

Univerzitet u Zenici

Politehnički fakultet

Softversko inženjeristvo

Operativni sistemi i računarstva u oblaku (OSiRuO)

Projekat iz Operativnih sistema i računarstva u oblaku

**Postavljanje Snake igrice pomoću docker + render**

Članovi tima:

Emir Genjac

Emin Ahbabović

Anes Ćurić

Faris Šolbić

Zenica 2025.

**Sadržaj**

[1. Uvod 3](#_Toc199522963)

[2. Koraci razvoja 3](#_Toc199522964)

[2.2. Kreiranje aplikacije 3](#_Toc199522965)

[2.3. Dockerizacija aplikacije 5](#_Toc199522966)

[2.4. Lokalno testiranje 5](#_Toc199522967)

[2.5. Postavljanje na GitHub 7](#_Toc199522968)

[2.6. Deploy na Render platformi 8](#_Toc199522969)

[2.7. Pristup igri 9](#_Toc199522970)

[3. Kod aplikacije 10](#_Toc199522971)

[4. Zaključak i refleksija 16](#_Toc199522972)

# Uvod

Ova dokumentacija opisuje proces izrade interaktivne web aplikacije – klasične igre Snake, razvijene kao dio projekta iz predmeta Operativni sistemi i računarstvo u oblaku (OSiRuO). Cilj projekta bio je spojiti frontend razvoj sa savremenim alatima za kontejnerizaciju i cloud hosting.

Aplikacija je izrađena korištenjem web tehnologija: HTML, CSS i JavaScript, kontejnerizirana uz pomoć Dockera, objavljena na GitHub platformi, te postavljena online korištenjem Render servisa za cloud hosting.

