

COIN HUNTER



COIN HUNTER

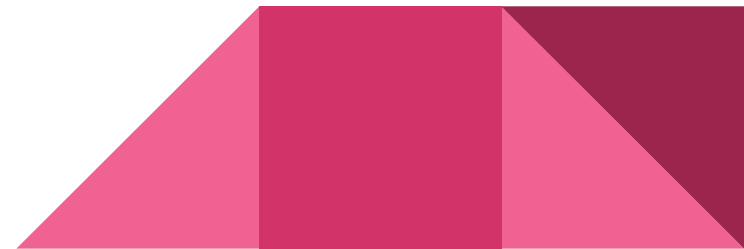


przeglądarkowa aplikacja z grą typu Endless Runner oraz warstwą społecznościową

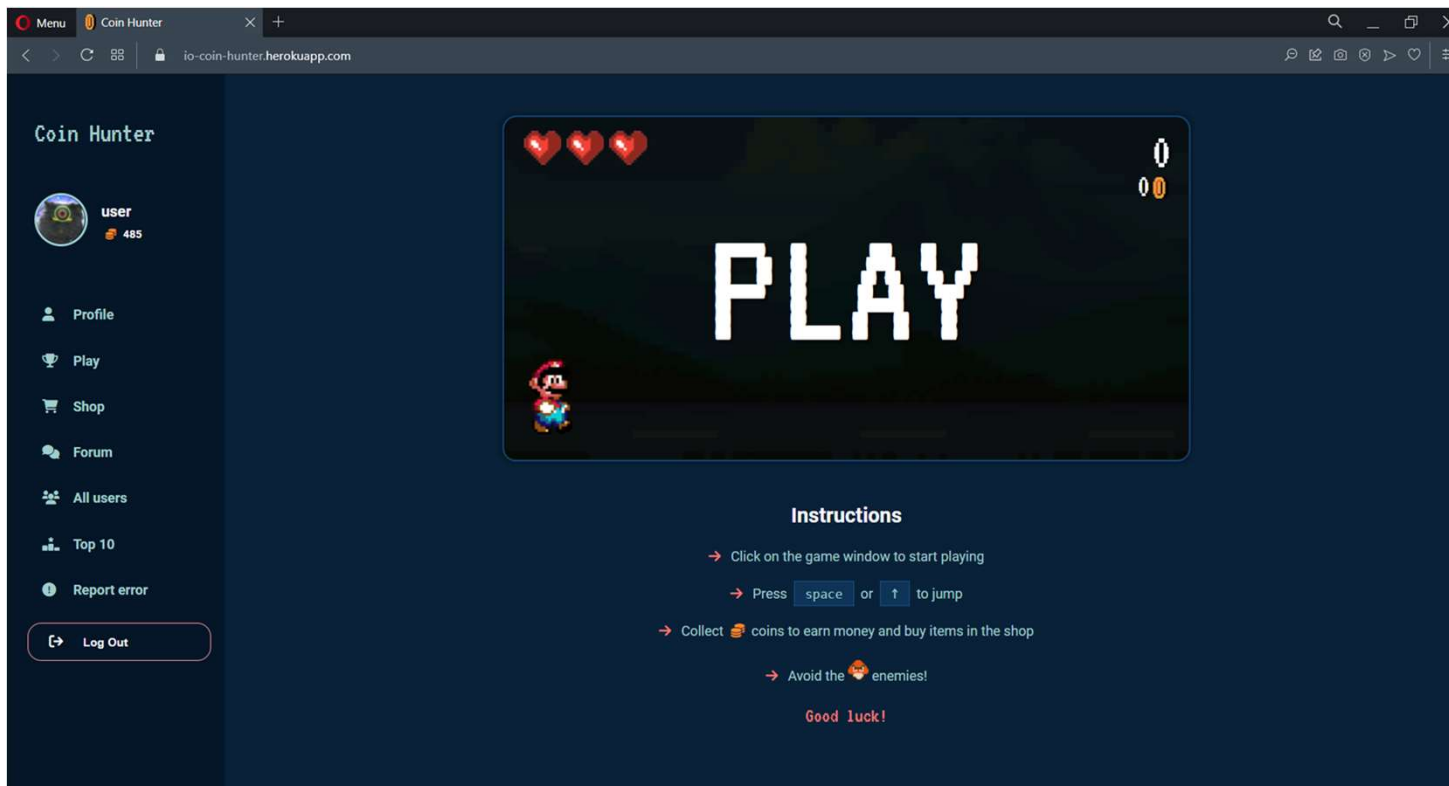
stworzona w ramach projektu z przedmiotu Inżynieria Oprogramowania

Twórcy:

- 01 Baran Dawid
- 02 Oberda Mikołaj
- 03 Pająk Julia
- 04 Piękosz Tomasz
- 05 Przybyła Agata
- 06 Wnuk Aleksandra

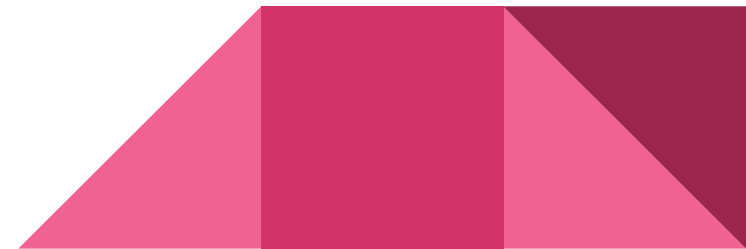


0 aplikacji

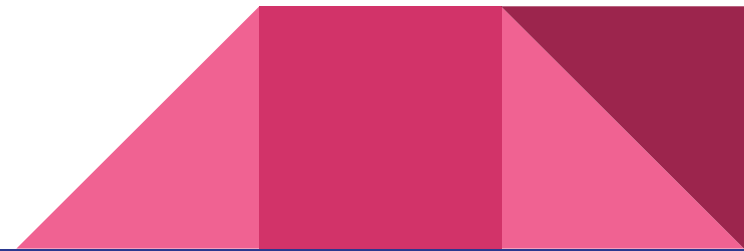


O aplikacji

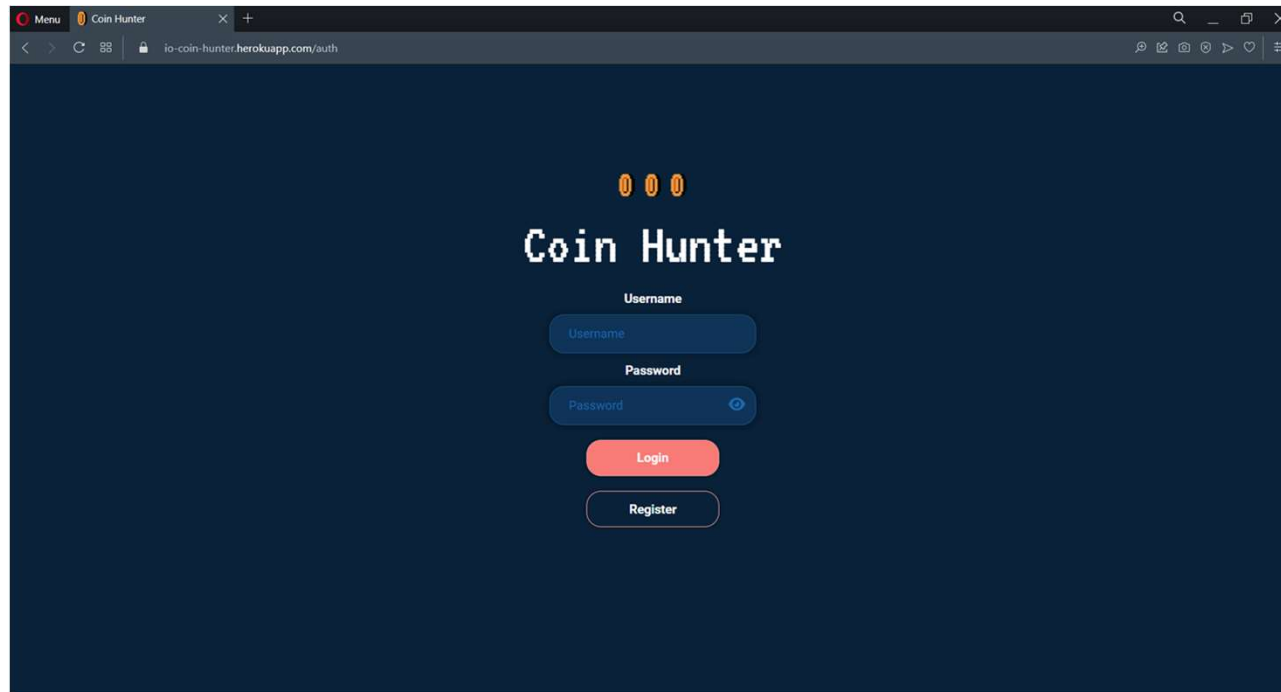
- 0 gra przeglądarkowa z warstwą społecznościową
- 0 typ gry: endless runner
- 0 warstwa społecznościowa
 - 0 profile użytkowników, dostępne po zalogowaniu (login i hasło)
 - 0 forum (możliwość publikowania postów)
 - 0 ranking
 - 0 sklep z awatarami
 - 0 katalog użytkowników
- 0 możliwość raportowania wykrytych błędów



