Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційна робота ступеня
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Ільющенка Данііла Сергійовича*
(ПІБ)

академічної групи *122-18-3*
(шифр)

спеціальності *122 Інформаційні технології*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка інформаційної системи для реселінгу хостингу серверів засобами фреймворку Ruby on Rails*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-3* (група) *Ільющенка Данііла Сергійовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інформаційної системи для реселінгу хостингу серверів засобами фреймворку Ruby on Rails*

затверджена наказом ректора НТУ «ДП» від 18.05.2022 № 268-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ (підпис) *доц. Спірінцев В.В* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Ільющенко Д.С.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2022*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2022*

ЗМІСТ

РЕФЕРАТ

Пояснювальна записка 87 с., 5 рис., 3 дод., 22 джерела.

Об’єкт розробки: інформаційна система з реселінгу хостингу серверів.

Мета кваліфікаційної роботи: створення інформаційної система для реселінгу хостингу серверів.

У вступі розглядається аналіз та сучасний стан проблеми , конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформу для розробки, виконано проектування і розробку веб-додатку засобами фреймворку Ruby on Rails, описана робота додатку, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні додатка, що надає можливість прихильникам малого і великого бізнесу легко та швидко обрати потрібні послуги з підключення до виділеного або віртуального сервера, подати заявку на відключення послуг задля створення власних проектів.

Актуальність інформаційної системи визначається великим попитом на оренду серверів Інтернет проектами, що інтенсивно розвиваються. Оскільки кожному бізнесу потрібна присутність в Інтернеті, то дана інформаційна система з реселінгу серверів якнайкраще вирішує подібні завдання.

Список ключових слів: ХОСТИНГ, РЕСЕЛІНГ, ІНФОРМАЦІЙНА СИСТЕМА, ВІРТУАЛЬНИЙ СЕРВЕР, АЛГОРИТМ, ПРОЕКТУВАННЯ, ВЕБ-ДОДАТОК ІНТЕРНЕТ.

ABSTRACT

Explanatory note: 87 p., 5 pictures, 3 extras, 22 sources.

The object of development: an information system for server hosting reselling.

The goal of the qualification work: creation of an information system for server hosting reselling.

The introduction examines the analysis and the current state of the problem, specifies the goal of the qualification work and the scope of its use, provides a substantiation of the relevance of the topic and clarifies the definition of the task.

The first section analyzes the subject area, identifies the relevance of the task and the purpose of the development, formulates the statement of the task, and specifies the requirements to the software implementation, technologies and software tools.

In the second section we analyzed the available solutions, turned around the platform for development, carried out the design and development of the web-device using Ruby on Rails framework, described the work of the add-on, algorithm and structure of its functioning, as well as the response and loading of the program, identified the input and output data, characterized the composition of the parameters of technical devices.

In the economic section, the labor intensity of the developed information system is determined, the cost of the work on the creation of the program is estimated and the time for its creation is calculated.

Practical importance lies in the creation of the add-on, which makes it possible for participants of small and large businesses easily and quickly turn to the required services for connection to a separate or virtual server, to apply for the inclusion of services for the creation of their own projects.

Relevance information system is determined by the great demand for server rental Internet intensively developing projects. Since every business requires presence on the Internet, this information system for reselling servers is the best way to solve such issues.

List of keywords: HOSTING, RESELLING, INFORMATION SYSTEM, VIRTUAL SERVER, ALGORITHM, DEVELOPMENT, WEB DEVICE, INTERNET.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

СУБД – система управління базами даних;

API – application programming interface;

CoC – convention over configuration;

CSS – cascading style sheets;

DRY – don’t repeat yourself;

HDD – hard disk drive;

HTTP – hypertext transfer protocol;

HTML – hypertext markup language;

JS – JavaScript;

IT – information technology;

ORM – object relational mapping;

PHP – hypertext preprocessor;

RoR – Ruby on Rails;

SASS – syntactically awesome style sheets;

SQL – structured query language;

SSD – solid-state drive;

UCS – unified computing system;

VPS – virtual private server.

ВСТУП

Розроблена інформаційна система призначена для застосування у сфері перепродажу хостингу серверів.

Коли ви орендуєте серверний простір і пропускну здатність у більшого хостинг-провайдера, а потім перепродаєте це своїм клієнтам, зберігаючи при цьому прибуток – це називається ріселінгом хостингу. Користувачеві надається можливість розподіляти дисковий простір і ресурси свого віртуального хостингу або серверу з метою розміщення на ньому сайтів третіх осіб, що можуть бути його клієнтами. Пакет послуг такого хостингу зазвичай включає спеціальне ПЗ для управління клієнтською базою, ресурсами, що надані клієнтам

Апаратне забезпечення та організацію його роботи (обслуговування фізичного сервера, співпраця з дата-центром) хостинговий провайдер бере на себе, а реселери оплачують виділені їм ресурси за фіксованою оплатою. Їх основним завданням є залучення якомога більшої кількості клієнтів на виділені ресурси, з метою отримання доходу.

Реселер хостингу надає великі повноваження та є ідеальним варіантом для адміністраторів, розробників, веб-дизайнерів або для тих, хто збирається побудувати бізнес у сфері хостинг-послуг без великих інвестицій.

Тому проблема, розглянута в даному дипломному проекті, є актуальною та має широке практичне значення.

Метою даної роботи є розробка інформаційної системи для реселінгу хостингу серверів засобами фреймворку Ruby on Rails.

Для досягнення поставленої мети необхідно вирішити основні завдання:

- здійснити аналіз існуючих хостингових компаній з метою виявлення їх властивостей, потенційних можливостей та недоліків;
- розробка алгоритму, бази даних і реалізації програми.

У відповідності до проведеного аналізу поставлені основні функціональні задачі перед системою:

- можливість ознайомитися з представленими послугами на сайті рісейлінгової компанії;
- надати можливість адміністраторам сайту додавати, видаляти та редагувати дані, які стосуються представлених послуг або товарів;
- надати можливість проектувальникам інфраструктури індивідуального автоматизованого доступу до серверів;
- підвищити швидкість створення проектів прихильниками малого та великого бізнесу завдяки зручному інтерфейсу.

Розроблену інформаційну систему можна використовувати в якості корпоративного сайту для надання послуг оренди віртуальних та виділених серверів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Основні концепції хмарних обчислень

Хмарні технології більше не є модною додатковою функцією для зберігання фотографій в Інтернеті. Тепер це є невід'ємною частиною бізнес-моделі, яка захоплює весь світ. Крім того, що хмарні технології дозволяють підприємствам зберігати та отримувати доступ до даних, вони змінюють методи роботи великих та малих підприємств.

Хмара – це спосіб абстрагування, об'єднання та спільного використання обчислювальних ресурсів – серверів, комутаторів, маршрутизаторів, операційних систем та програмного забезпечення безпеки – через мережу пристроїв на території або за її межами. Хмарні обчислення належать до використання цих ресурсів через Інтернет. Як правило, користувачі орендують обчислювальні ресурси, такі як простір для зберігання даних або потужність процесора, а постачальник хмарних обчислень активно керує обладнанням (наприклад, дисками зберігання даних та серверами), яке забезпечує ці обчислювальні ресурси. Хмарні провайдери також пропонують послуги вищого рівня: аналіз даних, підтримка аварійного відновлення чи середовище розробки – все це працює на обладнанні хмарного провайдера.

Хмарні проекти прагнуть максимально використати хмарні технології для досягнення бізнес-цілей. Щоб максимально використовувати функціональність хмари, компанії залучають фахівців із хмар, які займаються технологічними аспектами роботи з хмарами та допомагають вирішувати бізнес-завдання.

Хмара використовує суміш технологій, таких як операційна система, платформа управління та API. Кожна з цих технологій включає можливість

використання програм віртуалізації та автоматизації. В результаті існують сотні різних типів хмарних ресурсів та безліч варіантів для кожного з них.

Хмарні обчислення відкривають перед компаніями необмежені можливості надання послуг і товарів клієнтам. Ефективне використання хмарної системи починається з підбору відповідного типу хмари для робочих навантажень і даних вашої організації [1]. Щоб допомогти клієнтам підібрати робочі навантаження та дані, корисно знати трохи про основні характеристики та переваги хмарних обчислень, серед яких виділяють наступні:

– Надійність.

Хмара дає почуття упевненості в тому, що ваші сервери та дані завжди будуть на місці. Постачальники хмарних обчислень пропонують послуги резервного копіювання, аварійного відновлення та реплікації даних, щоб ваші дані завжди були у безпеці.

Крім того, постачальники хмарних обчислень пропонують різні варіанти відмови стійкості для ваших серверів: Вони надають дані та інформацію для моніторингу додатків, реагування на зміни продуктивності та створення єдиної картини працездатності.

– Масштабованість.

Масштабованість - це можливість збільшувати або зменшувати IT-ресурси в міру необхідності задоволення мінливого попиту. У хмарі підтримується як вертикальне, і горизонтальне масштабування, залежно від потреб клієнта.

Вертикальне масштабування передбачає збільшення розміру кожного ресурсу, наприклад, додавання більшої потужності процесора або пам'яті на існуючі сервери.

Горизонтальне масштабування передбачає додавання додаткових ресурсів до існуючих кластерів - швидше за все, того ж розміру, що й сервери.

– Безпека.

