Web žaidimas - Speland

AUTORIUS: SAULIUS ŠNIAUKŠTA IFZM-2

Žaidimo Idėja

- Žaidimas
- 2D
- Top-Down (liet. Iš viršaus)
- Vieno žaidėjo
- Pikselių grafikos
- „Dungeon Crawl“ (liet. Požemių tyrinėjimo) ir „Bullet Hell“ (liet. Šovinių pragaro) tipo
- Realaus laiko kovos

Sistemos:
Žaidėjo lygio kėlimas, inventorius, progreso išsaugojimas (registracija ir prisijungimas), priešų, požemių/lygių, apsimaunamų daiktų, klasės (tankas, žalos darytojas, magas) ir evoliucijos.

Žaidėjas gali rinkti klases, kovoti su priešais, rinkti daiktus ir tobulinti įgūdžius.

Panašios sistemos 1 - Realm Of The Mad God

- Top-Down
- Paprasta inventoriaus sistema
- Lygio kėlimo sistema
- Klasių sistema
- Paprastos pikselių grafikos
- Iš priešų išmetami daiktai
- Prie žaidėjo atsiranda atsitiktiniai požemiai
- Bullet Hell mechanikos

Tačiau dėmesys skiriamas MMO (daugelio žaidėjų) aspektui

[1]

[2]

Panašios sistemos 2 - Stein.World

- Pikselių grafikos 2D MMORPG
- Naršyklinis (sukurtas naudojant HTML5)
- Inventoriaus sistema
- Prisijungimo, saugojimo sistemos
- Daiktų sistema
- Atviro pasaulio

Patrauklaus dizaino, daugelio žaidėjų žaidimas.
Kova su priešais pusiau ėjimu paremta. Priešas ir žaidėjas puola tam tikru intervalu, labai lėtai

Žaidimas daug dėmesio skiria žaidėjų sąveikai

[3]

Panašios sistemos 3 - Heroes of Hammerwatch

- 2D pikselių grafikos požemių tyrinėjimo žaidimas
- Lygio kėlimo sistema
- Masinės priešų bangos ir kovos su bosais
- Žaidėjai gali rinktis klases
- Grupinis žaidimas 1-4 žaidėjams

Žaidėjas keliauja per nesikeičiančius požemius, kurie yra pripildyti priešais. Paskutinio požemio gale yra bosas.
Žaidėjas pakėlęs lygį gali išleisti taškus ir atrakinti naujus gebėjimus.
Klasės suskirstytos į tanko, žalos darytojo ir gydytojo.

[4]

Naudojamos technologijos

- HTML5 + Canvas - vizualizacija
- JavaScript - logika
- SQLite- duomenų bazė
- Node.js + Express - serverio pusės logika
- Aseprite - pikselių grafikos objektų kūrimas
- Garso efektų kūrimo įrankiai
- Socket.io - daugelio žaidėjų mažai integracaijai

Sistemos analizė

- Naršyklinis 2D RPG žaidimas, kuriame žaidėjas tyrinėja požemius, kovoja su priešais renka daiktus ir kelia lygį.

Pagrindinės funkcijos:

1. *Vartotojų registracija ir prisijungimas*
2. *Veikėjo kūrimas*
3. *Klasės ir papildomos klasės pasirinkimas*
4. *Kovos sistema*
5. *Priešai*
6. *Daiktų išmetimas*
7. *Inventorius*
8. *Patirties (XP) ir žaidėjo lygio sistema*
9. *Pagrindinis miestas ir atsitiktiniai požemių lygiai*

Sistemos analizė

Pagrindiniai vartotojai:

1. Žaidėjas - registruojasi, jungiasi, kurią veikėją, žaidžia.
2. Sistema - saugo progresą, duomenis, valdo logiką.

Naudojamos technologijos:
HTML5 - žaidimo atvaizdavimui.
CSS - žaidimo išvaizdai sukurti.
JavaScript - žaidimo funkcijų ir logikos įgyvendinimas.
Node.js + Express - serverio logikos sukūrimui ir palaikymui.
SQLite - lokali duomenų bazė prototipui.

Sistemos raktažodžiai

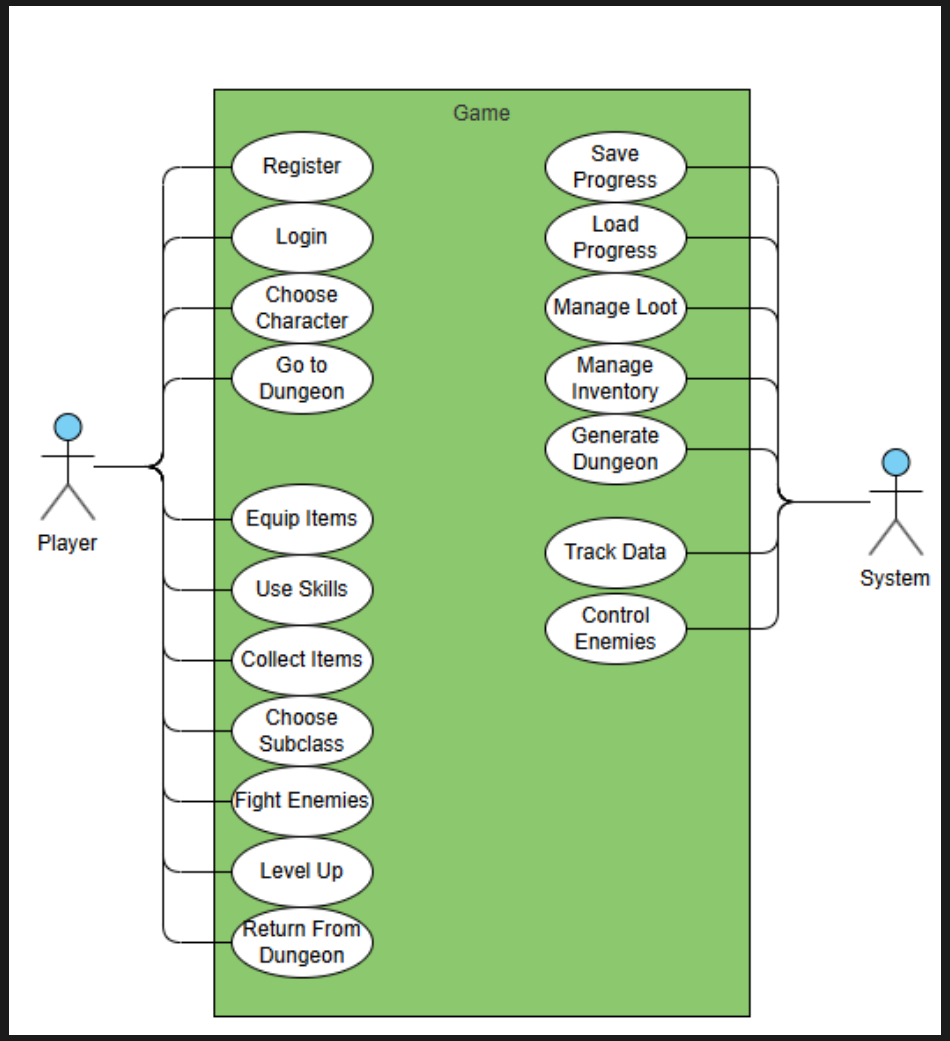
Tikslas - padidinti
žaidimo matomumą internete
ir pritraukti vartotojus.

