ovom slučaju ispisuje *'Hi, I'm a PHP script!'*). PHP kod nalazi se unutar posebnih oznaka. '<?php' označava početak, a '?>' kraj izvođenja PHP koda.

Jednostavan primjer korištenja PHP-a vidljiv je na slici 2.1.

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
            echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

Sl. 2.1. *Primjer PHP-a.*

Varijable u PHP-u predstavljene su znakom dolara, nakon čega slijedi naziv varijable. U nazivu varijabli, PHP razlikuje velika i mala slova. Valjano ime varijable započinje slovom ili znakom podcrtavanja ('_'), nakon čega slijedi bilo koji broj slova, brojeva ili znakova podcrtavanja. Na slici 2.2. prikazan je primjer korištenja varijabli.

```
<?php
$var = 'Bob';
$Var = 'Joe';
echo "$var, $Var";      // outputs "Bob, Joe"

$4site = 'not yet';     // invalid; starts with a number
$_4site = 'not yet';    // valid; starts with an underscore
$täyte = 'mansikka';    // valid; 'ä' is (Extended) ASCII 228.
?>
```

Sl. 2.2. *Primjer korištenja varijabli.*

U PHP-u polje je zapravo sortirana mapa (Sl. 2.3.). Mapa je tip koji povezuje vrijednosti s ključevima. Takav tip optimiziran je za nekoliko različitih primjena. Može ga se tretirati kao polje, listu (vektor), *hash* tablicu, rječnik, kolekciju, stog, red... Budući da vrijednosti polja mogu biti druga polja, moguća su i stabla, odnosno višedimenzionalna polja.

```php
$array = [
    "foo" => "bar",
    "bar" => "foo",
];
```

Sl. 2.3. *Primjer polja.*

Svaka PHP skripta izrađena je iz niza izjava. Izjava može biti dodjela, funkcijski poziv, petlja, uvjetna izjava ili čak izjava koja ne čini ništa (prazna izjava). Izjave obično završavaju točkom sa zarezom. Pored toga, izjave se mogu grupirati u grupu izjava pomoću kapsuliranja skupine izjava s vitičastim zagradama. [2]

Neke od naredbi za kontrolu toka su *if, else, while, for, foreach* (Sl. 2.4.).

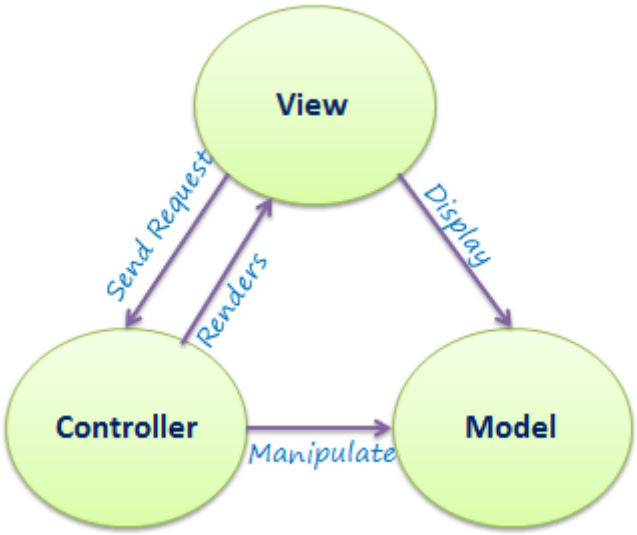
```
foreach (array_expression as $value)
    statement
foreach (array_expression as $key => $value)
    statement
```

Sl. 2.4. *Foreach naredba*.

2.3. LARAVEL

Laravel je *open source,* besplatan i jedan od najpopularnijih PHP *web frameworka* temeljen na *Model-View-Controller (MVC)* obrascu softverske arhitekture. Stvorio ga je Taylor Otwell, čiji je cilj smanjiti troškove inicijalnog razvoja i poboljšati kvalitetu koda definiranjem industrijskih standardnih praksi.

Obrazac softverske arhitekture *Model-View-Controller (MVC)* koristi se za odvajanje pojedinih dijelova aplikacije u komponente s obzirom na njihovu namjenu. Prvi od ta tri dijela je model. Model se sastoji od logike, podataka, poslovnih pravila te funkcija. Pogled (engl. *View*) je prikaz podataka koji može biti obrazac, dijagram ili tablica. Upravitelj (engl. *Controller*) povezuje inpute i povezuje ih s modelom ili pogledom. Na taj način olakšava se razvoj, održavanje i testiranje aplikacije. Odnos između komponenti prikazan je na slici 2.5.



Sl. 2.5. *Odnos između komponenti*.

Od svog prvog objavljivanja u lipnju 2011. godine, Laravel je kao i sam PHP konstantno napredovao i razvijao nove verzije *frameworka*. Verzija Laravela korištena za ovaj projekt je 5.5.

Laravel framework sadrži mnoge značajke uključujući ugrađenu autentikaciju, predložak prikaza (engl. *blade*), rutiranje, pristup bazi podataka, predmemoriranje, upravljanje datotekama...

2.3.1. Pristup bazama podataka

Laravel također pruža najsuvremeniji paket alata za interakciju s bazama podataka. Migracije baze podataka omogućuju jednostavno dizajniranje i modificiranje baze podataka neovisno o platformi. Migracije se mogu pokrenuti prema bilo kojoj vrsti baze podataka koju

Laravel podržava (*MySQL, PostfreSQL, MSSQL i SQLite*). *Laravel Fluent Query Builder* zatim služi za izradu i izvršavanje upita na bazu podataka.

Laravel implementacija *ActiveRecord* naziva se *Eloquent*. *Eloquent* pruža interakciju s bazom na objektno-orijentirani način. Pomoću *Eloquenta* možemo kreirati, ažurirati, preuzimati i brisati zapise baze podataka bez potrebe da napišemo liniju SQL koda. Također *Eloquent* pruža mogućnost definiranja odnosa među modelima.

2.3.2. *Command line alat*

Laravel dolazi s alatom za *command line* koji se zove Artisan. Uz Artisan, može se komunicirati s aplikacijom za pokretanje radnji poput pokretanja migracije podataka, zakazivanja poslova, *unit* testiranja. Artisan je lako nadograditi vlastitim naredbama za funkcionalnost kakvu želimo.

2.3.3. Rutiranje

Laravel omogućuje organiziranje svih URL-ova stranice kroz usmjerivač (engl. *router*). Ako želimo promijeniti vezu web stranica ili nekih API-ja, možemo promijeniti u jednoj datoteci dok u isto vrijeme web stranica funkcionira kako se očekuje.

2.3.4. Predložak pogleda (*View template*)

Blade je jednostavan, ali moćan predložak koji se isporučuje s Laravelom. Za razliku od drugih popularnih PHP predložaka, Blade ne ograničava na korištenje običnog PHP koda u pogledima. Svi Blade prikazi sastavljaju se u običan PHP kod i spremaju se dok ne budu izmijenjeni. Blade datoteke koriste ekstenziju .blade.php i obično su pohranjene u direktoriju *resources/views*.

