Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Інформаційна технологія віртуального ігрового автомата»

Виконав: студент 2 курсу,
групи 1КН-18м
спеціальності 122 «Комп'ютерні науки»
Хворостюк Є. В.
Керівник: к.т.н., ст. викл. Озеранський В.С.
Рецензент: к.т.н., доцент Войтко В.В.

Вінниця - 2019 року

РЕФЕРАТ

Дана магістерська кваліфікаційна робота, присвячена розробці інформаційної технології віртуального ігрового автомата. Технологія призначена для удосконалення процесів генерації виграшних комбінацій.

В ході роботи проведено аналіз предметної області віртуальних ігрових автоматів та програмні засоби генерації різних ігрових ситуацій для них. Спроектовано програмні засоби для можливості управління генерацією виграшних комбінацій з використанням нечіткої логіки. Визначено структурну організацію основних модулів та сервісів технології. Здійснено програмну реалізацію інформаційної технології віртуального ігрового автомата на мові програмування PHP та MYSQL для роботи з базою даних. Результати роботи можна використовувати в маркетинговій та розважальній сферах, за у мови проходження відповідного ліцензування.

ABSTRACT

Master's qualification work is devoted to the development of information technology of a virtual slot machine. The technology is designed to improve the processes of generating winning combinations.

During the work was analyzed the subject area of virtual slot machines and software tools for generating different game situations for them. Software is designed to control the generation of winning combinations using fuzzy logic. The structural organization of the main modules and services of technology is determined. There were done the software implementation of information technology of virtual gaming machine in programming language PHP and MYSQL to work with the database. The results of the work can be used in marketing and entertainment, if will be given the relevant licensing.

ЗМІСТ

ВСТУП 6
1 ОБРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВІРТУАЛЬНОГО ІГРОВОГО АВТОМАТА 10
1.1 Аналіз предметної області та програмних засобів для створення віртуальних ігрових ситуацій 10
1.2 Характеристика та аналіз аналогів 12
1.3 Алгоритми роботи віртуального ігрового автомата 15
1.4 Постановка задачі 17
1.5 Висновок 18
2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРТМІВ РОБОТИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВІРТУАЛЬНОГО ІГРОВОГО АВТОМАТА 19
2.1 Розробка структури інформаційної технології 19
2.2 Сервіс-орієнтовна архітектура 21
2.3 Розробка бази даних 25
2.4 Використання методів нечіткої логіки для генерації виграшних комбінацій 29
2.5 Розробка моделі та алгоритму генерації виграшних комбінацій для віртуального ігрового автомата 33
2.6 Висновок 36
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВІРТУАЛЬНОГО ІГРОВОГО АВТОМАТА 37
3.1 Обґрунтування вибору програмних засобів 37
3.2 Налаштування середовища розробки 40
3.3 Розробка структури модулів інформаційної технології 42
3.4 Програмна реалізація основних модулів інформаційної технології віртуального ігрового автомата 44
3.5 Тестування програмного забезпечення та аналіз результатів 48

3.6 Висновок 51

4 ЕКОНОМІЧНА ЧАСТИНА 52

4.1 Оцінювання комерційного потенціалу розробки 52

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи. 53

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки. 57

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності 58

4.5 Висновок 61

ВИСНОВКИ 63

ПЕРЕЛІК ПОСИЛАНЬ 65

Додаток А Інструкція користувача ..... **Error! Bookmark not defined.**

Додаток Б Лістинг програми ..... **Error! Bookmark not defined.**

Додаток В Графічна частина ..... **Error! Bookmark not defined.**

ВСТУП

Актуальність теми дослідження. На сьогоднішній день ринок розваг є одним з найприбутковіших. З моменту початку інформаційно-технічної революції світ стрімко рухається в майбутнє, створюючи все більш досконалі комп'ютерні системи, щоб полегшити життя людини, а так само зайняти його дозвілля. З появою персональних комп'ютерів, з кожним роком, їх роль в житті людей постійно зростає. Комп'ютер став незамінним помічником не тільки в сфері економічних розрахунків, а й є потужним центром розваг. Сфери впливу кіно і літератури відчувають потужний тиск з боку інтерактивних розваг, пристроїв доповненої реальності і інших, впроваджуваних завдяки розвитку комп'ютерних технологій. Сучасна людина іноді навіть не замислюється, що і його смартфон є аналогічним мобільним персональним комп'ютером. З ростом ролі комп'ютерів в житті людини, комп'ютерна техніка суттєво впливає і на модель поведінки людини. За останніми дослідження середній вік гравця комп'ютерних ігор становить 30 років і вище.

В сучасному світі віртуальні казино зайняли дуже стійкі позиції в сфері ігрового ринку. Кількість гравців, що віддають перевагу подібному формату, неймовірно швидко зростає. Це говорить про актуальність і доречність створення ігрових онлайн систем. Ігрові автомати в віртуальних казино адаптовані під такий формат і представлені в неймовірно величезній кількості. Велика популярність прийшла після офіційної заборони ігрового бізнесу, після чого багато закладів перейшли в мережу Інтернет.

Є кілька факторів, які є основними у формуванні популярності та актуальності сучасних онлайн ігрових автоматів. Фінансова система. Гравцям доступні різні способи, щоб поповнювати свій внутрішній рахунок або взагалі грати за віртуальні кошти і такі ж способи виведення зароблених коштів. Безпека. Всі дані шифруються і ретельно приховуються. Також ніхто

не буде спостерігати за ігровим процесом гравця, що може йому перешкодити або дезорієнтувати. Законодавчі обмеження. Реальні казино заборонені, але на онлайн додатки заборона не поширюється. Дані фактори можна вважати поштовхом до розвитку онлайн ігрових автоматів.

У лінійці ігрових автоматів «однорукий бандит» займає почесне місце ветерана класики азартного жанру. Все почалося більше ста років тому з винаходу першої механічної покерної машини Сітмана і Пітта, а слідом за нею з'явився і перший прототип трибарабанних автоматів з механічним важелем запуску маховиків. Перші ігрові апарати були влаштовані доволі тривіально – три барабана, які запускалися механічним важелем і зупиняючись, видавали горизонтальні комбінації з 3 символів, при збігу яких гравцеві присуджували приз. Весь процес повністю спирався на везіння, але оскільки випадання виграшної комбінації відбувалося не часто. Так автомат придбав своє прізвисько «однорукого бандита». Механіка поступово замінювалося електроприводами, що давало більше гарантій на чесність процесу.

Зв’язок роботи з науковими програмами, планами, темами. Магістрська кваліфікаційна робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 122 – «Інформаційна технологія віртуального ігрового автомата» та плану навчально-методичної та наукової роботи кафедри

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей програмного забезпечення для віртуальних ігрових автоматів.

Об’єктом дослідження є процеси генерації різних ігрових ситуацій для віртуальних ігрових автоматів.

Предметом дослідження є інформаційна технологія віртуальних ігрових автоматів та програмні засоби генерації різних ігрових ситуацій для віртуальних ігрових автоматів.

Для досягнення поставленої мети необхідно виконати такі задачі:

– аналіз предметної області та програм аналогів створення віртуальних ігрових ситуацій,
– розробка структури інформаційної технології віртуального ігрового автомата,
– розробка бази даних та алгоритмів роботи віртуального ігрового автомата,
– здійснення програмної реалізації віртуального ігрового автомата,
– економічне обґрунтування доцільності розробки віртульного ігрового автомата.

Методи дослідження. У роботі використані такі методи наукових досліджень: аналіз інформаційних систем віртуального ігрового автомата, нечітка логіка для реалізації алгоритму створення виграшних комбінацій, сервіс-орієнтована архітектура проектування для розробки й керування функціональних модулів системи, об’єктно-орієнтоване програмування для отримання подальших результатів роботи програми.

Наукова новизна. Удосконалено процес генерації виграшних комбінацій з використанням нечіткої логіки. Удосконалено інформаційну технологію віртуального ігрового автомата, що відрізняється від відомих можливістю управління генерацією виграшних комбінацій з використанням нечіткої логіки, що дало змогу керувати потенційним виграшем користувача, не показуючи цього.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено структурну схему віртуального ігрового автомата.
2. Розроблено алгоритм генерації виграшних комбінацій для віртуального ігрового автомата з допомогою нечіткої логіки;
3. Розроблено програмний засіб віртуального ігрового автомата на основі сервіс-орієнтованої архітектури.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним

застосуванням математичних методів під час доведення наукових положень, чітким та лаконічним виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У роботах, опублікованих у співавторстві, автору належать такі результати: [1] – розроблено структурну схему віртуального ігрового автомата та проаналізовано підходи до її організації.

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2020)» (м. Вінниця, Україна, 2019 р.).

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано 1 тезу доповіді на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2020)» [1].

1 ОБРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВІРТУАЛЬНОГО ІГРОВОГО АВТОМАТА

1.1 Аналіз предметної області та програмних засобів для створення віртуальних ігрових ситуацій

Ігровий автомат, або слот (англ. slot machine, slot), «однорукий бандит» – різновид ігрових автоматів для азартної гри. Ігри для таких автоматів також називаються «слотами» [2], у кожної з таких ігор є своя власна назва. У них можна грати не тільки на слот-машині, але й на комп'ютері, встановивши спеціальне програмне забезпечення або ж у режимі онлайн. Слоти користуються величезною популярністю в казино, як «реальних», так і в інтернеті. Slot англійською означає «щілина», «проріз» (для монети в автоматі). Раніше у всіх слот-машин була щілина, куди потрібно було опустити монету. Є такі «монетні» слоти і тепер. Оскільки, граючи в інтернеті, монету не опустиш, гравець спочатку повинен покласти гроші на свій рахунок у віртуальному казино, де він грає (якщо він грає на реальні гроші, а не режимі, без ставок зі справжніми грошима).

Слоти складаються з трьох і більше коліс (барабанів) [20], що обертаються. Вони обертаються у вертикальній площині, паралельно один одному. На барабани нанесені різні картинки-символи (фрукти, гроші, тварини тощо).

Кинувши в проріз одну або декілька монет (або іншим чином зробивши ставку), гравець натискає кнопку Spin, і барабани автомата починають обертатися. Після їх зупинки символи на них шикуються в випадковому порядку. Кожен такий символ має свою цінність. Якщо при цьому в один ряд вишикується комбінація з декількох однакових символів, то така комбінація є виграшною, і в такому випадку гравцеві виплачується виграш згідно з

таблицею. Мета гри – скласти комбінацію однакових символів найбільшої цінності.

Перший прототип грального автомату було винайдено 1887 року в Англії. У 1905 році в США створюється відомий Liberty Bell, і з цього часу процес доступності гри збільшується, що одразу відзначається на збільшенні кількості гравців по всьому світу.

Розробка гри має низку послідовних етапів [2]: розробка алгоритмів, програмного коду та бази даних, розробка контенту (малюнків, моделей, музики та анімацій). Їм передує проектування — генерування дизайнером ідей щодо майбутньої гри, вибір жанру, тематики, розробка сценарію та образів персонажів з оточенням, розробка правил та ігрових механік. Менеджер координує дії різних людей, залучених до розробки, складає план їхньої роботи, встановлює терміни її виконання, планує витрати. Індустрія ігор включає у себе багато людей з різними професіями та ролями: програмістів, які відповідають за технічні можливості гри, художників, моделювальників та аніматорів, які створюють графічний контент, композиторів та звукорежисерів, які створюють звукове оформлення та музичний супровід, який нерідко видається окремим накладом. За успішне завершення роботи над проектом відповідають продюсери.