Angliški raktažodžiai:
2D RPG
Dungeon Crawler
Pixel RPG
Browser RPG
Bullet Hell

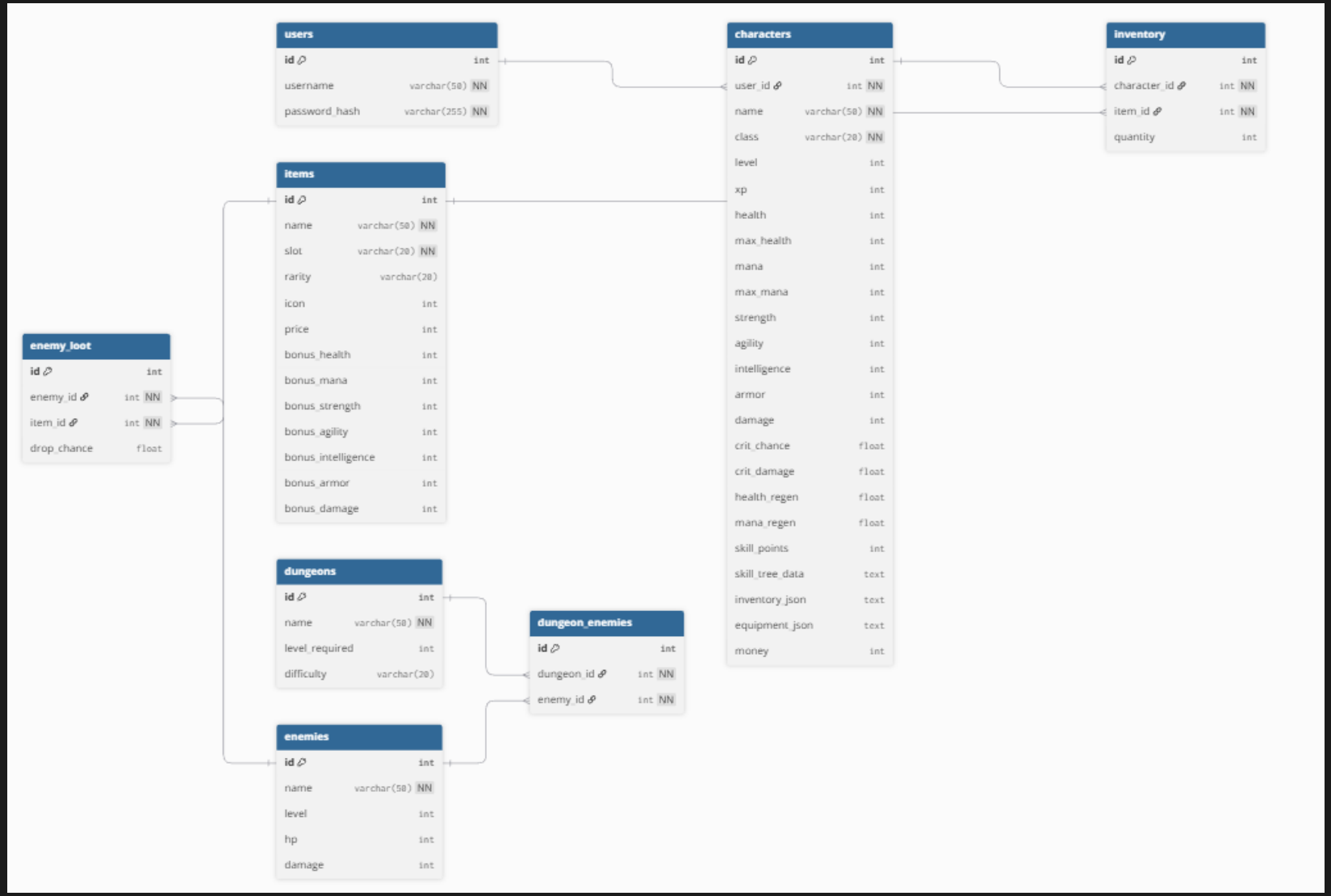
Jų panaudojimas:

1. Puslapio pavadinime
2. Puslapio aprašyme (meta).
3. Žaidimo pradiniame prisijungimo/registracijos lange.
4. Vaizdų (sprites) aprašymuose.