**Link ka aplikaciji:** https://snake-game-rch4.onrender.com

**GitHub repozitorij:** https://github.com/emirgenjac/snake-game

# Koraci razvoja

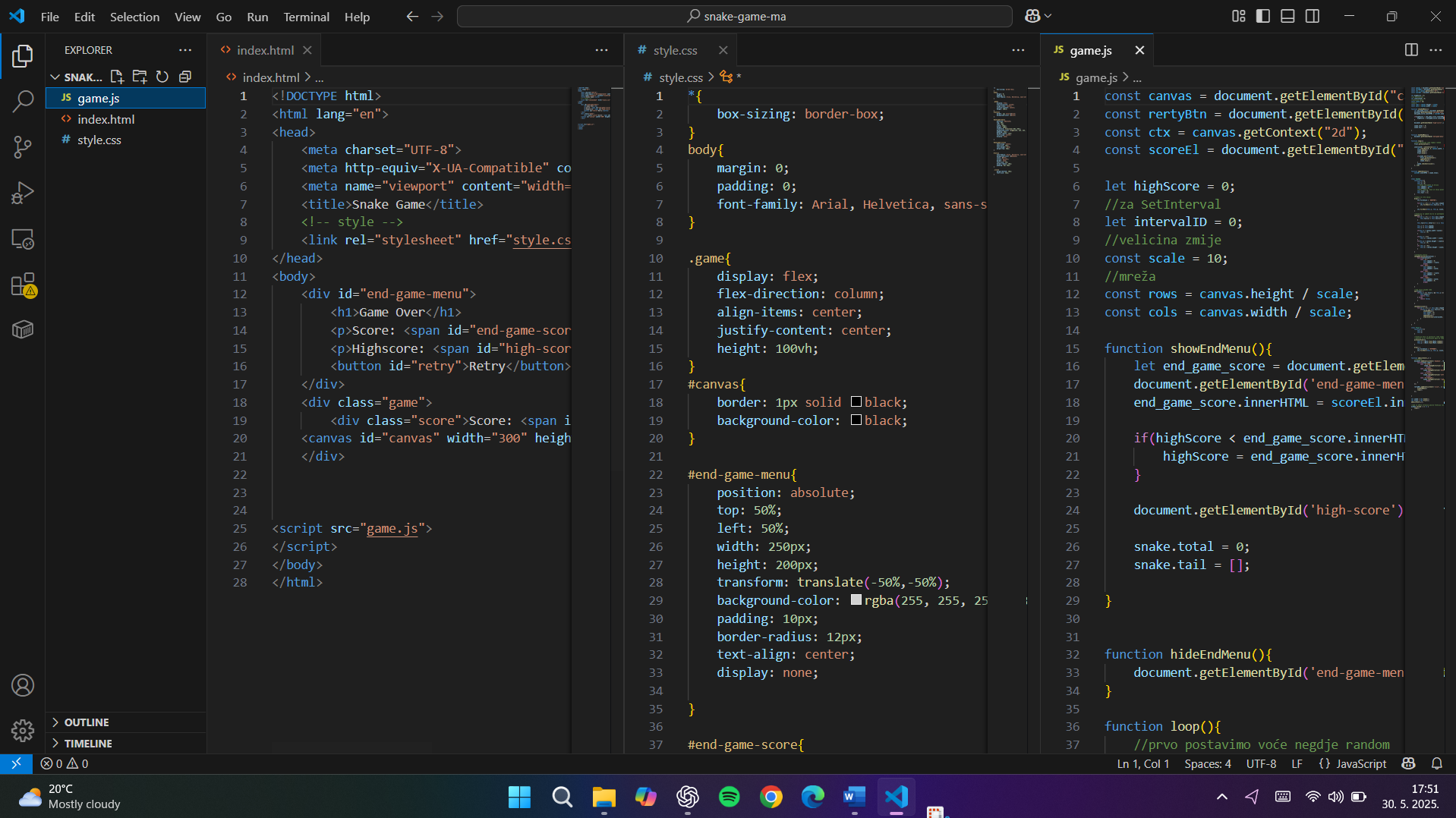
* 1. **Odabir cloud platforme**

Izabrana je Render platforma zbog svoje jednostavne integracije sa GitHub-om i podrške za automatski deploy pomoću Dockerfile-a.

## Kreiranje aplikacije

Napravljen je projekat u folderu snake-game-main sa sljedećim datotekama:

* game.html – Sadrži <canvas> element za prikaz igre
* style.css – Stilovi za igru (centralno poravnanje, dimenzije, boje)
* game.js – Glavna logika igre: kretanje zmije, poeni, kolizije, restart



Slika 1 – Struktura igrice u VS Code-u

## Dockerizacija aplikacije

A computer screen with text

AI-generated content may be incorrect.

Slika 2 - Dockerfile

A computer screen with text and images

AI-generated content may be incorrect.

Slika 3 - nginx.conf

A screenshot of a computer program

AI-generated content may be incorrect.

Slika 4 – bash skripta za pokretanje kontejnera

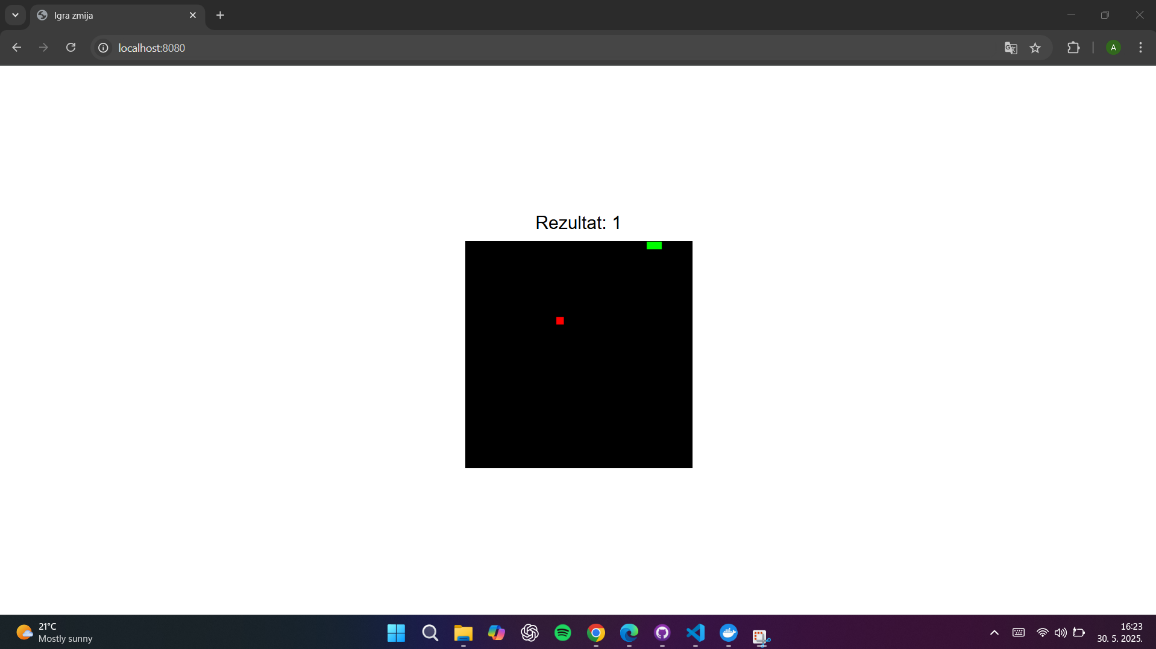
## Lokalno testiranje

Testiranje je izvršeno lokalno pomoću Docker komandi:

*docker build -t snake-game .*

*docker run -d -p 8080:80 snake-game*

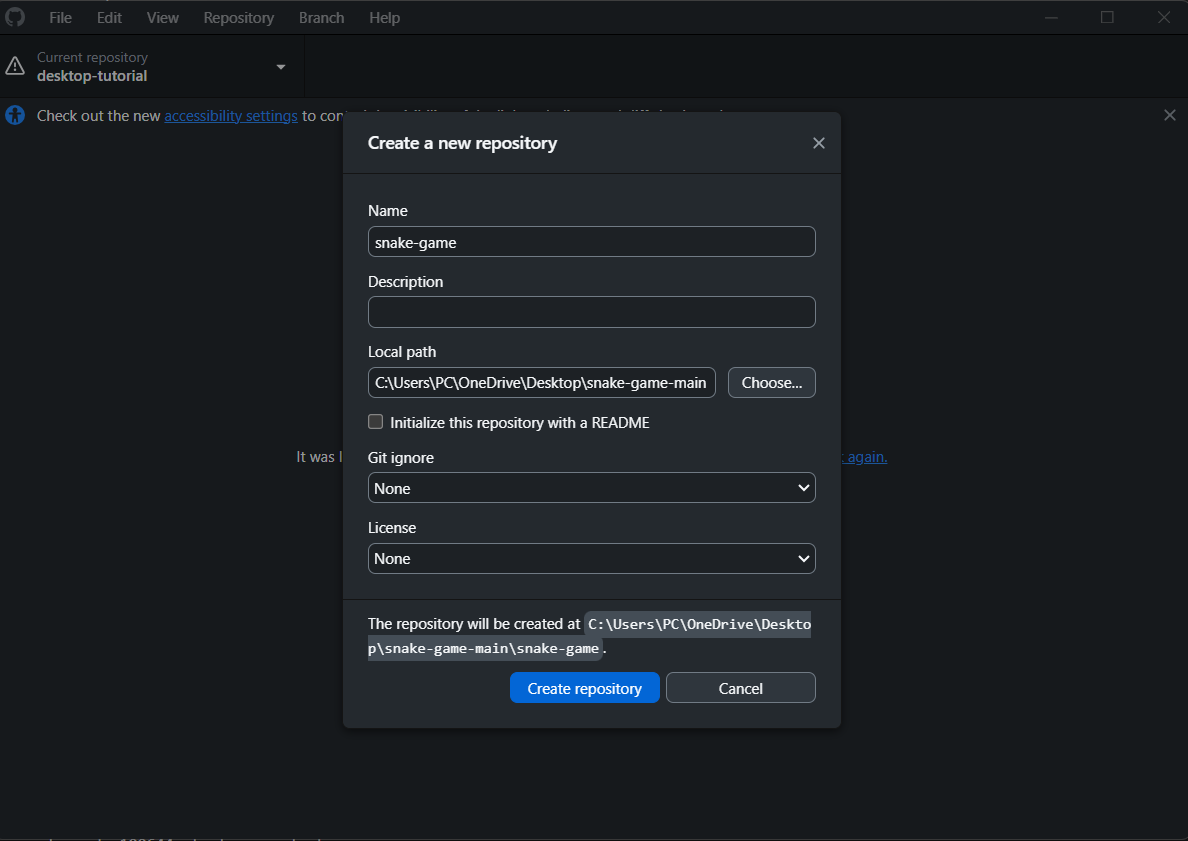
**Pristup aplikaciji**: <http://localhost:8080>



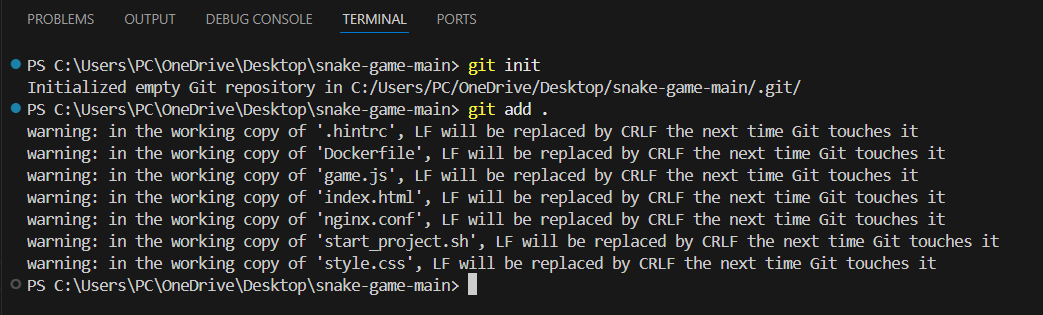
Slika 5 - Prikaz pokrenute Snake igre u lokalnom browseru na adresi localhost:8080