Omówienie wymagań funkcjonalnych projektu



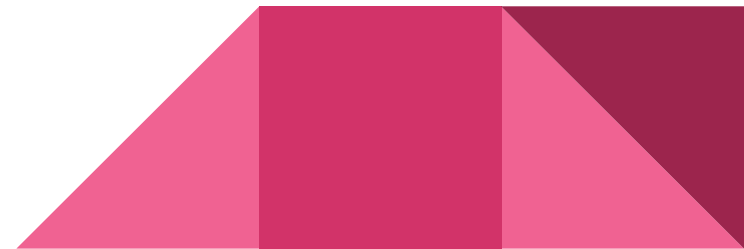
Gra wraz z warstwą społecznościową jest aplikacją przeglądarkową



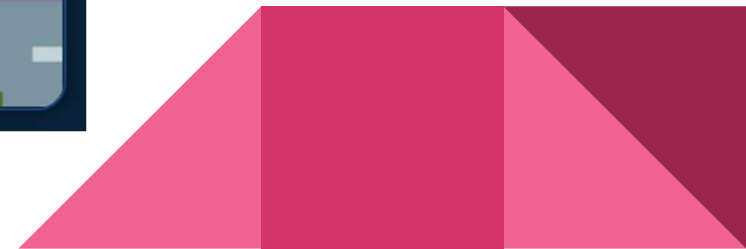
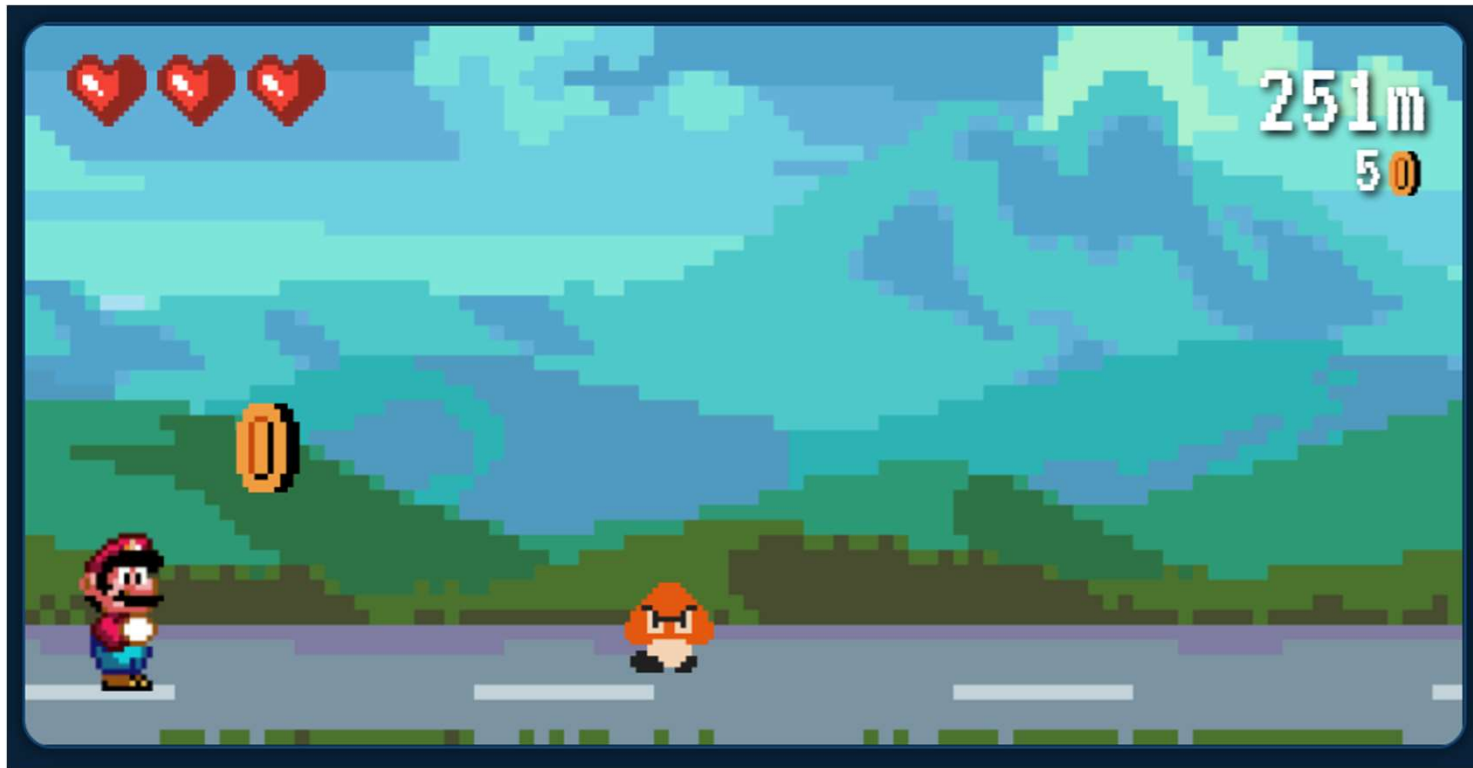
Dostęp możliwy jest po rejestracji i zalogowaniu, przy rejestracji wymagane są tylko login i hasło - demo

nazwa użytkownika ograniczona została do 16 znaków
hasło może być dowolnej długości
można używać wszystkich znaków UNICODE

hasła użytkowników szyfrowane są przy użyciu funkcji bcrypt



Grafika gry - pixel art



Pixel art na stronie

000

Coin Hunter

→ Collect 🪙 coins to earn money and buy items in the shop

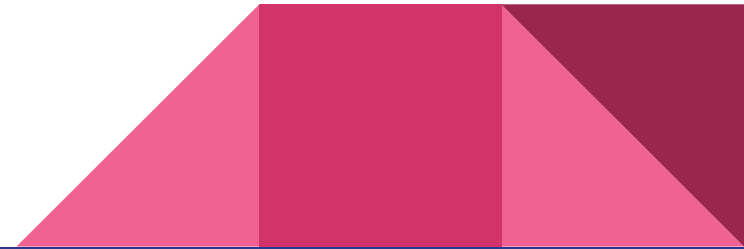
→ Avoid the 🧛 enemies!

Good luck!