Хмарні провайдери допомагають вашій організації підтримувати високий рівень безпеки. Завдяки широкому набору політик, технологій, засобів контролю та експертних технічних навичок, хмарні провайдери пропонують набагато кращий набір інструментів для забезпечення безпеки, ніж більшість організацій можуть досягти на своїй території. Компанії можуть використовувати ці інструменти та послуги для зміцнення безпеки.

– Економічна ефективність.

Хмарні обчислення передбачають модель ціноутворення за принципом "плати з використанням" або "споживання". Замість того, щоб платити вперед за певну кількість обчислювальних ресурсів або обладнання, ви можете орендувати те, що вам потрібно, і платити тільки за ті ресурси, які ви використовуєте.

Ця модель, що базується на споживанні, пропонує безліч переваг, включаючи:

- відсутність попередніх витрат;
- відсутність необхідності купувати, розуміти та керувати дорогою інфраструктурою;
- можливість збільшувати чи зменшувати обсяг ресурсів залежно від потреб;
- можливість змінити свою думку про типи чи розміри ресурсів.

Ці переваги допомагають організаціям мінімізувати фінансові ризики, особливо на початку проекту.

Як тільки проект запущено, постачальник хмарних послуг повинен надати звітність про витрати, прогнози витрат на основі поточних тенденцій та моделювання витрат на основі історичних даних. У міру стабілізації послуги ви зможете оптимізувати витрати, взявши на себе зобов'язання щодо мінімальних витрат або навіть зарезервувавши ресурси.

– Еластичність.

Еластичність означає здатність надавати або видаляти ресурси на вимогу залежно від потреб вашої організації. Завдяки процесу "автомасштабування" система хмарних обчислень може автоматично додавати або видаляти ресурси, щоб компенсувати коливання робочого навантаження.

– Різноманітність послуг.

Послуги хмарних провайдерів можуть істотно відрізнятися і включати широкий спектр можливостей. Хмарні провайдери можуть використовуватися для традиційних IT-інфраструктур, таких як сервери, системи зберігання даних та мережі. Вони також пропонують більш сучасні послуги та структури, такі як штучний інтелект (AI) та Інтернет речей (IoT).

1.1.2. Технологія віртуалізації серверів

Програмне забезпечення для віртуалізації серверів поділяє фізичні сервери на кілька віртуальних сегментів, якими можна управляти незалежно один від одного. Продукти віртуалізації серверів, зазвичай називають гіпервізорами, використовуються організаціями для поділу виділених серверів на масштабовані віртуальні екземпляри, звані віртуальними приватними серверами (VPS) [2]. Віртуальні приватні сервери, створені за допомогою програмного забезпечення для віртуалізації серверів, можуть містити унікальну операційну систему та керуватися незалежно один від одного за допомогою інтегрованої панелі керування хостингом. Організації використовують програмне забезпечення для віртуалізації серверів, щоб розподіляти ресурси сервера між віртуальними машинами для оптимізації робочого навантаження. Віртуальні машини, створені за допомогою програмного забезпечення для віртуалізації серверів, найчастіше гнучкіші та надійніші в управлінні, ніж сервери без розділів. Організації, які розміщують та керують серверами всередині компанії, використовують програмне забезпечення для віртуалізації серверів, щоб знизити навантаження на ресурси. Крім того, програмне забезпечення для віртуалізації серверів

зазвичай використовується постачальниками хостингу для надання послуг VPS-хостингу своїм клієнтам.

Для включення до категорії "Віртуалізація серверів" продукт повинен:

– дозволяти користувачам розділяти фізичні сервери на кілька віртуальних екземплярів;
– забезпечувати просте управління масштабованими віртуальними середовищами;
– надавати доступ до вбудованої або сторонньої панелі керування, за допомогою якої можна керувати віртуальними серверами.

Віртуалізація серверів використовується для маскування ресурсів сервера користувачів сервера. Це може включати кількість та ідентичність операційних систем, процесорів та окремих фізичних серверів, тож можна зробити висновок, що віртуалізація сервера - це процес поділу фізичного сервера на кілька унікальних та ізольованих віртуальних серверів за допомогою програмного додатка. Кожен віртуальний сервер може запускати власні операційні системи незалежно один від одного [3].

З основні переваг віртуалізації серверів виділяють:

– вища продуктивність сервера;
– нижчі експлуатаційні витрати;
– усунення складності сервера;
– підвищена продуктивність додатків;
– швидше розгортання робочого навантаження.

Віртуалізацію серверів також поділяють на три види:

1) Повна віртуалізація: Повна віртуалізація використовує гіпервізор, тип програмного забезпечення, яке безпосередньо взаємодіє з дисковим простором та процесором фізичного сервера. Гіпервізор контролює ресурси фізичного сервера і робить кожен віртуальний сервер незалежним і не знає про інші віртуальні сервери. Він також передає ресурси з фізичного сервера на потрібний віртуальний

сервер із запуском додатків. Найбільшим обмеженням використання повної віртуалізації є те, що гіпервізор має власні потреби в обробці даних. Це може сповільнити роботу програм та вплинути на продуктивність сервера.

2) Пара-віртуалізація: На відміну від повної віртуалізації, при пара-віртуалізації вся мережа працює як єдине ціле. Оскільки при паравіртуалізації кожна операційна система на віртуальних серверах знає один про одного, гіпервізору не потрібно використовувати стільки обчислювальної потужності для управління операційними системами.
3) Віртуалізація на рівні ОС: На відміну від повної та пара-віртуалізації, віртуалізація на рівні ОС не використовує гіпервізор. Натомість можливості віртуалізації, які є частиною операційної системи фізичного сервера, виконують всі завдання гіпервізорус. Однак при такому методі віртуалізації серверів всі віртуальні сервери повинні працювати під керуванням однієї і тієї ж операційної системи.

Віртуалізація серверів – це економічно ефективний спосіб надання послуг веб-хостингу та ефективного використання наявних ресурсів у IT-інфраструктурі. Без віртуалізації серверів сервери використовують лише невелику частину своєї обчислювальної потужності. Це призводить до того, що сервери простоюють, оскільки робоче навантаження розподіляється лише на частину серверів мережі. Центри обробки даних стають переповненими недовикористовуваними серверами, що призводить до нераціонального використання ресурсів та електроенергії.

Завдяки тому, що кожен фізичний сервер поділений на кілька віртуальних серверів, віртуалізація серверів дозволяє кожному віртуальному серверу діяти як унікальний фізичний пристрій. Кожен віртуальний сервер може запускати власні програми та операційну систему. Цей процес збільшує

використання ресурсів, змушуючи кожен віртуальний сервер діяти як фізичний сервер і збільшує потужність кожної фізичної машини.

1.1.3. Віртуальні та виділені сервери

Металеві сервери – це сервери, де всі ресурси сервера належать лише одному користувачеві. Саме тому такий сервер називають сервером з одним орендарем, оскільки в ньому немає спільного використання ресурсів, і він призначений тільки для одного клієнта [4]. Історія процесорів виділених серверів до сьогоднішнього дня:

– 1980-і роки: проста настільна серверна машина

Традиційно "голі метали" були простими настільними машинами, які розміщувалися у невеликій кімнаті всередині організації. Його конфігурація складалася з 256 МГц процесора та 2 ГБ диска.

– 1990-і: стійкий веб-сервер

Поступово, коли зріс попит на послуги веб-хостингу, стійкий сервер замінив настільні машини. Він мав Intel P2 Xeon 450Mhz, 256Mb RAM та 24X CDROM плеєр. 1998: був представлений перший сервер Google з 256 Мб ОЗУ та двома процесорами 200 МГц.

– 2000-і: блейд-сервери

У 2001-2002 роках з'явилися блейд-сервери – машини розміром із долоню. У 2008 році віртуалізація стала популярною, і сервери стали ґрунтуватися на технології розподілених обчислень. З розвитком технології віртуалізації виникли хмарні обчислення. Обговоривши історію процесорів для виділених серверів, ми повинні також поговорити про майбутнє виділених серверів, яким є революційна технологія під назвою Unified Computing System (UCS).

UCS дозволила оптимізувати ресурси центру обробки даних та виділеного сервера, об'єднавши їх у пул віртуалізованих серверів, систем зберігання даних та мережевих ресурсів. UCS має уніфіковану

інфраструктуру з централізованим підходом, що забезпечує більш швидку обробку даних та безперебійну роботу серверів. Ця інноваційна технологія нового покоління дозволяє автоматично конфігурувати апаратне забезпечення та використовує модель, орієнтовану на програми. У двох словах можна сказати, що UCS забезпечує маневровість, стабільність, гнучкість та високу швидкість.

Віртуалізація серверів, часто розташованих в автономному центрі обробки даних або хмарному середовищі, передбачає перетворення одного фізичного сервера на кілька віртуальних машин (ВМ). Віртуальний сервер конфігурується таким чином, щоб кілька користувачів могли спільно використовувати обчислювальну потужність. При порівнянні фізичного сервера та віртуального сервера віртуальний сервер є ефективним способом економії витрат на фізичне обладнання. Він також споживає порівняно менше енергії, що забезпечує додаткову економію коштів та екологічні переваги.

Віртуальний сервер відтворює функціональність виділеного фізичного сервера. Він існує прозоро для користувачів як розділений простір усередині фізичного сервера. Віртуалізація серверів дозволяє легко перерозподіляти ресурси та адаптуватися до динамічних робочих навантажень.