2.3.5. Raspored zadataka

Od verzije Laravel 5.0 uveden je dodatak Artisanu koji omogućuje programirano raspoređivanje periodično izvršavanih zadataka. Interno, Scheduler se oslanja na *cron daemon* za pokretanje pojedinog Artisan zadatka koji zauzvrat izvršava konfigurirane zadatke.

2.3.6. Sustav datoteka

Od verzije Laravel 5.0 uveden je apstrakcijski sloj koji omogućava lokalne datotečne sustave i usluge pohrane podataka temeljenih na oblaku omogućenih od strane Amazon S3 i Rackspace Cloud-a.

2.3.7. Autentikacija

S Laravelom u paketu dolazi osnovni sustav autentikacije. Osnovna autentikacija može se lako konfigurirati na web stranicu i API kroz nekoliko koraka. Od verzije 5.0 Laravel kao izborni paket uključuje i pojednostavljene mehanizme autentikacije za različite pružatelje Oauth usluge uključujući Facebook, Twitter, GitHub, Bitbucker i Google. [3]

2.4. JAVASCRIPT

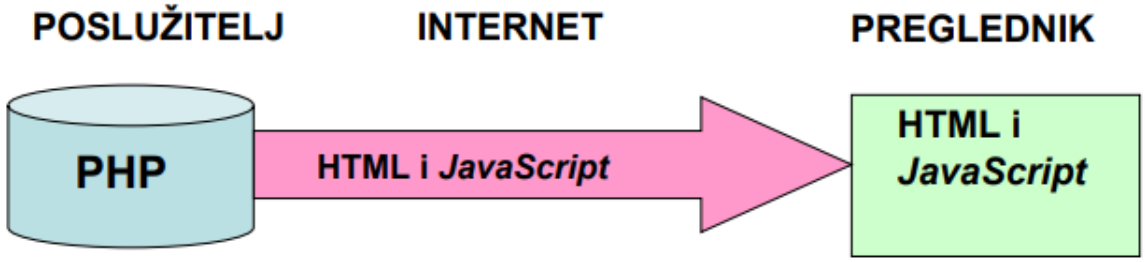
JavaScript je interpretiran, jednostavan programski jezik korišten ponajprije za razvoj interaktivnih HTML stranica. Jezgra JavaScripta sadržana je u većini današnjih internet preglednika (Google Chrome, Mozilla Firefox, Opera, Internet Explorer, Safari...). Prvu inačicu JavaScripta predstavio je Netscape 1995. godine. Nedugo zatim, Microsoft predstavlja jezik sličan JavaScriptu pod nazivom Jscript. Danas, za standardizaciju skriptnih jezika, uključujući i JavaScript zadužena je organizacija ECMA.

JavaScript se može uključiti u HTML dokument pisanjem koda između oznaka *<script>* i *</script>*, iz vanjske datoteke uporabom atributa *'src'* u oznaci *<script>*, preko obrade događaja navedenog kao HTML atribut (npr. *onmouseover, onclick*), u URL-u koji se koristi posebnim protokolom *javascript:*.

JavaScript se može pisati prema Unicode standardu, no preporučuje se pisanje prema ASCII standardu. Ključne riječi, varijable, funkcije i drugi nazivi moraju se pisati dosljedno jer JavaScript razlikuje mala i velika slova. Točka-zarez (;) označava kraj naredbe, ali, za razliku od PHP-a, nije obavezan znak. JavaScript na kraj retka ubacuje točku-zarez ako pojedini kod izgleda kao naredba što u nekim slučajevima nije ono što je programer htio.

Nazivi varijabli i funkcija moraju se pridržavati pravila gdje prvi znak u nazivu mora biti slovo, podcrta (_) ili znak dolara ($), dok znakovi koji slijede mogu biti slova, brojevi, dolarski znak ili podcrta. [4]

Primjer toka podataka od web poslužitelja do web preglednika za PHP i JavaScript prikazan je na slici 2.6.



Sl. 2.6. *Primjer toka podataka od web-poslužitelja do web-preglednika za PHP*.

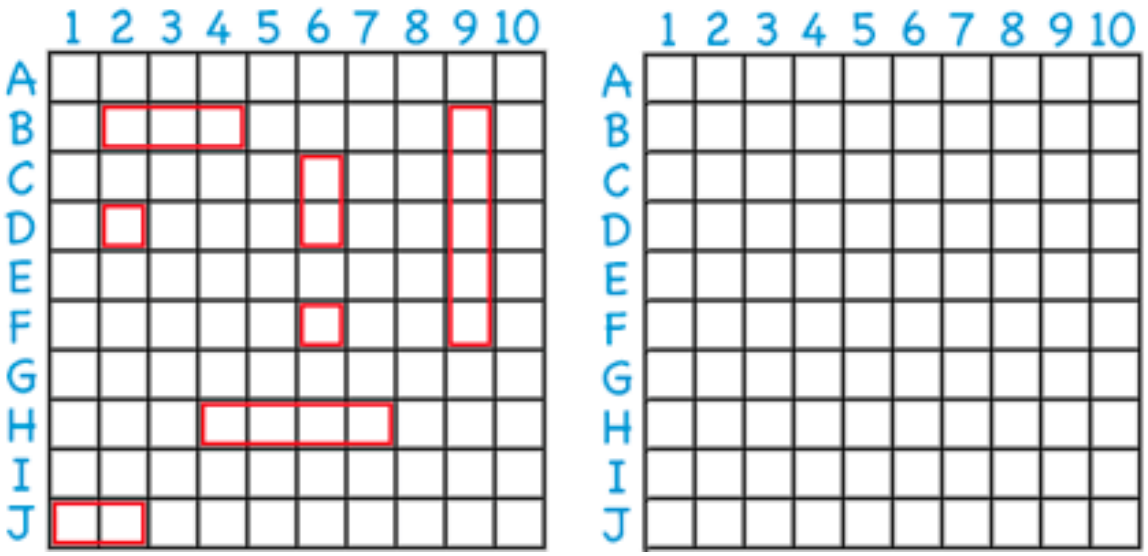
3. BATTLESHIP (POTAPANJE BRODOVA)

Battleship (Potapanje brodova) je igra, nastala početkom 19. stoljeća, u kojoj sudjeluju dva igrača. Prve inačice igre igrale su se pomoću papira i olovke. Milton Bradley, 1967. godine, predstavlja prvu verziju ove igre u obliku plastične kutije s figuricama. Prva računalna inačica ove igre predstavljena je deset godina kasnije pod imenom Electronic Battleship.

3.1. Uvod u Battleship (potapanje brodova)

Prve inačice igre igrale su se pomoću papira i olovke (Sl. 3.1.). Oba igrača na raspolaganju su imala dva papira s ucrtanim poljima jednake veličine. Na prvom polju svaki igrač ucrtavao je svoju kombinaciju brodova za koju smatra da će protivnik najteže pogoditi. Oba igrača na raspolaganju imala su jednak broj brodova različitih dužina. Kada su oba igrača ucrtali svoju kombinaciju brodova, naizmjenično su govorili koordinate na kojima smatraju da se nalazi dio protivnikovog broda. Rezultat uspješnosti pogađanja upisivali su u drugo, na početku igre prazno, polje. Pobjednik igre je igrač koji prvi uspije pogoditi cjelokupnu kombinaciju protivnikovih brodova. [5] Olovku i papir, nešto kasnije, zamjenila je modernija inačica igre u obliku plastične kutije s figuricama (Sl. 3.2.). Milton Bradley predstavio je 1967. godine prvu takvu verziju ove igre.

