ЗМІСТ

[Вступ 4](#_Toc28203066)

[1 Постановка задачі дослідження 5](#_Toc28203067)

[1.1 Аналіз предметної області 5](#_Toc28203068)

[1.2 Огляд існуючих рішень за обраною темою 5](#_Toc28203069)

[1.2.1 Гра для MicrosoftWindowsTerraria 5](#_Toc28203070)

[1.2.2 Гра «Limbo» 6](#_Toc28203071)

[1.2.3 Гра Angry Birds 8](#_Toc28203072)

[1.2.4 Гра Dustforce 8](#_Toc28203073)

[1.3 Постановка задачі 10](#_Toc28203074)

[2 Специфікація вимог до програмного забезпечення (srs – software requirement specification) 11](#_Toc28203075)

[2.1 Загальна інформація щодо веб-застосунку 11](#_Toc28203076)

[2.1.1 Огляд продукту 11](#_Toc28203077)

[2.1.2 Мета 11](#_Toc28203078)

[2.1.3Межі 12](#_Toc28203079)

[2.1.4 Масштаб проекту 12](#_Toc28203080)

[2.1.5 Обмеження реалізації 12](#_Toc28203081)

[2.1.6 Діаграми потоків даних 12](#_Toc28203082)

[2.2 Вимоги до веб-застосунку 13](#_Toc28203083)

[2.2.1 Функціональні вимоги 13](#_Toc28203084)

[2.2.2 Вимоги до зовнішнього інтерфейсу 15](#_Toc28203085)

[2.2.3 Нефункціональні вимоги 15](#_Toc28203086)

[2.3 Опис обраних технологій для розробки веб-застосунку 15](#_Toc28203087)

[2.3.1 Мова гіперрозмітки HTML5 15](#_Toc28203088)

[2.3.2 Каскадні стилі CSS3 17](#_Toc28203089)

[2.3.3 Мова програмування JavaScript 18](#_Toc28203090)

[2.3.4 Серверна платформа Node.js 20](#_Toc28203091)

[2.3.5Фреймворк Express 22](#_Toc28203092)

[2.3.6 База даних MongoDB 23](#_Toc28203093)

[2.4 Опис архітектури веб-застосунку 24](#_Toc28203094)

[2.4.1 Діаграма розміщення 24](#_Toc28203095)

[2.4.2 Діаграма компонентів 25](#_Toc28203096)

[2.5 Модель бази даних 26](#_Toc28203097)

[3 Опис веб-застосунку 29](#_Toc28203098)

[3.1 Тестування програмного забезпечення 29](#_Toc28203099)

[3.1.1 Валідація програмного продукту 29](#_Toc28203100)

[3.1.2 Навантажене тестування 30](#_Toc28203101)

[3.2 Керівництво користувача 32](#_Toc28203102)

[3.3 Використання веб-застосунку 33](#_Toc28203103)

[Висновки 38](#_Toc28203104)

[Список інформаційних джерел 39](#_Toc28203105)

# ВСТУП

З розвитком комп’ютерних технологій, комп’ютерні ігри почали здобувати велику популярність серед населення. Відеоігри, що застосовували власне відео для взаємодії з гравцем розповсюдилися ще у 60-х роках минулого століття, а з появленням перших аркадних ігрових автоматів та приставок галузь спричинила бум серед молоді того часу.

Зараз ми маємо великий вибір серед добре продуманих ігор, що не перестають дивувати сюжетом, графікою, технічними інноваціями з року в рік. В наш час знайти людину, що б здивувалася побачивши гру Mario, мабуть, неможливо, але ще не так давно, у 1980-х ця гра і подібні до неї аркади – відеоігри з навмисно примітивним ігровим процесом, здобули неабияку популярність. І зараз існує певний круг людей, що не будуть проти зайвий раз пройти Pacman, або пограти з другом в мультиплеэрі у MarioBros.

Метою виконання цієї роботи є розробка гри-акради «RunningWild», що не буде занадто складною і дозволить гравцю відпочити від повсякденних справ. Основна мета гри – вивчення рівней (за рахунок багаторазового проходження) задля скорішого проходження та попадання до таблиці лідерів. На етапі проектування будуть описані методи та етапи проектування, надані UMLдіаграми. Технології, що будуть використовуватися включають мову JavaScript, сервер NodeJS, HTML та CSS.

Результатом стане гра «RunningWild», що являє собою головоломку з декількома кімнатами, в яку можна пограти у своєму браузері.

# 1 ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

## 1.1 Аналіз предметної області

Предметною областю обраного проекту є аркадна гра – гра з примітивним ігровим процесом, а саме – платформер.

Актуальність обраної тематики, а саме жанру платформера (бігали) є безперечною. Такі ігри можна знайти в телефонах, комп’ютерах, приставках. Навіть зараз з розповсюдженням більш потужних процесорів, відео карт та появою 3D ігор – старі платформери користуються великою популярністю. Сьогодні їх також продовжують розробляти (наприклад: Terraria, Limbo, Trine 2, ShovelKnightі т. д).

Ці прості ігри не потребують багато часу для освоєння. У більшості випадках ми можемо керувати персонажем просто натискаючи відповідні стрілки на екрані, джойстику, клавіатурі. Вони не потребують багато пам’яті, швидко запускаються, тому в них часто грають щоб «вбити час».

## Огляд існуючих рішень за обраною темою

Огляд існуючих аркад, платформерів є важливою складовою роботи. Аналіз сильних та слабих сторін існуючих аналогів дозволить врахувати їх недоліки та використати переваги при розробці власної гри-аркади.

При оцінюванні аналогів будемо керуватись наступними критеріями: дизайн,простота і зрозумілість, реєстрація, сюжет, багатокористувацький режим.

### 1.2.1 Гра для MicrosoftWindowsTerraria

Інтерфейс гри можна побачити на рисунку 1.1.

Terraria (укр. Террарія, Землеріум) — інді-відеогра розроблена студією Re-Logic. Сенс гри полягає у дослідженні світу, виготовленні предметів, будівництві та битві з різними істотами у випадково створеному 2D-світі. Terraria часто порівнюється з Minecraft, через схожі механіки та графіку. Гра вийшла 16 травня 2011 року. За оцінками, було продано близько 50 000 копій в перший день випуску, і одночасно понад 17 000 гравців були онлайн першого дня. За тиждень було продано 200 000 копій гри, що робить її найбільш проданою грою за тиждень у Steam, таким чином Terraria випередила TheWitcher 2 та Portal 2.