Існує чимала кількість інструментальних засобів [3] для створення ігрових рішень, а також для полегшення їх розробки, покращення графіки, оптимізації фізики тощо. Проте різні засоби мають спецефічні сфери використання залежно від типу, жанру та платформи розгортання гри. Більшість інструментальних засобів для розробки ігор можна поділити на 3 группи:

– фреймворк – програмна платформа, яка визначає структуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту [17]. Фреймворк відрізняється від поняття бібліотеки тим, що бібліотека може бути використана в програмному продукті просто як набір підпрограм

близькою функціональності, не впливаючи на архітектуру програмного продукту і не накладаючи на неї ніяких обмежень. У той час як фреймворк диктує правила побудови архітектури додатку, задаючи на початковому етапі розробки, поведінки за умовчанням, каркас, який потрібно буде розширювати і змінювати відповідно до зазначених вимог;

– ігровий рушій - центральний програмний компонент комп'ютерних та відеоігор або інших інтерактивних додатків з графікою, оброблюваної в реальному часі [2]. Він забезпечує основні технології, спрощує розробку і часто дає грі можливість запускатися на декількох платформах, таких як ігрові консолі та настільні операційні системи, наприклад, GNU / Linux, Mac OS X і Microsoft Windows або бути розміщеними на серверах під керуванням цих систем і бути доступними для користувача в мережі інтернет;
– конструктор ігор - програма, яка об'єднує в собі ігровий рушій і інтегроване середовище розробки, і, як правило, включає в себе редактор рівнів. Такі програми значно спрощують процес розробки ігор, роблячи його доступним любителям-непрограмістів, і можуть бути використані в початковому навчанні програмуванню.

1.2 Характеристика та аналіз аналогів

В даний момент різноманітність ігрових автоматів досягло свого піку. Але незважаючи на це в Україні найвідомішими залишаються:

– класичні слоти [20] – хороший вибір для початківця, оскільки вважаються простими і інтуїтивно зрозумілими. Вважається що представники цього виду мають високий пейаут.
– відео-слоти – вони не мають обмежень в дизайні і кількість ліній часом перевалює за тисячу. Нові технології дозволяють інтегрувати високоякісну графіку, яка скрашує ігровий процес.

– фруктові машини – за своєю механікою схожі на класичні автомати онлайн, з однією явною відмінністю, яка полягає в присутності бонусних систем, а також вимагають більшої участі з боку гравця.
– відео-покер [20] – практично не відрізняється від онлайн покеру в якому потрібно збирати комбінації з карт, тільки тут в якості вашого основного противника буде комп'ютер.

При такому розмаїтті апаратів, вибір багатьох гравців часто падає на сучасні онлайн ігрові автомати Україна з наявністю мультиплікаторів і прогресивних джекпотів від розробників Microgaming, NetEnt і Endorphina у яких відсоток віддачі досягає 98%.

Також ігрові автомати класифікуються за кількістю барабанів:

– 3 барабана. Перевагою і недоліком цих автоматів є простота гри і найменша сума можливого виграшу. Відмінний варіант для новачка або людини з обмеженим бюджетом. Яскравим представником виступають класичні слоти.
– 5 барабанів [18]. Найпоширеніший вид, оскільки має різноманітне кількість ліній виплат, а значить і ймовірність перемогти вище. Вони мають високоякісну HD графіком, різноманітну бонусну систему і включають в себе різні міні ігри на віртуальні гроші.
– 7 барабанів. Крім двох додаткових барабанів і великої кількості ліній виграшу, нічим не відрізняються від своїх попередників.

До останнього часу в Україні топовими вважалися ігрові автомати онлайн на 3 барабана, так як вони позбавлені непотрібних надлишків і досить прості у використанні, але останнім часом на перше місце виходять більш іноваційні емулятори з 3d графікою.

Популярні бонусні символи:

– wild – символ при випаданні, якого може замінити відсутні зображення для виграшної комбінації.
– scatter – символ, при випаданні якого можна збільшити виграш або запуститися додаткова міні гра.

Дані бонуси характерні для слот-машин новітнього типу і в разі вдалого збору комбінації гравцеві завжди обіцяють велику перемогу [18]. Розглянемо популярні аналоги віртуальних ігрових автоматів: TwinSpin, Pyramid.

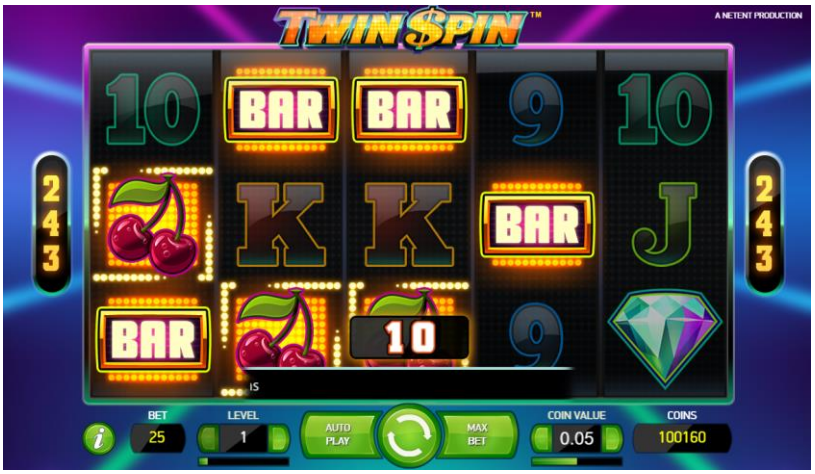

Рисунок 1.1 – Приклад ігрового вікна віртального ігрового автомата TwinSpin

Даний ігровий автомат є одним з перших та найпопулярніших у світі. Це приклад класичного ігрового слота. Кожний раунд починається з ідентичних суміжних подвійних барабанів, які пов'язані один з одним - звідси і назва Twin Spin. Під час обертання подвійні барабани можуть розширюватися і перетворюватися в потрійні, четверні або навіть п'ятирічні барабани. Унікальна функція синхронізації і зв'язування барабанів, яка з'являється на кожному обертанні. Відповідні символи в будь-якій позиції на трьох або більше суміжних барабанах, починаючи з крайнього лівого барабана і закінчуючи самим правим барабаном, є виграшною комбінацією. Виплачується тільки найдовша підходяща комбінація на символ. За один раунд може бути лише три виграшних комбінації.

Рисунок 1.2 – Приклад ігрового вікна віртального ігрового автомата Pyramid

Віртуальний ігровий автомат Pyramid є дещо складнішим за попередній аналог, так як тут присутній символ Wild який заміняє будь який інший елемент на ігровій площині. Даний символ може з’явитись лише на 2,3,4 барабанах і подовжити виграшну комбінацю. Але все ж, так як і у попередньому автоматі зарахуються лише співпадіння символів починаючи з крайнього лівого барабана і закінчуючи самим правим барабаном і можливі лише три виграшні комбінації.

1.3 Алгоритми роботи віртуального ігрового автомата

Всі ігрові автомати онлайн і оффлайн працюють на основі деякого алгоритму, RNG – random number generator. Цей термін можна перевести як генератор випадкових або рандомних чисел/значень. Це складний алгоритм, який лежить в основі абсолютно кожного слота. І при ліцензуванні онлайн казино і автоматів, перевіряється саме RNG [3]. Його задача – це генерація псевдовипадкових значень. Саме цей алгоритм робить гру непередбачуваною та справедливою. Рандомні значення відповідають за певні комбінації та символи. Ні один гравець, ні адміністрація онлайн казино не може прогнозувати результати нового повороту барабану.

Навіть незважаючи на те, що з моменту появи перших ігрових апаратів прогрес пішов вперед, принцип і надійність генератора випадкових чисел

залишаються незмінними. Механізм регулярно допрацьовується, враховуючи особливості сучасного ігрового програмного забезпечення. У створенні генератора випадкових чисел брали участь математики, програмісти і навіть психологи, і існували подібні генератори з давніх часів - їх надійність перевірена часом. Той генератор, який використовується сьогодні - це розробка професора Массачусетського технологічного університету, яка представляє собою 128-бітний алгоритм md5. При цьому використовується вона не тільки в онлайн-слотах, а й в охоронних системах і персональних комп'ютерах.

Віддача слота, RTP [5] гри, теоретичний відсоток повернення - це все один і той же параметр. RTP - return to player, означає повернення гравцеві. Термін є офіційним і використовується в наземних і онлайн казино для всіх типів азартних ігор. Зазвичай return to player виражається у відсотках і обчислюється за простою формулою: RTP = розмір виграшу / розмір ставок * 100%. Всі ігрові автомати в реальних казино і онлайн слоти в віртуальних казино зобов'язані вказувати точний показник віддачі [3]. Це одне з головних вимог, що пред'являється розробнику при ліцензуванні. При першій перевірці регулятор обов'язково перевіряє відповідність заявленого розробником значення RTP, і видає ліцензію грі, тільки якщо цей параметр вказаний вірно. Через час автомат проходить повторну перевірку. Мало знати відсоток їх віддачі, важливо ще й розуміти, як це працює на практиці. Так, RTP = 96,34% означає, що гравець поверне 96,34% всіх зроблених на автоматі ставок, але на довгій дистанції. Не вірно вважати, що якщо за 10 або 100 обертань витратити 100 монет, то обов'язково буде виграш в розмірі 96,34 монети. Мінімальна довжина дистанції становить 1000 спинив, а регулятори і самі розробники перевіряють віддачу на кілька мільйонів обертань. Протилежним показником до RTP є House Advantage - перевага казино. Воно також обчислюється за простою формулою: House Advantage = 100% - RTP.

Значення RTP слота закладається в програмне забезпечення, як ігрових автоматів в реальному казино, так і віртуальних слотів. Для реальних

автоматів створюють спеціальні чіпи, які містять програму всієї роботи гри, алгоритм генератора випадкових чисел і параметри RTP. Чіп програмується один раз і проходить перевірку. Якщо перевірка успішна, чіп монтується в автомат і пломбується. Що стосується віртуальних казино ігор, відсоток віддачі також є частиною програмного коду, створюваного девелопером. Коли розробка завершена, заявлений RTP [5] перевіряється регулятором. Тільки після успішного її проходження видається ліцензія, слот надходить у продаж і розміщується на ліцензованих ігрових майданчиках.

1.4 Постановка задачі

У даній роботі необхідно реалізувати віртуальний ігровий автомат, який буде працювати по за допомогою взаємодії користувача з веб додатком ігрового автомата. У вікні додатку для користувача будуть доступні такі дії:

- Вибір величини ставки
- Вибір величини значення ігрової монети
- Вибір автоматичного або ручного режиму прокрутки барабанів.

Після встановлення всіх налаштування користувач натискає на кнопку ”Spin”, після чого відправляється запит на сервер, подальша обробка вхідних даних раунду за допомогою алгоритмів системи, відповідь сервера з результатами раунду. Результати раунду будуються на основі інформації про інші раунди користувача за поточний день збережені у базі даних, а також за допомогою алгоритму випадкових чисел.

Ігровий автомат повинен працювати як Web-сервіс (Web API) через стандартні HTTP/HTTPS – запити та використовувати аутентифікацію по зашифрованому id користувача. Web-сервіс повинен підтримувати такий формат обміну даними як JSON. Веб-сервіс повинен працювати під управлінням операційної системи Windows XP/7/8/10/Server, Linux, які є найбільш поширеними серед операційних систем. Ємність ОЗУ залежить від навантаження на веб-сервер, але повинно бути не меншою за 512 Мбайт.

Надійність функціонування програми і її функціональну стійкість визначають вхідні дані, які передаються в тілі HTTP-запиту, тому необхідно передбачити перевірку вхідних даних на правильність.

1.5 Висновок

Під час роботи над розділом проаналізовано предметну область, розглянуто: основні поняття інформаційної технології віртуального ігрового автомата; процеси та етапи, які будуть використовуватись при організації технології віртуального ігрового автомата.

Проаналізувавши об’єкт дослідження визначено: ключові вимоги до інформаційної технології, вхідні та вихідні дані для програмних засобів та вимоги до програмно-апаратного забезпечення системи.