Prva računalna verzija ove igre pod imenom Electronic Battleship predstavljena je 1977. godine. Potapanje brodova (Battleship) jedna je od najranijih igara za koje je predstavljena računalna verzija.



Sl. 3.1. *Verzija igre Battleship s papirom i olovkom.*



Sl. 3.2. *Battleship igra u obliku kutije s figuricama*.

4. RJEŠAVANJE DIPLOMSKOG ZADATKA

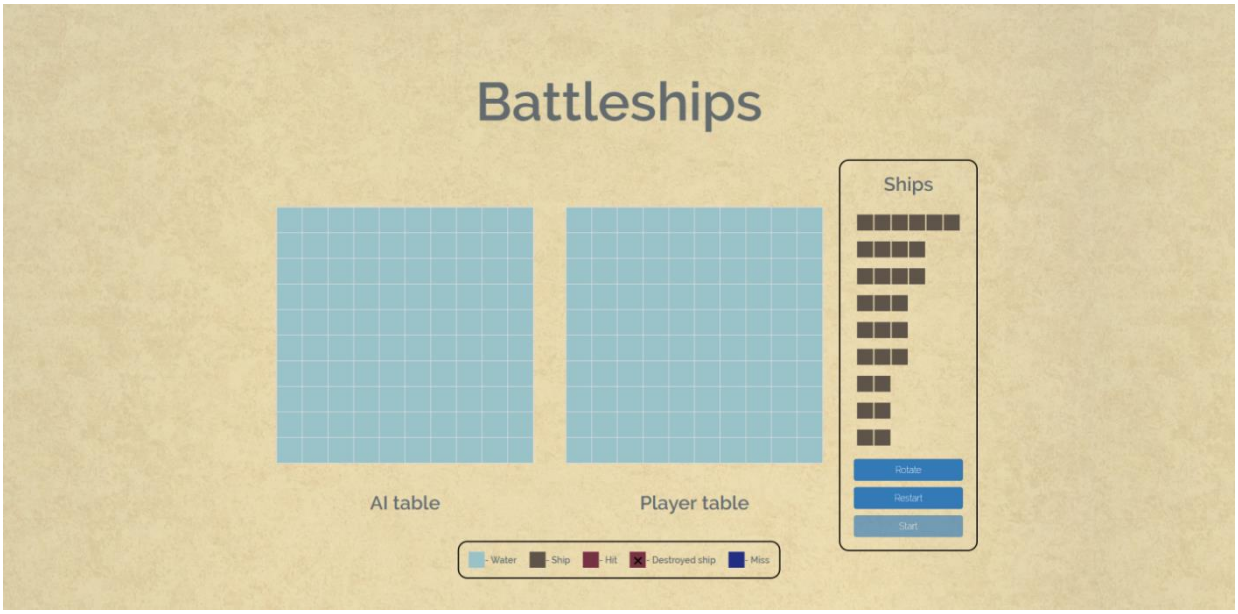
Ideja je napraviti web aplikaciju u kojoj će se igra odvijati između igrača i računala (AI). Obje strane na raspolaganju će imati jednak broj brodova različite veličine. Nakon što postave svoje kombinacije brodova, naizmjenično će pogađati koordinate na kojima se može nalaziti dio protivnikovog broda sve dok jedna strana ne pogodi cijelu kombinaciju.

4.1. Programski dio rada

Igra se odvija između igrača i računala. Igrač prije početka igre postavlja svoju kombinaciju brodova na polje 10x10 na način da se susjedni brodovi ne smiju dodirivati (osim dijagonalno). Računalo (AI) generira svoj nasumični uzorak brodova i postavlja ih na svoje polje, koje je igraču sakriveno i obojeno svijetlo plavom bojom. Uzorak brodova sastoji se od tri broda 1x2, tri broda 1x3, dva broda 1x4 te jednog broda 1x6. Nakon što je igrač postavio svoju kombinaciju brodova, pritiskom na tipku ‘Start’ započinje igra. Igrač prvi bira polje na kojem misli da se nalazi dio broda kojeg je postavilo računalo. Igrač i računalo igraju naizmjenično. U slučaju pogotka polje se boja u crvenu boju, u slučaju promašaja polje postaje tamno plave boje. Ako je računalo ili igrač pogodilo cijeli brod, polja na kojima se nalaze dijelovi broda postat će prekrižena, dok će sva polja koja dotiču taj brod (osim dijagonalno) biti obojena tamno plavom bojom. Kada je računalo na potezu, nasumično generira koordinate reda i stupca te provjerava nalazi li se na tim koordinatama dio broda. U slučaju pogotka računalo na svom idućem potezu generira nove koordinate, ali ovoga puta preferira koordinate koje su iznad, desno, ispod ili lijevo od prethodno pogođene koordinate. Ukoliko je računalo pogodilo cijeli brod, polja oko broda više neće gađati. Igrač nastoji pogoditi sve brodove prije računala i na taj način pobijediti u igri.

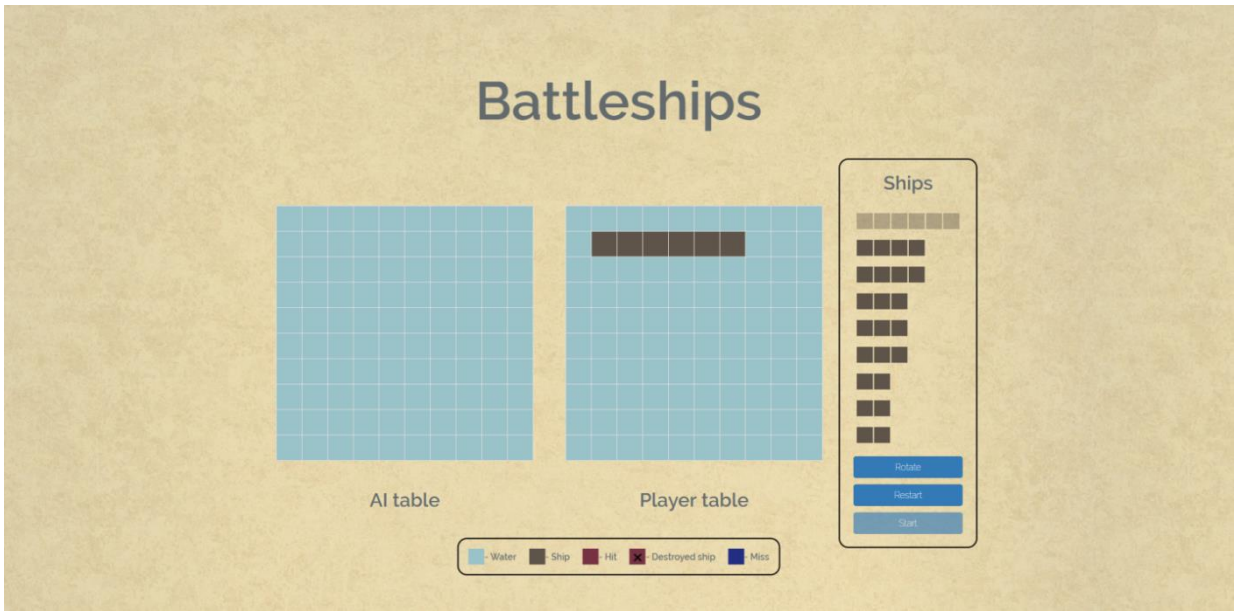
4.2. Rad programskog dijela

Program je napisan u programskom jeziku PHP te Laravel frameworku u obliku web aplikacije. Uz PHP, za izradu aplikacije korišteni su i JavaScript, HTML i CSS.