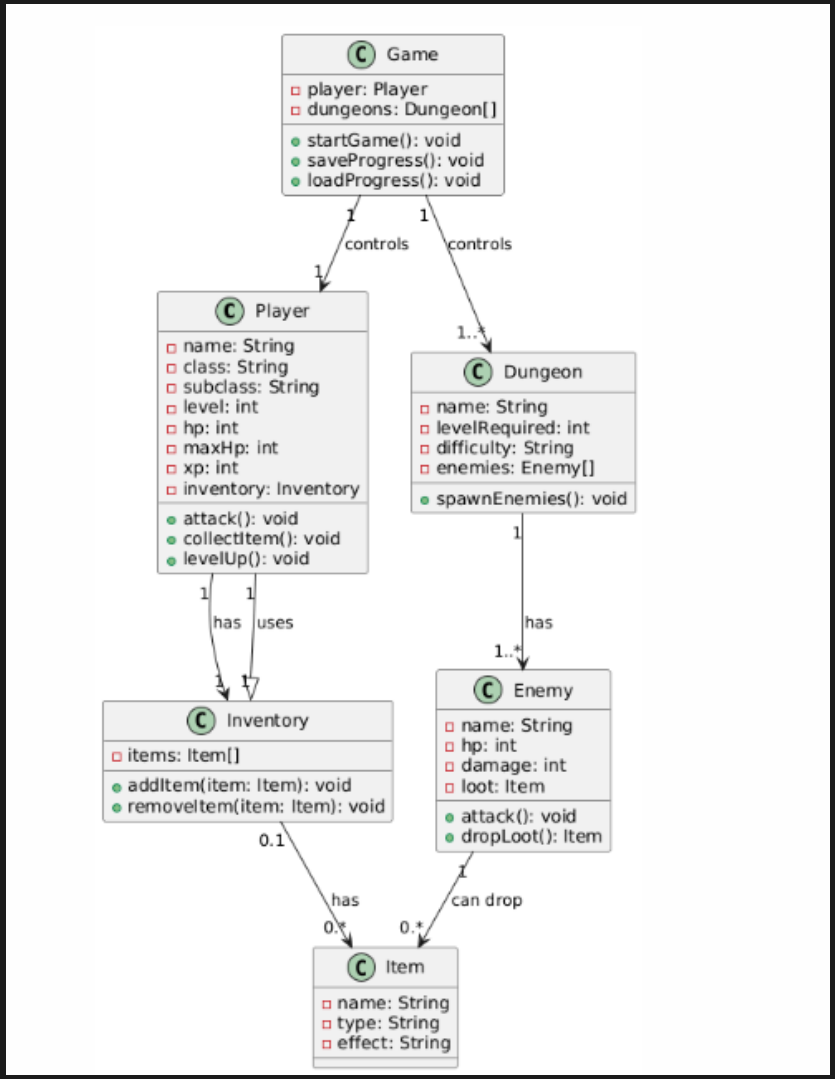
Panaudos atvejų diagrama



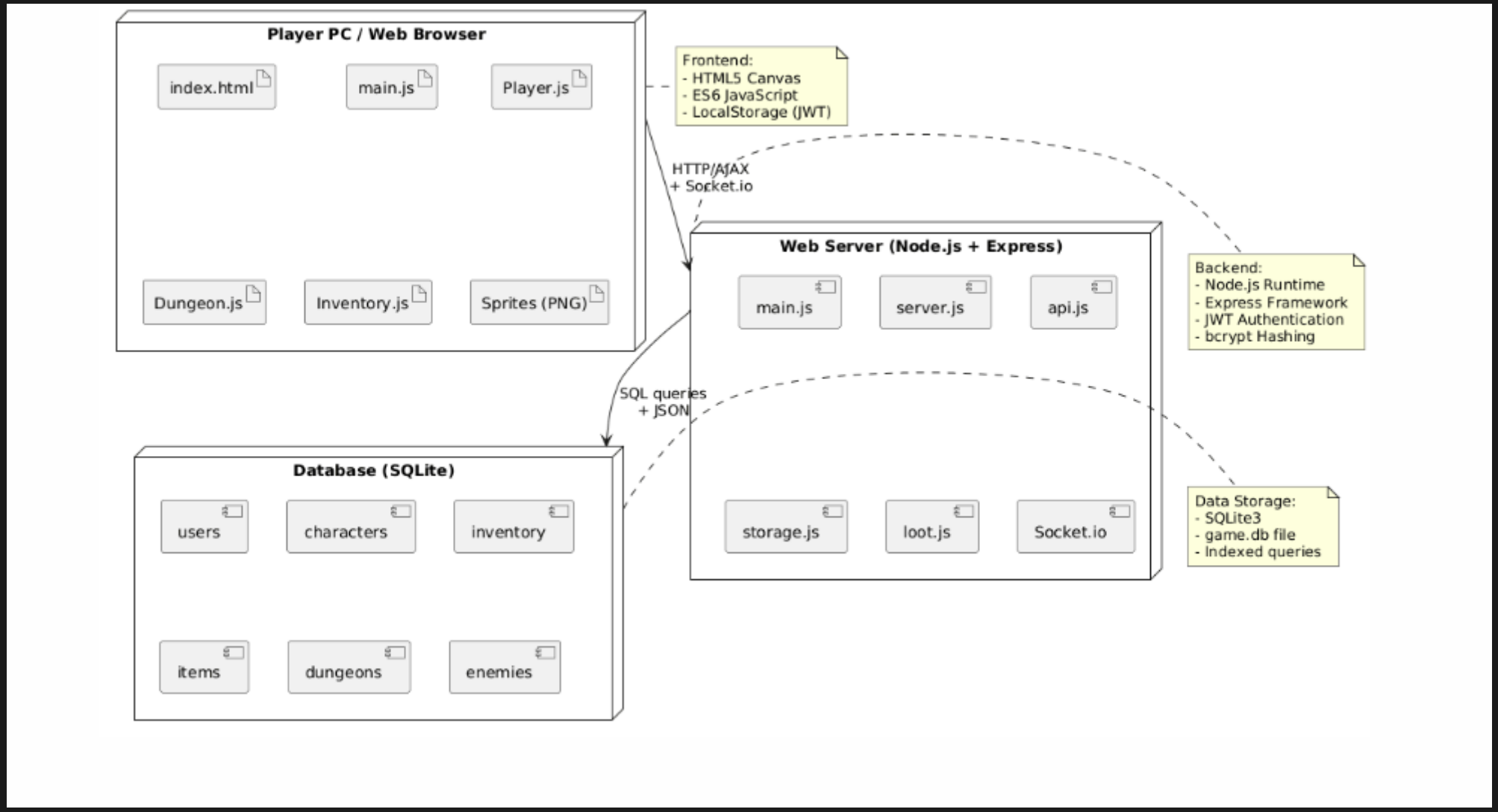
Duomenų bazių diagrama



Klasių diagrama



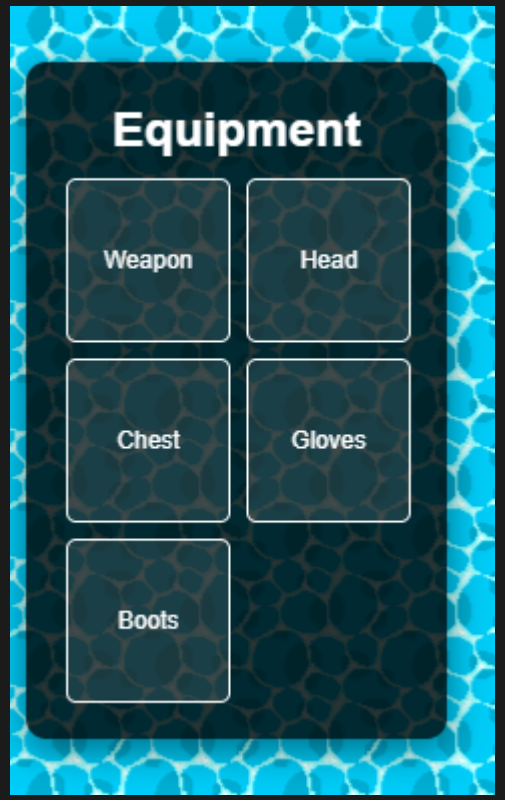
Diegimo diagrama



Projektavimo išvados

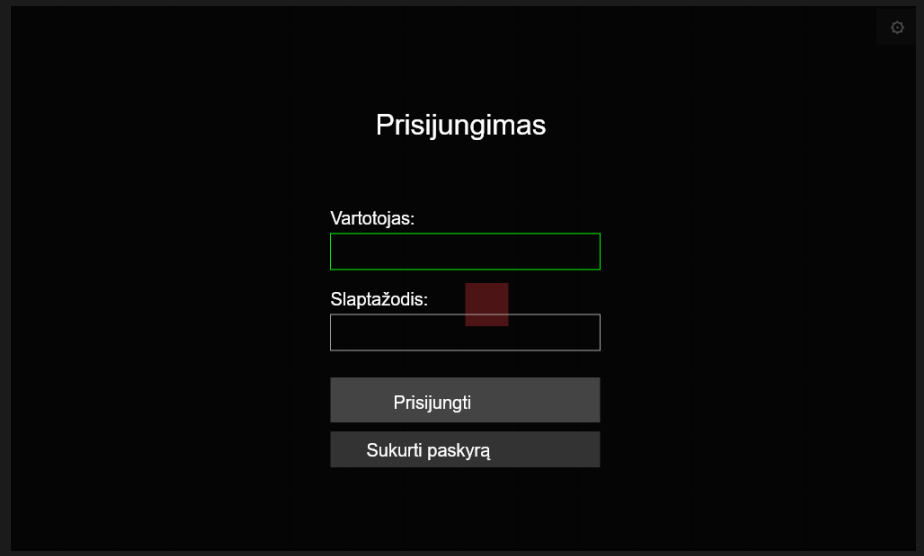
1. Panaudota modulinė architektūra palengviną plėtrą.
2. Atsitiktinių lygių ir priešų kūrimas didina žaidimo įvairovę, bet reikalauja optimizacijos, kad nebūtų per sunku ar per lengva.
3. Lokali duomenų bazė SQLite tinka prototipui, plėtimuisi geriausia naudoti serverį.
4. Raktažodžių naudojimas leidžia lengviau rasti žaidimą internete.
5. Didėjant sistemų kiekiui didėja implementacijos sudėtingumas, todėl reikia atsižvelgti kokias sistemas naudoti ir kaip jos siejasi tarpusavyje.