Перетворення одного фізичного сервера на кілька віртуальних серверів дозволяє організаціям більш ефективно використовувати обчислювальну потужність та ресурси за рахунок запуску декількох операційних систем та програм на одному розділеному сервері.

Більшість організацій використовують віртуальні сервери для скорочення витрат на серверне обладнання та зниження витрат на живлення та електроенергію. Віртуальні сервери відіграють ключову роль у створенні програм, інструментів або середовищ.

Виділений сервер – це тип віддаленого сервера, який повністю призначений для окремої людини, організації або програми. Він

розгортається, розміщується та керується постачальником послуг хостингу, хмарних чи керованих послуг [3].

Виділений сервер є ексклюзивним і не використовується спільно з іншими клієнтами, послугами або програмами.

Виділений сервер забезпечує функціональність, подібну до внутрішнього сервера, але належить, працює і управляється бекенд-провайдером. Користувач/клієнт віддалено підключається до виділеного сервера через Інтернет для виконання набору послуг на базі сервера.

Виділений сервер може використовуватися для розміщення програм та/або послуг, а також для зберігання даних та резервного копіювання. Виділений сервер також може використовуватися всередині компанії для розміщення та надання спеціалізованих послуг, наприклад для реалізації виділених файлових або мережевих серверів.

1.1.4. Реселінг хостингу, його переваги та недоліки

На сьогоднішній час багато веб-студій займаються реселінгом хостингу оскільки багато клієнтів, що замовляють розроблення сайту, погоджуються з подальшим технічним супроводом сайту: розміщення на хостингу, реєстрація доменного імені, технічна підтримка, антивірусний захист, внесення змін до сайту [6]. Як правило, багато пересічних клієнтів не стануть самостійно шукати хостинг і скористаються повним комплектом, а веб-студія отримує додаткові дивіденди від замовника.

Реселери надають послуги з розміщення інформації на серверах. Готова технічна база, напрацьована клієнтська аудиторія приносять непоганий заробіток реселерам.

Суть реселінгу полягає у поділі фізичного або віртуального хоста, щоб надалі перепродати його під своїм брендом. Фактично реселер користується готовою технічною базою компанії. За символічну плату підприємець отримує повний контроль за сайтами клієнтів.

Перепродаж хостингу здійснюється із застосуванням спеціального ПЗ, процес виглядає досить просто:

- Реселер орендує у великого провайдера сервер, який у напівавтоматичному режимі встановлюється програмне забезпечення, необхідне реселінгу хостингу. Зазвичай пакет входить cPanel і WHM/WHMCS.
- Реселер створює своїм клієнтам облікові записи та надає доступ до панелі cPanel.
- Клієнти працюють із реселером на встановлених ним умовах, оплачуючи послуги через панель в автоматичному режимі. Оренда хостингу для них не відрізняється від роботи зі звичайним провайдером верхнього рівня.

По суті, реселінг є посередницькою діяльністю, в рамках якої реселер надає послуги під своїм брендом і встановлює власні тарифи. Йому не доводиться займатися обслуговуванням програмно-апаратної частини сервера, але спілкування з кінцевими клієнтами та техпідтримка на рівні хостингу повністю лягає на його плечі.

Реселер хостинг надає великі повноваження та є ідеальним варіантом для адміністраторів, розробників, веб-дизайнерів або для тих, хто має намір побудувати бізнес у сфері хостинг-послуг без великих інвестицій. А при відповідальному ставленні до бізнесу, дохід реселера може стати більш стабільним та матиме тенденцію до зросту [7].

Тож, серед переваг реселінгу можна виділити наступні:

- відмінний спосіб заробити для тих, хто не має достатньо коштів для заснування власного дата центру, але бажає випробувати власні сили у сфері продажу хостингу;
- будь-який користувач може заробити без глибоких пізнань у веб-технологіях, опираючись лише на наявні навички та знання;

– провайдери можуть робити знижки для реселерів, що може позитивно позначитися на розвитку бізнесу;
– для підтримки бізнесу не потрібно утримувати велику кількість персоналу.

До недоліків реселінгу можна віднести:

– через можливість достатньо просто відкрити свій власний бізнес, у нішу залучилось величезна кількість непрофесіоналів та шарлатанів, чиї послуги не витримують жодної критики за якісними показниками, що ускладнює процес отримання надійної репутації;
– стабільний та дохід з’явиться лише з часом;
– провайдер послуг може в будь-який момент змінити вартість послуг або змінити правила обслуговування клієнтів, тож потрібно бути уважним та постійно слідкувати за оновленнями провайдера.

1.1.5. Класифікація хостингу та критерії його вибору

У світі веб-хостингу існує багато варіантів, які дозволять розмістити сайт в Інтернеті. Проте кожен із них відповідає безпосередньо потребам власників сайтів – як великих, так і малих.

Веб-сайти розміщуються на серверах – потужних апаратних засобах, на яких зберігаються веб-сайти та дані, пов'язані з ними.

Кожен компонент вашого сайту зберігається на сервері та доступний через веб-хост. Сюди входять такі елементи як файли, текст, зображення, відео - все.

Сервери фізично розташовані у центрах обробки даних, які зазвичай управляються різними хостинговими компаніями.

Веб-хостинги надають технології та серверний простір, необхідні для доступу до вашого сайту в Інтернеті [8]. Це дозволяє користувачам шукати ваш сайт та переглядати ваші веб-сторінки в Інтернеті.

Всі вони виступають як місце зберігання даних для сайту, але вони відрізняються обсягом пам'яті, контролем, необхідними технічними знаннями, швидкістю і надійністю сервера. Список типів хостингу, з якими найчастіше зіткнутися розглянемо нижче:

1. Віртуальний хостинг (Shared hosting).

Загальний хостинг ідеально підходить для хостингу веб-сайтів початкового рівня. В цьому випадку сайт зберігатиметься на тому ж сервері, що і кілька інших сайтів. При використанні плану віртуального хостингу всі домени використовують ті самі ресурси сервера, такі як оперативна пам'ять (Random Access Memory) і центральний процесор (Central Processing Unit). Однак, оскільки всі ресурси є загальними, вартість планів віртуального хостингу відносно низька, що робить їх чудовим варіантом для власників сайтів на початковому етапі.

У більшості випадків користувачі-початківці знаходять загальний хостинг найпростішим способом розміщення свого сайту; тому незалежно від того, чи є ви власником малого бізнесу, громадської групи або мамою, що сидить вдома, бажає вести блог, ваш сайт буде доступний в Інтернеті. Плани загального хостингу часто поставляються з багатьма корисними інструментами, такими як: конструктори сайтів (відкриється в новій вкладці), хостинг WordPress (відкриється в новій вкладці) та можливість роботи з поштовими клієнтами (відкриється в новій вкладці).

2. Керований хостинг.

Більшість пакетів хостингу, які ви знайдете в Інтернеті, швидше за все, є керованими. Хостингові компанії надають технічні послуги, такі як встановлення та настроювання апаратного та програмного забезпечення, технічне обслуговування, заміна обладнання, технічна підтримка, виправлення, оновлення та моніторинг. При керованому хостингу

провайдер піклується про повсякденне управління обладнанням, операційними системами та стандартними програмами.

Хоча існує безліч різних варіантів хостингу, все зводиться до вибору тарифного плану, який відповідає вашим потребам. Кожен тарифний план відповідає вимогам різних груп, і усвідомлення того, які потреби у вас є на веб-сайті, допоможе вам переконатися, що ви обрали саме той тарифний план, який підходить вам та вашому бізнесу.

3. Колокейшн.

Замість того, щоб тримати сервери всередині компанії або приватному центрі обробки даних, ви можете вибрати "спільне розміщення" вашого обладнання, орендувавши місце в центрі колокейшн. Центр забезпечить живлення, пропускну здатність, IP-адресу та системи охолодження, необхідні вашому серверу. Простір здається в оренду у стійках та шафах.

Колокейшн дає доступ до вищого рівня пропускної спроможності, ніж звичайна офісна серверна кімната за набагато нижчою ціною. Ви надані самі собі (в буквальному сенсі) і повинні будете подбати про все, включаючи обладнання, програмне забезпечення та послуги [9].

4. Хмарний хостинг:

Хмарний хостинг – актуальне слово у технологічній індустрії. Стосовно веб-хостингу він означає безліч комп'ютерів, що працюють разом і запускають програми, використовуючи об'єднані обчислювальні ресурси. Це рішення хостингу, яке працює через мережу та дозволяє компаніям споживати обчислювальні ресурси як комунальні послуги.

Це дозволяє користувачам використовувати стільки ресурсів, скільки їм необхідно, без необхідності створювати та підтримувати власну обчислювальну інфраструктуру. Ресурси, що використовуються, розподіляються між декількома серверами, що знижує ймовірність простоїв через несправність сервера.

Хмарний хостинг масштабується, тобто ваш сайт може зростати з часом, використовуючи стільки ресурсів, скільки йому потрібно, а власник сайту платить тільки за те, що йому необхідно.

5. Хостинг виділених серверів.

Виділений хостинг дає власникам сайтів найбільший контроль над сервером, на якому зберігається їхній сайт. Це тому, що сервер орендується виключно вами, і тільки ваш сайт зберігається на ньому. Це означає, що у вас є повний адміністраторський доступ, тому ви можете контролювати все – від безпеки до операційної системи, яку ви запускаєте. Проте за цей контроль доводиться платити.