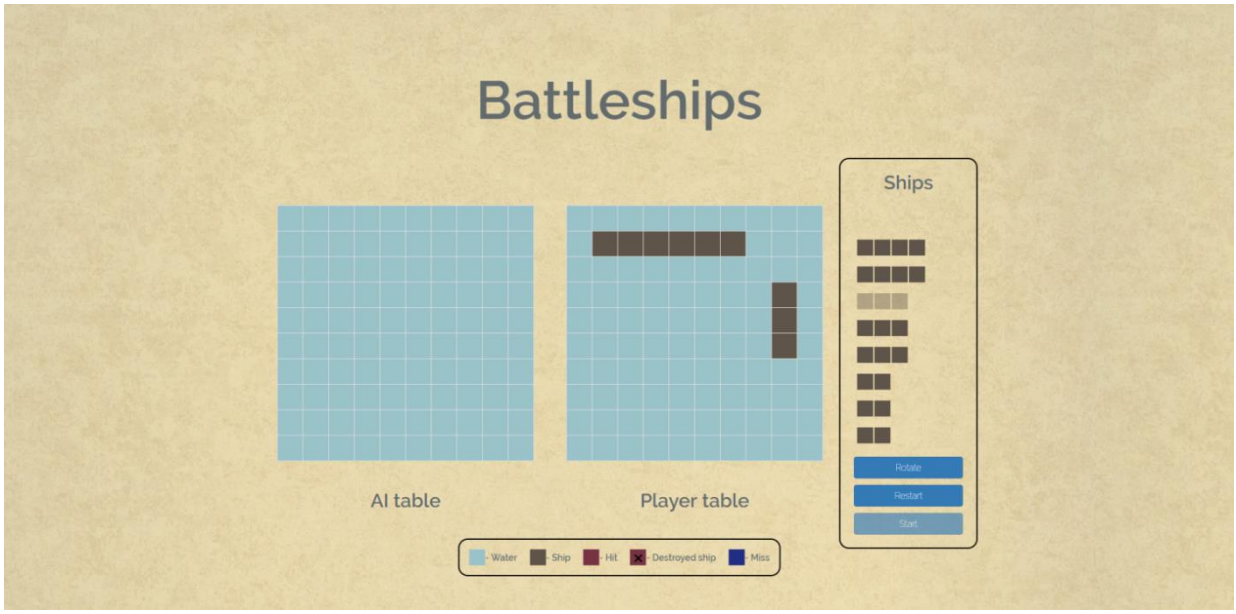
Sl. 4.1. *Izgled web aplikacije*.

Nakon što otvori web aplikaciju (Sl. 4.1.), pred igračem se nalaze dva polja, polje računala (*AI table*) i polje igrača (*Player table*). Desno od igračevog polja, unutar okvira, nalaze se brodovi koje igrač prije samog početka igre treba postaviti na svoje polje. Osim brodova, unutar okvira nalaze se i tri tipke (*'Rotate'*, *'Restart'* i *'Start'*). Ispod polja nalazi se legenda s objašnjenjem što pojedina boja bolja predstavlja u igri.



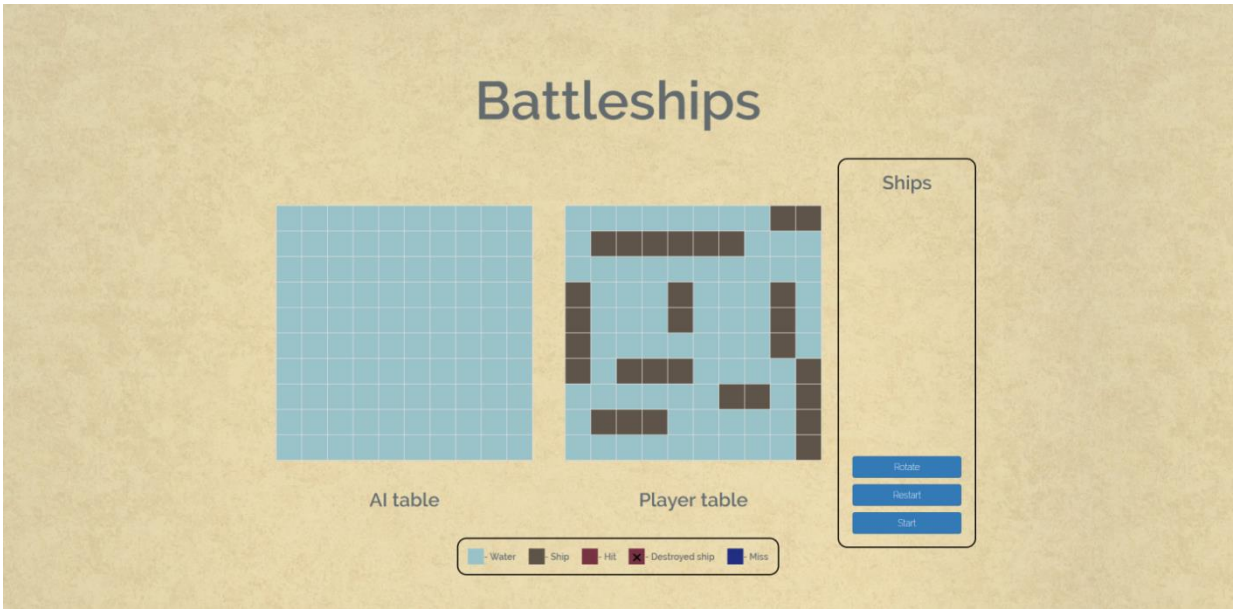
Sl. 4.2. *Postavljanje broda.*

Igrač na raspolaganju ima tri broda 1x2, tri broda 1x3, dva broda 1x4 te jedan brod veličine 1x6. Brodovi se postavljaju na način da igrač odabere željeni brod klikom na brod unutar okvira s desne strane svoga polja (Sl. 4.2.). Nakon toga odabrani brod postaje prozirniji te, nakon što igrač postavi brod na svoje polje, brod nestaje iz okvira.