Terraria підтримує багатокористувацький режим. Окрім, власне, гри з іншими гравцями, він не відрізняється від однокористувацької. Кожен гравець може увімкнути в інвентарі режим PvP (значок схрещених мечів та щита). Якщо у двох або декількох гравців одночасно увімкнений режим PvP, то вони можуть битись. Смерть від рук іншого гравця не карається втратою монет, якщо у персонажа «класичний» рівень складності. Приєднатись до іншого серверу можна через запрошення або ввівши IP-адресу.



Рисунок 1.1 – Інтерфейс Terraria

### 1.2.2 Гра «Limbo»

Інтерфейс гри Limbo можна побачити на рисунку 1.2.

Limbo (укр. Чистилище) — відеогра жанру пазл-платформера, перший проект незалежної ігрової датської студії Playdead. Гру було випущено 21 липня 2010 року як ексклюзив для Xbox LiveArcade. Порти цієї гри для Playstation 3 та Microsoft Windows було зроблено після закінчення терміну ексклюзивності (один рік). Версію для OS X було випущено у грудні 2011, порт для Playstation Vita та iOS відповідно у червні та липні 2013.

Гра представлена в чорно-білих тонах, з використанням освітлення, ефектів зернистої плівки та мінімальною кількістю навколишніх звуків, аби створити напружену атмосферу, яка часто асоціюється з жанром жахів. Глави оформлені як занедбані туманні місця, наповнені пастками чи просто покинутими і зламаними, від того небезпечними, механізмами, спорудами. При загибелі тіло хлопчика розриває на шматки, іноді зустрічаються трупи і рої мух, чуються раптові звуки.

Гра не дає чіткого уявлення про сюжет. Хлопчик отямлюється в лісі, де його переслідує велетенський павук. Далі він опиняється на болотах, місці, обжитому вороже налаштованими дітьми, минає міські руїни. Наприкінці хлопчик приходить в місце, наповнене механізмами, де сила тяжіння і все положення простору постійно змінюються. Врешті він опиняється в лісі, подібному на ліс з початку гри, де бачить дівчинку. Він робить кілька невпевнених кроків до неї, а та помічає його і на цьому гра завершується.



Рисунок 1.2 – Інтерфейс Limbo

### 1.2.3 Гра Angry Birds

Інтерфейс гри Limbo можна побачити на рисунку 1.3.

Angry Birds (укр. Сердиті птахи) — відеогра-головоломка, розроблена фінською компанією Rovio Entertainment, в якій гравець з допомогою рогатки повинен стріляти птахами по свинях, розставлених на різних конструкціях. З моменту першого релізу для AppleiOS, згодом вийшли версії гри для інших платформ, включаючи Windows Phone, Microsoft Windows, Mac OS X, Symbian, Android, Maemo 5, Mee Go Harmattan, Bada, Web OS а також ігрові консолі. З 11 травня 2011 року додано версію позначену як для браузера Google Chrome, але яка працює і в інших сучасних браузерах. Станом на 2015 момент гру завантажили понад 100 мільйонів разів тільки у Google Play, що робить її найчастіше скачуваним додатком для мобільних платформ.



Рисунок 1.3 – Інтерфейс гри AngryBirds.

### 1.2.4 Гра Dustforce

Інтерфейс гри Dustforce можна побачити на рисунку 1.4.

Дія Dustforce розгортається в світі, де бруд і пил буквально «заразили» собою живих істот і предмети інтер'єру. Основна локація надає доступ до більш ніж п'ятдесяти рівнів, проте велика частина з них недоступна на момент початку гри і відкривається «ключами», що видаються за успішне проходження відкритих рівнів. Персонажі Гравець може вибрати одного з чотирьох персонажів, всі вони мають подібні вміння, але відрізняються в динаміці.

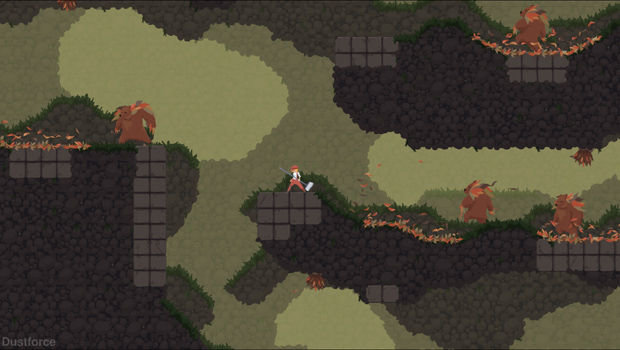


Рисунок 1.4 – Інтерфейс сервісу Dustforce

Таблиця 1.1 – Порівняння ігор.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерій порівняння | Terraria | Limbo | Angry Birds | Dustforce |
| Дизайн | Сучасний, зрозумілий | Зрозумілий, простий | Детальний, пророблений | Сучасний, якісний |
| Мультиплеєр | + | - | - | + |
| Реєстрація | + | + | + | + |
| Простота і зрозумілість | + | + | + | + |
| Вибір персонажу | **-** | **-** | **-** | + |

Всі представлені ігри задовольняють критеріям. Судячи з обсягу проданих копій ігор можна говорити про актуальність обраної тематики курсової роботи.

## 1.3 Постановка задачі

Метою виконання курсової роботи є розробка гри-платформера на мові JavaScript «RunningWild», що буде доступною з звичайного браузера кінцевому користувачу. На сервері повинні зберігатися база даних користувачів, куди будуть поміщатися їхні поставлені рекорди.

Для досягнення поставленої мети необхідно виконати наступні задачі:

* клієнт-серверний RESTful веб-застосунок;
* дані повинні зберігатися у документо-орієнтованій базі даних MongoDB;
* реалізація REST API бекенду за допомогою Express;
* реалізація клієнтського додатку до REST API за допомогою розмітки HTMLта таблиць стилів CSS;
* реалізація авторизації користувача – клієнта з обмеженим доступом до бази даних;
* створити меню гри: нова гра, вибір рівня складності, збереження часу проходження, вихід з гри, таблиця лідерів, Help – опис використання гри;
* при досягненні певного результату необхідно реалізувати перехідна інший рівень складності;
* розробити презентацію результатів виконання курсової роботи.