У результаті аналізу систем аналогів визначено їхні характеристики, основні критерії, переваги та недоліки, область використання, розглянуто основі проблеми, які виникають при розробці подібних систем та технології за допомогою яких, можна їх усунути.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРТМІВ РОБОТИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВІРТУАЛЬНОГО ІГРОВОГО АВТОМАТА

2.1 Розробка структури інформаційної технології

Усі веб-додатки в інтернеті використовують архітектуру клієнт-сервер. Вихідні файли додатку зберігаються на веб-сервері, який обробляє запити, що приходять через мережу інтернет від численних користувачів. На стороні клієнта веб-додаток працює в браузері, наприклад в Google Chrome або Mozilla Firefox. Користувальницький інтерфейс програми передається на клієнтську машину у вигляді сторінок написаних мовою HTML (Hypertext Markup Language), де браузер інтерпретує та відображає їх. Клієнтські комп'ютери забезпечують інтерфейс, який дозволяє користувачеві комп'ютера запитувати сервери та відображати результати, які сервер повертає. Веб-сервери чекають надходження запитів від клієнтів, а потім відповідають на них. Обмін даними здійснюється на апаратному рівні на основі протоколу TCP/IP і протоколу більш високого логічного рівня HTTP або захищеному HTTPS [4]. Хоча протокол HTTP був розроблений ще у 1990 році, він стрімко розвивався. Існує дві версії протоколу найголовнішою їх відмінністю стала можливість тільки одного підключення до серверу для завантаження сайту, що дало величезний приріс у швидкодії. Мультиплексування – дозволяється відправка декількох запитів одночасно, з тим самим з'єднанням. Раніше, за допомогою HTTP / 1.1, для оброки кожного нового запиту доводилось чекати завершення інших переказів. Пріоритезація – запитам призначаються рівні залежностей, які сервер може використовувати для швидшого надання ресурсів з більш високим пріоритетом.

В ідеалі, сервер надає стандартний прозорий інтерфейс для клієнтів, щоб клієнти не потребували інфромації про специфіку системи (наприклад,

апаратного та програмного забезпечення), яка надає послугу [4]. Приклад роботи моделі клієнт-сервер та переваги першою версії протоколу HTTP над другою зображено на рисунку 2.1.

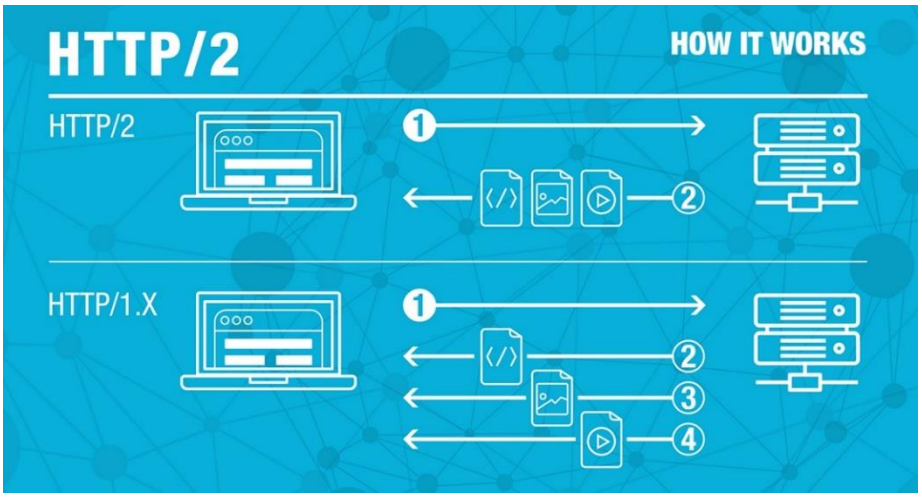

Рисунок 2.1 – Робота протоколу HTTP під управлінням клієнт-серверної моделі

Клієнти часто розташовуються на персональних комп'ютерах, а сервери знаходяться в інших місцях мережі, як правило, на більш потужних машинах. Ця обчислювальна модель особливо ефективна, коли клієнти та сервер мають окремі завдання, які вони виконують за замовчуванням [11].

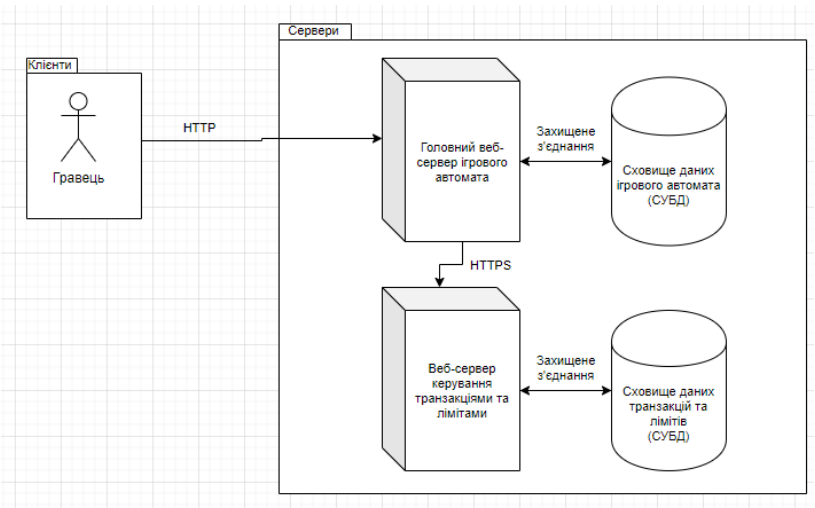

Рисунок 2.2 – Структурна схема віртуального ігрового автомата

В даній інформаційній технології віртуального ігрового автомата серверною частиною буде керувати Slim Framework з деякими модулями Laravel [12]. Тобто усі частини додатку, такі як веб-сервіси, управління базами даних, маршрутизація, кешування, аутентифікація, локалізація, управління групами користувачів, будуть підпорядковані фреймворку (рисунок 2.2). А генерація HTML документа та усі інші операції зв'язані з клієнтом будуть оброблятись на його стороні опираючись на відповіді сервера у JSON форматі. Таким чином знизиться навантаження на сервер і буде отримано приріст у швидкості роботи додатку.

2.2 Сервіс-орієнтовна архітектура

Сервіс-орієнтована архітектура (далі – СОА) – це концепція проектування, розробки й керування функціональних модулів (сервісів), кожний з яких доступний через мережу і здатний виконувати певні дії [14]. СОА створює комунікаційне середовище для модулів, що реалізують прикладну бізнес-логіку. Інформація про модулі публікується в такій формі, що їх використання не вимагає знань про використані в них рішення і технології. Розробнику не потрібно знати, як працює програма, необхідно лише розуміти, які вхідні і вихідні дані потрібні, і як викликаються ці програми для виконання. Сервіс-орієнтовані обчислення – це обчислювальна парадигма, яка використовує сервіси як фундаментальні елементи для розробки застосувань. Вони базуються на СОА і забезпечують виконання операцій керування сервісами. Розробка таких систем – це процес пошуку, дослідження та композиції сервісів, що задовольняють вимогам користувача [14]. Можливість композиції сервісів часто розглядають як одну з основних переваг їх використання. Вона полягає у знаходженні набору елементарних сервісів, необхідних для реалізації функцій, використовуваних у запиті користувача, і визначення порядку їх виконання. SOA – це архітектурний

шаблон програмного забезпечення, модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабко пов’язаних замінних компонентів, оснащених стандартизованими інтерфейсами для взаємодії за стандартизованими протоколами. Технологія SOA [14] для керування бізнес-процесами є великим кроком вперед з точки зору підвищення ефективності розробки систем; за значимістю її можна порівняти із створенням у кінці 50-х років компіляторів мови високого рівня. Цей підхід дозволяє спростити використання Web-сервісів з будь-якого місця розташування і їх виконання на основі бізнес-правил. SOA підштовхує до використання альтернативних технологій і підходів (таких як обмін повідомленнями) для побудови застосувань за допомогою зв’язування сервісів, а не за допомогою написання нового програмного коду. У цьому випадку, при належному проектуванні, застосування повідомлень дозволяє компаніям своєчасно реагувати на зміну ринкових умов – настроювати процес обміну повідомленнями, а не розробляти нові програми. SOA – це термін, який з’явився для опису виконуваних компонентів – таких як Web-сервіси – які можуть викликатися іншими програмами, які виступають як клієнти або споживачі цих сервісів. У найзагальнішому вигляді SOA припускає наявність трьох основних учасників:

– постачальника сервісу
– споживача сервісу
– реєстру сервісів.

Взаємодія учасників виглядає досить просто: постачальник сервісу реєструє свої сервіси в реєстрі, а споживач звертається до реєстру із запитом [14]. Для використання сервісу необхідно дотримуватися угоди про інтерфейс для звернення до сервісу – інтерфейс повинен не залежати від платформи. SOA реалізує масштабованість сервісів – можливість додавання сервісів, а також їх модернізацію. Постачальник сервісу і його споживач виявляються непов’язаними – вони спілкуються за допомогою повідомлень. Оскільки інтерфейс повинен не залежати від платформи, то і технологія,

використовувана для визначення повідомлень, також повинна не залежати від платформи. Тому, як правило, повідомлення є XML документами, які відповідають XML-схемі.

Концепція Web-сервісів виникла наприкінці 90-х років XX ст. і стала галузевим стандартом у сфері ІКТ [14]. Стандарти Web-сервісів розроблені такими компаніями, як IBM, Microsoft, Ariba, Sun Microsystems, SAP за підтримки Консорціуму W3C. У межах W3C було створено робочу групу Web Services Architecture Working Group, яка опублікувала глосарій термінів у сфері Web-сервісів. Web-сервіси використовують XML для обміну даними між застосуваннями, незалежно від використання операційної системи, апаратної платформи і розробника. Web-сервіс – це набір логічно пов'язаних функцій, які можуть бути програмно викликані через мережу Internet. Web-сервіс – це програма, що ідентифікується через URI, інтерфейс якої може бути подано у вигляді мови XML. Web-сервіси – це реалізована програмними засобами система для підтримки міжкомп'ютерної взаємодії телекомунікаційних мереж, що підтримується такими стандартами: SOAP (Simple Object Access Protocol) – протокол обміну повідомленнями; WSDL – мова опису програмних інтерфейсів Web-сервісів; UDDI (Universal Description, Discovery and Integration) – класифікатор Web-сервісів. Інформація про те, які функції пропонує конкретний Webсервіс, міститься в його описі – WSDL-документі. Інші системи взаємодіють з Web-послугами, використовуючи повідомлення у стандарті за протоколом SOAP, передані з використанням HTTP і XML і в поєднанні з іншими Web-стандартами [14]. Для пошуку Web-сервісів використовують спеціальні реєстри, що підтримують UDDI. Є два основні методи публікації Webсервісів для користувачів – UDDI і DISCO. UDDI – це централізований структурований сервіс реєстру, а DISCO пропонує вільну форму механізму пошуку через браузер. Реєстр UDDI – центральне сховище для специфікацій та інформації про підприємства, включаючи послуги, які компанії надають шляхом Internet.

SOAP Web-сервіси стають доступними через протоколи HTTP GET, HTTP POST, HTTP SOAP.

SOAP – стандарт передачі повідомлень через Internet, розроблений фірмою Microsoft для віддаленого виклику процедур (RPC, Remote Procйdure Call) через протокол HTTP [15]. Він дає змогу передавати інформацію мережею у форматі XML. Можуть використовуватися будь-яка мережа, будь-який протокол передачі даних, довільна інформація, різні обчислювальні пристрої (зокрема мобільні). Специфікація SOAP визначає XML – «конверт» для передачі повідомлень, метод для кодування програмних структур даних у форматі XML, а також засоби зв'язку через протокол HTTP. WSDL WSDL – заснований на XML стандарт опису того, як користуватися сервісом, запропонований Консорціумом W3C.

REST – підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів [15]. Був описаний і популяризований 2000 року Роєм Філдінгом, одним із творців протоколу HTTP. В основі REST закладено принципи функціонування інтернету і, зокрема, можливості HTTP. Філдінг розробив REST паралельно з HTTP 1.1 базуючись на попередньому протоколі HTTP 1.0. Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (HTML, XML, JSON). Будь-який REST протокол (HTTP в тому числі) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформації про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабовність системи і дозволяє їй еволюціонувати з новими вимогами [15]. При підході REST кількість методів і складність протоколу суворо обмежені, що призводить до того, що кількість окремих ресурсів має бути великою.