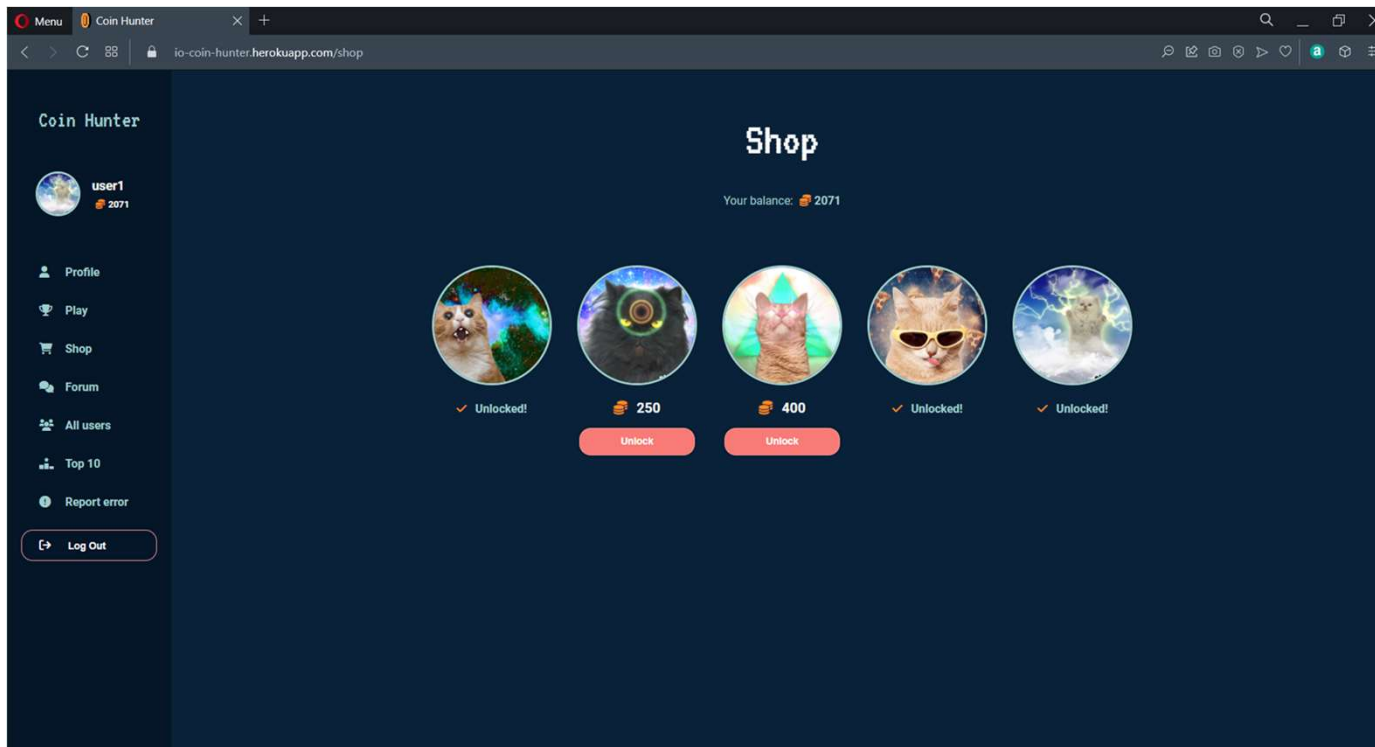
Shop

Your balance: 🪙 485

Gra jest typu Endless Runner, cel: najdłuższy dystans
- demo

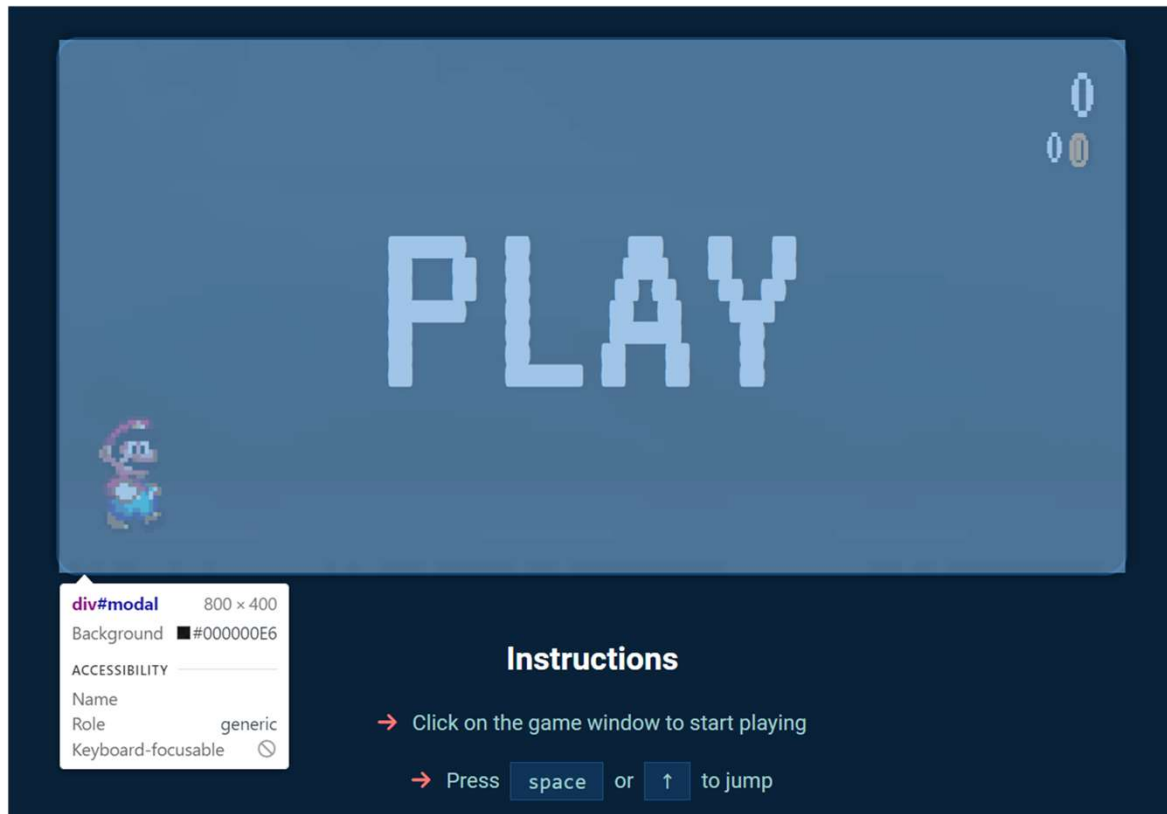


Do zdobycia w grze są monety, które można wymienić na grafiki - awatary (skins) w grze



Aktualnie dostępnych jest 5 grafik, które można odblokowywać poprzez zakup ich w zakładce *Shop* za uzbierane w grze monety. Odblokowane grafiki można wykorzystywać jako awatary na forum