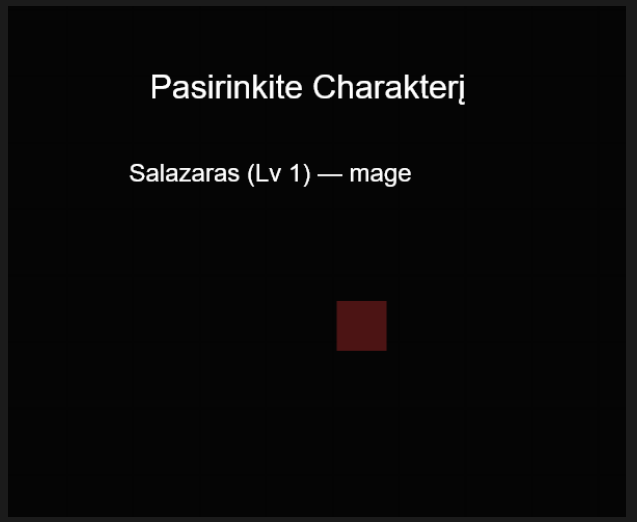
Prototipo ekranvaizdžiai



Ekranvaizdžiai 2.0

Žaidimo



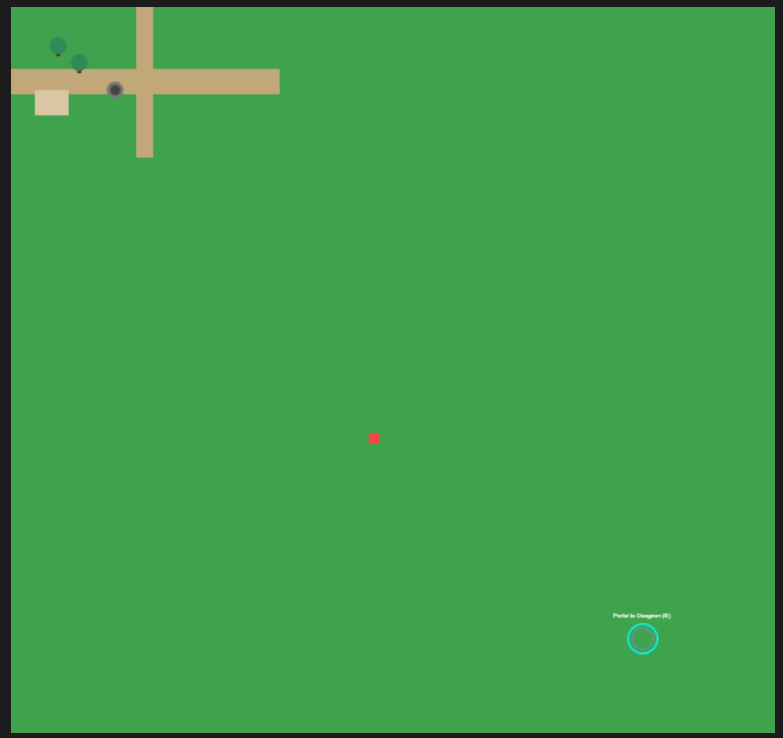


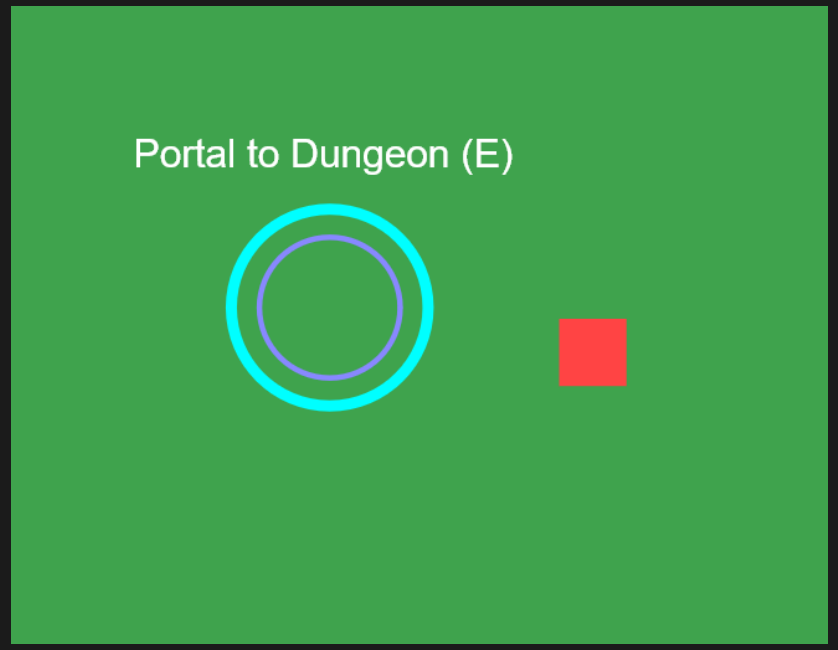
Ekranvaizdžiai 2.0

Žaidimo

Ekranvaizdžiai 2.0

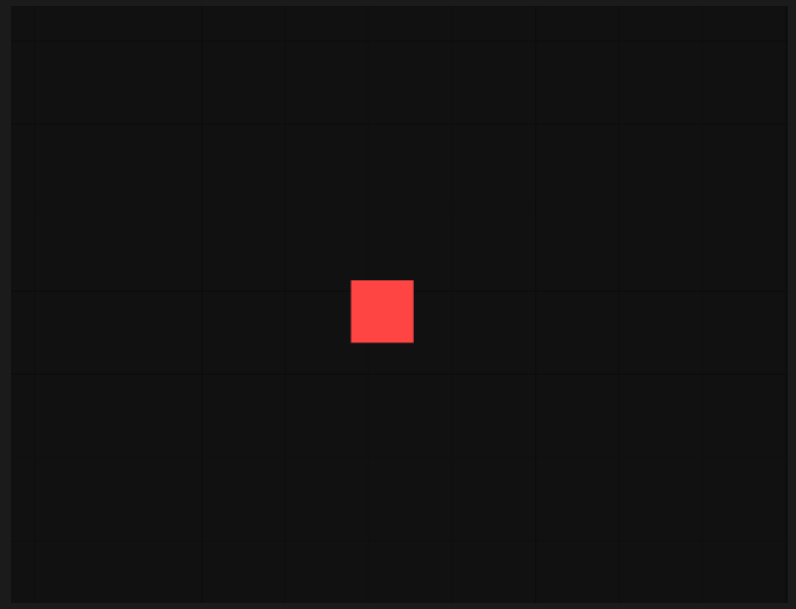
Žaidimo

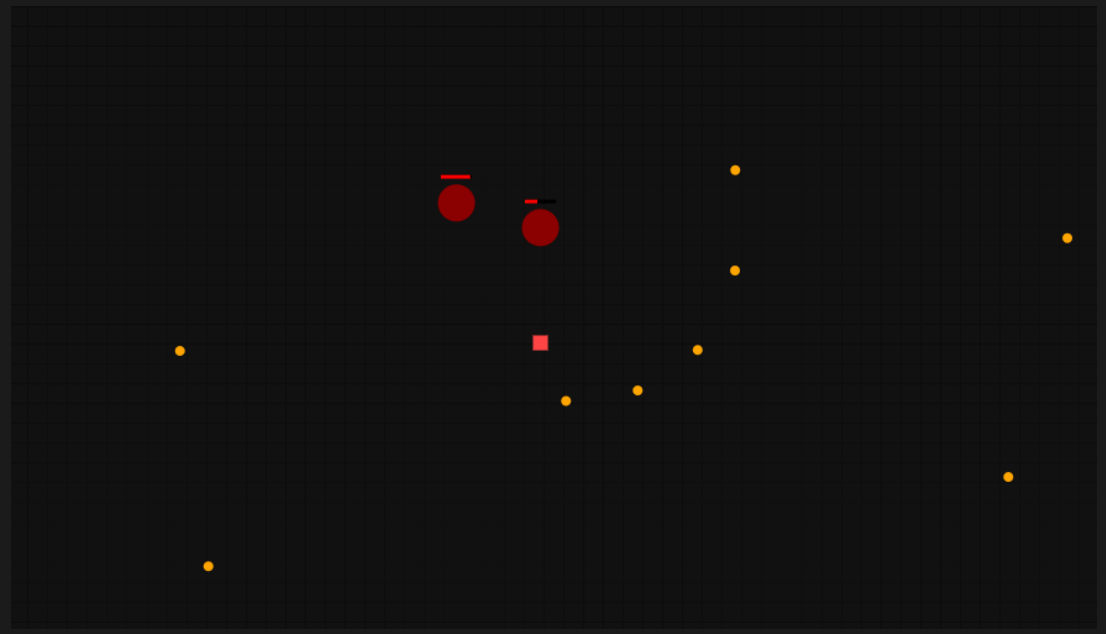




Ekranvaizdžiai 2.0

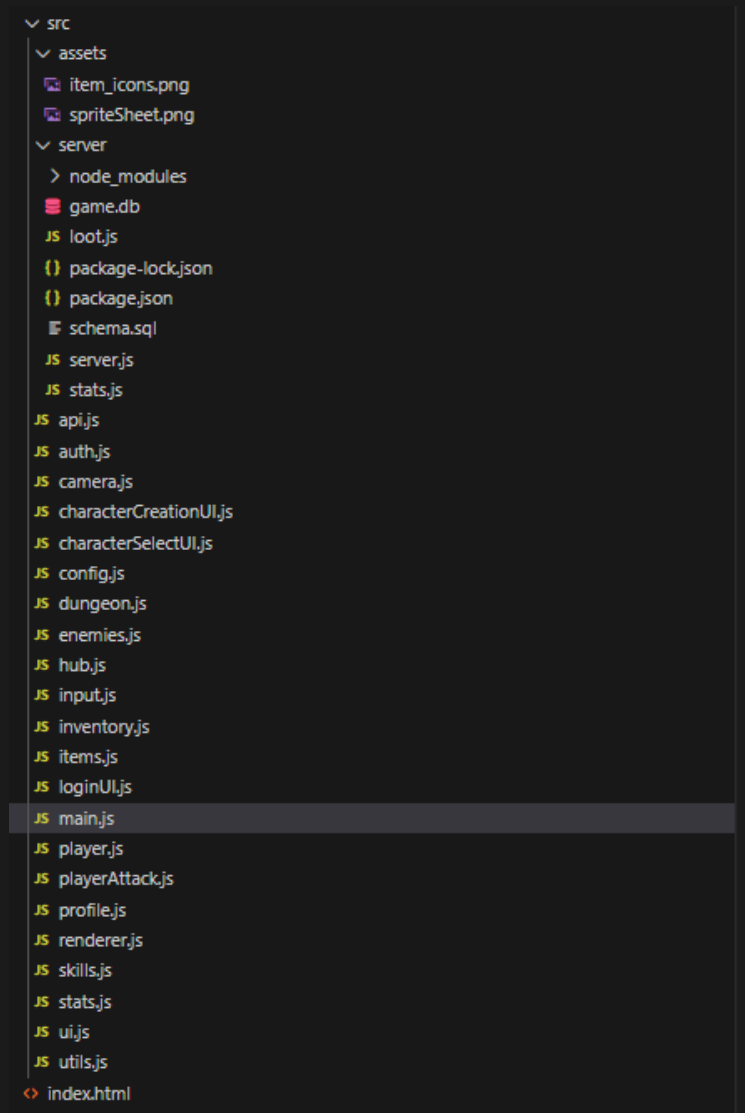
Žaidimo

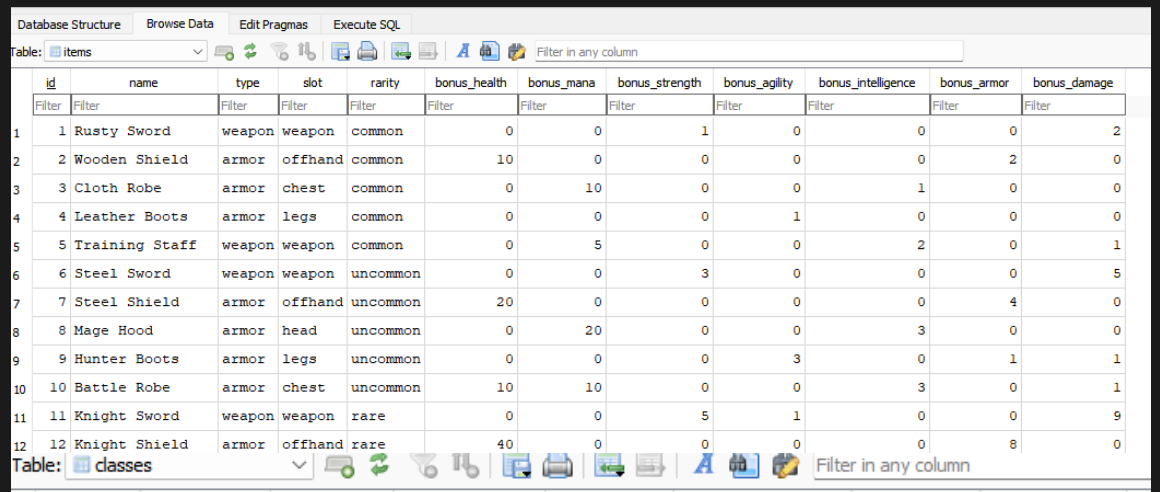




Ekranvaizdžiai 2.0

Failų





id	name	type	slot	rarity	bonus_health	bonus_mana	bonus_strength	bonus_agility	bonus_intelligence	bonus_armor	bonus_damage
1	Rusty Sword	weapon	weapon	common	0	0	1	0	0	0	2
2	Wooden Shield	armor	offhand	common	10	0	0	0	0	2	0
3	Cloth Robe	armor	chest	common	0	10	0	0	1	0	0
4	Leather Boots	armor	legs	common	0	0	0	1	0	0	0
5	Training Staff	weapon	weapon	common	0	5	0	0	2	0	1