Вартість виділених серверів - один із найдорожчих варіантів хостингу. Як правило, вони використовуються власниками сайтів з високим рівнем відвідуваності та тими, хто потребує повного контролю над своїми серверами. Крім того, для встановлення та поточного керування сервером потрібен високий рівень технічних знань.

6. Віртуальний приватний сервер (VPS):

VPS-хостинг – це щось середнє між загальним та виділеним сервером. Він ідеально підходить для власників сайтів, яким потрібно більше контролю, але не обов'язково потрібний виділений сервер.

VPS-хостинг унікальний тим, що кожен сайт розміщується у власному просторі на сервері, хоча він, як і раніше, ділить фізичний сервер з іншими користувачами. Хоча VPS-хостинг надає власникам сайтів більше можливостей для налаштування та простору для зберігання даних, він все ж таки не здатний обробляти неймовірно високі рівні трафіку або скачки у використанні, що означає, що на продуктивність сайту можуть впливати інші сайти на сервері.

Як правило, VPS-хостинг використовується власниками сайтів, які хочуть мати виділений хостинг, але не мають необхідних технічних знань. VPS-хостинг пропонує економічні переваги віртуального

хостингу із контролем виділеного хостингу. Це відмінний вибір для досвідчених користувачів та тих, хто хоче встановити спеціальне програмне забезпечення та пакети.

Вивчивши перераховані види та характеристики хостингу, ви цілком самі зможете зупинити вибір на одному з них, керуючись виключно своїми потребами та фінансовими можливостями. Тим більше, що всі переваги та недоліки, які має кожен тип хостингу, було перераховано. Кожен тарифний план відповідає вимогам різних груп, і усвідомлення того, які ваші потреби на веб-сайті, допоможе вам переконатися, що ви вибираєте правильний тарифний план для себе та свого бізнесу.

Тож для початківця вебмайстера, цілком можна використовувати звичайний віртуальний вид хостингу, варіантів якого в інтернеті безліч [10]. Ну а якщо потрібний майданчик для розміщення серйозного проекту, варто задуматися про один з дорожчих і надійних варіантів.

Найважливішими факторами, які слід враховувати при виборі хостингу, є тип вашого сайту, необхідні ресурси, бюджет і очікуваний трафік. Ось короткий огляд переваг кожного типу хостингу:

- Віртуальний хостинг: Найбільш економічний варіант для сайтів із низькою відвідуваністю.
- Керований хостинг: Ідеально підходить для нетехнічних користувачів, які воліють відкласти технічні завдання для експертів.
- VPS-хостинг: Простіше кажучи, це найкращий варіант для сайтів, що переросли віртуальний (загальний) хостинг.
- Хмарний хостинг: Найкраще підходить для сайтів, які швидко ростуть і потребують масштабованих ресурсів.
- Виділений хостинг: Дорогий варіант для великих сайтів, де вам потрібно все контролювати.

– Колокаційний хостинг: Найдорожчий варіант, який дає вам максимальний контроль над апаратним та програмним забезпеченням.

Також при виборі хостингу слід звертати увагу на наступні параметри:

– обсяг дискового простору, що надається для розміщення сайту;
– стек програм та інструментарій для управління сайтом;
– організація тих. підтримки хостера;
– співвідношення вартості/якості послуг хостингу.

Обсяг дискового простору - на сьогоднішній день це дуже недорогий ресурс, тому реально підібрати такі тарифні плани, на яких хостери виділяють його з надлишком. Скрипти займають, як правило, невеликий об'єм дискового простору. Основне місце використовується медіаконтентом: зображення та відеофайли. Тому, якщо у вас сайт-візитка або електронний магазин з невеликою кількістю товарів, 1ГБ місця на жорсткому диску для розміщення сайту буде достатньо.

Серед типів дискового накопичувача - виділяють три типи: HDD, SSD та NVMe SSD [11]:

– HDD - це найстаріший і найповільніший варіант дисків, але при цьому ціна одного гігабайта найнижча. Такий тип зберігання даних варто вибирати, якщо швидкість доступу до інформації не має значення і важливіший обсяг.
– SSD - диски цього типу в 4-5 разів швидше, ніж HDD. Особливо це позначається під час читання невеликих файлів. Місце на них коштує дорожче. SSD найчастіше використовуються на серверах завдяки сумісному інтерфейсу підключення з HDD дисками та економічним енергоспоживанням.
– NVMe SSD – найпродуктивніший тип дисків. Перевага у швидкості в порівнянні з HDD досягає 10-12 разів. Однак це найдорожчий тип носія. Для його підключення на сервер потрібно додаткове

обладнання, тому NVMe SSD менш поширені. Доцільно використовувати такі диски у місцях, де швидкість це пріоритет. Наприклад, на серверах бази даних.

Центральний процесор (CPU) та кількість виділених ядер – для недорого віртуального хостингу такий показник відсутній, його можна зустріти на бізнес-хостингу. Зазвичай CPU характеризується частотою процесора на сервері та наявністю виділених ядер. Що частота процесора, то швидше виконується операція. Якщо виділені ядра процесора не вказані, процесорний час поділяється між усіма користувачами сервера. Тому, якщо сусід по серверу надто активно використовує процесор, решта сайтів сервера працюватиме повільніше.

Регіон розміщення серверів провайдера - якщо цільова аудиторія сайту знаходиться в Україні, не варто розміщувати проект на серверах США. Чим більша відстань від відвідувача до сервера, тим повільніше завантажуватиметься сайт. Для пошукових систем локація сервера допомагає визначити регіон, до якого належить проект.

Далі слід звернути увагу на інструментарій та стек програмного забезпечення, який надає вам хостер, а саме:

- Трафік - обсяг інформації, який може бути переданий та отриманий сервером, де розміщено сайт. Для невеликого корпоративного сайту цей параметр можна ігнорувати, але для великих проектів він може бути важливим. Додатковий трафік або надаватися за доплату або швидкість доступу буде обмежена після досягнення ліміту.
- Підтримка необхідного програмного стеку - Стандартний віртуальний хостинг зазвичай підтримує такі технології: PHP, MySQL, PostgreSQL, phpMyAdmin. Цього достатньо для розміщення сайту, створеного на таких популярних CMS, як WordPress, Joomla, Drupal, "1С-Бітрікс". Іноді сайт може мати специфічні вимоги, наприклад, необхідну мову програмування Python, або брокер

повідомлень RabbitMQ, або потрібен певний набір модулів PHP. Тому перед вибором хостингу варто ознайомитись із технічними вимогами програмного продукту, розміщеного на хостингу.

- Доступ до сайту по FTP та SSH - FTP – стандартний метод роботи з файлами на хостингу. Для нього можна використовувати FTP клієнт або webftp. SSH та SFTP – використовуються більш просунутими користувачами. Ці протоколи дозволяють працювати з файлами безпосередньо на сервері. За допомогою SSH можна виконати додаткове налаштування та встановлення програм усередині свого віртуального середовища.
- Електронна пошта - Тут варто звернути увагу на кількість і розмір поштових скриньок, а також протоколи, за якими можна працювати з поштою. POP3 та SMTP – це стандартні протоколи приймання та надсилання пошти. Зручно, щоб ще був IMAP. На відміну від Pop3 цей протокол дозволяє працювати з поштою прямо на сервері, розподіляти її по папках і створювати фільтри. Є ще додаткові функції, які може продати провайдер по роботі з поштою. Наприклад, на хостингу FREEhost.UA доступні такі функції: автовідповідачі, переадресація вхідних листів, спам-фільтр, антивірус, web-пошта, спільний календар та записник для користувачів домену, імпорт пошти з іншого сервера.
- Резервне копіювання - Бекап треба робити завжди. Це золоте правило всіх досвідчених програмістів та адміністраторів. Тому, вибираючи хостинг, зверніть увагу на наявність цієї послуги на період зберігання резервних копій, і чи бекапи включені в дискову квоту акаунта.
- Безкоштовний SSL-сертифікат - Сертифікат SSL використовується для безпечного підключення до сайту. Це обов’язково для будь-якого

сайту, що працює з персональними даними користувачів та електронних магазинів.

- Технічна підтримка – один з основних факторів, що впливають на якість та вартість послуги. Зазвичай технічна підтримка включена до комерційних тарифних планів віртуального хостингу. Для послуг VPS та оренди сервера підтримка ділиться на кілька рівнів, залежно від типу та складності запитів. Базова підтримка зазвичай входить у вартість тарифного плану, а розширена є платною послугою.

Для VPS та оренди сервера варто уточнити, які саме послуги входять до базової та розширеної підтримки:

- Аптайм - це період безперервної доступності хостингу (сайту або сервера). Даунтайм (downtime) - це час простою веб-ресурсу через технічний збій на сервері. Звичайно, будь-який хостинг-провайдер прагне показника аптайму 100%, але досягти його на практиці нереально. Користувачеві варто вибирати хостинг із показником Uptime не менше 99,9%. Існує класифікація Tier для визначення рівня надійності ЦОД, яка допоможе вибрати найбільш надійний сервер для розміщення вашого веб-проекту. Найкращий показник аптайму (Tier 4) – це 99,995% (простий сервер приблизно 0,4 години на рік).
- Об'єм оперативної пам'яті — кількість оперативної пам'яті, яка виділяється під ваш веб-ресурс. Як правило, на дешевих тарифних планах Shared-хостингу може не вказуватися, але при замовленні VIP-планів, VPS або виділеного фізичного сервера фахівці хостинг-провайдера обов'язково вкажуть цей параметр. Для будь-якого (не статичного) сайту на CMS необхідно від 256 Мб RAM.
- PHP memory_limit (МБ) – обсяг даних, які PHP-процес може зберегти в оперативній пам'яті. Цей параметр може впливати на швидкість роботи сайту. Тут необхідно розуміти різницю та співвідношення з

попереднім параметром. Наприклад, високий показник PHP memory_limit дозволяє обробити складну операцію або великий набір класів, але якщо вам виділено невеликий загальний обсяг пам'яті (RAM), то на обробку другого процесу може ресурсів і не вистачить.