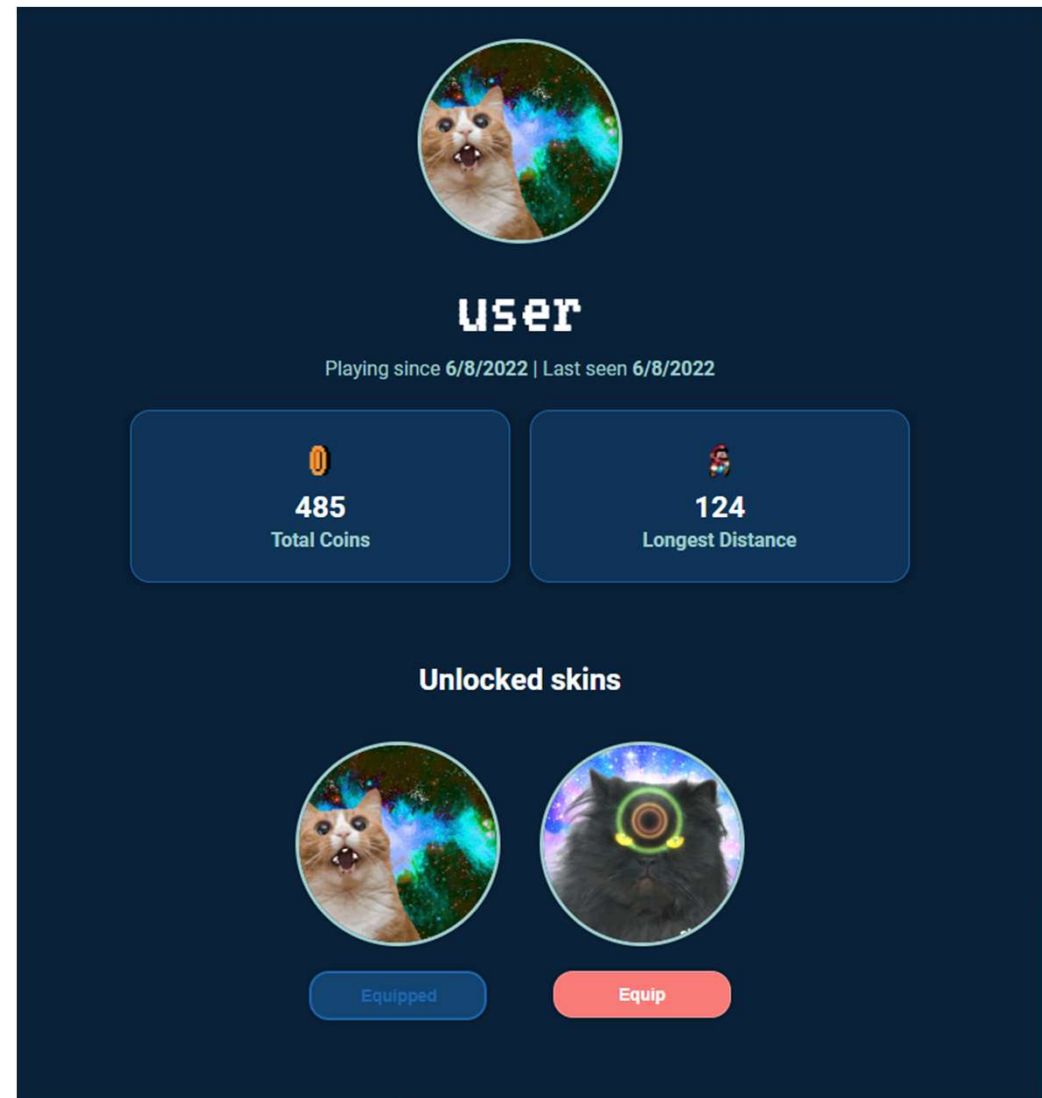
Rozmiar okna gry: 800 x 400 px



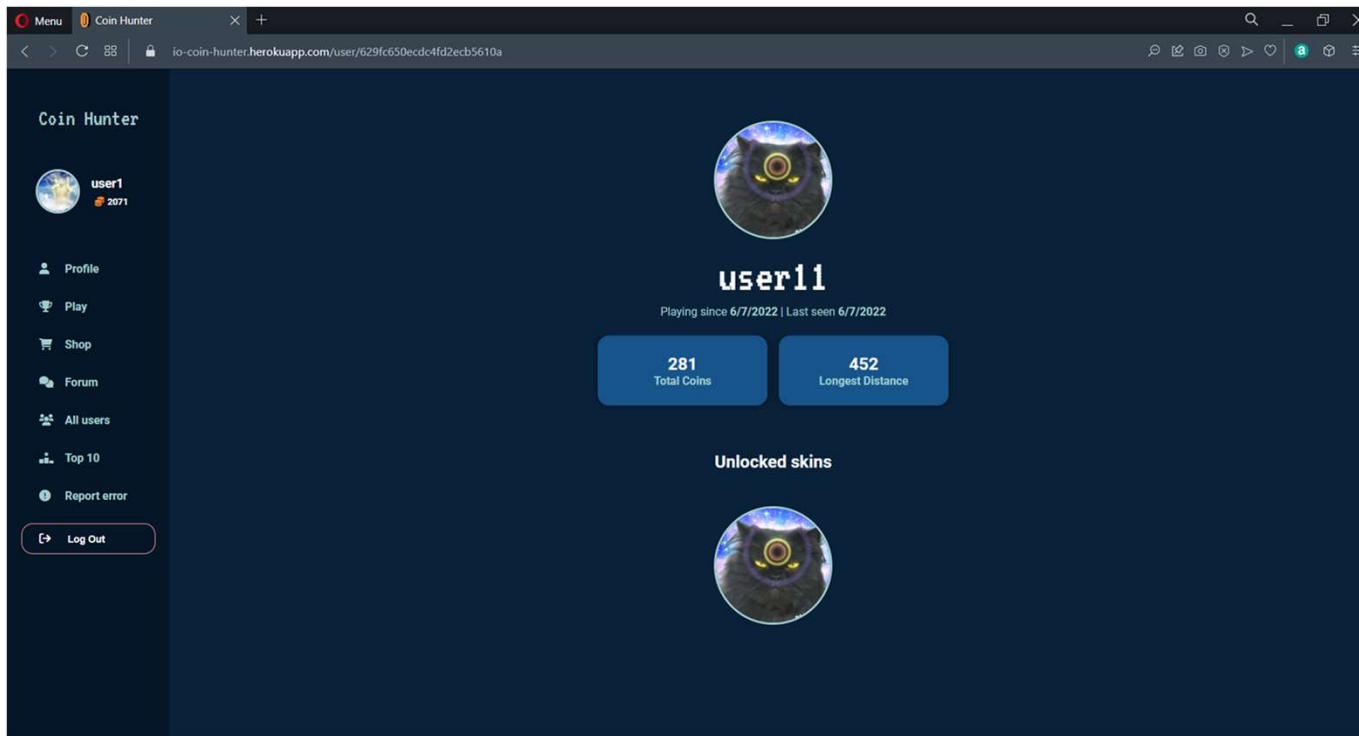
```
.game-container {  
  width: 800px;  
  height: 400px;  
  /* ... */  
}
```

W warstwie społecznościowej każdy ma swój profil, a na nim: rekord (*longest distance*), liczbę monet (*coins*), awatar (*skin*)

dodatkowo każdy użytkownik na swoim profilu ma dostęp do wszystkich odblokowanych przez siebie awatarów oraz może decydować o tym, którego awatara aktualnie używa

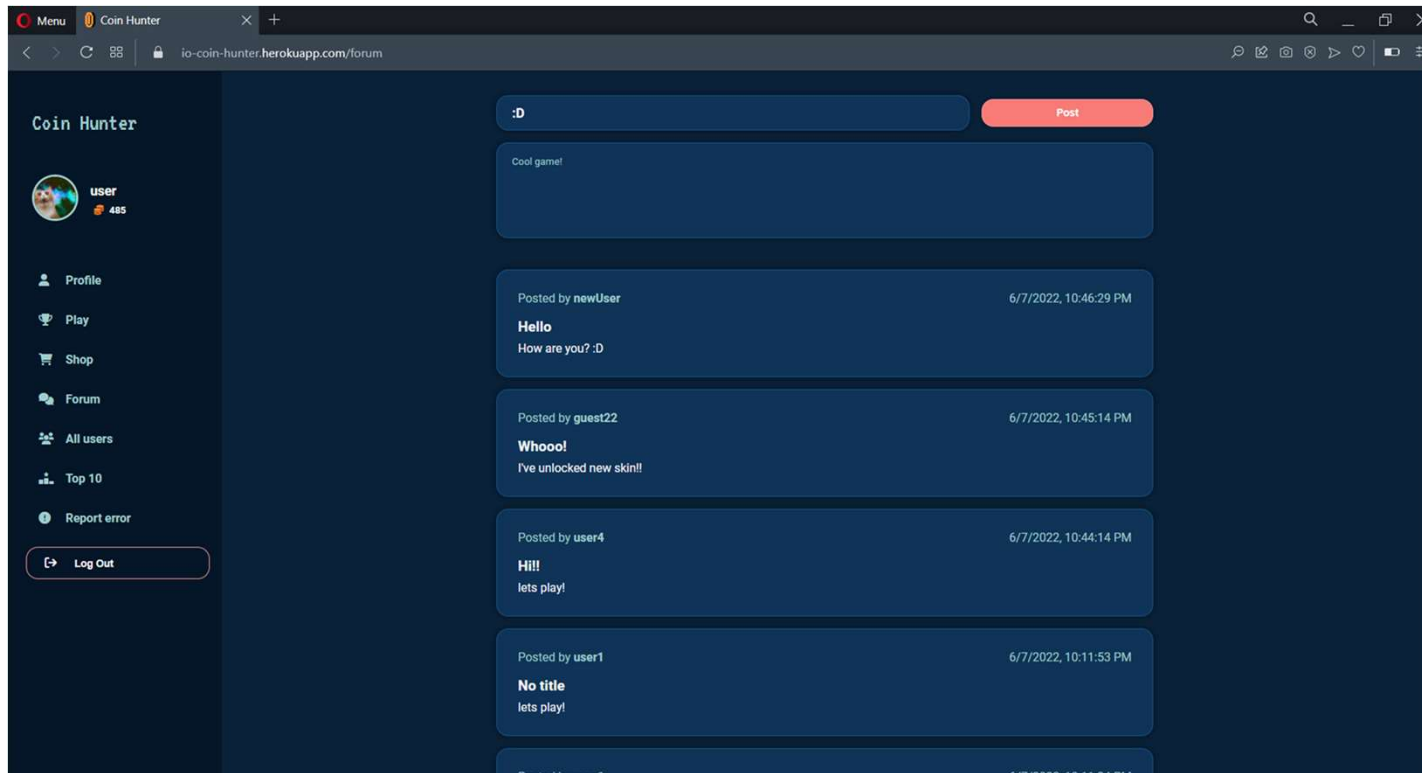


Każdy ma możliwość odwiedzenia profilu innego gracza



będąc na profilu innego gracza mamy wgląd m. in. w jego wyniki w grze (liczbę monet i maksymalny przebiegnięty dystans) oraz kolekcję odblokowanych przez niego awatarów

Każdy będzie mógł pisać posty na swoim profilu



posty umieszczać można w zakładce forum
w skład postu wchodzi tytuł (opcjonalny), oraz treść (możliwe jest umieszczanie textart)

każdy post opatrzony jest datą i godzinę publikacji oraz loginem autora, który po kliknięciu przekierowuje na jego profil