# 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (SRS – SOFTWARE REQUIREMENT SPECIFICATION)

## 2.1 Загальна інформація щодо веб-застосунку

У данному разі веб-застосунок це гра на мові JavaScript.Її суть полягає в проходженні рівней за якомога швидкий час аби побити попередні рекорди. Управління не повинно бути складним. Стандартне керування за допомогою стрілок, а також пара допоміжних клавіш для рестарту та відкриття меню повинні об’єднувати у собі доступ до повного функціоналу.

### 2.1.1 Огляд продукту

Мета проекту – створення браузерної гри-платформеру «Running Wild». Контекстна діаграма наведена на рисунку 2.1.

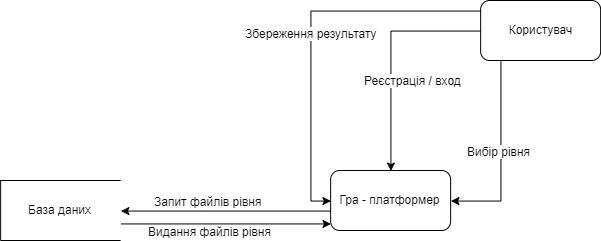


Рисунок 2.1 - Контекстна діаграма для розроблюваної інформаційної системи.

### 2.1.2 Мета

Мета – розробити просту гру де потрібно якомога швидше пройти усі рівні. Створення подібної гри повинно закріпити знання ООП, посприяти кращому засвоєнню мови JavaScript у виконавця курсової, стати відправним пунктом на шляху створення та відлаження відносно середніх та великих проектів.

### 2.1.3Межі

Застосунок має певну кількість рівней. Перший рівень має певну кількість кімнат (одинадцять) та відповідно до розмір у цих кімнат на екрані відображається 16 плиток в довжину та 16 плиток в висоту. Щодо другого рівня, то довжина та висота, що відображається така ж сама, але здійснюється прокрутка мапи адже її довжина перевищує 16 клітинок. Ця кількість може різнитися між рівнями.

### 2.1.4 Масштаб проекту

Застосунок дозволить користувачам запустити рівень та грати через інтернет не встановлюючи при цьому стороннє ПО на свій комп’ютер.

Функції, реалізація яких запланована у цій версії:

* створення, перегляд, зміна і видалення даних у базі даних;
* забезпечення доступу до системи через мережу інтернет для авторизованих користувачів;
* додання нових користувачів до бази даних;
* редагування інформації про користувача.

### 2.1.5 Обмеження реалізації

Увесь код HTML повинен відповідати стандарту HTML 5.0.

Весь код CSS повинен відповідати рівню CSS3.

### 2.1.6 Діаграми потоків даних

Для опису присутньої функціональності були побудовані діаграми потоків даних (DFD – data flow diagram) (рис 2.2).

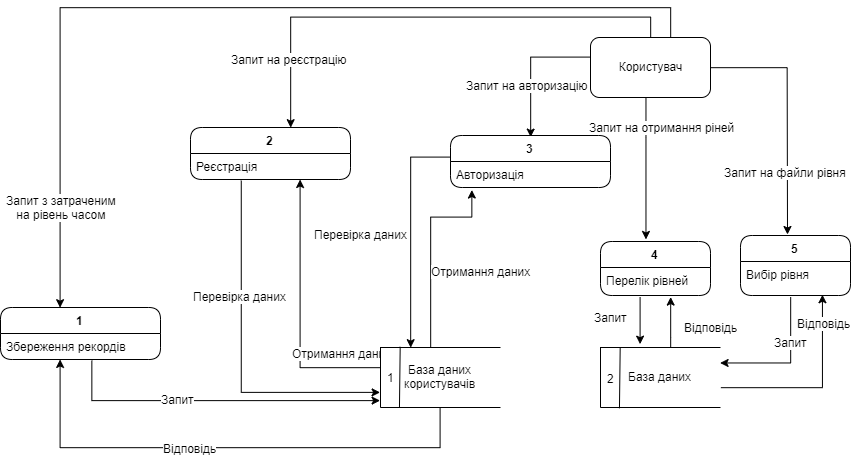


Рисунок 2.2 – Діаграма потоків даних (клієнт)

## 2.2 Вимоги до веб-застосунку

### 2.2.1 Функціональні вимоги

Функціональні вимоги наведені в таблиці 2.1.

Таблиця 2.1 – Опис функціональних вимог до веб-застосунку.

|  |  |
| --- | --- |
| Вимога | Опис |
| Реєстрація.Клієнт | Якщо користувач хоче пограти, то він має бути зареєстрованим. |
| Реєстрація.Клієнт.Дані | Для реєстрації клієнт має ввести необхідні дані в реєстраційну форму. |
| Реєстрація.Клієнт.Завершення | Після того як клієнт натисне кнопку «Зареєструватися», система завантажує доступні рівні для гри |
| Реєстрація.Клієнт.Перевірка | Після подання форми реєстрації клієнтом, система має провірити чи не існує даний клієнт в базі даних. |
| Реєстрація.Клієнт.Підтвердження | При невиявлені даного клієнта в базі даних, додати нового клієнта до бази даних з статусом «Реєстрація пройшла успішно». |

Кінець таблиці 2.1

|  |  |
| --- | --- |
| Реєстрація.Клієнт.Відмова | При виявлені даного клієнта в базі даних, завершити транзакцію зі статусом «Даний клієнт уже зареєстрований». |
| Вхід.Система | Для входу до системи, користувач має ввести логін і пароль в форму входу. |
| Вхід.Система.Підтвердження | При збігу даних завершити транзакцію зі статусом «Вхід виконано успішно». |
| Вхід.Система.Відмова | При розбіжності даних завершити транзакцію зі статусом «Невірні дані» |
| ОбліковийЗапис.Користувач | Клієнт має можливість вийти з власного облікового запису. |
| Рівень.ЗбереженняЧасу | Система повинна дозволяти клієнту, зареєстрованому в системі, розміщувати час затрачений на пройденя кожного рівня гри |
| Рівень.ЗавантаженняФайліс | Якщо користувач виконав вхід до системи та відправив запит на рівень передати потрібні для гри файли (картинки, jsonтощо) |

Діаграму варіантів використання наведено на рисунку 2.3



Рисунок 2.3 – Діаграма варіантів використання

### 2.2.2 Вимоги до зовнішнього інтерфейсу

Застосунок забезпечує доступ до певного функціоналу гри при умові успішної авторизації. Доступ виконується завдяки мережі Інтернет.