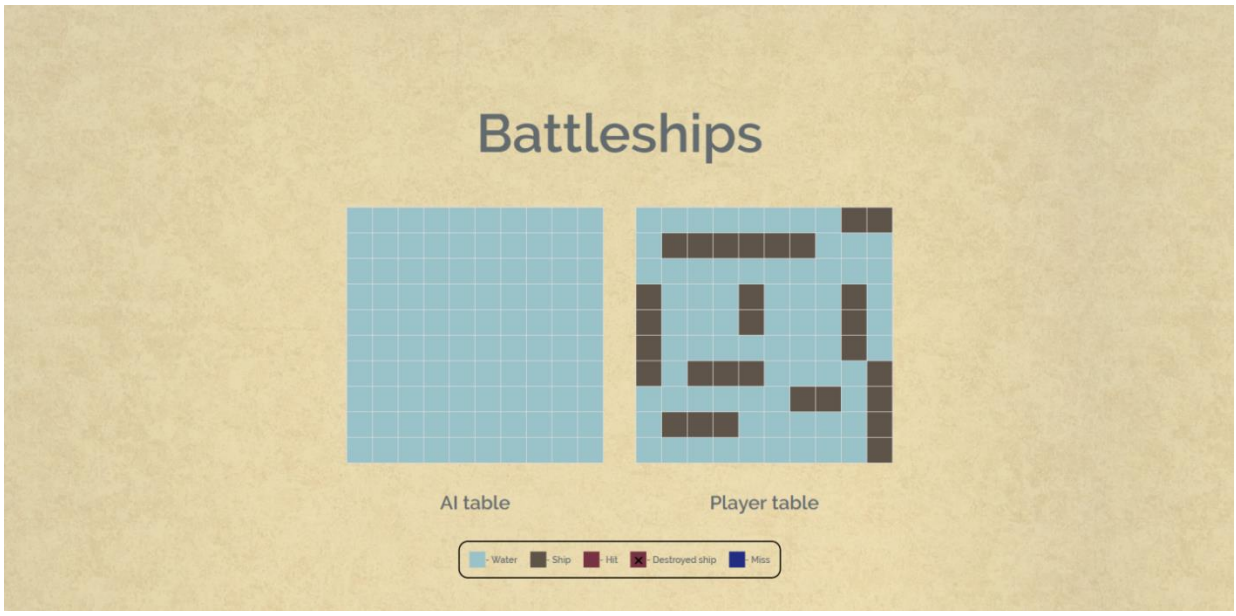
Sl. 4.3. *Rotiranje broda.*

Igrač u svakom trenutku postavljanja broda može promijeniti smjer postavljanja (zarotirati brod) klikom na tipku *'Rotate'* (Sl. 4.3.). Brod je moguće postaviti samo na igračevo polje i to u slučaju kada se brod koji se postavlja cijeli nalazi unutar igračevog polje te svojim stranama ne dodiruje susjedni brod i s njime se ne križa. Ukoliko je igrač nezadovoljan s načinom na koji je postavio sve ili neki od brodova klikom na tipku *'Restart'* može ponovo pokrenuti web aplikaciju te ispočetka krenuti s postavljanjem svojih brodova.

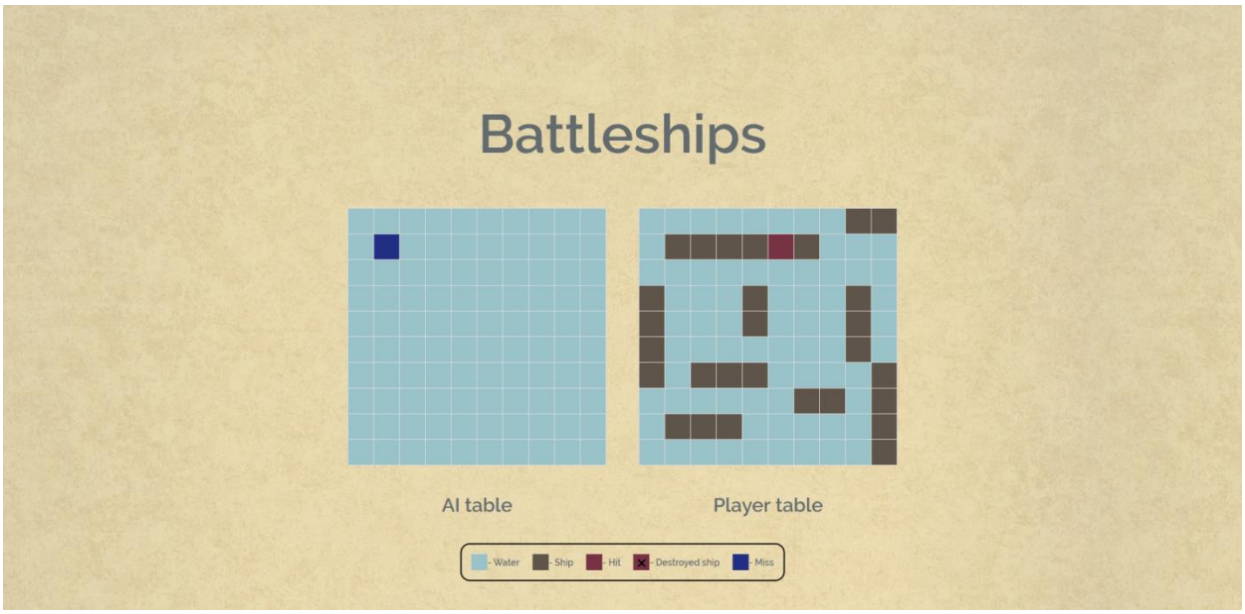


Sl. 4.4. *Aktivirana tipka 'Start'.*

Kada su svi brodovi postavljeni na igračevo polje, unutar okvira s desne strane igračevog polja, aktivira se tipka *'Start'* (Sl.4.4.). Klikom na tipku *'Start'* započinje igra i nestaje okvir s desne strane (Sl. 4.5).