Prowadzony jest ranking najlepszych dziesięciu graczy

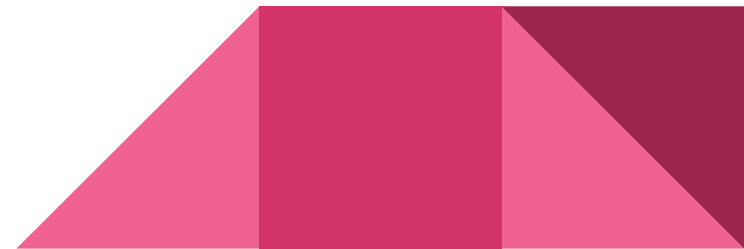
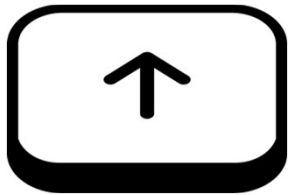
Top 10

| Coins | | Distance | | | |
|-------|---|---|-----|---|--------|
| 1. |  user25 |  6368 | 1. |  dawid | 1316 m |
| 2. |  guest22 |  6082 | 2. |  test2 | 1170 m |
| 3. |  user5 |  5825 | 3. |  miko | 1117 m |
| 4. |  user12 |  5234 | 4. |  user25 | 953 m |
| 5. |  user3 |  3492 | 5. |  agata | 870 m |
| 6. |  guest24 |  424 | 6. |  1 | 855 m |
| 7. |  user11 |  281 | 7. |  user12 | 753 m |
| 8. |  user2 |  171 | 8. |  guest24 | 732 m |
| 9. |  user4 |  53 | 9. |  user31 | 631 m |
| 10. |  user31 |  52 | 10. |  user11 | 452 m |

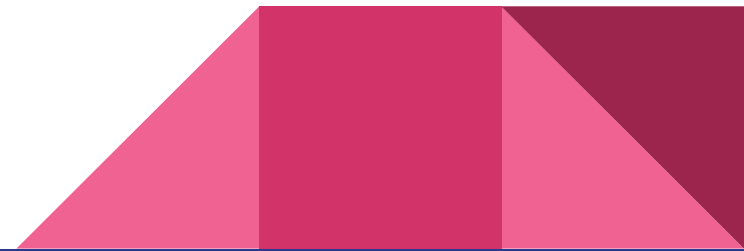
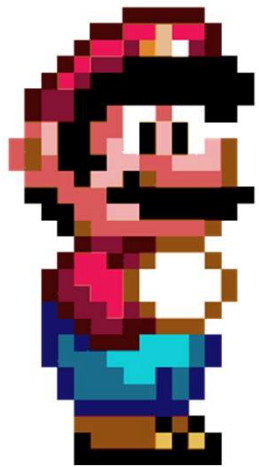
prowadzone są dwa rankingi:

- po ilości aktualnie zgromadzonych monet
- po najdłuższym przebiegniętym jednorazowo dystansie

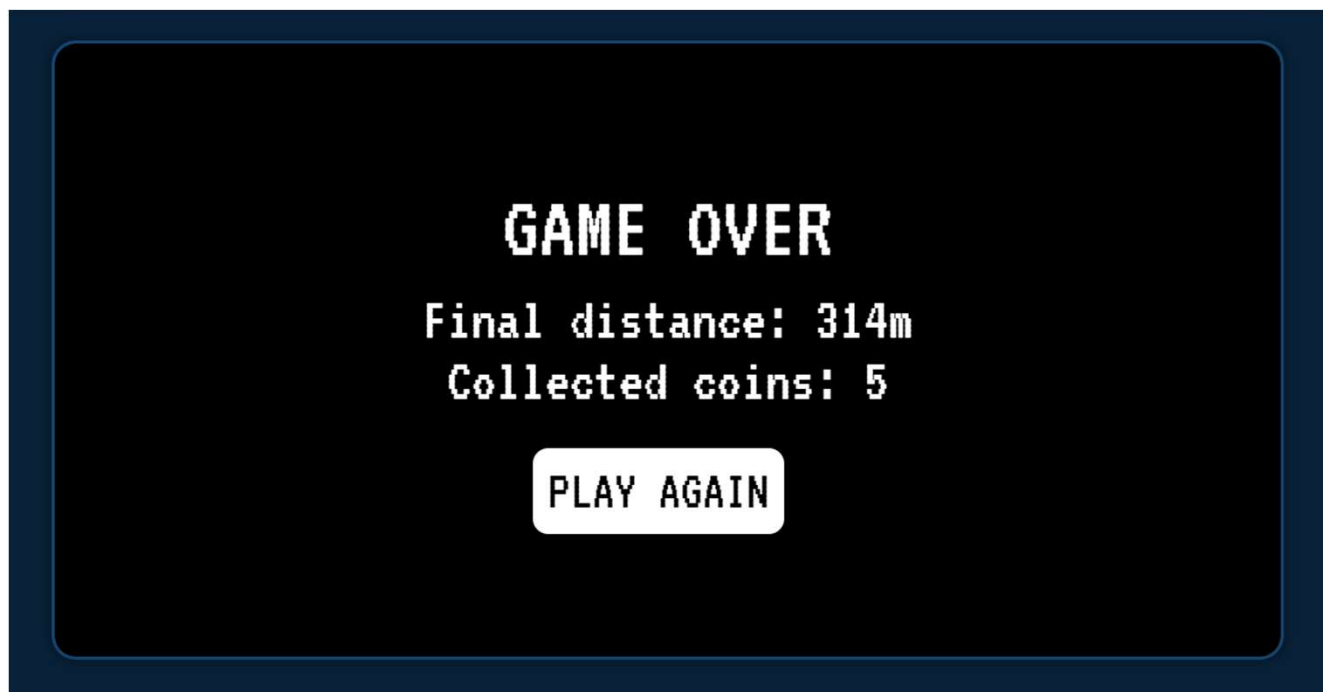
Sterowanie grą za pomocą strzałki w górę i spacji - demo



Gra przyspiesza z czasem - demo

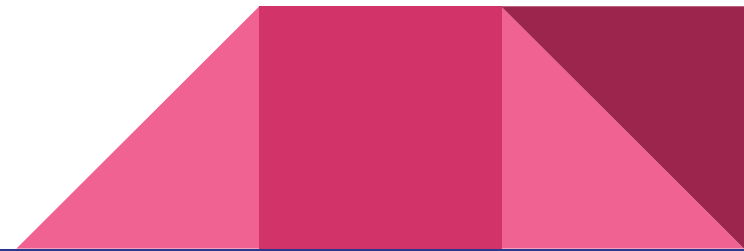


Po przegraniu wyświetla się wynik, jaki udało się uzyskać graczowi oraz opcja rozpoczęcia nowej gry.



po skończonej grze liczba zebranych monet jest dodawana do ilości już zgromadzonej oraz aktualizowany jest najdłuższy przebiegnięty przez gracza dystans