## 

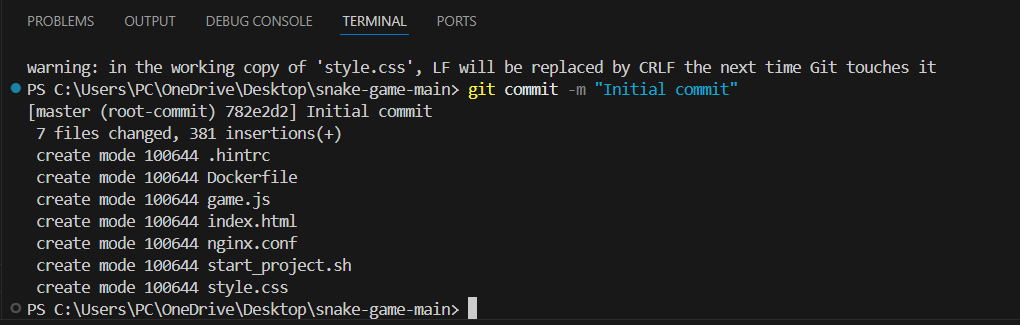
## Postavljanje na GitHub



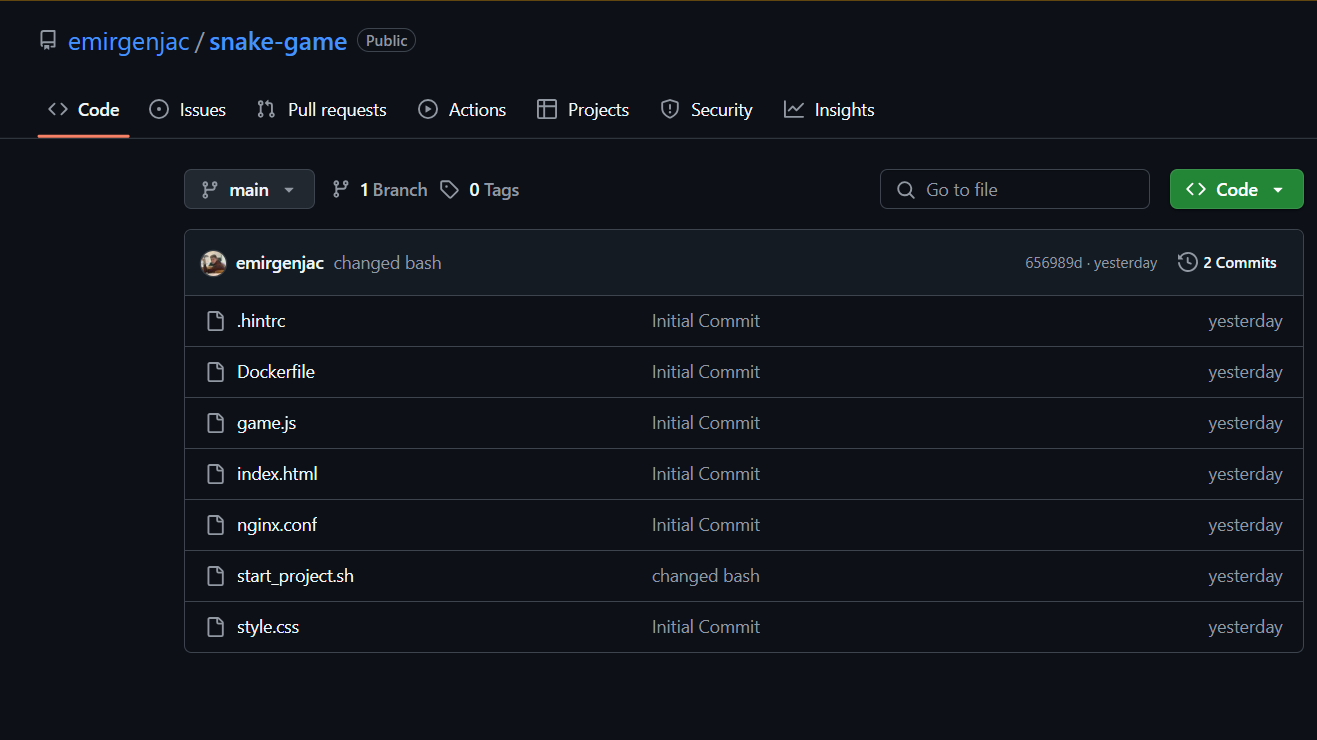
Slika 6 - Kreiranje repozitorija na GitHub-u