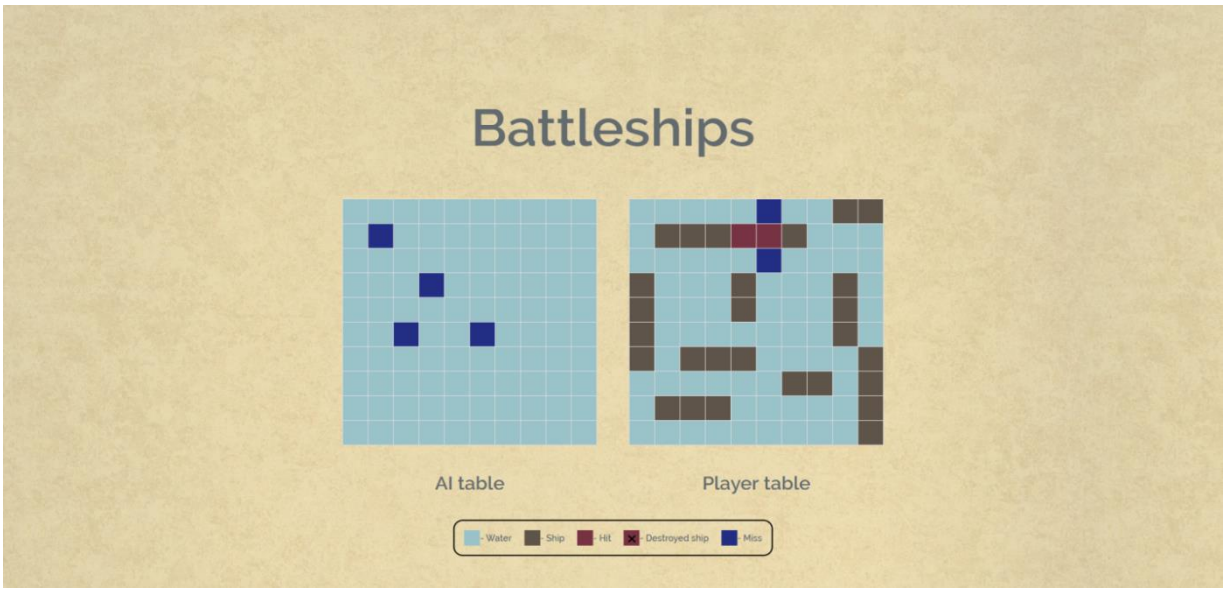


Sl. 4.5. *Početak igre.*



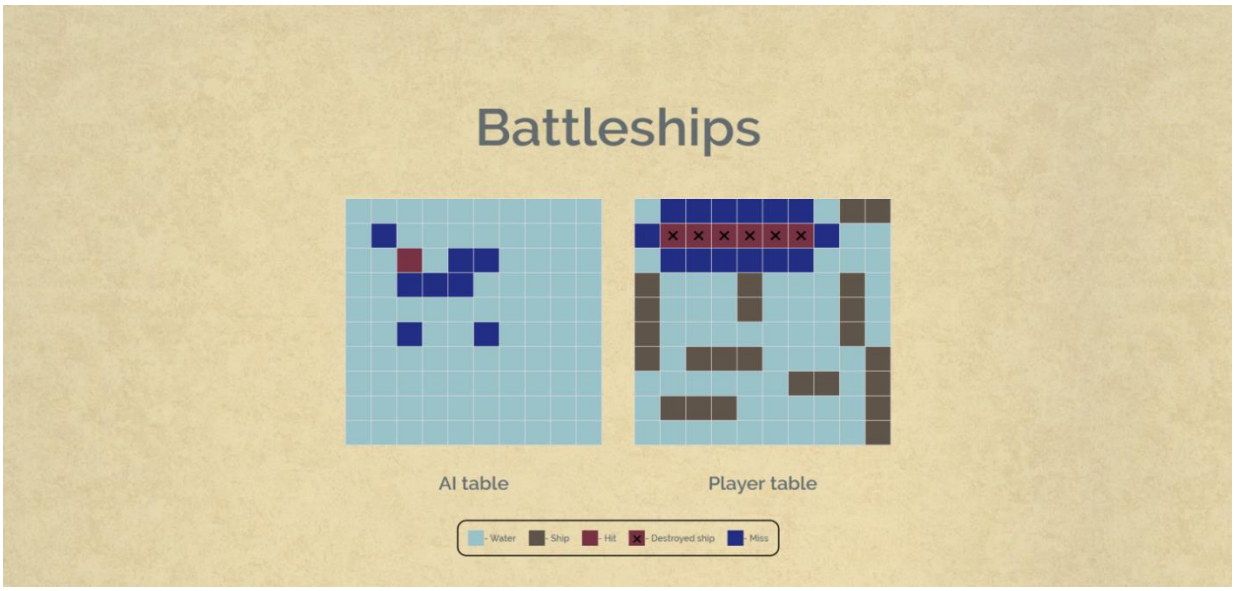
Sl. 4.6. *Prvi potez.*

Igrač je prvi na potezu. Klikom na polje računala (AI table), igrač bira koordinatu polja na kojoj smatra da se nalazi dio broda. Nakon njega igra računalo (AI) i nakon analize koordinata igrača i računala istovremeno se mijenjaju oba polja. U slučaju pogotka polje se mijenja u crvenu boju, dok se u slučaju promašaja polje mijenja u tamno plavu boju. Rezultat pogađanja igrača označava se na polje računala, a rezultat pogađanja računala označava se na polje igrača (Sl. 4.6.).



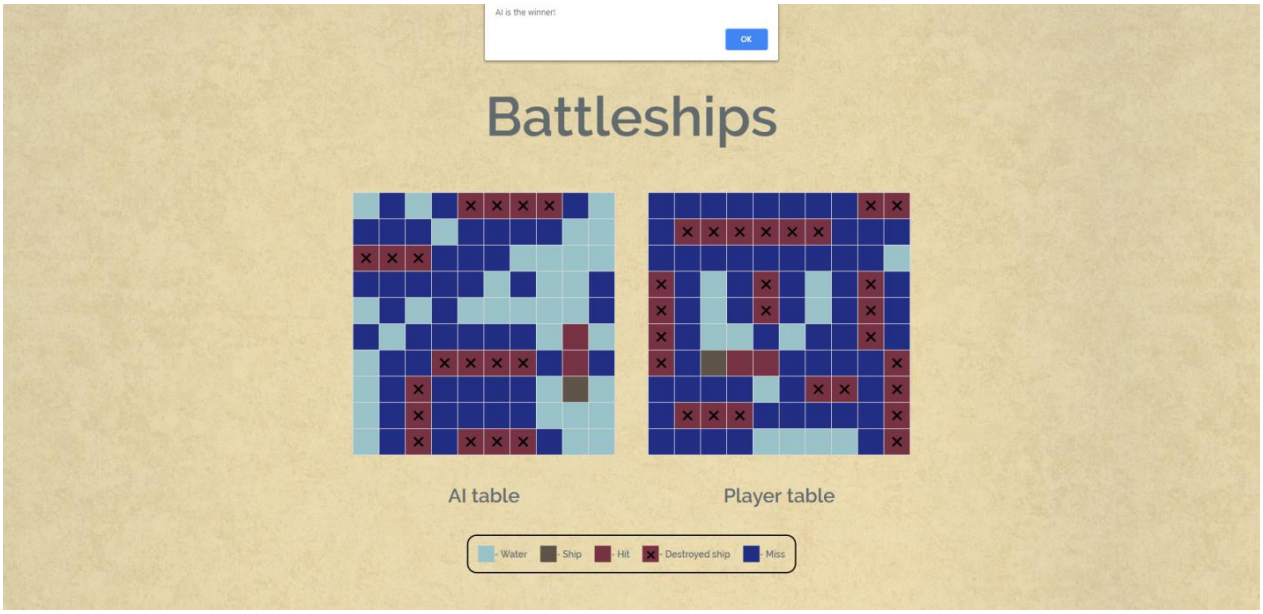
Sl. 4.7. *Računalo (AI) nakon pogotka*.

Nakon što je računalo (AI) pogodilo koordinatu polja na kojoj se nalazi dio broda, u svojim sljedećim potezima preferirat će koordinate koje se nalaze iznad, ispod, lijevo ili desno od prethodno pogođene koordinate sve dok ne pogodi idući dio broda (Sl. 4.7.). Zatim će nastaviti birati koordinate u smjeru pogođenog dijela broda sve dok rezultat pogađanja ne bude promašaj ili dok nije uništen cijeli brod. Ako je rezultat promašaj, računalo će okrenuti smjer i gađati koordinate u suprotnom smjeru sve dok ne uništi cijeli brod.



Sl. 4.8. *Uništen cijeli brod*.

Ukoliko je uništen cijeli brod, sva polja ispod i iznad broda bit će označena kao promašaj i računalo ih više neće gađati. Polja koja sadrže dijelove uništenog broda bit će označena crvenom bojom sa znakom 'x' u sredini (Sl. 4.8.) Nakon što je uništilo cijeli brod, računalo će birati novu nasumičnu koordinatu polja koja je dostupna za pogađanje.