Omówienie wymagań niefunkcjonalnych projektu



Aplikacja działa na trzech najpopularniejszych przeglądarkach w Polsce

- demo

1. Google Chrome

1. Mozilla Firefox

1. Opera

Źródło: <https://www.vd.pl/ranking-przegladowek-internetowych/>

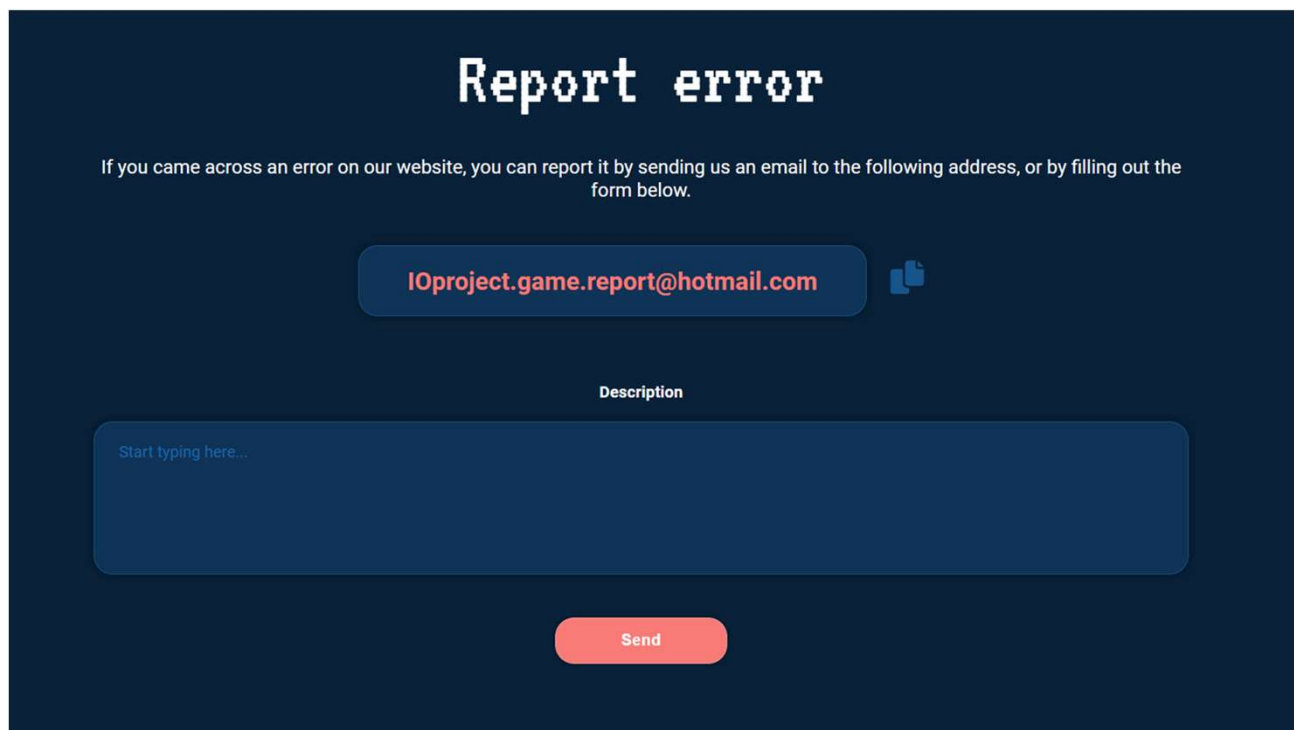


Aplikacja będzie dostępna dla każdego kto ma więcej niż 7 lat

- 🕒 brak reklam i nieprzystwoitych treści na stronie i w grze
- 🕒 regulamin forum dostępny pod linkiem w zakładce *Forum*




Użytkownicy mogą zgłaszać propozycje zmian/błędów na dedykowany adres mailowy



Report error

If you came across an error on our website, you can report it by sending us an email to the following address, or by filling out the form below.


IOproject.game.report@hotmail.com 

Description

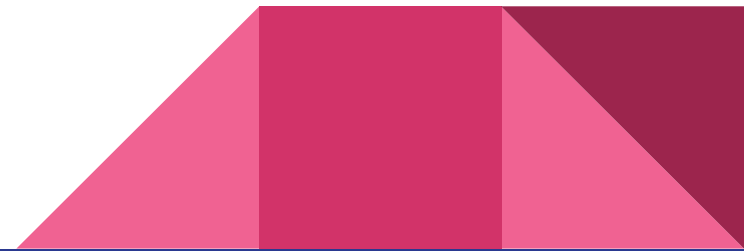
Start typing here...

Send

adres mailowy dostępny jest w zakładce Report error

zgłosić błąd można kopiując adres mailowy (poprzez kliknięcie na ikonkę ) i wysyłając wiadomość z własnej poczty lub przez wypełnienie dostępnego na stronie formularza

Proces projektowy



Od pomysłu do realizacji

PROFILE PAGE

link to main page with info about the game (introduction)

GAME INFO

skins in use

starts: coins | points | star

username

playing since: date, last seen: date

quote? | add quote

unlock skins:

skin name ← current

skin name

skin name

visible when opening profile owner?

update when logged in

style

→ profile

→ forum

→ index

→ top 10

→ shop

→ report err

→ log out

SHOP PAGE

ATTRIBUTE

SHOP

BUDGET: 250

skins | price

100 coins | UNLOCKED

150 coins | BUY button

200 | UNLOCKED

300 | BLOCKED

need more selection

blocked: not enough coins | buy: add skins to cart | unlocked skins

FORUM PAGE

GAME INFO

title ← not required (default: no title)

content

post

posts:

username | title | date, time

content

→ forum

TOP 10

TOP 10 PLAYERS

link to user page?

1. username | points | star

2. | star

3. | star

10. (or more when 2 is 0)

Pierwsze wersje projektu - szkic layoutu

INDEX PAGE

type username | find

list of matching results

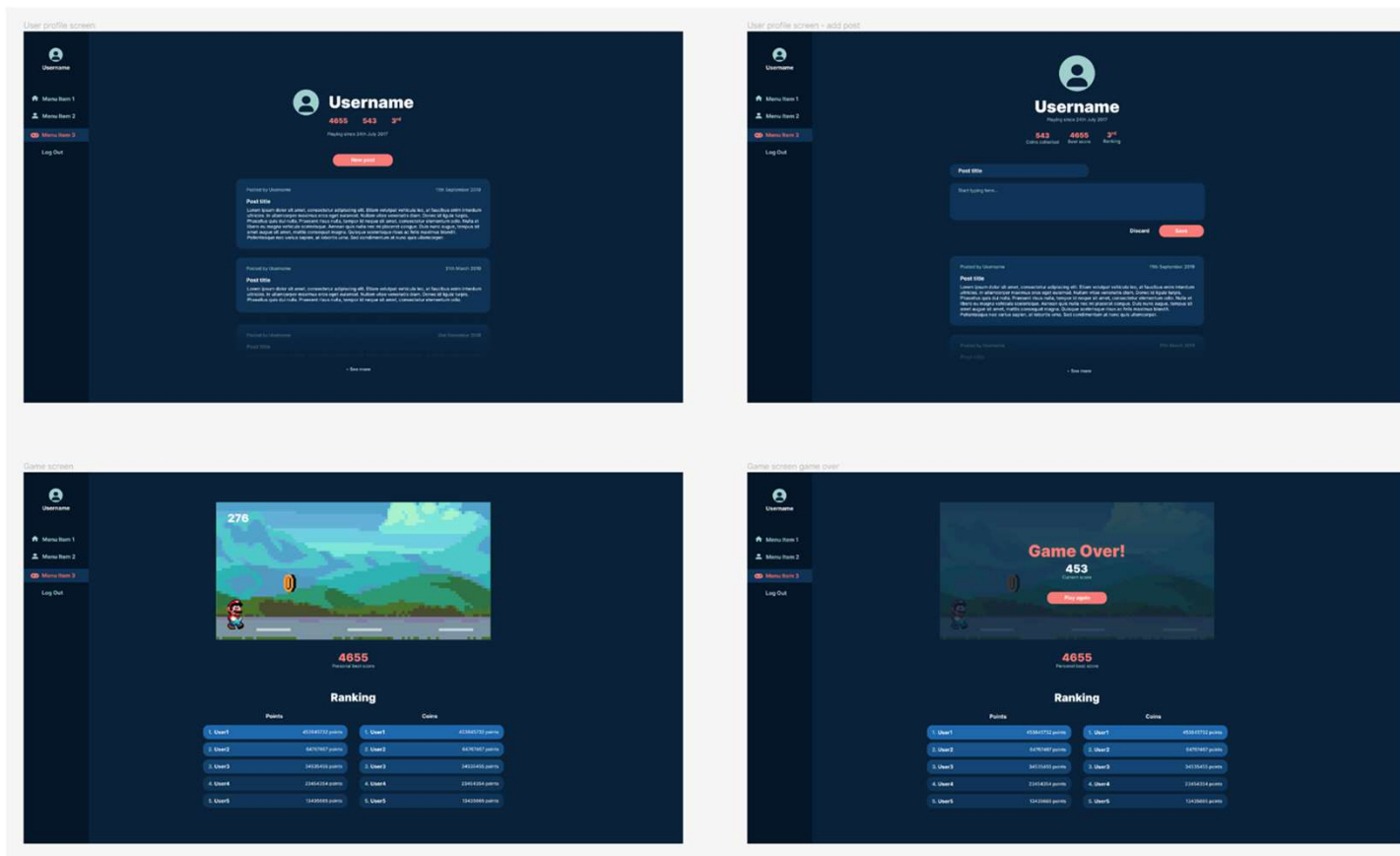
← link to user profile page

← " " " "

← " " " "

INDEX

Od pomysłu do realizacji

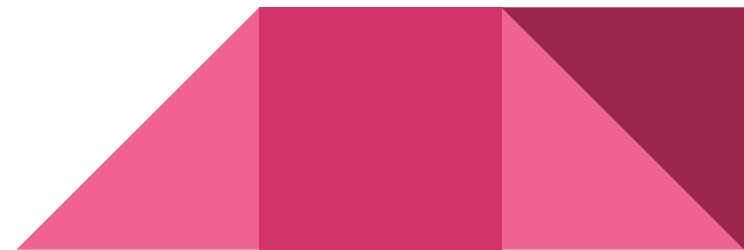


Pierwsze wersje projektu - Figma

Podział na zespoły

| Zespół programistów webowych | Zespół programistów gier | Zespół specjalistów ds. dokumentacji |
|---|---|--|
| <ul style="list-style-type: none">- Pająk Julia- Wnuk Aleksandra | <ul style="list-style-type: none">- Baran Dawid- Piękosz Tomasz | <ul style="list-style-type: none">- Oberda Mikołaj- Przybyła Agata |
| projekt oraz implementacja strony internetowej stworzenie serwera i bazy danych oraz zapewnienie komunikacji pomiędzy nimi | projekt oraz implementacja gry testowanie gry przygotowanie dokumentacji testowej | pełna dokumentacja projektu: <ul style="list-style-type: none">- opracowanie wymagań i analizy zagrożeń- analiza projektu od strony biznesowej- diagramy UML |