Slika 7 - Inicializacija repozitorija

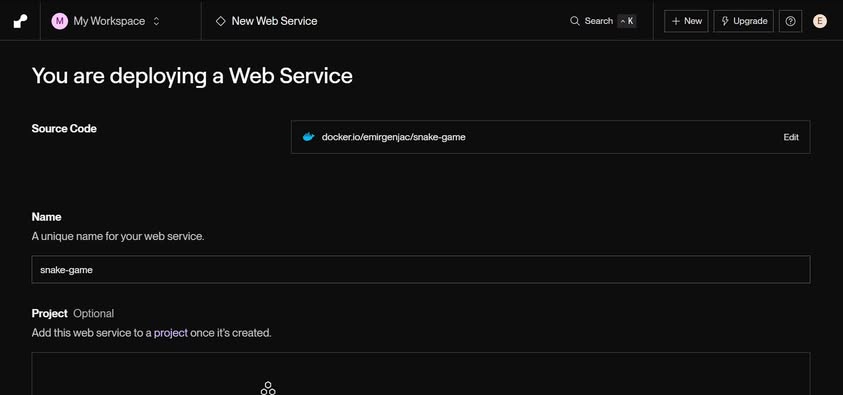


Slika 8 - Commit i verzionisanje

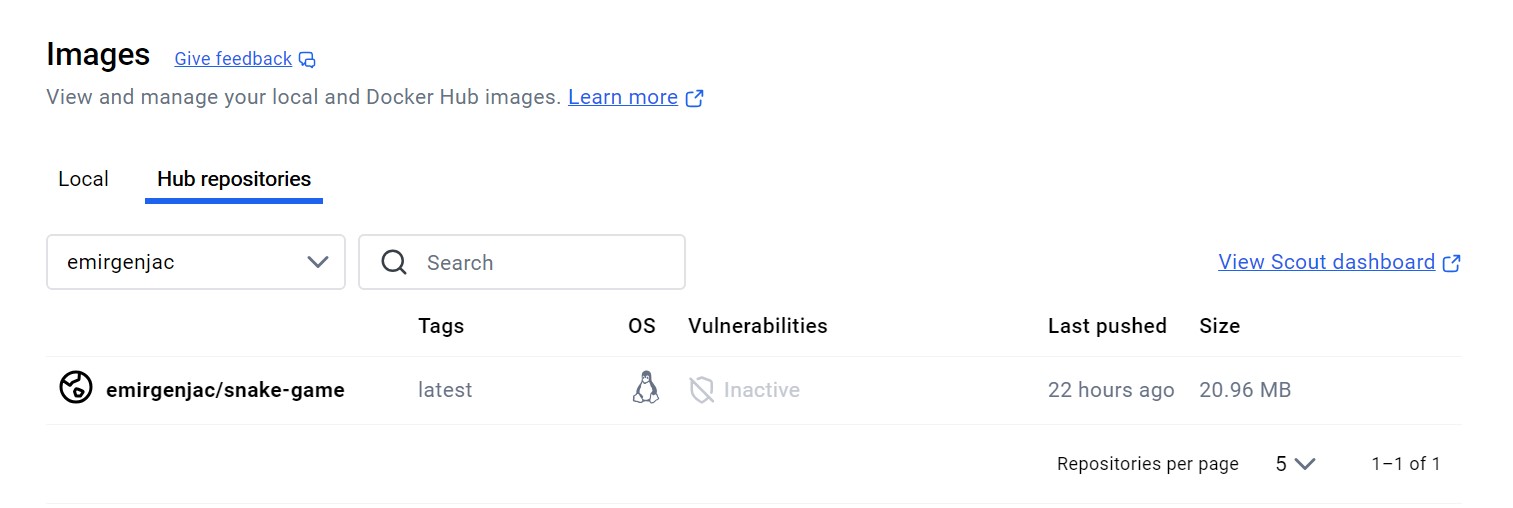


Slika 9 - Prikaz repozitorija

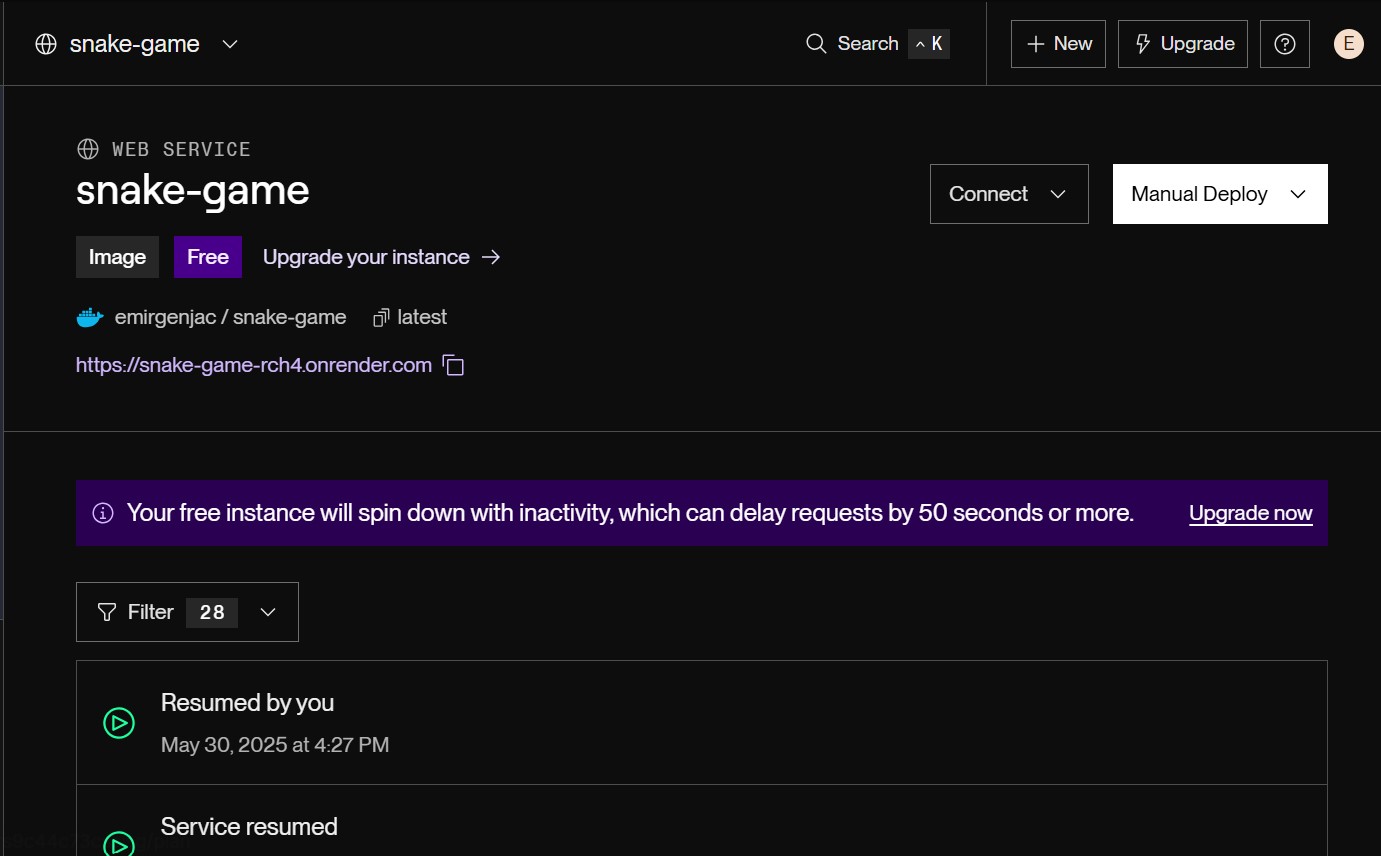