Опис Web-сервісу мовою WSDL містить технічні деталі, необхідні для інтеграції Web-сервісу у застосування (формат повідомлень, операції). На сьогодні WSDL підтримують продукт від Microsoft – SOAP Toolkit 2.0

(WSDL Generator) і продукт від IBM – WSDL Toolkit. Мова опису Web-сервісів (Web Services Description Language (WSDL)) визначає синтаксис того, як Web-сервіс може бути викликаний.

UDDI Стандарт [14]. UDDI надає механізм виявлення Web-сервісів. UDDI формує бізнес-реєстр (UDDI Business Registry), в якому провайдери Web-послуг можуть реєструвати свої послуги, а розробники – відшуковувати необхідні їм сервіси. Компанії реєструють себе в Business Registry, який є базою даних загального користування. UDDI дає можливість описувати, інтегрувати і публікувати сервіси. UDDI сам є спеціалізованим Web-сервісом, що дає змогу користувачам і застосуванням знаходити необхідні їм сервіси. Інтелектуальний пошук та автоматичне компонування Webсервісів можуть бути здійснені за допомогою можливостей семантичного опису Web-сервісів, запропонованих у OWLS.

2.3 Розробка бази даних

База даних – упорядкований набір логічно взаємопов'язаних даних, що використовуються спільно, та призначені для задоволення інформаційних потреб користувачів. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Базою даних можна вважати будь-який впорядкований набір даних. В даному випадку інформацію про склади, товари, категорії до яких вони можуть відноситись також інформацію про клієнтів та історію їх замовлень.

MySQL – вільна система керування реляційними базами даних [7]. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. Зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сайтів, оскільки має чудову підтримку з боку різноманітних мов програмування. MySQL надає багатий набір функціональних можливостей, які підтримують безпечне середовище

для зберігання, обслуговування і отримання даних, а також характеризується великою швидкістю, стійкістю і простотою у використанні, була розроблена для підвищення швидкодії обробки великих баз даних.

Для ефективного функціонування інформаційної технології віртуального ігрового автомтата необхідне сховище інформації про користувачів, транзакції раундів в яких вони брали участь та значення ліміту для контролю випадіння виграшних комбінацій. Також потрібно зберігати інформацію про сесії гравців, звичайні та бонусні раунди (спіни), занчення ставок та виграшів для контролю виграшних комбінацій та ведення звітності у системі. Для реалізації сервісу технічної підтримки у системі, а також для вирішенню можливих конфліктів і відтворення результатів раундів необхідно зберігати повну інформацію про відповіді сервера клієнту гри. На основі цих потреб було створено реляційну базу даних з такими таблицями: users, sessions, limits, transactions, spins, freespins, bets, wins, games.

Нормалізацією називається формальна процедура, в ході якої атрибути даних групуються в таблиці, а таблиці групуються в БД.

Задачами нормалізації є:

– вилучення з таблиць інформації, що повторюється;
– створення структури, в якій передбачена можливість майбутніх змін;
– створення структури, в якій вплив структурних змін на додатки, що використовують дані цієї БД, зведено до мінімуму.

Для першої нормальної форми [7] потрібно, щоб таблиця була двовимірною і не містила груп, що повторюються. У таких таблиць є тільки дві характеристики – довжина (кількість записів або рядків) та ширина (кількість полів або стовпців). Вона не повинна містити комірок, що включають кілька значень. Для того, щоб в одній комірці містилося кілька величин, необхідно ввести третій вимір – глибину, за допомогою якої можна зберігати в одній комірці одразу декілька значень.

Для другої нормальної форми потрібно, щоб дані у всіх не ключових стовпцях повністю залежали від первинного ключа і кожного елемента

(стовпця) первинного ключа, якщо ключ є складеним. Під повною залежністю розуміються те, що значення в кожному не ключовому стовпці однозначно визначається значенням первинного ключа. Якщо одне з полів не залежить від величини первинного ключа, то необхідно включити в ключ доповняльні таблиці. Перед перевіркою на відповідність другій нормальній формі таблиця повинна бути приведена до першої нормальної форми.

Для третьої нормальної форми [7] потрібно, щоб всі неключові стовпці таблиці не тільки залежали від первинного ключа таблиці, але були незалежними один від одного, тобто, щоб були відсутні транзитивні функціональні залежності між стовпцями. Для цього потрібно, щоб таблиці були попередньо приведені до першої, другої нормальної форми.

Найпростішим відношенням між таблицями є відношення “один-до-одного”. В такому відношенні одному запису однієї таблиці відповідає тільки один запис у іншій. Таблиці, що зв’язані відношенням “один-до-одного” можна об’єднати в одну таблицю, яка складається з полів обох таблиць. Наприклад, це може бути потрібним для того, щоб скоротити час перегляду полів, що містять певний набір даних. В деяких випадках необхідно керувати доступом до частин таблиць, які містять важливі або конфіденційні дані.

Відношення “один-до-багатьох” [7] зв’язує один запис першої таблиці з декількома записами другої за допомогою первинного ключа базової таблиці і відповідного йому зовнішнього ключа зв’язаної таблиці. Зовнішній ключ таблиці, що містить велику кількість відношень, може входити до складеного первинного ключа, але він є зовнішнім по відношенню до базової таблиці. Відношення “один-до-багатьох” використовується найбільш часто.

На рисунку 2.3 продемонстровані зв’язки таблиць “spins”, “bets”, “wins”, “freespins”. В даному випадку використані такі типи зв’язків: “Один-до-одного” – таблиця “spins” зв’язана з “bets” за допомогою індексного поля sid, тобто одному раунду може належати одна ставка. Таким же чином таблиця “spins” зв‘язана із таблицею “wins”, тобто кожний раунд може мати

лише один виграш. “Багато-до-одного” – з іншого боку таблиця “freespins” зв’язана з “spins” по тому ж полю sid, що означає приналежність одного або більше бонусних раундів до однієї звичайного. Таким же зв’язком, “Багато-до-одного”, зв’язані транзакції з раундами, тобто кожний раунд може мати дві транзакції, які розділяються за типом, у випадку якщо це ставка або виграш.

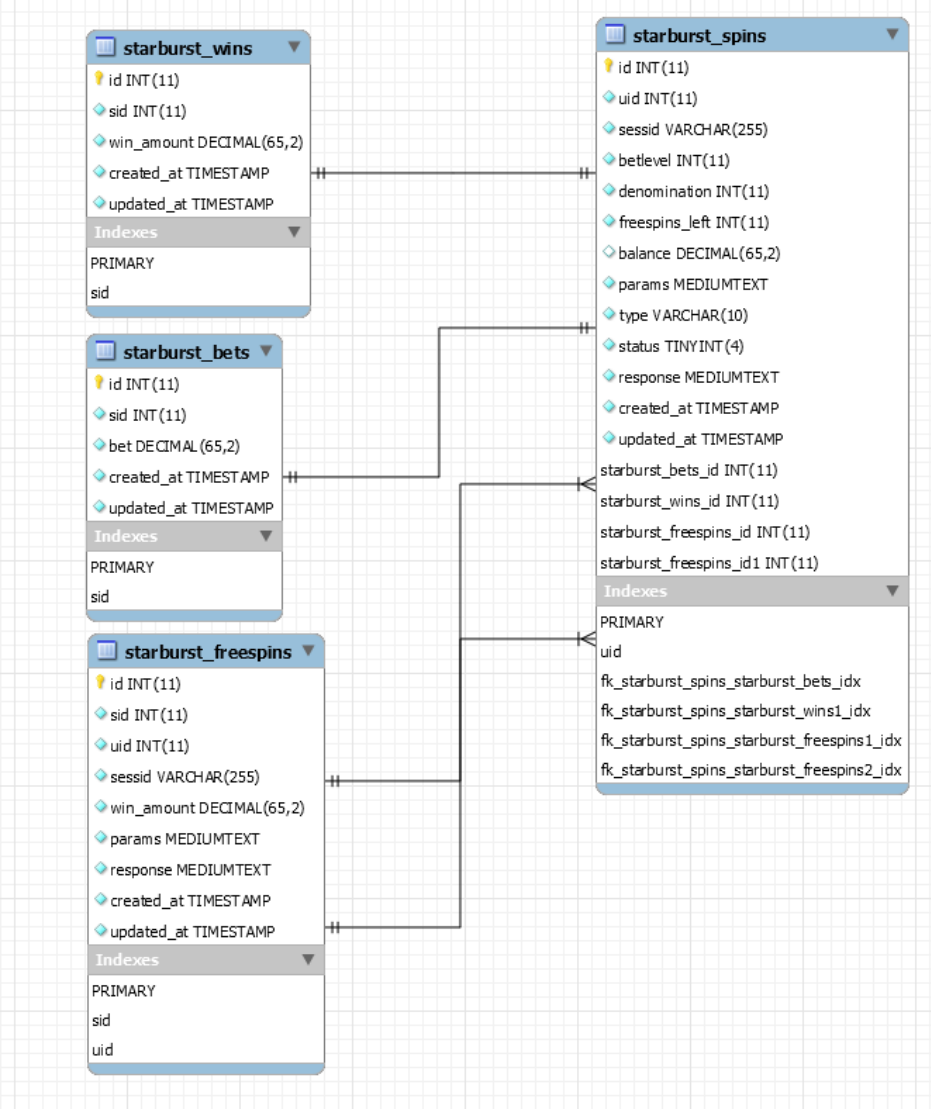

Рисунок 2.3 – Схема зв’язків таблиць “spins”, “bets”, “wins”, “freespins”

Відношення “багато-до-одного” протилежне відношенню “один-до-багатьох”. Якщо вибір відношення “багато-до-одного” або “один-до-багатьох” не має великої ролі, то відношення між таблицями називається

рефлексивним [7]. Відношення “багато-до-одного” є відображенням відношення “один-до-багатьох”.

Таким чином за допомогою різних типів зв’язків була побудована база даних для інформаційної технологіх віртуального ігрового автомата. Завдяки даній концепції бази було розроблено багато потрібних функцій у системі. Наприклад зберігання транзакцій кожного з раундів гравця, тобто збереження фінансового сліду у системі, для подальшого створення алгоритму для контролю за виграшними комбінаціями, а також ведення звітності для аналізу окупності гри та відслідковування за користувачами. Також дуже важливим рішенням було створення таблиць “spins”, “freespins”, “bets” та “wins” для зберігання інформації про ставки та виграші у раундах. Розділення об’єкта раунда на чотири таблиці дало змогу надійно зберігати інформацію та швидко доступатись до неї у разі необхідності не втрачаючи цілісності даних.

2.4 Використання методів нечіткої логіки для генерації виграшних комбінацій

Прийняття рішень у проблемно-орієнтованих інформаційних системах та системах керування здійснюється в умовах апріорної невизначеності, обумовленої неточністю або неповнотою вхідних даних, стохастичною природою зовнішніх впливів, відсутністю адекватної математичної моделі функціонування, нечіткістю мети, людським фактором та ін.

Невизначеність системи призводить до зростання ризиків від прийняття неефективних рішень, результатом чого можуть бути негативні економічні, технічні та соціальні наслідки [16]. Невизначеності у системах прийняття рішень компенсують за допомогою різноманітних методів штучного інтелекту. Для ефективного прийняття рішень при невизначеності умов функціонування системи застосовують методи на основі правил нечіткої логіки. Такі методи ґрунтуються на нечітких множинах і

використовують лінгвістичні величини і висловлювання для опису стратегій прийняття рішень.

Методи нечітких множин [19] особливо корисні за відсутності точної математичної моделі функціонування системи. Теорія нечітких множин дає можливість застосувати для прийняття рішень неточні та суб'єктивні експертні знання про предметну область без формалізації їх у вигляді традиційних математичних моделей.

З використанням теорії нечітких множин вирішуються питання узгодження суперечливих критеріїв прийняття рішень, створення логічних регуляторів систем. Нечіткі множини дають змогу застосовувати лінгвістичний опис складних процесів, встановлювати нечіткі відношення між поняттями, прогнозувати поведінку системи, формувати множину альтернативних дій, виконувати формальний опис нечітких правил прийняття рішень.