- Кількість сайтів - кількість сайтів, які ви можете розмістити на хостингу (не враховуючи піддомени).
- Час відгуку - це тимчасовий період (у мілісекундах) між моментом відправлення першого байта інформації (за протоколом http методом GET) і моментом отримання першого байта у відповідь від сервера.
- Бази MySQL - число баз даних MySQL, яке хостер дозволяє розмістити згідно з вашим тарифним планом (як правило, дорівнює кількості сайтів, але може бути і більше за нього). Це важливий параметр, який варто звертати увагу, оскільки весь динамічний контент сайту зазвичай зберігається в базі даних. Наприклад, сайт на CMS WordPress або Joomla запустити без бази MySQL не можна. Зазвичай, для кожного сайту використовується окрема база даних. Тому кількість баз даних у тарифі не повинна бути меншою за кількість сайтів.
- HTTPS (видача безкоштовних сертифікатів SSL) — підтримка передачі даних із захищеним протоколом https.
- Memcache – сервіс для додаткового кешування інформації в оперативній пам'яті. Використовується для підвищення швидкості доступу до даних, що часто використовуються, і можливості обміну інформацією між скриптами.
- IPv6 – наявність підтримки нової версії протоколу IP.
- Індекс продуктивності - це параметр в умовних одиницях, що визначає обсяг обчислювальної потужності, доступний для конкретного облікового запису на хостингу. Чим більший цей

параметр, тим більше ресурсів можна використовувати для сайтів користувача, щоб зробити їх роботу швидше.

Також, на першому етапі необхідно визначитися, які завдання вирішуватиме новий веб-проект (інтернет-магазин, корпоративне веб-додаток, резервне копіювання, робота з БД, поштовий сервер, веб-портал, сховище офісних документів тощо). Залежно від виду завдань та підбирається обладнання. Можливо, вам знадобиться більша частота процесора або кількість ядер більше або більше оперативної пам'яті, тут необхідно врахувати всі аспекти.

Оперативна пам'ять: останнім часом ціни на даний ресурс стали менш високими, тому радимо вибирати конфігурації з великим обсягом оперативної пам'яті - так додатків буде простіше зберігати найбільшу кількість даних.

Пропускна спроможність мережі: як правило, хостинг-провайдери можуть запропонувати порти Ethernet на 100 Мбіт/с та 1 Гбіт/с. Ці порти можна використовувати як виходу в інтернет, так організації локальної мережі. У цьому випадку використання портів на 1G або 10G також залежить від ваших завдань. Наприклад, ви рознесли базу даних, сайт і корпоративну пошту по окремих серверах, в результаті навантаження на кожен сервер однотипне, і програми не крадуть один в одного ресурси, тому сервери ефективно працюють. Але після винесення сервера БД на окрему машину може знизитися швидкість взаємодії програми з базою через латентність мережі (під "мережевою латентністю" мається на увазі тимчасова затримка, яка виникає при передачі даних або запиту, що посилається по мережі). Проблема вирішується заміною мережевих інтерфейсів з 1G на 10G у тому, щоб знизити що така затримки.

Вибір процесора: у разі необхідно враховувати 2 характеристики — частоту і кількість ядер. Є завдання, які вирішуються саме на процесорах із великою кількістю ядер. Наприклад, для веб-сервера NGINX важлива саме

кількість ядер. Для вирішення завдань із віртуалізації або обробки відео також радимо вибирати сервер із великою кількістю ядер. Якщо ви плануєте запускати процеси, де потрібна багатопоточність та паралельні обчислення, то кількість ядер процесора для вас дуже значущий параметр, а ось частота процесора вже не така важлива.

З іншого боку, є завдання та процеси, де кількість ядер зовсім не важлива, а от частота процесора має пріоритет. Наприклад, MySQL погано працює з багатопоточністю, тому краще вибрати процесор із більшою частотою, а кількість ядер може бути невеликою. Аналогічним чином можна орієнтуватися при роботі з 1С: вибирайте процесори з більшою частотою, але меншою кількістю ядер.

Якщо ж у вас проект дуже бюджетний і для сервера не потрібен потужний процесор, то можна заощадити кошти та вибрати менш продуктивні сервери з попередніх поколінь.

Об'єм дискового простору (HDD/SSD): при виборі дискових накопичувачів слід звернути увагу на такі фактори: час доступу до диска, його ємність та ціну. Ці фактори взаємопов'язані між собою. Наприклад, вартість SATA-дисків невисока, хостери пропонують дешево SATA-диски на кілька терабайт, проте такі диски мають великий час доступу. SSD мають невеликий час доступу, але поки що дорогі. Тому наша порада: для організації завантажувального розділу слід вибрати невеликі за обсягом SSD, те саме можна зробити і для зберігання БД. А ось для організації файлових сховищ з великими обсягами статичних даних або резервних копій можна сміливо вибирати великі за обсягом SATA.

Якщо для вас важлива збереження ваших даних, то варто звернути увагу на RAID-масиви, які служать для об'єднання декількох дисків в один модуль, для підвищення рівня стійкості до відмов і швидкості запису, а також читання даних. При відмови одного або навіть кількох SATA або SSD, вам все одно будуть доступні ваші дані.

Можна вибрати апаратну реалізацію RAID (HARD RAID) або програмну (SOFT RAID). Також є стандартні рівні RAID-масивів, наприклад, RAID 0, RAID 1, RAID 5, RAID 10.

Перш ніж вибирати сервер, користувач повинен ознайомитися з порадами, а також:

- перед замовленням послуги хостингу звернути увагу на надійність хостингової компанії, наявність у неї свого дата-центру та обладнання, що надається, а також на співвідношення ціни та якості;
- визначитись із завданнями, які вирішуватиме ваш інтернет-проект;
- вивчити основні параметри хостингів, характеристики серверів та тарифні плани хостерів, виробити вимоги до сервера та скласти оптимальну конфігурацію.

1.2. Призначення розробки та галузь застосування

Як об'єкт впровадження розроблюваної спеціалізованої системи подачі заявки на підключення сервера розглядається універсальне програмне забезпечення на базі мови програмування Ruby з використанням фреймворку Ruby on Rails та системи управління базами даних PostgreSQL для виконання операцій вибірки, сортування та фільтрації даних для розширення основного функціоналу сайту.

Запропоноване програмне забезпечення дозволяє:

- ознайомитися зі списком послуг та товарів, представлених на сайті, сортуючи в будь-якому зручному порядку;
- оформити заявку на підключення виділеного або віртуального сервера;
- додавати, видаляти та редагувати будь-які дані, які стосуються представлених на сайті послуг завдяки адміністративній панелі ActiveAdmin;

- встановити зв'язок з білінговою панеллю WHMCS для зручного перегляду статистики, зміни характеристик виділеного або віртуального серверу та здійснення оплати послуг
- надати можливість проектувальникам інфраструктури індивідуального автоматизованого доступу до серверів;
- підвищити швидкість створення проектів прихильниками малого та великого бізнесу завдяки зручному інтерфейсу;
- надати можливість швидкої масштабованості серверу;
- скоротити витрати, час та ресурси шляхом придбання повністю керованих та самоврядних серверів;
- дозволити розробникам зосередитись на дизайні та коді проекту, а не на налаштуванні та підтримці серверу.

Запропоноване програмне забезпечення дозволяє прихильникам малого та великого бізнесу легко та швидко обрати потрібні послуги з підключення до виділеного або віртуального сервера, подати заявку на відключення послуг, зв’язатися з представниками технічної підтримки задля вирішення будь-яких питань або проблем засобами інтеграції API WHMCS.

1.3. Підстава для розробки

Відповідно до ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- ОПП за спеціальністю 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18.05.2022 р;

– завдання на дипломний проект на тему «Розробка інформаційної системи для реселінгу хостингу серверів засобами фреймворку Ruby on Rails».

1.4. Постановка завдання

Останніми роками спостерігається динамічне зростання попиту на віддалені сервери, які є альтернативою фізичному обладнанню, яким компанія оснащує офіс, але більше така тенденція поширюється на представників крупного бізнесу. Робота з ними передбачає наявність віддаленого доступу для різних користувачів за допомогою Інтернету. Сам сервер, який, по суті, є комп'ютером, може розташовуватися або в самій фірмі, або в будь-якому іншому місці, наприклад, у хостинг-провайдера.

Крім того, відмінність видаленого серверу у тому, що він підвищує рівень безпеки, оскільки знаходиться у захищених data-центрах. Підключаючись до нього, компанія переносить у нього та зберігає всю інформацію, а також програмне забезпечення, листування та ін.