## Deploy na Render platformi



Slika 10 - Deployanje Web Servisa



Slika 11 – Docker hub repositorij



Slika 12 - Pregled i Deployanje Servisa

## Pristup igri

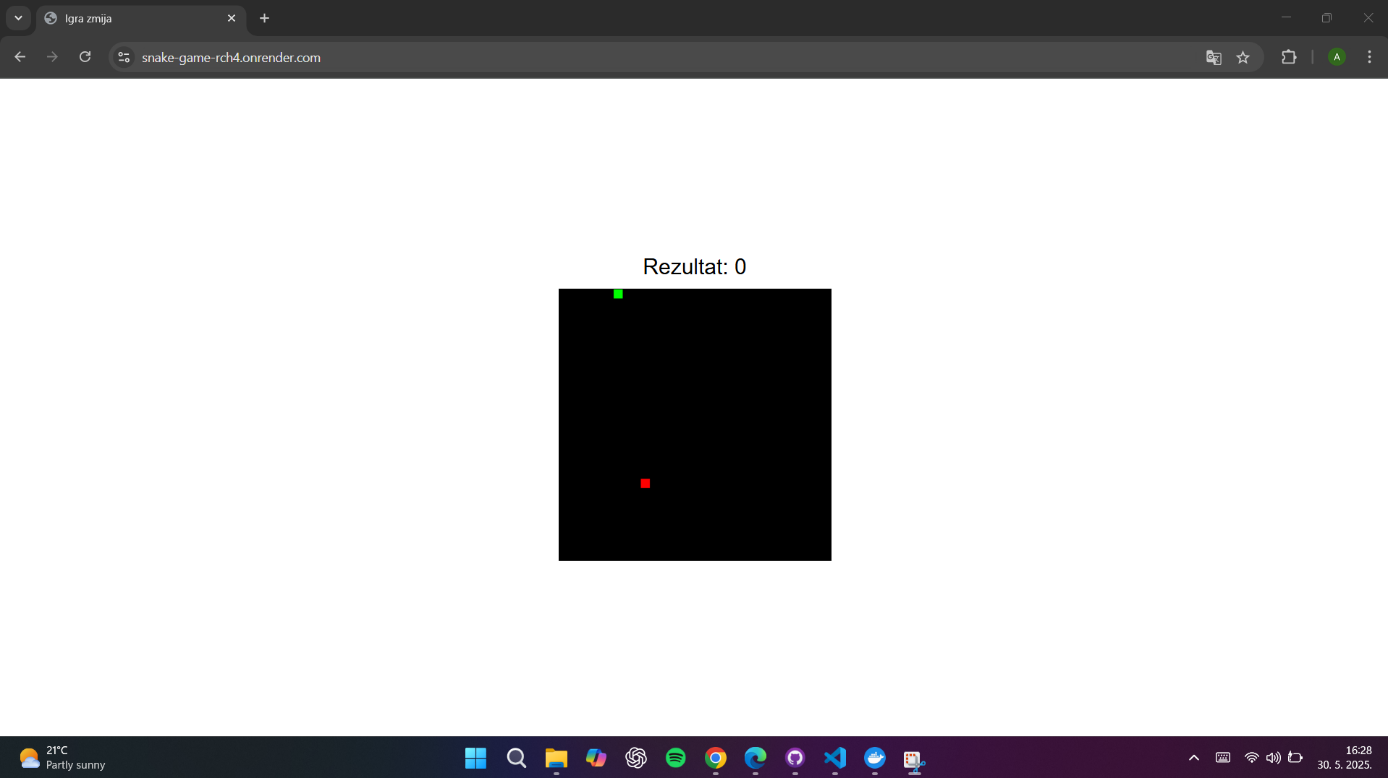
Aplikacija je javna i ne zahtijeva autentikaciju. Pristup je moguć bez logina putem linka:

<https://snake-game-rch4.onrender.com>

A screenshot of a computer

AI-generated content may be incorrect.

Slika 13 – Root path



Slika 14 – Online pristup

# Kod aplikacije

index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Snake Game</title>

    <!-- style -->

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <div id="end-game-menu">

        <h1>Game Over</h1>

        <p>Score: <span id="end-game-score"></span></p>

        <p>Highscore: <span id="high-score"></span></p>

        <button id="retry">Retry</button>

    </div>

    <div class="game">

        <div class="score">Score: <span id="score">0</span></div>

    <canvas id="canvas" width="300" height="300"></canvas>

    </div>

<script src="game.js">

</script>

</body>

</html>

style.css

\*{

    box-sizing: border-box;

}

body{

    margin: 0;

    padding: 0;

    font-family: Arial, Helvetica, sans-serif;

}

.game{

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    height: 100vh;

}

#canvas{

    border: 1px solid black;

    background-color: black;

}

#end-game-menu{

    position: absolute;

    top: 50%;

    left: 50%;

    width: 250px;

    height: 200px;

    transform: translate(-50%,-50%);

    background-color: rgba(255, 255, 255, 0.8);

    padding: 10px;

    border-radius: 12px;

    text-align: center;

    display: none;

}

#end-game-score{

    text-align: center;

    font-size: 16px;

    font-weight: bold;

    margin-top: 20px;

}

#retry{

    font-family: Arial, Helvetica, sans-serif;

    background-color: #2e2e2e;

    color: #fff;

    cursor: pointer;

    border: none;

    font-size: 1em;

    border-radius: 10px;

    padding: 6px 20px;

}

.score{

    margin-bottom: 10px;

    font-size: 24px;

}

game.js

const canvas = document.getElementById("canvas");

const rertyBtn = document.getElementById("retry");

const ctx = canvas.getContext("2d");

const scoreEl = document.getElementById("score");

let highScore = 0;

//za SetInterval

let intervalID = 0;

//velicina zmije

const scale = 10;

//mreža

const rows = canvas.height / scale;

const cols = canvas.width / scale;

function showEndMenu(){

    let end\_game\_score = document.getElementById('end-game-score');

    document.getElementById('end-game-menu').style.display = 'block';

    end\_game\_score.innerHTML = scoreEl.innerHTML;

    if(highScore < end\_game\_score.innerHTML){

        highScore = end\_game\_score.innerHTML;

    }

    document.getElementById('high-score').innerHTML = highScore;

    snake.total = 0;

    snake.tail = [];

}

function hideEndMenu(){

    document.getElementById('end-game-menu').style.display = 'none';

}

function loop(){

    //prvo postavimo voće negdje random

    fruit.pickLocation();

    intervalID = setInterval(()=> {

        ctx.clearRect(0, 0, canvas.width, canvas.height);

        snake.update()

        snake.draw();

        fruit.draw();

        if(snake.eat(fruit)){

            fruit.pickLocation();

            updateScore();

            snake.draw();

        }

        snake.checkCollision();

    }, 120)