Методи теорії нечітких множин є зручним засобом проектування інтерфейсів у людиномашинних системах. На основі нечіткого логічного виведення [19] будуються системи керування, подання знань, підтримки прийняття рішень, апроксимації, структурної та параметричної ідентифікації, розпізнавання образів, оптимізації. Нечітка логіка знаходить застосування у побутовій електроніці, діагностиці, різноманітних експертних системах. Нечіткі експертні системи для підтримки прийняття рішень знаходять широке застосування у військовій справі, медицині та економіці. З їх допомогою здійснюють бізнес-прогнозування, оцінювання ризиків та прибутковості інвестиційних проектів. На основі нечіткої логіки досліджують глобальні політичні рішення та моделюють кризові ситуації.

Важливим застосуванням теорії нечітких множин є контролери нечіткої логіки, які використовуються у різноманітних системах керування, зокрема у побутових приладах. Замість математичної моделі для опису системи такі контролери використовують інтегровані знання експертів, які за структурою

подання наближаються до розмовної мови і описуються за допомогою лінгвістичних змінних та нечітких множин.

Загальна структура fuzzi-контролера [16] містить у своєму складі такі складові: блок фазифікації; база знань; блок рішень; блок дефазифікації.

Блок фазифікації перетворює чіткі величини, виміряні на виході об'єкта керування, на нечіткі величини, описані лінгвістичними змінними у базі знань.

Блок рішень використовує нечіткі умовні (if – then) правила, закладені у базі знань, для перетворення нечітких вхідних даних на необхідні керуючі впливи, що мають також нечіткий характер.

Блок дефазифікації перетворює нечіткі дані з виходу блоку рішень на чітку величину, яка подається на виконавчий пристрій для керування об'єктом.

З огляду на широке поширення систем штучного інтелекту з інтегрованою нечіткою логікою, розроблення ефективних систем прийняття рішень на їх основі є актуальною науково-практичною проблемою.

Перспектива застосування нечіткої логіки [19] полягає у розробленні гібридних методів штучного інтелекту, до яких можна віднести нечіткі штучні нейронні мережі, адаптивне поповнення баз нечітких правил, підтримка нечітких запитів до баз даних, побудова нечітких когнітивних карт, нечіткі графи, нечіткі мережі Петрі, нечіткі дерева прийняття рішень, нечітка кластеризація.

У прикладних дослідженнях з проблем керування, в технічних науках, медицині, соціології, економіці, психології тощо широко використовуються експертні оцінки, які формулюються у термінах природної мови [16]. З цією метою використовуються нечіткі лінгвістичні змінні. Нечітка лінгвістична змінна X (наприклад, тиск) задається набором термів, які позначають якісні ознаки станів системи (наприклад, „низький”, „середній”, „високий”). Нехай X визначається m лінгвістичними термами:

$$\tilde{X} = \{A_j \mid j = 1..m\}. \tag{2.1}$$

Тоді лінгвістична змінна X є нечітким образом носія X. Кожен із термів є нечіткою множиною.

Функції належності однієї лінгвістичної змінної X визначаються в одному вимірювальному просторі X:

Операції нечіткої логіки. Над нечіткими множинами можна визначити логічні операції, аналогічні операціям звичайної (однозначної) логіки, наприклад, NOT, AND, OR Множиною значень функції істинності звичайної логіки є {0,1}. Функції істинності нечіткої логіки набувають значення на відрізку [0,1] . У нечіткій логіці формула (висловлювання) може бути істинною зі значенням [0,1].

Функції та структура нечіткої системи [19]. Нехай нечітка система здійснює вибір варіантів рішень на основі залежності вихідної величини від декількох вхідних величин. Припустимо, що математична модель залежності виходу від входів відсутня і замість неї використовується база експертних правил у вигляді нечітких висловлювань "if then − " у термінах лінгвістичних змінних та нечітких множин. Тоді функціональність нечіткої системи прийняття рішень визначається такими кроками:

– перетворення чітких вхідних змінних на нечіткі, тобто визначення ступеня відповідності входів кожній із нечітких множин;
– обчислення правил на основі використання нечітких операторів та застосування імплікації для отримання вихідних значень правил;
– агрегування нечітких виходів правил у загальне вихідне значення;
– перетворення нечіткого виходу правил на чітке значення.

Система побудована за схемою багатошарової штучної нейромережі, яка складається з вхідного, двох прихованих та вихідного шару. Перший шар зображає входи системи, другий шар – нечіткі лінгвістичні змінні, третій шар – правила над нечіткими змінними, четвертий шар – виходи правил. Ваги

усіх шарів, крім останнього, дорівнюють 1. Ваги зв’язків між шаром правил та вихідним шаром визначаються алгоритмом навчання.в

Нечітке логічне виведення [19]. Нечіткі вхідні значення системи перетворюються на вихідні на основі правил нечіткої логіки, що характерно для експертних систем прийняття рішень. На практиці для нечіткого виведення використовується максимінна композиція, а нечітка імплікація реалізується знаходженням мінімуму функцій належності. Для імітації роботи експертної системи за схемою імплікації використовується множина нечітких продукційних правил, кожне з яких будується у вигляді умовного оператора: if логічний вираз then оператор, де логічний вираз – висловлювання, побудоване на основі базових логічних операцій над нечіткими величинами; оператор – результуюче рішення.

Використання нечітких умовних правил є природнім для подання знань експертами і спрощує їх машинне опрацювання. Загалом до правила можуть входити усі можливі комбінації лінгвістичних термів для усіх вхідних змінних, об’єднаних логічними операціями.

2.5 Розробка моделі та алгоритму генерації виграшних комбінацій для віртуального ігрового автомата

Для створення виграшних комбінацій було розроблено алгоритм, робота якого складається з декілького частин поділених на мікро-сервіси.

Перш за все відбувається збір та підсумування інфомації з бази даних про ставки та виграші користувача за поточний день (рисунок 2.4). Далі за відповідною формулою (2.2) розраховується допустимий ліміт на виграш користувача.

$$Max = T_b * (1 - Limit / 100) - T_w \quad (2.2)$$

де T_b – сума ставок гравця за поточний день;

T_w – сума виграшів гравця за поточний день;

Limit – закріплене значення – коефіцієнт заробітку гри.

На рисунку 2.3 зображено роботу алгоритму збору інформації про ставки та виграші користувача. Він представляє собою sql запит у базу даних, у якому використовуються додаткові методи Sum() для підсумування значення поля, та перевірка isNull().

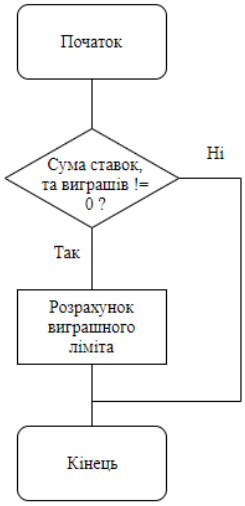

Рисунок 2.4 – Загальна схема алгоритму розрахунку виграшного ліміта віртуального ігрового автомата

Наступним кроком, на основі результатів розрахунку виграшного ліміта будується ігрове поле та шукаються співпадіння за встановленими правилами, у випадку успіху, виграш оцінюється за трьома категоріями (низький, середній, високий) і у випадку задоволення розміру ліміта гравцеві дозволяється виграти або весь цикл запусається знову до моменту випадіння “вірної” комбінації.

На рисунку 2.5 зображено повний алгоритм генерації ігрового поля та результату раунду, тобто виграшних комбінацій.

1. Спочатку веб-сервіс гри робить HTTP запит до окремого сервісу, який відповідає за керування транзакціями користувачів та лімітами гри.
2. Запускаєтсья цикл for в якому не залежно від значення ліміта відпрацьовує метод створення ігрового поля (createReels), який працює

на основі методу випадкових чисел та відсоткових співвідношень в залежності від ваги кожного символа на полі.

3. У випадку знаходження співпадінь за допомогою методу match(), розраховується майбутній виграш гравця, який в подальшому оцінюється, як низький, середній чи високий. Низький та середній виграші пропускаються системою, у випадку високого виграшу гравця, цикл створення ігрового поля та генерації виграшу повториться, але уже з параметром, для зниження відсотків генерації символів із великою вагою і далі порівняється з доступним лімітом. Якщо ліміт перевищено основний цикл алгоритму буде повторюватись до тих пір поки не буде задовонеий ліміт. Таким чином ми контролюємо ігровий процес і заробіток гравця.

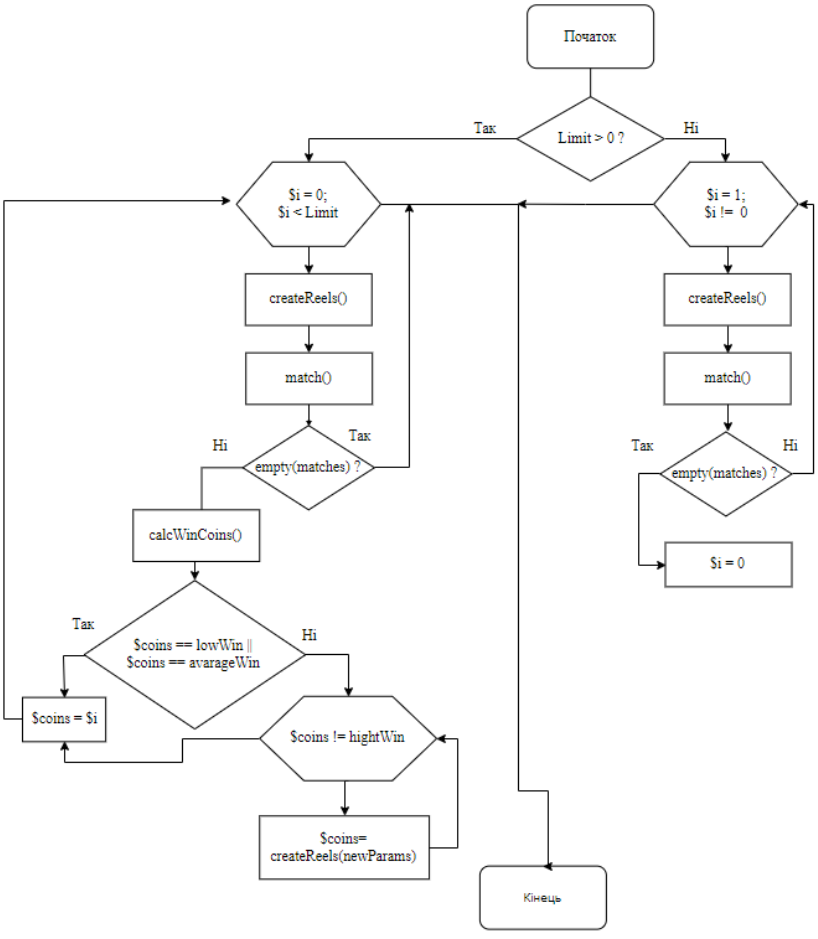

Рисунок 2.5 – Загальна схема алгоритму генерації виграшної комбінації віртуального ігрового автомата

4. Якщо ліміт рівняється нулю, основний цикл алгоритму буде воторюватиць поки не згенерується програшна комбінація. У випадку програшу, навіть з доступним лімітом для виграшу, гра завершає раунд з від’ємним результатом.

2.6 Висновок

У даному розділі визначено, якими будуть особливості розробленого програмного продукту, а саме використання сервіс-орієнтованї архітектури, використання нечіткої логіки та теорії приняття рішень у розрбці алгоритмів системи. Сформульовано структуру програмного продукту та розроблено основні алгоритми для функціонування віртуального ігрового автомата.

Розглянуто і проаналізовано основні підходи сервіс-орієнтованої архітектури проектування основних функціональних модулів. Проаналізовано переваги та не доліки найпопулярніших підходів проектування Web-сервісів, обрано та застосовано REST підхід, перевагами якого є масштабовність системи, що дозволяє їй еволюціонувати з новими вимогами, а також не має прив’язки до одного якось одного формату передачі данних (HTML, XML, JSON).