Перехід на віддалений сервер дозволяє вирішити низку завдань:

- організувати роботу IT-інфраструктури максимально ефективно;
- максимально підвищити рівень захисту корпоративної інформації;
- уникнути витрат на модернізацію застарілого обладнання та загалом знизити витрати на обслуговування IT-інфраструктури;
- швидко реагувати на позаплановий аудит, наприклад, готувати інформацію;
- скоротити штат системних адміністраторів;
- скоротити витрати на придбання обладнання, у тому числі серверів баз даних, файлових, термінальних, різноманітних сховищ;

При цьому сервер віддаленого доступу у хмарі підтримує два режими роботи:

1) Віддалене управління. Клієнти дистанційно керують роботою сервера (його називають сервером терміналів).
2) Віддалений доступ до мережі. Користувачі підключаються так, ніби він знаходиться у спільній локальній мережі. Функціонально він працює так само, як усі комп'ютери в корпоративній мережі. У цьому режимі з віддаленим доступом до мережі користувачі використовують встановлені на комп'ютері клієнтські програми доступу та здійснюють підключення. Він перевіряє права доступу, потім дозволяє або забороняє його. Потім обслуговує сеанс зв'язку на період підключення, перш ніж його перерве адміністратор або користувач.

Для підключення у більшості випадків використовується Remote Desktop Protocol (скорочено RDP). Його популяризувала Microsoft, оскільки корпорація рекомендує саме його підключення термінальних машин. Клієнти при цьому розроблені з урахуванням всіх версій ОС компанії, навіть мобільних. Але навіть якщо ви не використовуєте Windows, цей протокол може підійти, тому що є версії для інших ОС - Android, iOS, Linux і т.д. Якщо робочий стіл знаходиться на віддаленому сервері в інтернеті, можна скористатися функціоналом, розробленим корпорацією Microsoft. Але для цього потрібно знати IP-адресу сервера.

Тож для компанії дуже важливо забезпечити безпечне зберігання даних і водночас надати доступу до них великому числу користувачів, тобто співробітникам. Видалений сервер — це можливість дотримуватися обох цих вимог. Він дозволяє отримати доступ за допомогою глобальної мережі.

Метою кваліфікаційної роботи є розробка інформаційної системи хостингу для перепродажу серверів з використанням фреймворку Ruby on Rails та системи управління базами даних PostgreSQL. Розроблений веб-застосунок допоможе власникам малого та великого бізнесу дати можливість зручного вибору віртуальних та видалених серверів для подальшої оренди,

підвищити швидкість та якість розробки власного продукту, зменшити фінансові витрати завдяки відсутності потреби утримувати штат адміністраторів, гнучкого масштабування завдяки зміненню характеристик орендованого серверу через білінгову панель WHMCS.

Для досягнення поставленої в роботі мети необхідно виконати наступні завдання:

- здійснити аналіз предметної галузі з метою виявлення сучасного стану обраної проблематики;
- описати архітектуру бази даних та виконати аналіз щодо її нормалізації;
- здійснити аналіз існуючих хостингових компаній з метою виявлення їх властивостей, потенційних можливостей та недоліків;
- здійснити аналіз існуючих білінгових панелей з метою виявлення можливостей взаємодії з нею через API-інтерфейс;
- здійснити вибір інструментальних засобів для використання API-інтерфейсу білінгової панелі;
- розглянути існуючий механізм взаємодії власного веб-сайту з білінговою панеллю.

1.5. Вимоги до програми або програмного засобу

Вимоги до програмного забезпечення - набір вимог до якості, властивостей та функцій програмного забезпечення, що буде знаходитися у процесі розробки або вже розробляється. Вимоги визначаються у процесі аналізу вимог та фіксуються у специфікації вимог, діаграмах прецедентів та інших артефактах аналізу та розробки вимог.

В якості головних критеріїв надійності та ефективності запровадження програмного продукту є якість та надійність - недопустимість виникнення помилок та перебоїв під час роботи системи.

Слідуючи уявленням про кінцевий продукт, можна сформувати список вимог до програми:

- продукт має бути зручним у використанні, мати зрозумілі повідомлення про помилки, поводити себе відповідно до очікувань користувача, легко та швидко виконувати поставлені задачі та мати інтуїтивно зрозумілий інтерфейс для користувача;
- продукт має бути зрілим та завершеним, мати великий показник часу роботи без помилок та збоїв, підтримувати заданий рівень працездатності при порушенні правил взаємодії з інтерфейсом користувачем, відновлювати зазначений рівень цілісності даних після критичних помилок та запобігати неавторизованому і недозволеному доступу до даних;
- продукт має вирішувати потрібний набір задач та виконувати свою головну роль;
- продукт має мати якісну кодову базу задля запобігання небажаних збоїв та гнучкості системи, використовувати нові технології задля спроможності впровадження трендових та актуальних до часу технічних рішень.

1.5.1. Вимоги до функціональних характеристик

В програмі для досягнення поставленої мети повинні бути реалізовані наступні пункти:

- можливість ознайомитися з представленими послугами на сайті рісейлінгової компанії;
- можливість сортування за необхідними характеристиками серверів, представлених на сайті, завдяки зв’язку з системою управління базами даних PostgreSQL;

- можливість перенаправлення користувача до білінгової панелі WHMCS задля перегляду більш детальної інформації, зміни та оплати необхідних тарифів;
- можливість навігації користувача по сторінкам сайту;
- можливість перенаправити користувача до білінгової панелі WHMCS з метою зареєструватися або увійти до свого акаунту.

1.5.2. Вимоги до інформаційної безпеки

Задля уникнення критичних помилок, збоїв та переривань системи необхідно звернути увагу на наступні пункти та реалізувати:

- обробку виняткових ситуацій;
- використання https з’єднання;
- використання SSL сертифікату;
- можливість повторного введення даних після невдалої спроби;
- виведення повідомлень про наявність помилок;
- перехоплення невдалих відповідей з бази даних та сповіщення про це користувачу;
- написання інтеграційних та модульних тестів задля впевненості у коректній роботі застосунку.

1.5.3. Вимоги до складу та параметрів технічних засобів

Задля нормального функціонування веб-застосунку, повинні виконуватися наступні вимоги до технічних засобів, на яких планується взаємодія з додатком:

- клавіатура;
- маніпулятор типу «миша»;
- монітор;

- доступ до мережі «Інтернет»;
- операційна система Windows 7, Windows 8, Windows 8.1, Windows 10 або пізніші версії та процесор Intel Pentium 4 або пізнішої версії із підтримкою SSE3;
- відеопам’ять: 64 MB;
- оперативна пам'ять: 512 MB;
- жорсткий диск: 512 MB
- операційна система macOS X El Capitan (10.11) або пізніші версії;
- операційна система Ubuntu 18.04 (64-розрядна версія) або пізнішої версії, Debian 10 або пізнішої версії, openSUSE 15.2 або пізнішої версії, Fedora Linux 32 або пізнішої версії та процесор Intel Pentium 4 або пізнішої версії із підтримкою SSE3;
- операційна система Android 6.0 Marshmallow або пізнішої версії;
- операційна система iOS 12 або новішої версії.

Завчасно повинен бути встановлений браузер задля перегляду веб-додатку (Google Chrome, Internet Explorer, Edge Browser, Opera, FireFox або інший).

Наведені вище технічні характеристики є рекомендованими, тож при наявності технічних засобів не нижче зазначених, розроблений веб-застосунок буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування веб-застосунку необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати застосунок, відповідало наступним вимогам:

- Windows 7, Windows 8, Windows 8.1, Windows 10 або пізніші версії;
- macOS X El Capitan (10.11) або пізніші версії;

- Ubuntu 18.04 (64-розрядна версія) або пізнішої версії, Debian 10 або пізнішої версії, openSUSE 15.2 або пізнішої версії, Fedora Linux 32 або пізнішої версії;
- Android 6.0 Marshmallow або пізнішої версії;
- iOS 12 або новішої версії.

Веб-додаток має бути реалізовано на мові програмування Ruby з використанням фреймворку Ruby on Rails та системи управління базами даних PostgreSQL.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Сьогодні кожному бізнесу потрібна присутність в Інтернеті. Незалежно від типу пропонованих вами товарів та послуг чи галузі, в якій ви працюєте, ефективний бізнес-сайт має вирішальне значення для вашого успіху, тож він являється обов'язковим атрибутом ведення будь-якого виду бізнесу та є повноцінним представленням компанії в інтернеті. Корпоративний сайт або сайт компанії належить підприємству для спілкування з відвідувачами сайту. Корпоративний сайт містить такі матеріали, як профіль компанії, опис пропонованих продуктів та послуг, новини або оголошення компанії, а також контактну інформацію.

Такий сайт формує авторитет компанії, залучає нових клієнтів, розширяє ринок збуту, допомагає оперативно отримати дані про ціни товарів або послуг в зручному вигляді. А завдяки наявності функціоналу здійснення покупки, це збільшить прибутковість. Корпоративний сайт є повноцінним представленням Вашої компанії в інтернеті. Мета створення корпоративного сайту – довести інформацію про компанію, її діяльність, товари або послуги, до відома якомога більшої кількості людей, в яких зацікавлена компанія як в майбутніх споживачах або партнерах.