Struktura projektu



Technikalia

JavaScript, HTML (EJS), CSS: wykorzystane języki programowania

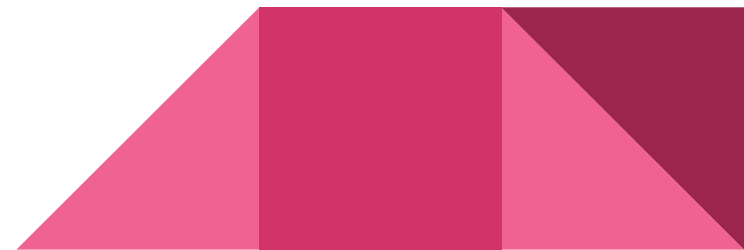
Node.js: wieloplatformowe środowisko uruchomieniowe do tworzenia aplikacji typu server-side napisanych w języku JavaScript. Umożliwia tworzenie aplikacji w obrębie jednego języka programowania. Zalety: asynchroniczność, wysoka skalowalność, duża popularność

Server.js: bardzo popularna biblioteka Node.js, pozwala tworzyć nawet bardzo skomplikowane aplikacje webowe, automatyzuje operacje związane z systemami API, zarządza żadaniami HTTP i HTTPS, sesjami oraz routingiem, obsługą błędów (wspiera proces debugowania), służy do organizacji projektu w oparciu o architekturę MVC (Model-View-Controller)

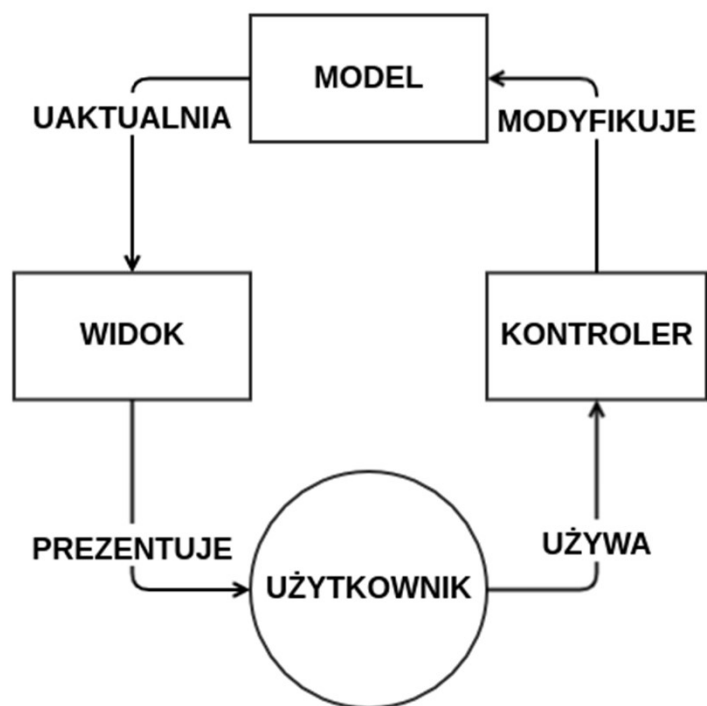
Repozytorium: <https://github.com/araignee19/IO-Endless-Runner>



Struktura strony internetowej



Architektura Model-Widok-Kontroler (MVC)



Pozwala rozbić duże aplikacje na moduły mające odrębne funkcjonalności.

Model:

determinuje to jak ustrukturyzowana jest baza danych. Jest to fragment aplikacji, który definiuje w jaki sposób komunikuje się ona z bazą danych. Kontrolery korzystają z modeli aby uzyskać dostęp do bazy danych.

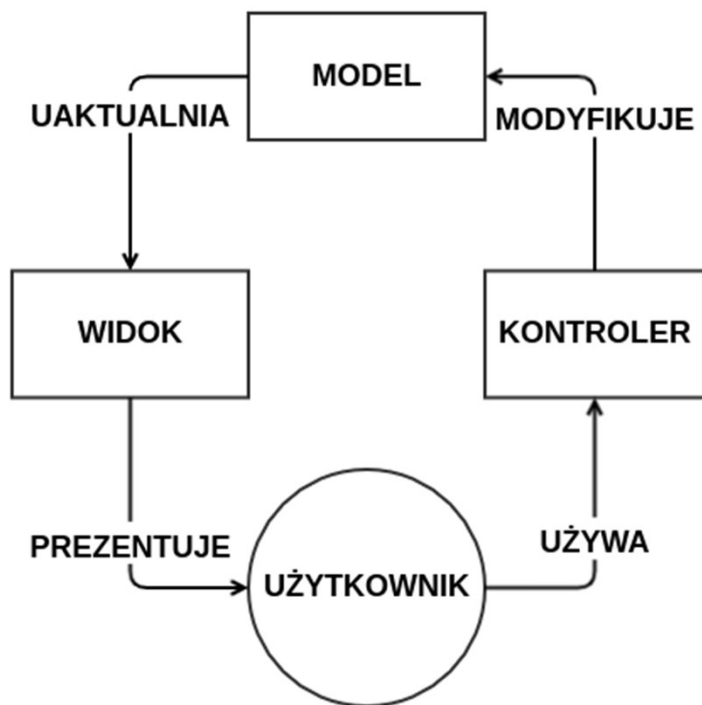
Widok:

część aplikacji odpowiedzialna za interakcję z użytkownikiem

Kontroler:

korzysta z modelu i wysyła odpowiedź/funkcjonalność to widoku

Architektura Model-Widok-Kontroler (MVC)



```
▼ IO_PROJECT
> .vscode
> models
> node_modules
> public
> routes
> tests
> views
⚙ .env
🔒 .gitignore
{} package-lock.json
{} package.json
JS passport-config.js
📖 README.md
JS server.js
JS utils.js
```

Baza danych - MongoDB, Mongoose

MongoDB - nierelacyjna baza danych - często używana w zestawieniu z pozostałymi technologiami w projekcie, prostota obsługi

Mongoose - umożliwia tworzenie schematów, które mają spełniać poszczególne kolekcje

```
const postSchema = new mongoose.Schema({
  title: {
    type: String
  },

  content: {
    type: String,
    required: true
  },

  author: {
    type: mongoose.Schema.Types.ObjectId,
    required: true,
    ref: 'User'
  },

  authorUsername: {
    type: String,
    required: true
  },

  date:{
    type: Date,
    required: true,
    default: Date.now
  }
})
```

Baza danych

W bazie obecne są kolekcje **users** oraz **posts**.

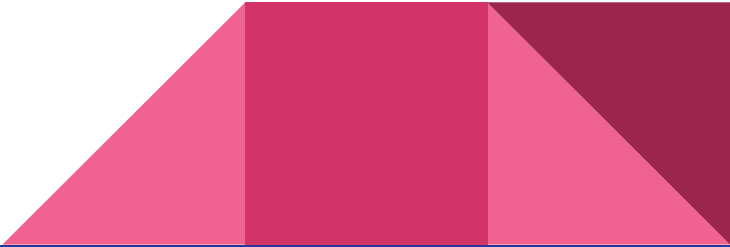
Komunikacja z bazą w poszczególnych routes za pomocą wbudowanych funkcji, przykładowo:

Wyszukiwanie użytkownika po ID:

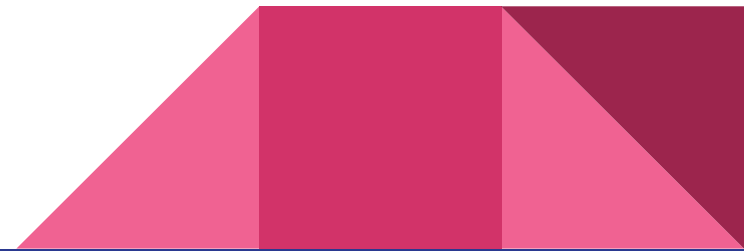
```
let user = await User.findById(req.user._id)
```

Zapisywanie zmodyfikowanego obiektu użytkownika w bazie:

```
const savedUser = await user.save()
```