Розглянуто і проаналізовано методи реалізації процесу генерації ігрового поля та виграшних комбінацій з допомогою нечіткої логіки, що дає змогу управління потенційним виграшем користувача.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВІРТУАЛЬНОГО ІГРОВОГО АВТОМАТА

3.1 Обґрунтування вибору програмних засобів

Веб-програмування стрімко розвивається, і перед серверними розробниками постає питання вибору між такими потужними мовами як Java, C, Perl і сучасними веб-орієнтованими мовами, такими як PHP, JavaScript, Ruby, Go.

PHP – був створений в 1994 році Расмусом Лердфордом. Він створив програмну оболонку (інтерпретатор), яка встановлюється в якості модуля для веб-сервера Apache або Nginx [8]. Спочатку розроблявся як препроцесор гіпертекстових сторінок, тому PHP може бути легко інтегрований в HTML код, проте, такий підхід зараз не є доброю практикою, але все ж для новачків такий підхід був очевидний. Це сприяло популярності мови, тому 80% сайтів в мережі інтернет написані на PHP.

Програма, яка переводить код, написаний на одній мові програмування, на інший називається транслятором. Компілятор – це теж транслятор. Він переводить код, написаний на мові високого рівня, в машинний код. В результаті процесу компіляції створюється двійковий виконуваний файл, який вже можна запускати без компілятора. Інтерпретатор – це зовсім інша категорія. Інтерпретатор не переводить в код, а виконує його [6]. Інтерпретатор аналізує код програми і виконує кожну його рядок. Кожен раз при виконанні такого коду, необхідно користуватися інтерпретатором. За продуктивністю інтерпретатори значно поступаються компіляторам, оскільки двійковий код виконується набагато швидше. Зате інтерпретатори дозволяють повністю контролювати програму під час її виконання. Що стосується PHP, то він не є ні компілятором, ні інтерпретатором. PHP є чимось середнім, між компілятором і інтерпретатором. На вхід PHP

подається сценарій. Він переводить його, перевіряючи синтаксис, в спеціальний байт-код (внутрішнє уявлення). Потім PHP виконує байт-код, а не код самої програми, при цьому він не створює виконуваний файл. Байт-код значно компактніше звичайного коду програми, тому його легше і швидше інтерпретувати, тобто виконувати. Переваги PHP:

- працює на всіх найбільш поширених операційних системах (Windows, MacOS, Linux), є свої версії пакетів розробки на PHP, а це значить, що можна створювати веб-сайти на будь-якій з цих операційних систем;
- PHP може працювати в зв'язці з різними веб-серверами: Apache, Nginx, IIS;
- простота і легкість освоєння. Як правило, вже маючи невеликий досвід в програмуванні на PHP, можна створювати прості веб-сайти;
- синтаксис PHP схожий на синтаксис мови C, тому, знаючи C або одну з мов з сіподібним синтаксисом, буде простіше опанувати PHP [17];
- PHP підтримує роботу з великою кількістю систем баз даних MySQL, MSSQL, Oracle, Postgre, MariaDB, MongoDB тощо;
- постійний розвиток PHP, виходять все нові версії, які несуть нові функції, адаптуючи мову програмування до нових викликів. І, як правило, перейти на нову версію не складає труднощів.

На даний момент веб-сайти розробляються на двох версіях PHP: більш застарілій 5.6 та відносно новій 7 версії мови. Основні відмінності нововведення такі:

- рефакторинг основних структур даних;
- покращена конвенція виклику віртуальної машини;
- новий API парсинга параметрів;
- новий диспетчер пам'яті;
- численні поліпшення виконавця віртуальної машини;
- істотно зменшено використання пам'яті;
- поліпшено функції __call () і __callStatic () [17];
- покращена конкатенація рядків;

– покращено пошук символів в рядках;

Фреймворки – це програмні продукти, які спрощують створення і підтримку технічно складних або навантажених проектів [17]. Фреймворк, як правило, містить тільки базові програмні модулі, а всі специфічні для проекту компоненти реалізуються розробником на їх основі. Тим самим досягається не тільки висока швидкість розробки, а й велика продуктивність і надійність рішень. Фреймворк відрізняється від бібліотеки тим, що бібліотека може бути використана в програмному продукті просто як набір підсистем близькою функціональності, не впливаючи на архітектуру основного програмного продукту і не накладаючи на неї ніяких обмежень. Фреймворк диктує правила побудови архітектури додатку, задаючи на початковому етапі розробки поведінку за замовчуванням, формуючи каркас, який потрібно буде розширювати і змінювати відповідно до зазначених вимог. Фреймворк може включати допоміжні програми, бібліотеки, мови сценаріїв і інше програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту.

Laravel – безкоштовний, PHP-фреймворк з відкритим кодом, створений Taylor Otwell і призначений для розробки веб-додатків відповідно до шаблону model-view-controller (MVC) [12]. Особливістю Laravel є модульна система упакування з виділеним менеджером залежностей, різні способи для доступу до реляційних баз даних, утиліти, які допомагають у розгортанні додатків і технічного обслуговування, а також багатий набір функціональних можливостей, що включає в себе основні функції PHP-структур.

Для взаємодії з базою даних Laravel використовує Eloquent ORM [9], яка являє собою дуже просту реалізацію шаблону проектування Active Record. Використання ORM дозволяє уникнути написання запитів на мові SQL. Суть її полягає в тому, що кожна сутність в базі даних, має свою об'єктну модель. Програміст модифікує цю модель, а вона вже, у свою чергу, взаємодіє з базою. Використання ORM набагато компактніше і швидше, ніж

написання SQL запитів. При створенні будь-якого веб-додатку завжди потрібно писати багато однотипних запитів до бази даних. ORM в свою чергу дозволяє уникнути написання дублюючого коду. Крім того, є можливість легко підтримувати цілість бази даних не тільки на рівні базу, але на рівні коду. Це значно підвищує надійність додатку. Однак ORM не завжди використовує самі найкращі варіанти об'єднання таблиць, що підвищує навантаження на сервер. Дуже часто вибирається велика кількість даних, що також знижує швидкість виконання запитів. Однак, ці недоліки сповна виправдані тим, наскільки сильно використання об'єктної моделі збільшує швидкість розробки. В Eloquent ORM передбачено механізм оптимізації запитів до бази даних, що називається "Active record" [9], який призначений для вирішення проблеми зайвих запитів. Відношення буде активно завантажено, якщо це було вказано при визові методу with(). Таким чином всі необхідні дані будуть завантажені з використанням мінімальної кількості запитів. Не буде необхідності звернутися до бази даних повторно, адже вони уже будуть присутні в моделі.

3.2 Налаштування середовища розробки

При розробці будь-якої системи слід використовувати апаратне і програмне забезпечення, схожі з тим, що буде використовуватися при роботі готової системи. Щоб відтворити умови роботи, максимально приближені до реального, для розробки був використаний Vagrant.

Vagrant – вільне та відкрите програмне забезпечення для створення та конфігурування віртуального середовища розробки [11]. Крім того, фреймворк Laravel надає власний пакет програми для Vagrant – Homestead, в який включено все необхідне програмне забезпечення. Для розгортання Homestead, необхідно виконати наступні дії:

– встановити Virtual Box
– встановити Vagrant

– відкрити консоль
– перейти в директорію проекту
– виконати команди: vagrant box add laravel/homestead, git clone https://github.com/laravel/homestead.git Homestead, bash init.sh.

Конфігурація Homestead [12] відбувається за допомогою редагування файла Homestead.yaml. Є можливість встановлювати наступні налаштування:

– вказати IP-адресу віртуального сервера;
– вказати кількість пам'яті;
– вказати кількість ядер;
– вказати провайдера для створення віртуальної машини;
– вказати шляхи для ssh ключів, щоб використовувати зовнішній репрозиторий для контролю версий;
– співставити шляхи до проектів на віртуалній машині та на host машині;
– вказати назви баз даних.

Приклад файлу Homestead.yaml зображено на рисунку 3.1.

```
---
ip: "192.168.10.10"
memory: 2048
cpus: 1
provider: virtualbox

authorize: ~/.ssh/id_rsa.pub

keys:
    - ~/.ssh/id_rsa

folders:
    - map: ~/Documents/code
      to: /home/vagrant/code

sites:
    - map: games.test
      to: /home/vagrant/code/slim_slots
    - map: dispatcher.test
      to: /home/vagrant/code/dispatcher

databases:
    - games
    - dispatcher
```

Рисунок 3.1 – Файл Homestead.yaml

Після налаштування віртуальної машини, необхідно додати адресу віртуального сервера в файл hosts в операційній системі. Це потрібно зробити, щоб адреса сервера стала доступною для служби DNS. У Windows 10 даний файл знаходиться у директорії C:\Windows\System32\drivers\etc. Потрібно лише додати туди строки: “192.168.10.10 games.test” та “192.168.10.10 dispatcher.test”. Після цього необхідно виконати команду vagrant up у директорії ~\Homestead. Після чого web-сервіси будуть запущені і доступні за адресами, введеними у hosts файлі.

3.3 Розробка структури модулів інформаційної технології

При розробці web-сервісів системи віртуального ігрового автомата, основною задачею є чітке і правильне розділення усіх функцій на окремі модулі, які в подальшому можна буде зручно використовувати та маштабувати, навіть у випадку дадання декількох різних ігрових автоматів. Для виконання цієї задачі було розроблено 2 web-сервіси з такими модулями:

- модуль маршрутизації для створення посилання на відображення клієнта ігрового автомата та єдиної точки відправки клієнтом всіх HTTP запитів;
- модуль відображення клієнта ігрового автомата;
- модуль розмежування ігрових ситуацій (dispatcher);
- конфігураційний модуль, для встановлення списку ігрових автоматів та базового скелету відповіді сервера на ігрову ситуацію;
- модуль ігрових ситуацій та логіки їх опрацювання ігровим автоматом;
- модуль зв’язку з базою данних, де зберігається інформація про ігрові ситуації, побудований на основі паттерну програмування DataMapper;
- додаткові універсальні сервіси для обрахування грошових операцій
- додаткові універсальні сервіси для формування стандартної відповіді від сервера;

– модуль створення та перевірки сисій та керування лімітами ігрового автомата.

Вигляд зв’язку між модулями програмного засобу зображено на рисунку 3.2.