Sl. 4.9. *Kraj igre.*

Igrač i računalo naizmjenično pogađaju koordinate polja sve dok netko ne pogodi cijelu protivnikovu kombinaciju brodova. Kada se to dogodi, u iskočnom prozoru ispisuje se ime tko je pobjednik (Sl. 4.9). Klikom na tipku 'OK' u iskočnom prozoru igra se vraća na početak i igrač može započeti novu igru postavljanjem kombinacije brodova i pritiskom na tipku *'Start'*.

4.3. Analiza programskog dijela

Web aplikacija izrađena je u MVC frameworku Laravel. Korišteni su programski jezici PHP i JavaScript. Osim PHP-a i JavaScripta korišteni su i HTML i CSS. Kreirane su 3 rute. Osnovna ruta '/' vraća pogled na kojem se nalazi web aplikacija. Ruta '*playerTable*' aktivira se klikom na tipku *'Start'* i poziva istoimenu metodu unutar kontrolera. Ruta '*playerShotInput*' također poziva istoimenu metodu unutar kontrolera, a aktivira se nakon što igrač odabere polje na kojem misli da se nalazi dio broda računala.

Pogled koji vraća ruta '/' naziva se *'main'* i nalazi se unutar 'main.blade.php' datoteke. U njemu su definirani svi vidljivi elementi. Elementi su stilizirani unutar <style> </style> oznaka korištenjem CSS-a (Sl. 4.10.).

```
.water-legend div {
    width: 25px;
    height: 25px;
    background: #9AC2C9;
}
.ship-legend div {
    width: 25px;
    height: 25px;
    background: #5F5449;
}
.hit-legend div {
    width: 25px;
    height: 25px;
    background: #773344;
}
.sunk-legend div {
    width: 25px;
    height: 25px;
    background: #773344;
}
```

Sl. 4.10. *Dio CSS koda za stiliziranje legende*.

Prikazane su dvije 10x10 tablice koje označavaju polje igrača i polje računala. Unutar okvira s desne strane igračevog polja nalaze se brodovi koje igrač prije početka igre mora postaviti na svoje polje. Sva funkcionalnost ovog dijela web aplikacije definirana je pomoću JavaScripta unutar 'list.js' datoteke. Definirane su funkcije za sve tri tipke. Tipka *'Rotate'* aktivira se klikom i ukoliko je aktivirana mijenja smjer postavljanja broda. Tipka *'Restart'* ponovno pokreće web aplikaciju i također se aktivira klikom. Za razliku od prethodne dvije tipke, tipku *'Start'* nije moguće odmah aktivirati. Uz nju je vezana klasa 'disabled' sve dok igrač ne postavi cijelu kombinaciju brodova na svoje polje. Tek tada moguće ju je aktivirati klikom.

Prilikom postavljanja brodova radi se validacija polja na koje se brod želi postaviti. Brod je moguće postaviti samo na igračevo polje. Nakon što je označen brod koji se želi postaviti, unutar okvira označeni brod mijenja prozirnost, a prelaskom strelice miša preko igračevog polja radi se validacija i ukoliko su zadovoljeni svi uvijeti sijenčaju se polja gdje je moguće postaviti brod. Validacija za postavljanje broda odrađena je unutar 'checkIfValid' konstante pomoću arrow

funkcije. Provjerava se je jesu li koordinate na koje se želi postaviti brod barem jedno polje udaljene od prethodno postavljenog broda te nalazi li se cijeli brod unutar igračevog polja. Ukoliko su svi uvjeti zadovoljeni polja na kojima bi se nalazio postavljeni brod mijenjaju klasu i sjenčaju se. Nakon što je validacija uspješna brod je moguće postaviti klikom. Koordinate postavljenih brodova spremaju se u posebnu tablicu. Svaki brod ima jedinstvenu oznaku. Polja oko postavljenog broda također su označena oznakom broda kojem pripadaju. Kada su svi brodovi postavljeni, igrač klikom na aktivnu tipku *'Start'* započinje igru.

Tipka *'Start'* poziva rutu *'playerTable'* i metodi *'getPlayerTable'* preko *request* parametra šalje tablicu s koordinatama brodova koje je prethodno postavio igrač. Ta tablica koristit će se za provjeru koordinata koje je biralo računalo (AI) u svom potezu. Unutar metode *'getPlayerTable'* u *session instancu requesta* postavljaju se svi parametri potrebni za daljnji razvoj igre. *Session instanca* se koristi kako bi se pamtilo stanje varijabli između poteza. Nadalje, poziva se metoda *'placeAllShips'* koja preko pomoćnih metoda generira nasumični uzorak brodova koji zadovoljava sve potrebne uvjete (jednako kao prilikom postavljanja igračevih brodova). Povrat metode je tablica s koordinatama brodova računala i ona se sprema unutar *session instance requesta* pod ključem *'ai_table'*. Ta tablica koristit će se za provjeru koordinata koje je prilikom svog poteza označio igrač. Uz dvije prethodno navedene tablice s koordinatama igrača i računala, još se spremaju dvije prazne tablice u koje će se upisivati rezultat pogađanja igrača i računala. Kada je sve spremno nestaje okvir s desne strane i tablice se pomiču u sredinu.

Igrač je prvi na potezu. Klikom na polje računala (*AI table*) označava polje na kojem smatra da se nalazi dio broda računala. Klik na polje tablice računala poziva rutu *'playerShotInput'* koja unutar *BoardController-a* poziva metodu *'getPlayerShotInput'*. Metodi *'getPlayerShotInput'* preko *request* parametra predane su koordinate na koje je kliknuo igrač. Zatim, metoda *'contains'* provjerava nalaze li se odabrane koordinate unutar tablice računala (*'ai_table'*) i rezultat pogađanja upisuju u tablicu za praćenje uspješnosti pogađanja igrača. Također poziva se i metoda *'AIPlays'* koja prima tablicu s koordinatama brodova igrača. Unutar te metode definiran je potez računala. Računalo najprije provjerava postoji li pogodak iz prethodnog poteza. Ukoliko postoji nastavlja istraživati oko pogođenog polja provjeravajući je li već utvrđen smjer broda te ako je nastavit će pogađati u tom smjeru sve dok ne naiđe na promašaj ili ne uništi cijeli brod. Tada će okrenuti smjer i nastaviti pogađati sve dok brod nije u potpunosti pogođen. Međutim, ako nakon prvog pogotka smjer nije poznat, nasumično će provjeravati sva 4 smjera (ispod, iznad, lijevo, desno) oko pogođene koordinate broda sve dok

rezultat pogađanja ne bude pogodak. Ako je cijeli brod uništen, ili ako je rezultat gađanja prve koordinate promašaj u idućem potezu računalo će nasumično odabrati dostupnu koordinatu. Kako bi igra bila zanimljivija, računalo ne može promašiti brod u više od pet uzastopnih poteza. Svi rezultati pogađanja upisuju se u tablicu za praćenje uspješnosti pogađanja računala. Preko te tablice provjerava se dostupnost odabrane koordinate za gađanje. Između svakog pogađanja igrača i računala, unutar metode *'checkIfShipIsSank'* provjerava se je li cijeli trenutno gađani brod uništen. Ukoliko je rezultat te metode istinit, u pripadajućoj tablici za praćenje uspješnosti pogađanja označavaju se koordinate oko pogođenog broda. Te koordinate automatski će biti označene kao promašaj i više neće biti dostupne za pogađanje ni igraču ni računalu. Rezultat svakog poteza igrača i računala prikazuju se igraču s odgodom od 100ms. Povrat metode *'getPlayerShotInput'* su tablice prema kojima se dodjeljuju klase poljima igrača i računala i na taj način prikazuje igraču razvoj igre.