В грі повинне бути присутнім основне меню, та меню вибору рівня. В основному меню потрібно розмістити корисну для користувача інформацію, таку як таблиця лідерів, допомога.

### 2.2.3 Нефункціональні вимоги

Користувачі повинні обов'язково реєструватися для можливості виконання операцій перегляду статистики та гри.

Тільки користувачі, позначені як адміністратори повинні мати змогу змінювати інформацію про інших користувачів (не адміністраторів).

Читабельність коду можна розглядати як важливу складову супроводжуваності. Код повинний бути зручним для аналізу, змін та тестування.

Гра не потребує інсталяції, так як вона буде доступна користувачеві за допомогою мережі інтернет і тому буде мати відмінну швидкість доступу до бази даних та завантаження файлів, не буде сильно завантажувати процесор та оперативну пам'ять.

## 2.3 Опис обраних технологій для розробки веб-застосунку

### 2.3.1 Мова гіперрозмітки HTML5

HTML5 є найновішим стандартом HTML. Цей термін представляє дві різні концепції:

* це нова версія мови HTML, з новими елементами, атрибутами і поведінкою;
* це великий стек технологій, котрі надають більшого різноманіття та потужності Веб-сайтам та додаткам. Цей набір інколи називають HTML5 & friends, але часто скорочують до HTML5.

Спроектована, для використання всіма розробниками Відкритого Вебу (Open Web), ця сторінка надає посилання до численних ресурсів про HTML5 технологї, розсортованні в декілька груп, в залежності від їх функцій.

* семантика: дозволяє описати якомога точніше з чого складається ваш контент;
* зв'язок: дозволяє вам взаємодіяти з сервером новим та інноваційним шляхом;
* оффлайн та сховище: надає можливість веб-сторінкам зберігати дані на стороні клієнта локально та оперувати оффлайном ефективніше;
* мультимедіа: робить відео та аудіо першокласними мешканцями Відкритого Вебу;
* 2D/3D графіка та ефекти: дозволяє безліч презентаційних варіантів;
* ефективність та інтеграція: надає більшу оптимізацію швидкості та кращого використання заліза (hardware) комп'ютера;
* доступ до пристрою: дозвляє використання різноманітних пристроїв вводу/виводу;
* стилі: Дозволяє авторам створювати більш складні і витончені теми.

Переваги HTML5:

* легкість створювання програмного код;
* використання елементів article, footer, nav, header та ін.;
* у вікні браузера можна відтворювати Flash (анімації, відеоролики, музику) і для цього не потрібні драйвера і утиліти;
* процес обміну даними з сервером став поліпшеним;
* отримання оповіщень про помилки при роботі сайту;
* Недоліки HTML:
* щоб створювати якісні проекти, доводиться серйозно попрацювати;
* при замовленні сайтів на HTML5, доведеться платити більше;
* деякі фахівці стверджують, що безпека після застосування нової технології стала нижче;
* розробники браузерних платформ намагаються не відставати від сучасних технологій, тим не менш, є деякі програми, в яких не вірно відображаються сайти на HTML5.

В цілому дана технологія є відмінною заміною застарілого HTML.

### 2.3.2 Каскадні стилі CSS3

CSS (англ. Cascading Style Sheets – каскадні таблиці стилів) – формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки.

Специфікація CSS3 - це незаперечне майбутнє в області декоративного оформлення веб-сторінок, і її розробка ще далека від завершення. Більшість модулів все ще продовжує удосконалюватися і модифікуватися, і жоден браузер не підтримує всі модулі. Це означає, що CSS3 зазнає таких самих складностей, як і HTML5. Веб-розробникам потрібно вирішувати, які можливості використовувати, а які ігнорувати, а також яким чином заповнити прогалини в браузерній підтримці.

Специфікація CSS3 не є частиною специфікації HTML5. Ці два стандарти були розроблені окремо один від одного, різними людьми, які працювали в різний час в різних місцях. Але навіть організація W3C закликає веб-розробників використовувати HTML5 та CSS3 разом, як частина однієї нової хвилі сучасного веб-дизайну.

Переваги CSS:

1. Простота самої мови CSS разом з принципом відділення оформлення від змісту дає можливість скоротити час на розробку і підтримку сайту.
2. Є кілька варіантів дизайну сторінки для перегляду на різних пристроях. Наприклад, дизайн на екрані комп'ютера розрахований на одну ширину, і буде повністю виводитися на екран, а на мобільних пристроях він буде підлаштовуватися до розмірів екрану і деякі елементи будуть виключені від показу, також і при друці, буде друкуватися потрібний текст, без зайвого (наприклад, без шапки меню).
3. Зменшується час завантаження сторінок web-сайту за рахунок перенесення правил представлення даних в окремий CSS-файл. Завдяки цьому браузер завантажує тільки структуру документа, а також дані, що зберігаються на сторінці, а представлення цих даних завантажується браузером тільки один раз і може бути закешований, - завдяки цьому зменшується трафік, час завантаження, а також навантаження на сервер.
4. Простота зміни дизайну. Один CSS управляє відображенням безлічі HTML-сторінок. Коли виникає необхідність змінити дизайн сайту, то нема чого правити кожну сторінку. Для подальшої зміни дизайну всього лише потрібно змінити CSS-файл, і як результат, зміна дизайну робиться швидше.
5. CSS надає додаткові можливості форматування, про яких при використанні тільки самих атрибутів навіть і не мріяли.
6. Підвищення сумісності з різними платформами за рахунок використання web-стандартів.

Недоліки CSS:

1. Різне відображення верстки в різних браузерах. Якщо браузери застарілі, то можливо, що одні й ті ж дані CSS по-різному ними інтерпретуються.
2. Необхідність виправляти не тільки один CSS-файл, але і теги HTML. Часто зустрічається на практиці необхідність виправляти не тільки один CSS-файл, але і теги HTML, які пов'язані з селекторами CSS. Іноді це значно збільшує час редагування, а також і тестування.

Отже, CSS – це сильний інструмент і один з основних складових практично будь-якого web-сайту. Без CSS просто неможливо уявити сучасну діяльність web-розробників. Сьогодні CSS – це загальноприйнятий стандарт розробки, який приймається усіма без винятку компаніями-розробниками, що явно показує його значимість і необхідність.

### 2.3.3 Мова програмування JavaScript

JavaScript ("JS" скорочено) це повноцінна динамічна мова програмування, яка, у застосуванні до HTML документу, може надати динамічну інтерактивність на веб-сайтах. Вона була винайдена Бренданом Eйхом, співзасновником проекту Mozilla, the Mozilla Foundation, та Mozilla Corporation.