6	Steel Sword	weapon	weapon	uncommon	0	0	3	0	0	0	5
7	Steel Shield	armor	offhand	uncommon	20	0	0	0	0	4	0
8	Mage Hood	armor	head	uncommon	0	20	0	0	3	0	0
9	Hunter Boots	armor	legs	uncommon	0	0	0	3	0	1	1
10	Battle Robe	armor	chest	uncommon	10	10	0	0	3	0	1
11	Knight Sword	weapon	weapon	rare	0	0	5	1	0	0	9
12	Knight Shield	armor	offhand	rare	40	0	0	0	0	8	0

Table: classes — Filter in any column

name	base_health	base_mana	base_strength	base_agility	base_intelligence	base_armor	base_damage
warrior	150	30	10	5	2	5	10
mage	80	150	2	4	12	0	4
tank	200	40	6	3	3	10	6

Table:

id	user_id	name	class	evel	xp	health	max_health	mana	max_mana	strength	agility	intelligence	crit_chance	crit_damage	armor	damage	health_regen	mana_regen
1	1	Salazaras	mage	1	0	80	80	150	150	2	4	12	0.1	2.0	0	4	0.3	1.5
2	2	labas	tank	1	0	200	200	40	40	6	3	3	0.02	1.2	10	6	1.5	0.3
3	3	ryt	mage	1	0	80	80	150	150	2	4	12	0.1	2.0	0	4	0.3	1.5
4	4	ane	mage	1	0	80	80	150	150	2	4	12	0.1	2.0	0	4	0.3	1.5

Sistemos analizė 2.0

1. Sistema sukurta naudojant „canvas“ pagrindu veikiančią architektūrą, todėl ji yra lengvai pritaikoma mobiliems įrenginiams, nes nereikalauja sudėtingų struktūrų.

2. Dėl modulinių failų ir atskirtų logikos sluoksnių (rendering, input, UI, enemies, projectiles) sistema gali būti optimizuojama atskirai, nekeičiant visos struktūros.

3. Greitaveika užtikrinama naudojant vieną piešimo ciklą (requestAnimationFrame) ir ribotą objektų kiekį ekrane, todėl žaidimas išlieka sklandus net silpnesniuose įrenginiuose.

4. Kamera, žaidėjo judėjimas ir priešų AI yra apskaičiuojami naudojant paprastas matematines operacijas, todėl sistema išlieka efektyvi ir nereikalauja didelių resursų.