}

function updateScore(){

    scoreEl.innerHTML = snake.total;

}

class Snake{

    constructor(){

        this.x = 0;

        this.y = 0;

        // dvije varijable za brzinu

        this.xSpeed = scale;

        this.ySpeed = 0;

        //broj voćkica  koje je zmija pojela

        this.total = 0;

        this.tail = [];

    }

    //ovdje se crta zmija:

    draw() {

        ctx.fillStyle = '#00FF00';

        for(let i = 0; i < this.tail.length; i++){

            ctx.fillRect(this.tail[i].x, this.tail[i].y, scale, scale);

        }

        ctx.fillRect(this.x, this.y, scale, scale);

    }

    //funkcija za update da bi se postepeno mijenjala pozicija zmije

    update() {

        for(let i = 0; i < this.tail.length - 1; i++){

            this.tail[i] = this.tail[i+1];

        }

        this.tail[this.total-1] = { x: this.x, y: this.y}

        this.x += this.xSpeed;

        this.y += this.ySpeed;

        if(this.x > canvas.width -scale){

            this.x = 0;

        }

        if(this.x < 0){

            this.x = canvas.width - scale;

        }

        if(this.y > canvas.height - scale){

            this.y = 0;

        }

        if( this.y < 0){

            this.y = canvas.height - scale;

        }

    }

    //promjena pravca

    changeDirection(direction) {

        switch(direction){

            case 'up':

                this.xSpeed = 0;

                this.ySpeed = -scale;

                break;

            case 'down':

                this.xSpeed = 0;

                this.ySpeed = scale;

                break;

            case 'left':

                this.xSpeed = -scale;

                this.ySpeed = 0;

                break;

            case 'right':

                this.xSpeed = scale;

                this.ySpeed = 0;

                break;

        }

    }

    //kad zmija pojede voće

    eat(fruit) {

        if(this.x === fruit.x && this.y === fruit.y){

            this.total++;

            return true;

        } else{

            return false;

        }

    }

    checkCollision(){

        for(let i = 0; i < this.tail.length; i++){

            if(this.x === this.tail[i].x && this.y === this.tail[i].y){

                showEndMenu();

                this.total = 0;

                this.tail = [];

                updateScore();

                clearInterval(intervalID);

            }

        }

    }

}

class Fruit {

    constructor(){

        this.x;

        this.y;

    }

    //funkcija koja ce postaviti voće negdje random, zato je gore this.x i this.y

    //odnosno (x,y) koordinate gdje će voće biti postavljeno

    pickLocation(){

        this.x = (Math.floor(Math.random() \* cols)) \* scale;

        this.y = (Math.floor(Math.random() \* rows)) \* scale

    }

    draw() {

        ctx.fillStyle = "#FF0000";

        ctx.fillRect(this.x, this.y, scale, scale)

    }

}

function addListeners\_() {

    //kretanje

    document.addEventListener('keydown', (event) => {

        switch(event.code){

            case 'ArrowUp':

                 snake.changeDirection('up');

                    break;

            case 'ArrowDown':

                snake.changeDirection('down');

                    break;

            case 'ArrowLeft':

                snake.changeDirection('left');

                    break;

            case 'ArrowRight':

                snake.changeDirection('right');

                    break;

        }

    });

    rertyBtn.addEventListener('click', () => {

        hideEndMenu()

        loop();

    });

}

let snake = new Snake();

let fruit = new Fruit();

addListeners\_();

//kad se učita stranica poziva funkciju loop()

window.onload = () => {

    loop();

}

# Zaključak i refleksija

Tokom izrade ovog projekta savladali smo niz važnih i korisnih vještina:

* Kreirali smo interaktivnu web igru koristeći HTML5 <canvas> element
* Upravljali smo logikom igre putem JavaScript game loop-a i događaja
* Dockerizirali smo aplikaciju i napisali vlastiti Dockerfile
* Konfigurisali smo jednostavni Nginx server unutar Docker containera sa rutiranjem
* Koristili smo Git i GitHub za verzionisanje i postavljanje koda
* Deployali smo aplikaciju na cloud koristeći Render platformu
* Stekli smo osnovno razumijevanje DevOps principa u razvoju softvera

Ovaj projekat nas je naučio kako povezati frontend razvoj sa konceptima kontejnerizacije i cloud hostinga. Po prvi put smo samostalno kontejnerizirali vlastitu aplikaciju i uspješno je postavili na internet koristeći GitHub i Render. Kroz rad sa Linux komandnom linijom, Docker-om i konfigurisanjem cloud servisa, postali smo sigurniji u korištenje savremenih alata koji se danas koriste u profesionalnom razvoju softvera. Iskustvo nam je pomoglo da shvatimo kompletan tok rada – od pisanja koda do objave funkcionalne aplikacije dostupne online.