MVC - Model

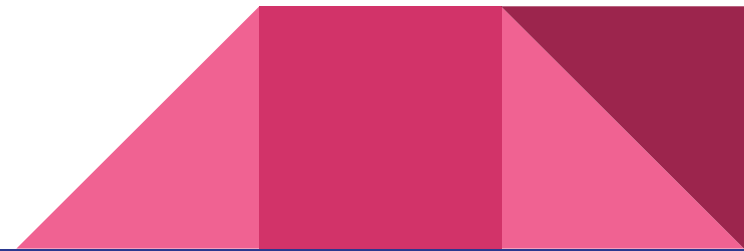


Modele - pliki .js w katalogu models

```
models > JS postjs > ...
1  const mongoose = require('mongoose')
2
3  const postSchema = new mongoose.Schema({
4    title: {
5      type: String
6    },
7
8    content: {
9      type: String,
10     required: true
11   },
12
13   author: {
14     type: mongoose.Schema.Types.ObjectId,
15     required: true,
16     ref: 'User'
17   },
18
19   authorUsername: {
20     type: String,
21     required: true
22   },
23
24   date: {
25     type: Date,
26     required: true,
27     default: Date.now
28   }
29 })
30
31
32 module.exports = mongoose.model('Post', postSchema)
```

```
▼ IO_PROJECT
  > .vscode
  ▼ models
    JS post.js
    JS user.js
    > node_modules
    > public
    > routes
    > tests
    > views
    ⚙ .env
    🔍 .gitignore
    {} package-lock.json
    {} package.json
    JS passport-config.js
    ⓘ README.md
    JS server.js
    JS utils.js
```

MVC - Kontroler

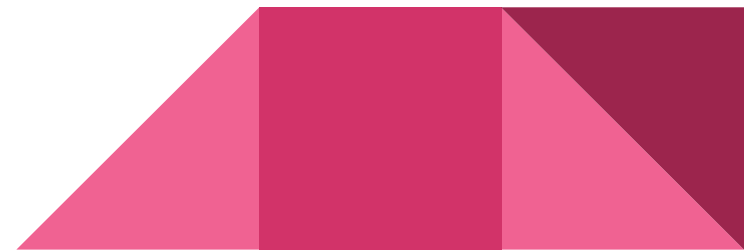


Kontrolery - pliki .js w katalogu routes

```
routes > JS ranking.js > ...
1  const express = require('express')
2  const router = express.Router()
3  const utils = require('../utils')
4  const User = require("../models/user")
5
6
7  router.get('/', utils.checkAuthenticated, async (req, res) => {
8    let topCoins
9    let topPoints
10   try {
11     topPoints = await User.find().sort({points: 'desc', username: 'asc'}).limit(10).exec()
12     topCoins = await User.find().sort({coins: -1, username: 1}).limit(10).exec()
13
14     res.render('ranking/ranking', {
15       topCoins: topCoins,
16       topPoints: topPoints,
17       user: req.user
18     })
19   } catch {
20     res.redirect('/')
21   }
22 })
23
24 module.exports = router
```

```
▼ IO_PROJECT
  > .vscode
  > models
  > node_modules
  > public
  ▼ routes
    JS auth.js
    JS catalog.js
    JS forum.js
    JS game.js
    JS index.js
    JS ranking.js
    JS report.js
    JS shop.js
    JS user.js
  > tests
  > views
  ⚙ .env
  🔍 .gitignore
  {} package-lock.json
  {} package.json
  JS passport-config.js
  ⓘ README.md
  JS server.js
  JS utils.js
```

MVC - Widok



Widoki - pliki .ejs w katalogu views

views > auth > login.ejs

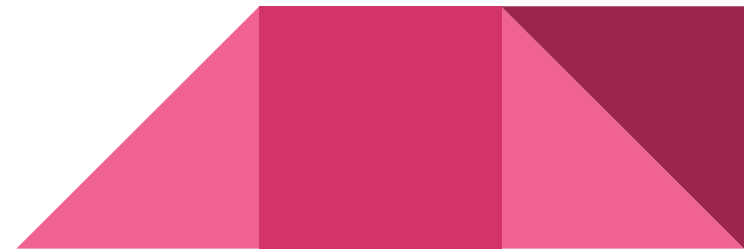
```
1  <%- include('functions.ejs') %>
2  <div class="auth">
3    
4    
5    
6    <h1>Coin Hunter</h1>
7    <% if(messages.error) { %>
8      <p class="error-msg"><i class="fa-solid fa-circle-exclamation"></i><%= messages.error %></p>
9    <% } %>
10   <form action="/" method="post">
11     <label for="username">Username</label>
12     <div >
13       <input name="username" placeholder="Username" type="text" maxlength="16">
14     </div>
15     <label for="password">Password</label>
16     <div class="container--relative">
17       <input class="peek-password" name="password" id="password" placeholder="Password" type="password" class="password--peek">
18       <i class="fa-solid fa-eye password-peek" onclick="peekPassword('password')" style="cursor: pointer;"></i>
19     </div>
20     <input type="submit" value="Login" class="button-link"></input>
21     <a href="/auth/register" class="button-link outline">Register</a>
22   </form>
23 </div>
```

IO_PROJECT

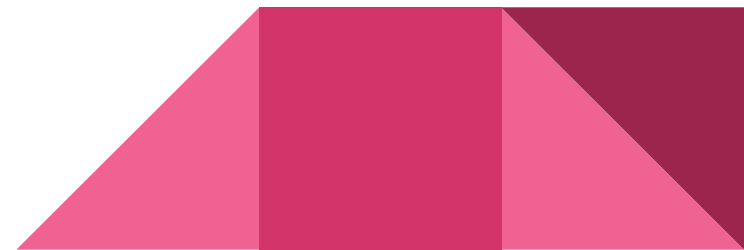
- > .vscode
- > models
- > node_modules
- > public
- > routes
- > tests
- views
 - auth
 - functions.ejs
 - login.ejs
 - register.ejs
 - catalog
 - forum
 - game
 - layouts
 - partials
 - ranking
 - report
 - shop
 - user
 - index.ejs
- .env
- .gitignore
- package-lock.json
- package.json
- passport-config.js
- README.md
- server.js
- utils.js

Inne użyte biblioteki

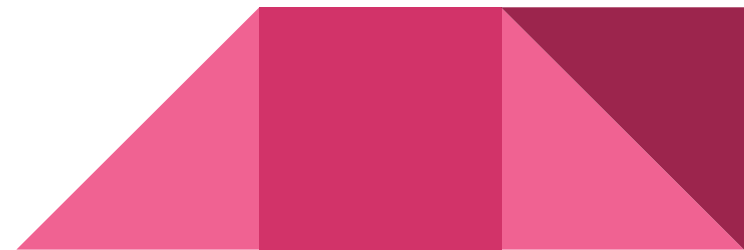
- Autentykacja użytkowników - **Passport.js**
 - Łatwa obsługa błędów, prostota implementacji, możliwość rozszerzenia opcji logowania na Google, Facebook itp.
- Szyfrowanie haseł - **bcrypt**
 - Bezpieczeństwo, odporność na hackowanie
- Obsługa formularza mailowego - **nodemailer**
 - Popularność modułu, łatwość implementacji



Struktura gry



Dokumentacja

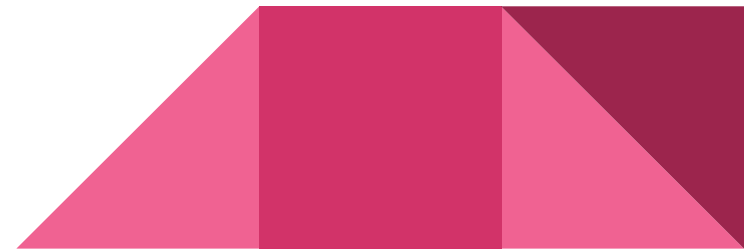


Demonstracja gotowej aplikacji

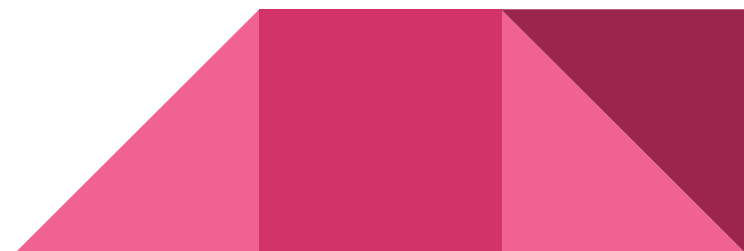
Aplikacja została umieszczona na platformie Heroku i jest dostępna dla każdego użytkownika internetu pod adresem:

<https://io-coin-hunter.herokuapp.com/>

zapraszamy na stronę!



Dziękujemy za uwagę!



Linki:

Repozytorium z kodem źródłowym projektu:

<https://github.com/araignee19/IO-Endless-Runner>

Aplikacja:

<https://io-coin-hunter.herokuapp.com/>