JavaScript має надзвичайно багато застосувань. Ви можете розпочати з малого: створити "каруселі", галереї зображень, динамічні макети сторінок, відповіді на натиски кнопок, тощо. Із досвідом, ви зможете створювати ігри, 2D та 3D графіку, складні застосунки з використанням баз даних та багато іншого!

JavaScript доволі компактна та гнучка мова. Розробники забезпечили велике розмаїття інструментів, що доповнюють основу мови JavaScript, які відкривають величезну кількість додаткового функціоналу з мінімальними зусиллями. Серед них:

Програмні інтерфейси (APIs) для браузерів — API, які вбудовані у браузери, що надають функціонал на зразок динамічного створення HTML та застосування CSS-стилів, збір та обробка відео-потоків з вебкамери користувача, генерація 3D-графіки та аудіо-семплів.

API третіх осіб, що дозволяють розробникам інтегрувати у власні сайти функціонал інших провайдерів, таких як Twitter або Facebook.

Фреймворки та бібліотеки третіх осіб, які ви можете застосувати до вашого HTML, щоб прискорити створення сайтів та застосунків.

Мова JavaScript використовується для:

* написання сценаріїв веб-сторінок для надання їм інтерактивності;
* створення односторінкових веб-застосунків (ReactJS, AngularJS, Vue.js);
* програмування на стороні сервера (Node.js);
* стаціонарних застосунків (Electron, NW.js);
* мобільних застосунків (ReactNative, Cordova);
* сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe CreativeSuite чи ApacheJMeter);
* всередині PDF-документів тощо.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови С, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

Переваги JavaScript:

1. Жоден сучасний браузер не обходиться без підтримки JavaScript.
2. З використанням написаних на JavaScript плагінів і скриптів впорається навіть не фахівець.
3. Корисні функціональні настройки.
4. Постійно удосконалюється мова – зараз розробляється бета-варіація проекту, JavaScript2.
5. Взаємодія з додатком може здійснюється навіть через текстові редактори – Microsoft Office і Open Office.
6. Перспектива використання мови в процесі навчання програмуванню і інформатиці.

Недоліки JavaScript:

1. Знижений рівень безпеки через повсюдний і вільного доступу до вихідного коду популярних скриптів.
2. Безліч дрібних дратівливих помилок на кожному етапі роботи. Велика частина з них легко виправляється, але їх наявність дозволяє вважати цю мову менш професійною, порівняно з іншими.
3. Повсюдне поширення. Своєрідним недоліком можна вважати той факт, що частина активно використовуваних програм (особливо додатків) перестануть існувати при відсутності мови, оскільки цілком базуються на ньому.

### 2.3.4 Серверна платформа Node.js

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є РаянДал (RyanDahl). Якщо раніше Javascript застосовувався для обробки даних в браузері на сторонні користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їх виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

Node.js має наступні властивості:

* асинхронна однопотокова модель виконання запитів;
* не блокуючий ввід/вивід;
* система модулів CommonJS;
* рушій JavaScriptGoogle V8.

Для керування модулями використовується пакетний менеджер npm (nodepackagemanager).

Платформа Node.js призначена для виконання високопродуктивних мережевих додатків, написаних мовою програмування JavaScript. Платформа окрім роботи із серверними скриптами для веб-запитів, також використовується для створення клієнтських та серверних програм.

В платформі використовується розроблений компанією Google рушій V8.

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів (callback). Як способи мультиплексування з'єднань підтримується epoll, kqueue, /dev/poll і select. Для мультиплексування з'єднань використовується бібліотека libuv, для створення пулу потоків (threadpool) задіяна бібліотека libeio, для виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Всі системні виклики, що спричиняють блокування, виконуються всередині пула потоків і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали (pipe).

За своєю суттю Node.js схожий на фреймворки Perl Any Event, Ruby Event Machine і Python Twisted, але цикл обробки подій (eventloop) у Node.js прихований від розробника і нагадує обробку подій у веб-застосунку, що працює в браузері. При написанні програм для Node.js необхідно враховувати специфіку подієво-орієнтованого програмування.

Переваги Node.js:

* швидкість;
* використання мови JS;
* доступність;
* активна спільнота підтримки;
* асинхронність.

Недоліки Node.js:

* не підходить для завдань, інтенсивних для ЦП;
* неефективний при широкомасштабному застосуванні;
* невідповідність, проблеми в узгодженості;
* погано підтримується реляційними базами даних.

### 2.3.5Фреймворк Express

Фреймворк Express - мінімалістичний і гнучкий фреймворк для Node.js, щозабезпечує широкий набір надійнийх функцій для розробки веб- і мобільнихдодатків. Це полегшує швидкий розвиток веб-додатків на основі Node.js. ФреймворкExpress має ряд переваг:

1. Дозволяє налаштовувати та контролювати HTTP-запити між клієнтською ісерверною частинами додатку;
2. Визначає таблицю маршрутизації, яка використовується для виконаннярізних дій на основі методу HTTP і URL-адреси;
3. Дозволяє динамічно відтворювати HTML-сторінки на основі передачіаргументів шаблонам;
4. Асинхронне та подієве управління – це означає, що сервер на основіфреймворку Express ніколи не чекає на повернення даних API. Серверпереходить до наступного API після виклику, а механізм сповіщень Eventsдопомагає серверу отримувати відповідь від попереднього виклику API.
5. Дуже швидкий – побудований на основі Google Chrome Engine V8,бібліотека фреймворку Express дуже швидко виконує код;
6. Однопотоковий, проте високо маштабований – Express використовує однупотокову модель з циклом подій. Механізм подій допомагає серверуреагувати неблокуючим способом і робить сервер дуже масштабованим, навідміну від традиційних серверів, які створюють обмежені потоки дляобробки запитів. Express використовує одну потокову програму, і одна й тасама програма може надавати послуги набагато більшу кількість запитів,ніж традиційні сервери, такі як HTTP-сервер Apache.
7. Без буферизації - програми Express ніколи не буферують будь-які дані. Всідані у програмі просто передаються від однієї функції до іншої.Фреймворк Express дозволяє розробника з легкістю працювати з POST і GETзапитами, підтримує більшість функціоналу доступного в JavaScript, тому він надзвичайно зручний для комплексної розробки разом із клієнтською частиною наоснові JavaScript.