Ekranvaizdžiai 3.0

Žaidimo

Ekranvaizdžiai 3.0

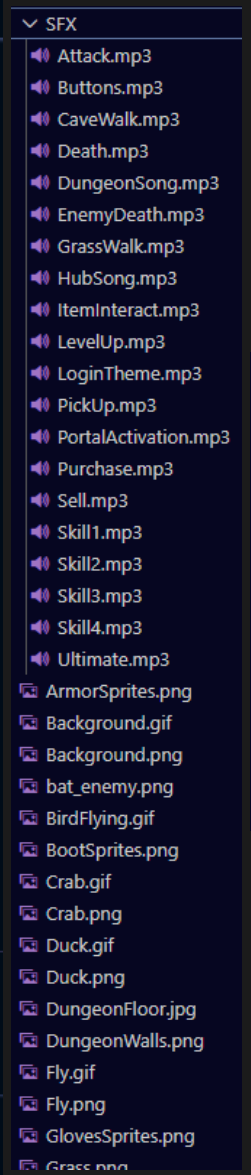
Žaidimo

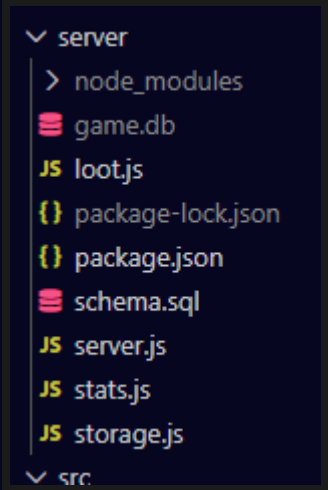
Ekranvaizdžiai 3.0

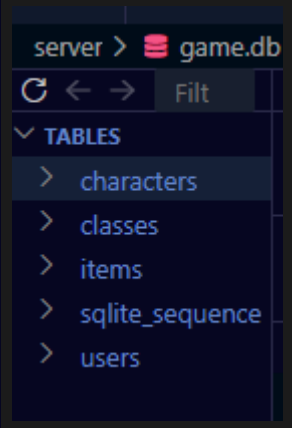
Žaidimo

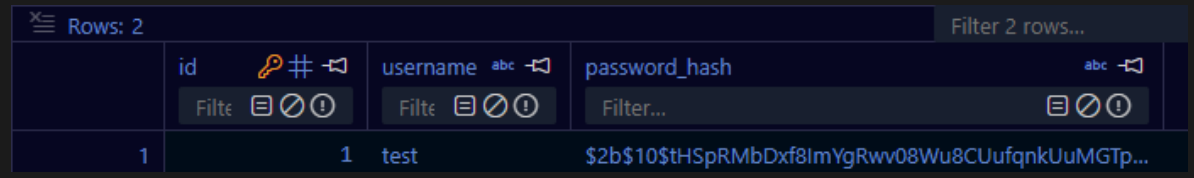
Ekranvaizdžiai 3.0

Failų









Sistemos analizė 3.0

1. Sistema sukurta naudojant Canvas API ir modulinę ESb architektūrą (30+ nepriklausomų modulių), todėl kiekviena sistema (skills, enemies, UI, inventory) gali būti plečiama ir optimizuojama atskirai nekeičiant visos struktūros.

2. Greitaveika užtikrinama naudojant vieną piešimo ciklą (requestAnimationFrame), objektų pooling'ą (projectiles clear) ir dinaminį priešų atkūrimą, todėl žaidimas išlieka sklandus 60 FPS net su 20+ priešų ekrane.

3. Multiplayer funkcionalumas realizuotas per Socket.io su realtime komunikacija, leidžiantis žaidėjams susitikinėti hub'e, naudojant efektyvų „event-driven“ architektūrą be serverio apkrovos.

4. Duomenų bazės optimizacija su 8 indeksais, JSON kešavimo sistema (inventory_json, equipment_json) ir JWT autentifikacija užtikrina greitą duomenų prieigą ir saugų vartotojų valdymą.

Sistemos analizė 3.0

1. Funkcijų struktūra laikosi Single Responsibility principo - 114 eksportuojamų funkcijų, kiekviena atlieka vieną aiškiai apibrėžtą veiksmą (pvz., updatePlayer(), drawEnemies(), clearProjectiles()).

2. HTML5 semantinė struktūra su <meta> SEO optimizacija (raktažodžiai, OpenGraph), <canvas> elementu ir tvarkingas kodas pagal W3C standartus.
3. SQLite duomenų bazė su 4 pagrindinėmis lentelėmis (users, characters, items, classes) ir JSON formato inventoriaus ir užsidėtų daiktų saugojimas užtikrina duomenų vientisumą ir greitą prieigą be papildomų operacijų.

Išvados

1. Sukurta sistema yra pilnai funkcionali ir užbaigta su primityviu daugelio žaidėjų funkcionalumu, realaus laiko kovos, įgūdžiu medžio, klasių evoliucijomis, inventoriaus valdymu ir SEO optimizacija.

2. Modulinė ESĮ architektūra leidžia lengvai plėsti funkcionalumą nekeičiant esamo kodo, tačiau didelis sistemų kiekis reikalauja atidaus integravimo tolimesniam vystymuisi.

3. Projektas parodė, kad „Single Responsibility" principas, „Socket.io" realaus laiko komunikacija ir „JWT" autentifikacija užtikrina stabilų, saugų ir sklandų žaidimą, nors didėjant žaidėjų skaičiui reikalinga papildoma optimizacija.

Darbo laiko lentelė

- Didžiausia darbo laiko dalis buvo skirta sistemos architektūros kūrimui ir pagrindinių modulių (žaidėjo, priešų, UI) integracijai.
- Reikšminga laiko dalis skirta klaidų paieškai ir taisymui, ypač susijusioms su koordinatėmis, kamera ir UI būsenų valdymu.
- Papildomas laikas skirtas funkcijų plėtrai – priešų AI, įvairios kovos sistemos, išmetamų daiktų mechanikai ir požemių logikai.
- Darbo eiga buvo iteracinė: kiekvienas modulis buvo kuriamas, testuojamas ir tobulinamas atskirai, taip užtikrinant stabilų progresą.

Darbo laiko lentelė

Darbo dalis	Aprašymas	Laikas
Projekto planavimas ir architektūra	Sistemos struktūros kūrimas, moduliai, failų išdėstymas, technologijų pasirinkimas	6h
HTML5 + Canvas UI pagrindas	Canvas inicializacija, render loop, input sistema, UI sluoksniai	10h
Login / Register / Logout sistema	Canvas formos, serverio validacija, API integracija	10h
Serverio kūrimas (Node.js + SQLite)	API endpoint'ai, DB schema, saugus duomenų apdorojimas	12h
Inventoriaus sistema	DB lentelės, API, canvas inventorius, item ikonų atvaizdavimas	10h
Žaidimo variklis (player movement, kamera)	Judėjimas, delta time, kamera, zoom_in/zoom_out	10h
Dungeon generavimas	Procedūrinis išdėstymas, sienos, portalai, perėjimai	8h
Sviedinių sistema	Pelės koordinatės → pasaulio koordinatės, skridimas, collision	3h
Testavimas ir derinimas	Bug'ų taisymas, koordinatės, kamera, UI būsenos	12h

Šaltiniai

1. https://www.realmeye.com/wiki/magic-mushroom
2. https://lostgarden.com/2012/06/19/goodbye-realm-of-the-mad-god
3. https://pg5-studio.itch.io/steinworld
4. https://store.steampowered.com/app/677120/Heroes_of_Hammerwatch

Nuorodos

Projekto GITHUB nuoroda: https://github.com/ZORAS/HTML_RPG.git