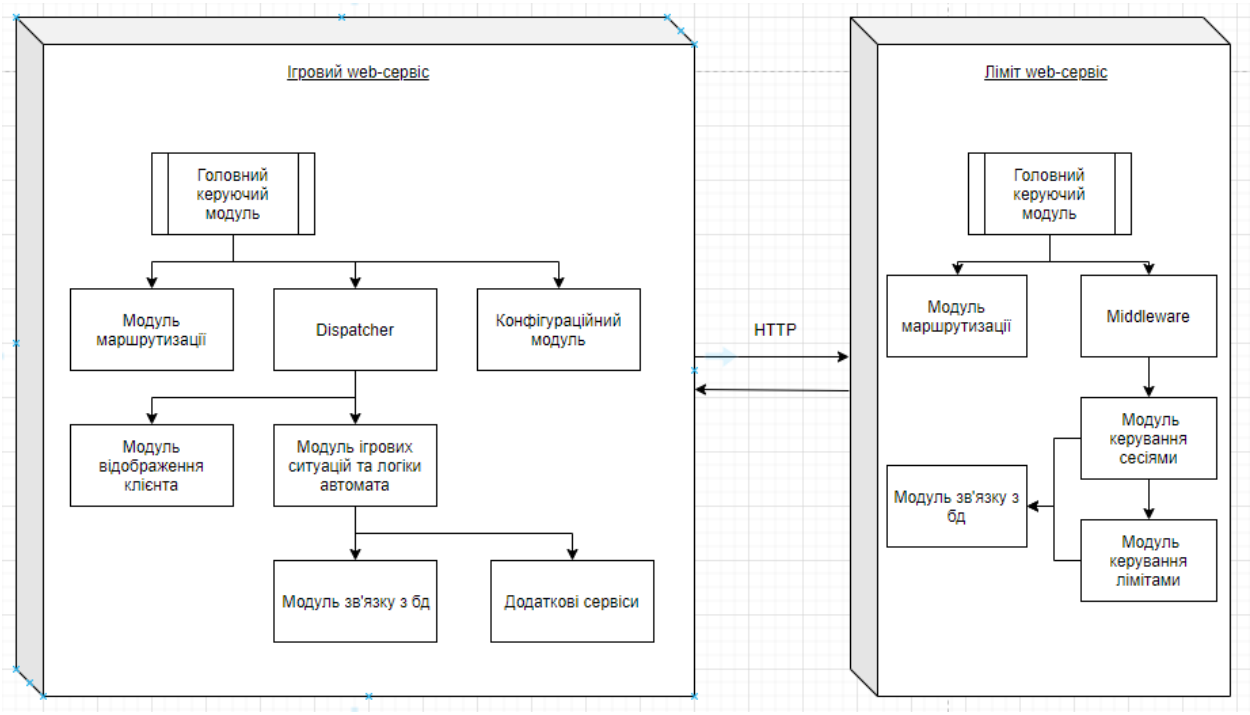

Рисунок 3.2 – Схема модулів програмного засобу

З схеми зв’язку модулів виходить, що застосунок після завершення матиме у собі як необхідний функціонал, так і можливість його розширення легкого масштабування. Також з схеми чітко видно розділення функцій. Ігровий web-сервіс відповідає за створення на керування різними ігровими ситуаціями та ігровим автоматом в цілому, а ліміт web-сервіс керує сесіями гравців та обрахуванням ліміту наступного виграшу. Дані сервіси спілкуються за допомогою HTTP запитів. Оскільки у тілі запиту часто передаються персональні дані та результати грошових операцій, необхідно використовувати HTTP аутентифікацію.

Схема "Базової" HTTP – аутентифікації визначена в стандарті RFC 7617, який передає облікові дані у вигляді пар ідентифікаторів ігрового автомата / пароля, кодованих за допомогою base64. Код перевірки запита представлений далі:

```
$validateGameApiRequest = function ($request, $response, $next) {
    $authHeader = $request->getHeader('X-AUTH');
    if ( ! $authHeader ) {
        return $response->withJson([
            'errors' => [
                ['status' => '401', 'detail' => 'No auth header']
            ]
        ], 401);
    }
    list($authType, $authKey) = explode(' ', $authHeader[0]);
    if ( $authKey !== base64_encode(getenv('GAME_API_USERNAME') . ':' . getenv('GAME_
API_PASSWORD')) ) {
        return $response->withJson([
            'errors' => [
                ['status' => '401', 'detail' => 'Unauthorized']
            ]
        ], 401);
    }
    $data = $request->getParsedBody();
    if ( ! isset($data['token']) ) {
        return $response->withJson([
            'errors' => [
                ['status' => '422', 'detail' => 'No token provided']
            ]
        ], 422);
    }
}
```

Оскільки ідентифікатор ігрового автомата та пароль передаються по мережі як чіткий текст, хоча і закодований base64, цей механізм є обротним, тому також необхідно використовувати SSL – сертифікат.

3.4 Програмна реалізація основних модулів інформаційної технології віртуального ігрового автомата

Першим кроком після взаємодії користувача з ігровим автоматом, а саме натискання кнопки “Крутити барабан”, є розрахунок максимального

допустимого виграшу. Для цього виконується HTTP запит, методом GET, на web-сервіс, який керує сесіями та лімітами:

```
$guzzle = new Guzzle();
$request_data = [
 'token'     => $session['attributes']['token'],
 'spin_id'   => $spin->id,
 'amount'    => $bet['amount'],
];
try {
 $api_request = $guzzle->get("{$container['settings']['api_url']}balance/bet",
 [
    'headers' => [
        'X-AUTH'        => 'Basic c2VjcmV0OnNlY3JldA==',
        'Cache-Control' => 'no-cache',
        'Content-Type'  => 'application/json'
    ],
    'json' => $request_data
 ]);
 $response_data = json_decode($api_request->getBody(), TRUE)['data'];
} catch (\GuzzleHttp\Exception\RequestException  $e) {
 $container['starburst_log']->error('', [
    'request'   => json_decode($e->getRequest()->getBody(), TRUE),
    'response'  => json_decode($e->getResponse()->getBody(), TRUE),
 ]);
 return \GameError::send(0);
}
```

Ліміт розраховується за допомогою SQL запиту:

```
$sql = "SELECT
 (SELECT IFNULL(SUM(amount), 0)
  FROM transactions
  WHERE created_at > CURDATE()
  AND status = 1
  AND type = 0) AS 'total_bets',
 (SELECT IFNULL(SUM(amount), 0)
  FROM transactions
  WHERE created_at > CURDATE()
  AND status = 1
  AND type = 1) AS 'total_wins'
";
$stmt = $this->container['db']->prepare($sql);
```

```
$stmt->execute();
$result = ($stmt->fetch());
```

де із бази даних збирається інформація про ставки та виграші користувача за поточний день. За допомогою методів IFNULL та SUM значення підсумовується або встановлюється у нуль по замовчуванню, у випадку його пустоти. Далі розраховуєтсья і вертається значення максимального можливого наступного виграшу:

```
$next_max_win=$result['total_bets']*(1 - $limit->limit/100) - $result['total_wins'];
return $next_max_win > 0 ? round($next_max_win, 2, PHP_ROUND_HALF_DOWN) : 0;
```

Другим кororком запускається алгоритм створення ігрового поля, генерації ігрової комбінації та визначення суми виграшу.

```
$limit = $response_data['attributes']['max_next_win'];
$lowReelsPercents = false;
```

Опираючись на значення ліміту запускається цикл, який буде працювати, поки не згенерує програшну ігрову ситуацію у випадку нульового значення максимального виграшу, або:

```
if ($limit > 0) {
 do {
```

генеруємо ігрове поле:

```
   $reels = $this->logic->generateReels($lowReelsPercents);
```

розраховуємо бонусний безкоштовний раунд:

```
   $overlaied = $this->logic->overlayReels($reels);
   $respin = $overlaied['respin'];
```

та перевіряємо наявність співпадіння, тобдо чи згенерувалась на полі виграшна комбінація:

```
   $matches = $this->logic->match($overlaied['reels']);
```

У випадку, якщо виграшних комбінацій немає – виходио з циклу і закінчуємо алгоритм.

```
   if (empty($matches)) {
     $i = 0;
     break;
   }else{
```

Якщо все ж виграшні комбінації присутні, розраховуємо суму потенційного виграшу,

```
    $win_coins = $this->logic->calculateWonCoins($matches);
```

```
    $bet_results = $container['moneyManager']-
>countWinning($betlevel, $denomination, $win_coins);
```

та перевіряємо тип виграшу, якщо тип середній чи низький закінчуємо алгоритм, тобто дозволяємо користувачу виграти,

```
    if ($this->determineWinType($bet_results['amount']) === 'low' ||
        $this->determineWinType($bet_results['amount']) === 'avarage') {
        $i = $bet_results['amount'];
    } else {
```

але якщо виграш високий, встановлюємо прапор пониження коефіцієнтів для генерації “дорогих символів”, та проходимо цикл ще раз

```
      $lowReelsPercents = true;
      $i = $limit + 1;
    }
   }
  } while($i > $limit);
 } elseif ($limit == 0) {
  $i = 1;
  while($i != 0) {
  $reels = $this->logic->generateReels();
  $overlaied = $this->logic->overlayReels($reels);
  $respin = $overlaied['respin'];
  $matches = $this->logic->match($overlaied['reels']);
  if (empty($matches)) {
    $i = 0;
  }
}
}
```

Далі, наступними кроками, по раніше заданим у конфігураціних модулях базовим параметрам фурмується тіло відповіді для клієнта та усі результати про ігрові ситуації зберігаються у базу даних. У випадку якщо змінна `$respin` має позитивне значення, для даного раунда виставляється мітка і наступною ігровою ситуації буде бонусний раунд (всього можливо три бонусних раунда). Алгортим ігрової ситуації бонусного раунда такий самий, як і для звичайного,

```
$won_coins = $this->logic->calculateWonCoins($matches);
$freespin_results = $container['moneyManager']->countWinning(
 $spin->betlevel,
```

```
 $spin->denomination,
 $won_coins
);
$previous_wins = $freespin_mapper->sumWinAmount($spin->id);
if (!is_null($spin_win)) {
 $i = $spin_win->win_amount + $previous_wins + $freespin_results['amount'];
} else {
 $i = $previous_wins + $freespin_results['amount'];
}
if ($this->determineWinType($freespin_results['amount']) !== 'high') {
 $lowReelsPercents = true;
 $i = $limit + 1;
}
```

тільки виграш накопичується і акумулюється.

3.5 Тестування програмного забезпечення та аналіз результатів

Інструкція для користувача допоможе у використанні росробленої інформаційної технології віртуального ігроовго автомата. Для змоги грати користувач повин володіти такими навичками:

- володіння навичками роботи користувача з операційними системами на яких можливі відтворення програмного забезпечення, або ж інших відворювальних засобів;
- володіння навичками роботи з встановленими браузерами у системі на базовому рівні.

Вікно додатку віртуального ігрового автомата складається з декількох блоків: основного ігрового поля (матриця 3x5); блоку вибору розміру ставки, визначення ваги ігрової монети та кнопки запуску барабанів ігрового поля (рисунок 3.3).

Рисунок 3.3 – Вікно додатку віртуального ігрового автомата

Також присутня спеціальна кнопка яка відкриває сторінку з описом різних функцій ігрового автомата та з правилами гри.

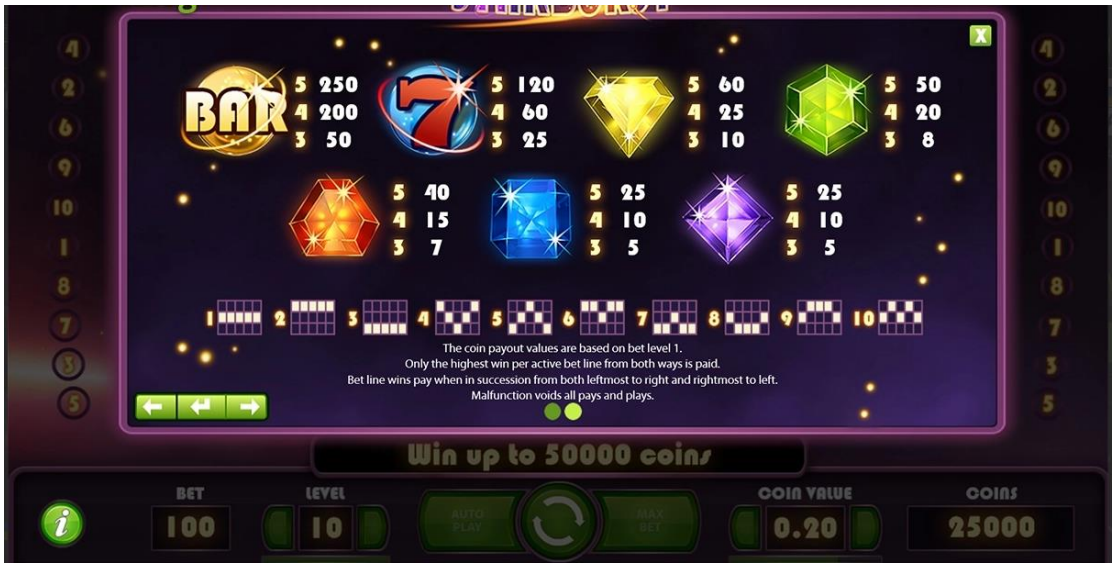

Рисунок 3.4 – Вікно з правилами віртуального ігрового автомата

Якщо відкрити вікно з правилами гри (рисунок 3.4), натиснувши на відповідну кнопку, користувач побачить вікно зі списком ігрових символів по центру екрана. Виграшна комбінація кожного з них може бути з 3, 4 або 5 символів підряд. Кожна комбінація, відповідно, має свою вагу, яка вимірюється в ігрових монетах. Інформація про даний список зберігається на сервері, і при роботі алгоритму генерації виграшної комбінації, на основі цього списку формується ігрове поле (Додаток Б). Символ має свій шанс на

випадіння у відсотковому співвідношенні, так як ціна за комбінацію кожного з них, різна.