### 2.3.6 База даних MongoDB

MongoDB – документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

Код MongoDB написаний на мові C++ і поширюється в рамках ліцензії AGPLv3.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин.

Основні можливості MongoDB:

* документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних);
* досить гнучка мова для формування запитів;
* динамічні запити;
* повна підтримка індексів;
* профілювання запитів;
* швидкі оновлення «на місці»;
* ефективне зберігання бінарних даних великих обсягів, наприклад, фото та відео;
* журналювання операцій, що модифікують дані в БД;
* підтримка відмовостійкості і масштабованості: асинхронна реплікація, набір реплік і шардінг;
* може працювати відповідно до парадигми MapReduce.

Недоліком MongoDB є використання пам'яті.

## 2.4 Опис архітектури веб-застосунку

### 2.4.1 Діаграма розміщення

Діаграми розгортання (діаграма розміщення) (рис. 2.4) представляє фізичне розташування системи, показуючи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення.

Вузол (node) – це те, що може містити програмне забезпечення. Вузли бувають двох типів. Пристрій (device) – це фізичне обладнання: комп'ютер або пристрій, пов'язаний з системою. Середовище виконання (execution environment) – це програмне забезпечення, яке саме може включати інше програмне забезпечення, наприклад операційну систему або процес-контейнер.

Вузли можуть містити артефакти (artifacts), які є фізичним уособленням програмного забезпечення; зазвичай це файли. Такими файлами можуть бути виконувані файли (такі як файли .eхе, виконавчі файли, файли DLL, файли JAR, складання або сценарії) або файли даних, конфігураційні файли, HTML-документи і т. д. Перелік артефактів всередині вузла вказує на те, що на даному вузлі артефакт розгортається в систему, яка запускається.

Артефакти можна зображувати у вигляді прямокутників класів або перераховувати їх імена всередині вузла.

Артефакти часто є реалізацією компонентів. Це можна показати, задавши значення-мітки всередині прямокутників артефактів.

Інформаційні шляхи між вузлами представляють обмін інформацією в системі. Можна супроводжувати ці шляхи інформацією про використання інформаційних протоколів.

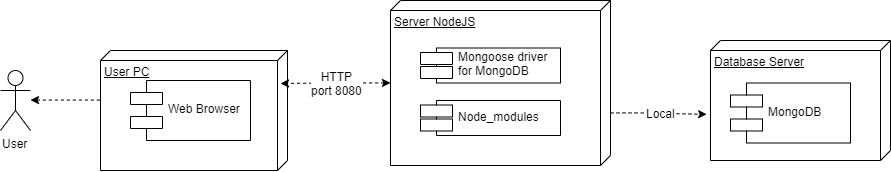


Рисунок 2.4 – Діаграма розміщення.

### 2.4.2 Діаграма компонентів

Діаграма компонентів зображена на рисунку 2.5.

Відносини залежності на діаграмі компонентів є відношення між різними видами компонентів. У цьому проекті такі відносини спостерігаються між виконуючими файлами js.

Так файл main.js має залежність від файлів game.js, display.js, requests.js, controller.js. Зміни в структур і описів цих класів можуть призвести до зміни компонента. Інший приклад – залежність компоненту server.js від компоненту node\_modules де зберігаються усі необхідні модулі для забезпечення роботи NodeJS.

Компонент main.js включає до себе три об'єкти: resize – для вирівнювання площини гри вздовж вікна браузера, render – для прорисовки гри, update – для оновлення стану гравця і гри взагалом.

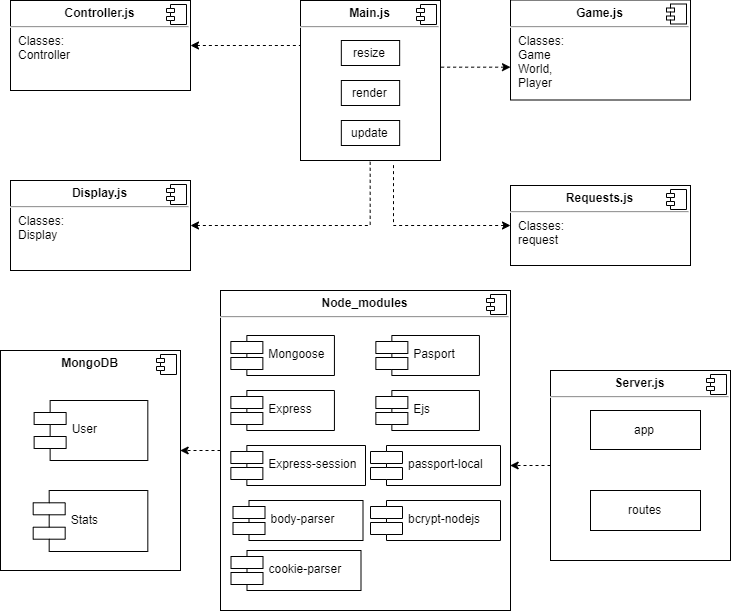


Рисунок 2.5 – Діаграма компонентів.

## 2.5 Модель бази даних

База даних, представлена у форматі схеми, складається з двох схем (рис. 2.6), які мають наступну структуру:

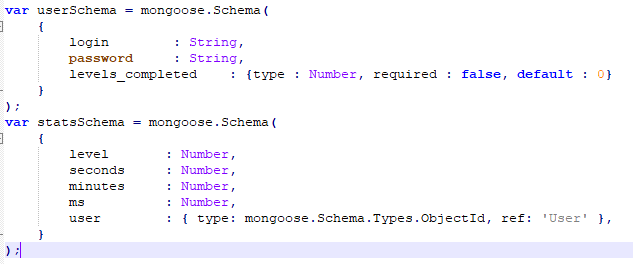


Рисунок 2.6 – Структура схеми бази даних.

Таблиця «statsSchema» призначена для зберігання інформації про затрачений на проходження рівня час. Запис складається з наступних полів, опис яких наведений в таблиці 2.2.

Таблиця 2.2­­ – Опис структури таблиці «statsSchema»

|  |  |  |
| --- | --- | --- |
| Ім’я поля | Тип даних | Опис |
| level | Числовий | Індекс рівня |
| seconds | Числовий | Секунди |
| minutes | Числовий | Хвилини |
| ms | Числовий | Мілісекунди |
| user | ObjectId | Id користувача |

Таблиця «userSchema» призначена для зберігання інформації про зареєстрованих клієнтів. Кожний запис складається з наступних полів, опис яких наведений в таблиці 2.3.