Також сайт містить розділи, на яких міститься корисна для потенційного користувача інформація, а саме:

- «Главная»: сторінка, де користувач може ознайомитися з основними послугами сайту, перейти на інші сторінки, дізнатися більш детальну інформацію про певну послугу або отримати відповіді на загальні поширені питання з приводу наданих послуг;
- «Выделенные серверы»: розділ, де користувач може дізнатися про основні переваги використання виділених серверів, обрати потрібний

сервер завдяки зручному інструменту фільтрації послуг та перейти до білінгової панелі для подальшої оплати послуги;

- «Виртуальные серверы»: розділ, де користувач може дізнатися про основні переваги використання віртуальних серверів, ознайомитися з тарифами та отримати відповіді на поширенні питання щодо VPS-хостингу;
- «Контакты»: розділ, що містить різноманітні варіанти для зв’язку;
- «Про нас»: розділ, що включає інформацію про діяльність хостингової платформи, містить посилання на офіційні акаунти у соцмережах та посилання на сторінки для ознайомлення з політикою конфіденційності та публічною офертою;
- «Политика конфиденциальности»: розділ, який описує, як може збиратися та використовуватися інформація про незареєстрованих відвідувачів та зареєстрованих користувачів, включаючи клієнтів, у зв'язку з їх доступом та використанням сайту та його під доменів;
- «Публичная оферта»: розділ, який є публічним договором приєднання та відповідно до чинного законодавства України має належну юридичну силу;

Тож серед можливостей корпоративного сайту можна виділити наступні:

- підтвердження серйозності і перспективності, формує позитивний імідж;
- репрезентація компанії;
- оперативний зворотний зв’язок з клієнтами по всьому світу;
- найоптимальніший спосіб реклами, оскільки вартість реклами в Інтернеті значно нижче, ніж вартість реклами в ЗМІ;
- можливість залучення нових клієнтів і партнерів;
- підвищення лояльності існуючих клієнтів, вигідно позиціонує компанію в очах майбутніх клієнтів;

- зручність доступу до комерційної інформації;
- відвідувач може прямо на сайті переглянути каталог продукції компанії, детально дізнатися про товари, подивитися фотогалерею, прайс-лист.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області розв'язуваної задачі не передбачають застосування математичних методів, при розробці веб-орієнтованої інформаційної системи для реселінгу хостингу серверів математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

В основі розробки веб-застосунку було використану мову програмування Ruby версії 2.7.2 та фреймворк Ruby on Rails. Це динамічна мова програмування з відкритим вихідним кодом, орієнтована на простоту та продуктивність. Він має елегантний синтаксис, який звичайно читати і легко писати.

Ruby входить у першу десятку за більшістю індексів, що вимірюють зростання та популярність мов програмування у світі (наприклад, індекс TIOBE). Певною мірою це зростання пояснюється популярністю програмного забезпечення, написаного на Ruby, зокрема веб-платформи Ruby on Rails. У Ruby реалізовано незалежну від ОС потокову обробку даних. Таким чином, для всіх платформ, на яких працює Ruby, ви також маєте багатопоточність, незалежно від того, підтримує її ОС чи ні, навіть у MS-DOS. Ruby дуже портативний: він розроблений в основному на GNU/Linux, але працює на багатьох типах UNIX, MacOS, Windows, DOS, BeOS, OS/2 і т.д. Ruby також повністю безкоштовний. Не тільки

безкоштовний, але й вільний для використання, копіювання, зміни та розповсюдження.

Тож відповідаючи на питання, чому саме Ruby, можна привести наступні пункти:

- об'єктно-орієнтована мова;
- простота та лаконічність мови;
- багато різних бібліотек високої якості;
- висока швидкість розробки;
- культура та стандарти;
- TDD/BDD;
- велике ком'юніті.

З приводу Ruby on Rails можна сказати, що це серверний фреймворк для розробки веб-додатків, написаний мовою програмування Ruby. Він розроблений для спрощення програмування веб-застосунків, роблячи припущення про те, що необхідно кожному розробнику для початку роботи, тож розробка веб-додатків стає більш цікавою. Він дозволяє писати менше коду і при цьому досягати більшого, ніж багато інших мов і фреймворків. Цей фреймворк використовує архітектурний шаблон MVC (Model-View-Controller) для проектування та розробки програмного забезпечення.

Rails заохочує та полегшує використання веб-стандартів, таких як JSON або XML для передачі даних та HTML, CSS та JavaScript для взаємодії з користувачем. На додаток до MVC, Rails підкреслює використання інших відомих моделей та парадигм програмної інженерії, включаючи «конвенцію над конфігурацією» (CoC), «не повторюйся» (DRY) та модель активного запису (Active Record).

Оскільки Rails активно використовує архітектурний патерн MVC (рис. 2.1), то він є невід’ємною частиною процесу розробки. Цей шаблон передбачає поділ системи на три взаємопов’язані компоненти – модель, представлення та контролер [19].

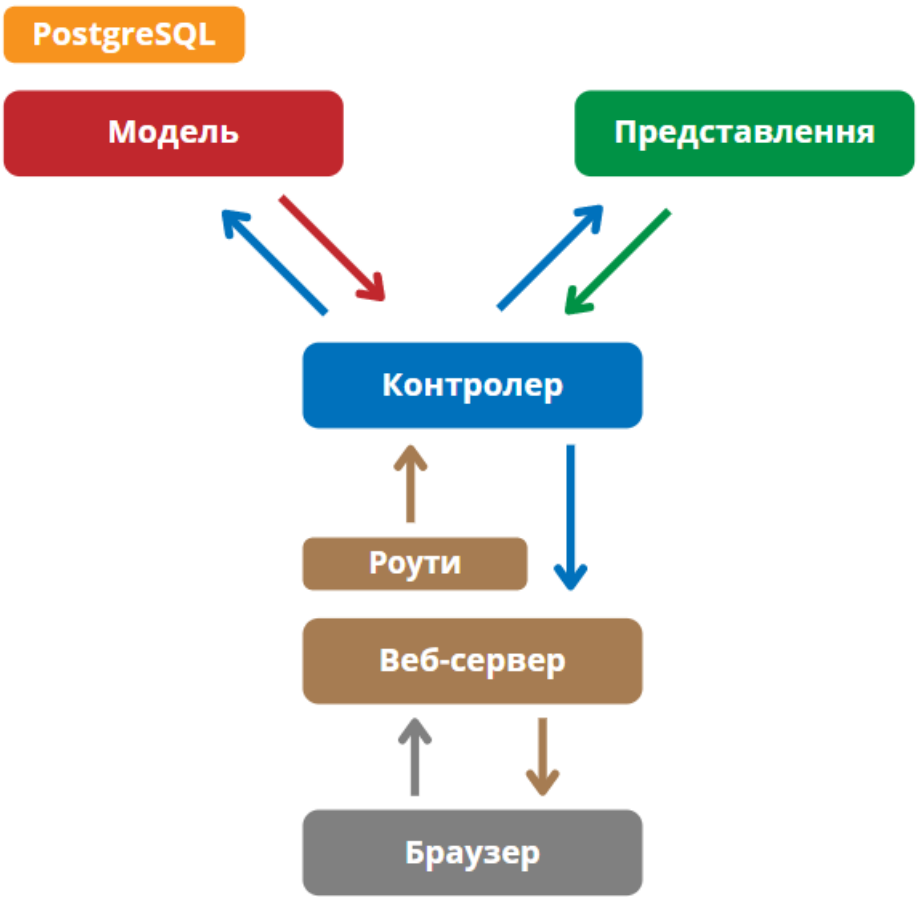

Рис. 2.1. Зображення взаємодії компонентів MVC

Active Record - це «M» в MVC - модель - шар системи, що відповідає за представлення бізнес-даних та логіки. Це реалізація патерна Active Record, який є описом системи об'єктно-реляційного відображення (ORM). Active Record дає нам кілька механізмів, найважливішим з яких є можливість:

– представляти моделі та їх дані;
– представляти асоціації між цими моделями;
– представляти ієрархії наслідування через пов'язані моделі;
– перевіряти моделі перед їх збереженням у базі даних;
– виконувати операції з базою даних об'єктно-орієнтованим способом.

Під час розробки веб-застосунку роль моделі виконує база даних серверу WHMCS.

Представлення - це «V» у MVC. Представлення відповідає за відображення даних та взаємодію користувача. Воно не виконує жодних

логічних операцій. Тільки відображення даних. Тобто код компонента view визначає зовнішній вигляд програми та способи її використання.

Контролер - це "C" у MVC. Цей компонент відповідає за зв'язок між моделлю та представленням. Після того як маршрутизація визначила, який контролер використовуватиме для запиту, ваш контролер відповідає за осмислення запиту та створення відповідного виводу. Як правило, на рівні контролера здійснюється фільтрація отриманих даних та авторизація - перевіряються права користувача на виконання дій або отримання інформації.

Одним з принципів розробки програмного забезпечення, який нав’язує Rails є «DRY» (Don't Repeat Yourself). Він припускає, що писати той самий код знову і знову - це погано, тож націлений на зниження повторення різноманітної інформації, особливо у системах з безліччю шарів абстрагування.

Інший принцип називається «CoC» (Convention over Configuration). Конвенція замість конфігурації - означає, що Rails робить припущення про те, що ви хочете зробити і як ви збираєтеся це зробити, замість того, щоб вимагати від вас вказівки кожної дрібниці через нескінченні конфігураційні файли. Це дуже поліпшує та прискорює процес розробки програмного забезпечення.