Ukoliko su igrač ili računalo uništili sve protivnikove brodove, u skočnom prozoru prikazuje se tko je pobjednik. Pritiskom na tipku 'OK' unutar skočnog prozora web aplikacija se vraća na početak.

5. ZAKLJUČAK

U ovom diplomskom radu realizirana je matematičko logička zagonetka potapanje brodova u obliku web aplikacije.

Cilj diplomskog rada bio je opisati matematičko logičku zagonetku potapanje brodova te napraviti programsku realizaciju. Prikazan je razvoj programskih jezika te su opisane tehnologije potrebne za realizaciju rada. Ukratko su opisani programski jezici JavaScript i PHP te MVC framework Laravel. Pojašnjena je sintaksa PHP-a i opisani su osnovni elementi potrebni za izradu web aplikacije u Laravelu.

Nadalje, pojašnjena je matematičko logička zagonetka "Potapanje brodova" i napravljen je kratak osvrt na razvoj i povijest same igre i načina igranja. Programska realizacija napravljena je u obliku web aplikacije u kojoj igrač i računalo međusobno jedno drugom pogađaju unaprijed postavljenu kombinaciju brodova sve dok netko ne uništi cijelu kombinaciju.

LITERATURA

[1] https://www.cs.princeton.edu/courses/archive/fall09/cos109/06langs.pdf [datum zadnje posjete stranici: 02.12.2018.]

[2] http://php.net/manual/en/ [datum zadnje posjete stranici: 02.12.2018.]

[3] http://stacktips.com/tutorials/laravel/intro-to-laravel-php-framework-and-features [datum zadnje posjete stranici: 02.12.2018.]

[4] D., Stančer, Osnove JavaScripta, Sveučilište u Zagrebu, Sveučilišni računski centar, Zagreb, 2015, dostupno na: https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c501_polaznik.pdf [datum zadnje posjete stranici: 02.12.2018.]

[5] http://boardgames.about.com/od/battleship/a/Rules-of-Battleship.htm [datum zadnje posjete stranici: 02.12.2018.]

[6] https://laravel.com/docs/5.5 [datum zadnje posjete stranici: 02.12.2018.]

[7] https://en.wikipedia.org/wiki/Battleship_(game) [datum zadnje posjete stranici: 02.12.2018.]

[8] D., Goodman, JavaScript Bible, Hungry Minds, New York, 2001, dostupno na: https://media.wiley.com/product_ancillary/28/07645334/DOWNLOAD/all.pdf?fbclid=IwAR05DGjA2IxPH3YhnN8AbXii3L05N6aOzkdmz9F8SVqXxV_pWWrnGLw2x1k [datum zadnje posjete stranici: 02.12.2018.]

[9] https://laracasts.com/ [datum zadnje posjete stranici: 02.12.2018.]

SAŽETAK

Web aplikacija za brodovlje

Cilj ovog rada je napraviti programsku realizaciju matematičko logičke zagonetke "*Battleships*" u obliku web aplikacije. Napravljen je kratak osvrt na povijest i razvoj programskih jezika. Opisane su tehnologije korištene prilikom realizacije programskog dijela rada uključujući programske jezike JavaScript i PHP te MVC framework Laravel. Predstavljeni su elementi frameworka koji se koriste u gotovo svakom projektu. Nakon toga, napravljen je kratki osvrt na povijest i način igranja igre "*Battleships*". Programski dio rada sadrži opis načina izrade web aplikacije popraćen komentarima i slikama izvedbe. Igrač i računalo igraju međusobno jedno protiv drugog. Prije samog početka igre, igrač postavlja kombinaciju brodova različite dužine na svoje polje. Kada je to uspješno napravio, pritiskom na tipku 'Start' započinje se igra. Računalo u tom trenutku generira svoj nasumičan uzorak brodova. I igrač i računalo na raspolaganju imaju jednak broj brodova. Igrač je prvi na potezu. Klikom na polje računala označava mjesto na kojem misli da se nalazi dio protivnikovog broda. Rezultat pogađanja bilježi se u tablicu i prikazuje igraču na mjestu koje je označio. Zatim, računalo generira svoje koordinate i ispituje nalazi li se na tom mjestu na igračevom polju dio broda te prikazuje rezultat pogađanja igraču na njegovoj tablici. Računalo i igrač igraju naizmjenično sve dok jedno od njih ne pogodi cijelu kombinaciju protivnikovih brodova. Strana kojoj to prvoj uspije pobijedila je u igri. Programska realizacija matematičko logičke zagonetke u obliku web aplikacije uspješno je realizirana.

Ključne riječi: PHP, JavaScript, Laravel, potapanje brodova, kombinacija, web aplikacija

ABSTRACT

Web application for logic puzzle 'Battleships'

This papers aim was to develop a program implementation of logic puzzle 'Battleships' in form of web application. Paper starts with brief summary of programming languages history. It is followed by basic information about JavaScript, PHP and MVC framework Laravel. Examples of basic elements used in almost every Laravel project are shown. Summary of programming language PHP and MVC framework Laravel is followed by review of history and rules of logic puzzle 'Battleships'. In programming chapter of paper, web application development is described along with pictures of finished product. The game takes place between player and computer. The player sets his battleship combination, while computer generates random pattern of battleships. Both, player and computer use the same number of battleships. Player and computer alternately guess each others battleship coordinates. The winner is the one who guesses complete opponents pattern first. Using accuired knowledge, program implementation of logic puzzle in form of web application is succesfully made.

Keywords: PHP, JavaScript, Laravel, battleship, pattern, code, web application

ŽIVOTOPIS

Filip Dumančić je rođen je 7. lipnja 1992. godine u gradu Đakovu, Hrvatska. Dolazi iz Semeljaca gdje je završio Osnovnu školu Josipa Kozarca. Paralelno s osnovnom školom upisuje se u školu stranih jezika Memoria u Đakovu gdje završava dva stupnja njemačkog jezika. Nakon završene osnovne škole upisuje Gimnaziju A.G. Matoša u Đakovu gdje završava opći smjer. Po završetku srednje škole upisuje se na Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva s kojeg se ispisuje nakon 2 godine. Godine 2013., kao redovan student, upisuje se na Sveučilište Josipa Jurja Strossmayera u Osijeku Elektrotehnički fakultet gdje 2016. godine završava preddiplomski i upisuje diplomski studij. Tijekom školovanja stekao je znanje engleskog jezika. Od 2011. godine posjeduje vozačku dozvolu B kategorije.

potpis: _____