Таблиця 2.3­­ – Опис структури таблиці «userSchema»

|  |  |  |
| --- | --- | --- |
| Ім’я поля | Тип даних | Опис |
| login | Текстовий | Пошта користувача |
| password | Текстовий | Пароль користувача |
| levels\_completed | Числовий | Скільки рівней вже пройдено |

ERмодель бази даних представлена на рисунку 2.7.

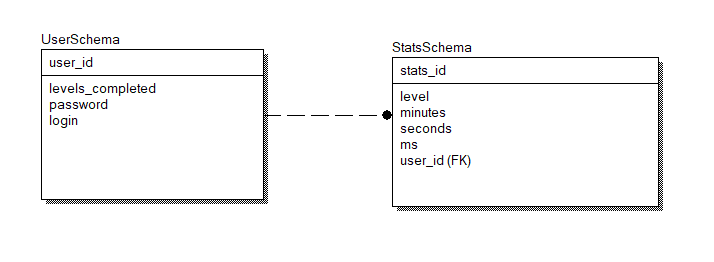


Рисунок 2.7 – ERмодель

# 3 ОПИС ВЕБ-ЗАСТОСУНКУ

## 3.1 Тестування програмного забезпечення

### 3.1.1 Валідація програмного продукту

Валідація програмного забезпечення — це процес визначення відповідності розроблюваного програмного забезпечення між очікуваннями і потребами користувача, вимогам до системи. Валідація є одним із основних етапів тестування програмного забезпечення.

Процес валідації може проводитися самим виконавцем або іншою особою, наприклад, замовником, що здійснює дії з впровадженню і проведенню цього процесу за планом, у якому відбиті елементи і задачі перевірки. При цьому використовуються методи, інструментальні засоби і процедури виконання задач процесу для встановлення відповідності тестових вимог і особливостей використання програмних продуктів проекту на правильність реалізації вимог. Верифікація і валідація полягає в перевірці специфікацій і правильності виконання програм відповідно до заданих вимог і формального опису програми.

Валідація дозволяє встановити здійснимість заданих вимог шляхом їх перегляду, інспекції і оцінки результатів проектування на процесах життєвого циклу для підтвердження того, що здійснюється коректна реалізація вимог, дотримання заданих умов і обмежень до системи. Верифікація і валідація забезпечують перевірку повноти, несуперечності і однозначності специфікації і правильності виконання функцій системи.

Очікування користувача:

* можливість зареєструватися;
* можливість обрати рівень та грати;
* можливість переглянути таблицю лідерів.

Результати проведення валідації:

* було створеноменю з функціями реєстрації/авторізаціїї, яке зберігає дані у базі даних на сервері;
* була створено меню з рівнями після вибору якого завантажуються необхідні файли;
* було створено меню для перегляду таблиці лідерів на кожному рівні.

У результаті валідації програмний продукт є повнофункціональним, що пройшов верифікацію та валідацію.

**3.1.2 Навантажене тестування**

Сучасне програмне забезпечення просто зобов’язано безвідмовно працювати під колосальним навантаженням. Будь-якого роду проблеми, які пов’язані з поганою продуктивністю, можуть стати причиною відмови клієнтів від використання того чи іншого продукту. У зв’язку з цим, проведення якісного навантаженого тестування повинно стати обов’язковим, для забезпечення стабільності роботи продукту.

Apache JMeter − є Java-додатком з відкритим кодом, призначеним для навантажувального тестування не тільки веб-додатків і їх окремих компонентів (скрипти, сервлети, Java об’єкти і ін.), але також FTP-серверів, баз даних (з використанням JDBC) і мережі. Функціональність розширюється за допомогою плагінів.

Проведемо навантажене тестування для програмного продукту, зробимо кілька сесій запитів на сайт з різними параметрами часу та потоків за допомогою Apache Jmeter (рис. 3.1-3.5).

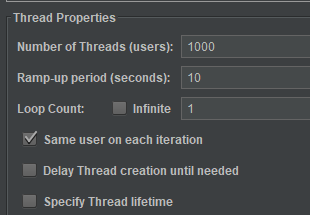




Рисунок 3.1 – Налаштування для тестування.



Рисунок 3.2 – Перший тест, 10 секунд, 1000 потоків.



Рисунок 3.3 – Другий тест, 10 секунд, 50 потоків.

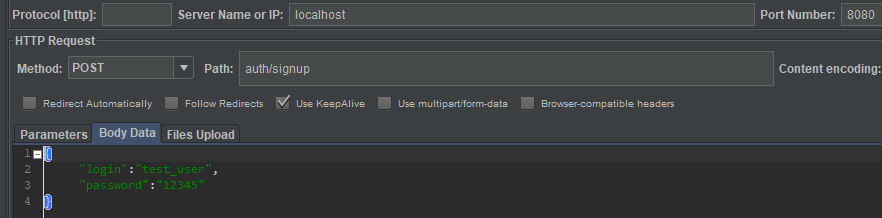
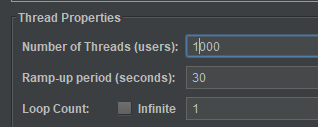


Рисунок 3.4 – Post запит (реєстрація тисячі користувачів за 30 секунд)

Якщо відправляти post запити з додаванням нових результатів до таблиці лідерів отримаємо відповідь зі статусом 400 та повідомлення про обмеження в доступі, бо користувач повинен спершу авторизуватися – рисунок 3.5.

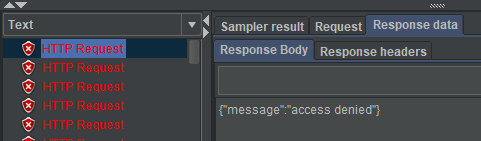
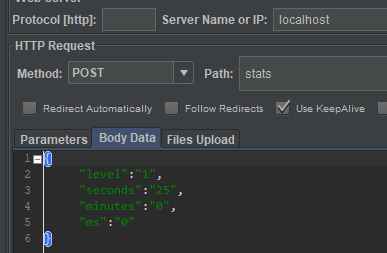
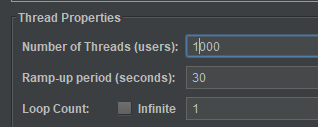


Рисунок 3.5 – Post запит - додавання нового результату до списку лідерів

Як можемо бачити за результатами тестування сервер проходить навантажене тестування на відмінно. З двох тестів не один не вивів сервер з ладу і також не втратив жодного запиту.