В нижній частині екрану видно список із 10 ліній по яким можуть генеруватись виграшні комбінації. Представлений ігровий автомат може співставляти комбінації як зліва на право так і справа на ліво. У разі виграшу, користувачеві буде зараховано найдорожча комбінація (рисунок 3.5). Список виграшних ліній також зберігається на сервері.

Рисунок 3.5 – Вікно віртуального ігрового автомата з виграшною комбінацією

Також при генерації ігрового поля з невеликим шансом може згенеруватись спеціальний символ – wild, який при порівнянні символів на виграшній лінії заміняє будь який з них. За правилами, wild може генеруватись лише на 2, 3 та 4 барабанах (рисунок 3.6).

Рисунок 3.6 – Вікно віртального ігрового автомата зі спеціалним символом wild

Якщо на всьому барабані згенерувались лише wild символи, користувач отримує додаткову бонусну прокрутку барабанів. Для бонусного раунда діють всі основні правила, лише попередньо згенерований барабан (який складається з бонусних символів), залишається на місці. Максимальна кількість бонусних раундів – 3.

3.6 Висновок

У даному розділі було проаналізовано мови програмування та середовище розробки. В ході аналізу для розробки інформаційної технології віртуального ігрового автомата було обрано мову програмування PHP, Slim та Laravel Framework, а також середовище програмування Visual Studio Code. Для розробки та тестування системи було встановлено та налаштовано віртуальну машину Homestead. Детально описано програмну реалізацію основних модулів та алгоритмів системи, таких як генерація ігрового поля та пошук виграшних комбінацій, HTTP аутентифікація і т.д.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Озеранський В.С. та Войтко В.В.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	3	4
4	4	3
5	3	3
6	4	4
7	4	3
8	4	4
9	3	3
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 43$	$СБ_2 = 42$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 42{,}5$	

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (4.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 21 день;

t - число днів роботи розробника, t = 45 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5400	257,14	8	2057,12
Інженер-програміст	3800	180,95	45	8142,75
Всього:				10199,87

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 10199,87 = 1019,98 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}= (З_о + З_р)\cdot\frac{\beta}{100}, \quad (4.2)$$

$$Н_{зп}= (10199,87+1019,98)\cdot\frac{36,3}{100} = 4072,80 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot\frac{Т}{12}, \quad (4.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	11000	25	3	687,5
Всього:				687,5

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	170	1	170
Пачка паперу	уп.	125	1	125
Ручка	шт.	5	1	5
Всього з урахуванням транспортних витрат				330

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п ; \quad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=190 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0,7$).

$$В_е= 1,7 \cdot 0,6 \cdot 190 \cdot 0,7 = 135,66 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (10199,87 + 1019,98) = 11219,85 \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 10199,87+1019,98+4072,80+135,66+687,5+330+11219,85= 27665,66 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (4.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{27665,66}{1} = 27665,66$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{27665,66}{0,9} = 30739,62 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 30 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 30 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 180 користувачів, протягом третього року – 150 користувачів. Реалізація

інформаційної технології до впровадження результатів наукової розробки складала 500 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 200 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 30 \cdot 500 + (200 + 30) \cdot 200 = 61000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 30 \cdot 500 + (200 + 30) \cdot (200 + 180) = 102400 \text{грн.}$$

Протягом третього року:

$$\Delta П_3 = 30 \cdot 500 + (200 + 30) \cdot (200 + 180 + 150) = 136900 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$ вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \qquad (4.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

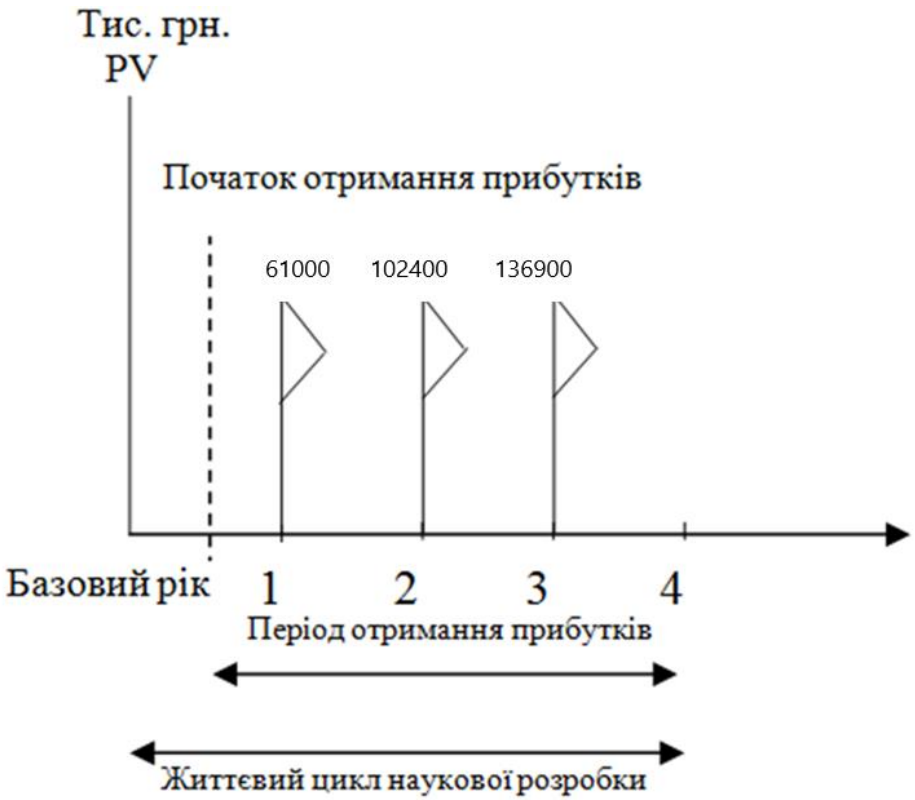

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{30739,62}{(1+0,1)^0} + \frac{61000}{(1+0,1)^2} + \frac{102400}{(1+0,1)^3} + \frac{136900}{(1+0,1)^4} = 251592,01 \text{(грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$E_{абс} = 251592,01 - 30739,62 = 220852,39 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (4.12)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{220852,39}{30739,62}} - 1 = 1,01 \text{ або } 101\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0,2 + 0,1 = 0,3$$

Оскільки $E_в$ = 101% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{1,01} = 0,99 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

4.5 Висновок

В даному розділі було проведено оцінювання комерційного потенціалу розробки програмного засобу.

Проведено технологічний аудит з залученням двох незалежних експертів та визначено, що рівень комерційного потенціалу розробки вище середнього.

Аналіз комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками вносить новизну в предметну галузь та випереджає аналогічні програмні продукти і є перспективною розробкою. Він має кращі цільові функціональні показники, а тому є конкурентоспроможним товаром на ринку. Існуючі переваги нової розробки дозволять швидко її поширити та популяризувати.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальні витрати на розробку складають 30739,62 грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі 220852,39 грн свідчить про отримання прибутку інвестором від комерціалізації програмного продукту.

Щорічна ефективність вкладених в наукову розробку інвестицій складає 101 %, що вище за мінімальну бар’єрну ставку дисконтування, яка складає 90%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки. Термін окупності вкладених у реалізацію проекту інвестицій становить 0,99 року, що також свідчить про доцільність фінансування нової розробки.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи розроблено інформаційну технологію віртуального ігрового автомата. Аналізуючи предметну область визначено, що є чимала кількість аналогічних розробок і саме ця інформаційна технологія вважається найбільш актуальною на ринку. Визначено та описано основні етапи, з яких складається процес розробки віртуального ігрового автомата.

Також під час розгляду магістерської кваліфікаційної роботи проведено аналіз аналогів, та визначено їх основні проблеми, в результаті чого прийнято рішення розробити власну інформаційну технологію віртуального ігрового автомата, використовуючи сучасні засоби веб-програмування.

У другому розділі було розглянуто різні підходи для проектування і обрано сервіс-орієнтовну архітектуру, основану на REST підході. Це дало змогу виконати усі поставлені задачі та навіть зробити доступною легку масштабованість системи, яка зможе включати в собі декілька ігрових автоматів. Також у ході розробки основних алгоритмів та моделей функціонування інформаційної технології було виявлено, що доцільним буде використати нечітку логіку для розрахунку виграшних комбінацій гравця. Зпроектовано базу даних для зберігання інформації про ігрові ситуації та їх подальшого використання для генерації максимального потенційного виграшу.

У третьому розділі обґрунтовано вибір мови програмування PHP, основними перевагами якої є: надійність, простота та універсальність. Представлено та детально описано роботу основних алгоритмів інформаційної технології.

У ході економічного обгрунтування розробки проведено оцінювання економічного потенціалу розробки (220852,39 грн), спрогнозовано витрати на виконання науково-дослідної, дослідно-конструкторської та

конструкторсько-технологічної роботи (30739,62 грн), спрогнозовано комерційні ефекти від реалізації результатів розробки, розраховано ефективність вкладених інвестицій та періоду їх окупності (0,99 року).

Задачу магістерської кваліфікаційної роботи виконано в повному обсязі: проаналізовано наявні ризики та загрози, розроблено та описано структуру програмного продукту та алгоритми основних блоків його роботи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Хворостюк Є.В. "Інформаційна технологія віртуального ігрового автомата" [Текст] / Хворостюк Є.В. Озеранський В.С. // Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2020)»: Тез. доп. – Вінниця, 2019. – с.1

2. Офіційний сайт Construct2 – [Електронний ресурс]. – Режим доступу: http://www.scirra.com.

3. Офіційний портал розробників XNA – [Електронний ресурс]. – Режим доступу: https://msdn.microsoft.com/ruRU/games-development-msdn.

4. Порівняння HTTP та HTTTPS – [Електронний ресурс]. – Режим доступу: https://ssl.com.ua/blog/http-vs-https.

5. Портал розробників Html5 ігор – [Електронний ресурс]. – Режим доступу https://html5gameengine.com/

6. Кузнецов М. – «Самоучитель PHP» –Петербург, 2009 р.

7. Paul Dubois – MySql Cookbook Solutions for database developers and administrators. – O'Reilly.

8. М.Зандстра – PHP объекты, шаблоны и методики програмирования 4-видання.

9. Eloquent ORM. [Електронний ресурс]. – Режим доступу: https://eloquentbyexample.com/

10. Visual Studio Code. [Електронний ресурс]. – Режим доступу: https://tproger.ru/news/visual-studio-code-1-22/

11. Integrated development environment. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Інтегроване середовище розробки.

12. Laravel for php artisans. [Електронний ресурс] – Режим доступу: https://www.laravel.com.

13. Нидерст Д.Р. – «HTML5, CSS3 и JavaScript. Исчерпывающее руководство» – Москва, 2014 р.

14. Сервісно-орієнтована архітектура. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Сервісно-орієнтована_архітектура.

15. Порівняння SOAP і REST. [Електронний ресурс] - Режим доступу: https://medium.com/@nanotexnolagiya/сравнение-soap-и-rest-c-json-2019-779fef6eba9b.

16. В.М.Антоненко, С.Д.Мамченко, Ю.В.Рогушина – Сучасні інформаційні системи і технології: управління знаннями.

17. PHP, MySql та інші технології. [Електронний ресурс] – Режим доступу: http://www.php.su/.

18. Алгоритм і система роботи онлайн-слотів. [Електронний ресурс] – Режим доступу: https://www.softgamings.com/ru/blog/algoritm-slotov/.

19. П. Кравець, Р. Киркало – Системи прийняття рішень з нечіткою логікою.

20. Інструкція для слот автоматів, історія, типи, принцпи роботи. [Електронний ресурс] – Режим доступу: http://mukachevo.net/ua/news/view/