## 3.2 Керівництво користувача

Розглянемо усю функціональність сервісу для користувача:

На наступному скріншоті ми бачимо головну сторінку гри. (рис 3.6).



Рисунок 3.6 – Головна сторінка.

Форма реєстрації для створення аккаунта щоб увійти в систему (рис. 3.7).



Рисунок 3.7 – Форма реєстрації.

Форма входу до системи (рис 3.8).



Рисунок 3.8 – Форма входу.

Меню гри відкривається після нажаття клавіши Esc – рисунок 3.9.



Рисунок 3.9 – Меню гри

**3.3 Використання веб-застосунку**

Розглянемо процедуру входу в аккаунт та безпосередньо гри для користувача. По-перше користувач має ввійти в існуючий аккаунт щоб отримати список усіх доступних рівней – рисунок 3.10.



Рисунок 3.10 – Доступні рівні

Мета першого рівня – знайти вихід з лабіринту. Після вибору рівня починається ігровий процес – рисунок 3.11.

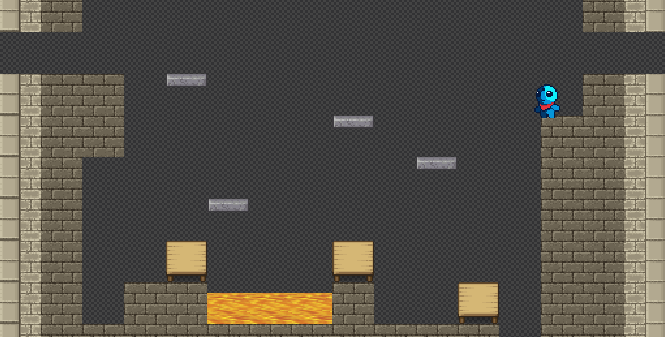


Рисунок 3.11 – Кадр з гри

Під час гри гравцю попадаються підказки, які він може відкривати – рисунок 3.12.



Рисунок 3.12– Підказка

Щоб увійти у деякі кімнати потрібно збирати ключі, що їх відкривають – рисунок 3.13.



Рисунок 3.13 – Гравець знайшов ключ

По всій карті можна знайти лавові озера. Коли гравець потрапляє в одне з таких озер – гра та таймер зупиняється – рисунок 3.14. Щоб відновити ігровий процес – потрібно нажати клавішу R – restart, або обновити сторінку і почати гру знову.

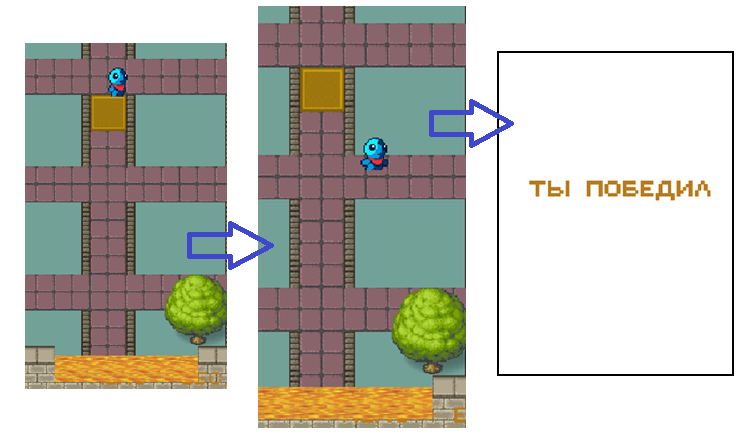


Рисунок 3.14 – Гравець потрапляє в лаву

Коли гравець досягає останніх дверей (рисунок 3.15) – гра завершується і час зберігається у базі. Побачити свій результат можна у таблиці лідерів – рисунок 3.16.

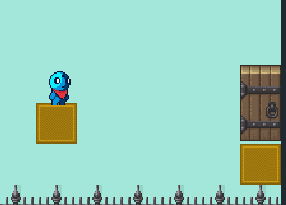


Рисунок 3.15 – Останні двері

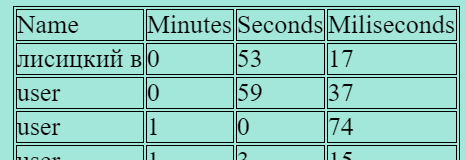


Рисунок 3.16 – Таблиця лідерів

Після проходження рівня (досягнення певної точки на карті) гра закінчується, а час гравця зберігається у базі.

# ВИСНОВКИ

В ході виконання курсової роботи було виконано наступні етапи.

1. Було проаналізовано предметну область індивідуального завдання курсової роботи.
2. Розроблено модель даних у виді схем у документо-орієнтованій базі даних.
3. Проведено первинну розробку серверної архітектури додатку.
4. Проведено проектування архітектури інформаційної системи з використанням технології «клієнт-сервер».
5. Проведено програмну реалізацію та налагодження розробленої інформаційної системи.

Отже, в ході виконання курсової роботи, послідовно застосовуючи принципи об'єктно-орієнтованого аналізу, документо-орієнтованого моделювання даних, а також використовуючи програмний засіб СУБД MongoDB, середу розробки Visual Studio Code для програмування веб-додатку на серверній платформі NodeJS в операційній системі MS Windows 10, було розроблено, реалізовано і задокументовано гру-платформер на мові програмування Java Sript.

# СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. How to Write a Java Script Platformer [Електронний ресурс]. - Режим доступу: <https://www.youtube.com/playlist?list=PLcN6MkgfgN4CpMUgWEM5d70ANMWgcmBp8> (21.12.2019)
2. Debug your web site on a mobile device [Електронний ресурс]. - Режим доступу: <https://blog.campvanilla.com/debug-website-on-mobile-device-5c27c8809d39> (21.12.2019)
3. MDNWebDocs [Електронний ресурс]. - Режим доступу: <https://developer.mozilla.org/bm/docs/Web/JavaScript> (21.12.2019)
4. Node.js Authentication using Passport.js [Електронний ресурс]. - Режим доступу: <https://medium.com/@bmshamsnahid/node-js-authentication-using-passport-js-78386be1f518> (21.12.2019)
5. Opengameart.org [Електронний ресурс]. - Режим доступу: <https://opengameart.org/>(21.12.2019)