Також під час розробки було використано «BDD» (Behaviour Driven Development) та «TDD» (Test Driven Development) принципи. Розробка на основі тестування (TDD) зазвичай включає написання тесту для певної функціональності, виконання тесту, щоб побачити його невдачу, а потім написання коду, щоб тест пройшов (рис. 2.2). Таким чином, розробники можуть бути впевнені, що вони написали код, який виконує свою роботу, інші розробники, що використовують компоненти, можуть запустити тест, щоб бути впевненими, що їх власний код функціонуватиме належним чином.

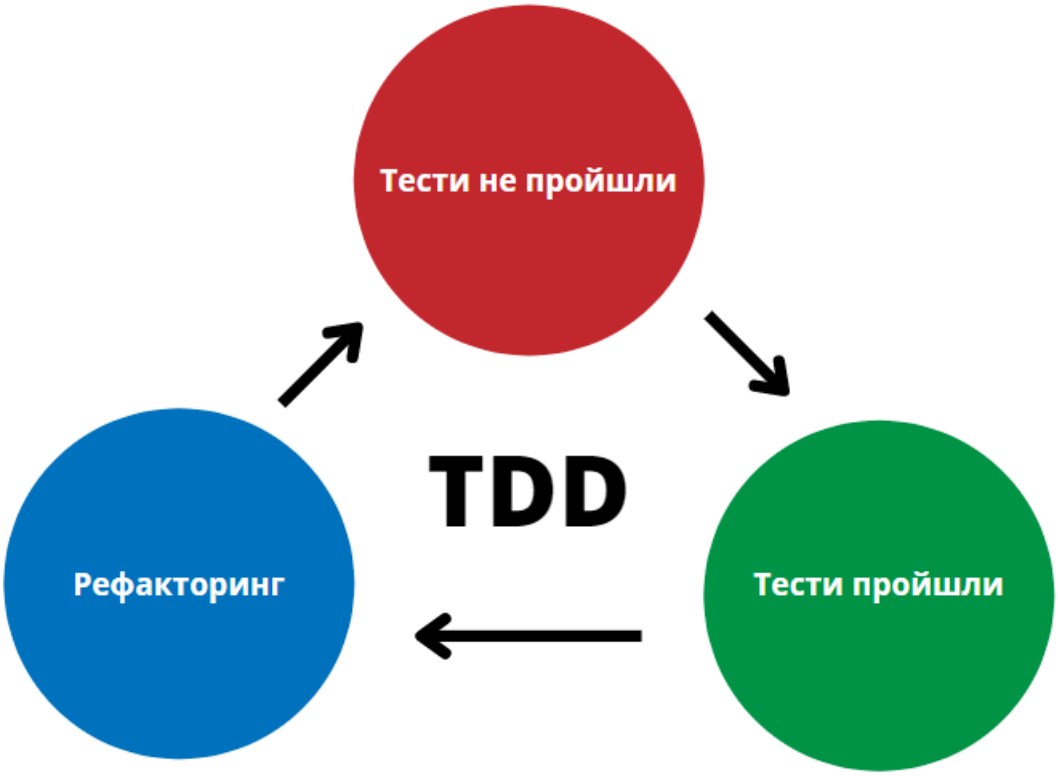

Рис. 2.2. Процес розробки програмного забезпечення з використанням підходу «TDD».

У поведінково-орієнтованій розробці (BDD) зазвичай беруть участь розробник, інженер з тестування та менеджер продукту (і, можливо, інші зацікавлені особи). Група збирається, щоб придумати конкретні приклади критеріїв приймання в історії користувача. Ці приклади описуються за допомогою мови, специфічної для конкретної області, і поміщаються у файл характеристик. Файл характеристик перетворюється на специфікацію, на основі якої розробники можуть написати реальний виконуваний тест.

BDD і TDD можуть здатися дуже схожими, оскільки вони обоє являють собою стратегії тестування програмної програми [21]. В обох випадках розробник пише тест до написання коду, щоб пройшов тест. І в обох випадках тести можуть бути використані як частина автоматизованої системи тестування для запобігання помилкам. BDD призначений для тестування поведінки програми з погляду кінцевого користувача, тоді як TDD сфокусований на тестуванні невеликих фрагментів функціональності в ізоляції.

У BDD беруть участь менеджери з продукту, розробники та інженери з тестування, які спільно розробляють конкретні приклади бажаної функціональності. Існує високий рівень комунікації перед будь-якою реалізацією. Для порівняння, TDD може бути проведена розробником-одинаком без будь-якого зовнішнього вкладу з боку менеджерів продукту або зацікавлених сторін.

Важливо відзначити, що BDD та TDD не виключають один одного - багато команд Agile використовують TDD без застосування BDD. Однак BDD гарантує, що більшість сценаріїв використання програми працюють на вищому рівні та забезпечують більший рівень впевненості.

Наприклад, команда розробників може використовувати BDD для розробки тестів вищого рівня, які підтверджують поведінку програми. При реалізації специфіки розробники можуть створювати окремі модульні тести для забезпечення надійності компонентів, тим більше, що ці компоненти можуть повторно використовуватися в інших місцях.

Rspec - відмінний приклад принципів TDD і BDD, об'єднаних у єдиний фреймворк додатків на Ruby.

Одним з найважливіших пунктів при розробці будь-якого програмного забезпечення являється безпека, тож задля того, щоб впевнитися у тому, що застосунок працює так, як очікується, було використано фреймворк для написання інтеграційних та юніт тестів – RSpec. У RSpec тести - це не просто скрипти, що перевіряють код вашої програми. Це також специфікації (або специфікації, для стислості): докладні пояснення того, як повинен поводитися додаток, виражені простою англійською мовою. Він дозволяє писати лаконічні та зрозумілі тести для будь-якої частини застосунку: контролер, модель, роути, представлення та ін. Тож, під час розробки було використано наступні типи тестування:

– юніт-тестування - це тип тестування програмного забезпечення, у якому тестуються окремі одиниці чи компоненти програмного

забезпечення. Мета – перевірити, що кожна одиниця програмного коду працює так, як очікується. Юніт-тестування проводиться під час розробки (фази кодування) програми розробниками. Тести блоків ізолюють ділянку коду та перевіряють його правильність. Одиницею може бути окрема функція, метод, процедура, модуль чи об'єкт;

– інтеграційне тестування - визначається як тип тестування, при якому програмні модулі логічно інтегруються та тестуються як група. Типовий програмний проект складається з кількох програмних модулів, тому мета цього рівня тестування - виявити дефекти у взаємодії між цими програмними модулями, коли вони інтегровані [20]. Інтеграційне тестування фокусується на перевірці обміну даними між цими модулями. Тому його також називають "I&T" (інтеграція та тестування), "тестування рядків" та іноді "тестування потоків".

В якості системи управління базами даних було використано PostgreSQL. PostgreSQL - це потужна об'єктно-реляційна система баз даних з відкритим вихідним кодом, яка використовує та розширює мову SQL у поєднанні з безліччю функцій, що дозволяють надійно зберігати та масштабувати найскладніші робочі навантаження даних. Серед переваг можна виділити наступні:

– відкритий вихідний код - це дозволяє вільно використовувати, змінювати та впроваджувати його відповідно до потреб бізнесу;
– безпека - існує безліч функцій для підвищення безпеки завдяки легкій розширюваності;
– зниження витрат - PostgreSQL нічого не вартий і абсолютно безкоштовний для використання;
– надійність - безліч компаній та приватних осіб роблять свій внесок у проект і стимулюють інновації вже понад 25 років. Сильна спільнота гарантує, що помилки виправляються без затримок;

– масштабованість - база даних PostgreSQL може зростати разом зі зростом проекту та його потреб і бути настільки великою, наскільки вам це потрібно оскільки існує багато технічних можливостей для масштабної експлуатації PostgreSQL.

Щоб запобігти написанню поганого коду та уникнути класичних помилок були використані статичні аналізатори коду. Це методи налагодження комп'ютерної програми, які здійснюються шляхом аналізу коду без виконання програми. Цей процес забезпечує розуміння структури коду і допомагає переконатися, що код відповідає галузевим стандартам. Автоматизовані інструменти можуть допомогти програмістам та розробникам у проведенні статичного аналізу. Процес ретельного вивчення коду лише шляхом візуального огляду (наприклад, шляхом перегляду роздруківки) без допомоги автоматизованих інструментів іноді називають розумінням програми або осмисленням програми. Тож у роботі були використані наступні інструменти:

– RuboCop - статичний аналізатор коду Ruby (лінтер), заснований на посібнику стилю спільноти Ruby. З коробки він виконуватиме багато рекомендацій, описаних у посібнику за стилем спільноти Ruby. Більшість аспектів поведінки можуть бути налаштовані за допомогою різних опцій конфігурації, що робить його дуже гнучким в руках розробника. Крім повідомлення про проблеми у коді, RuboCop може автоматично виправити деякі з них за вас автоматично;
– rubocop-rails - інструмент автоматичної перевірки стилю коду Rails. Розширення RuboCop, орієнтоване на впровадження найкращих практик Rails та угод з кодування;
– rubocop-rspec - перевірка стилю коду для файлів RSpec. Плагін для інструменту RuboCop, що забезпечує дотримання та лінтинг стилів коду;

- Fasterer - примушує програму працювати швидше за допомогою цього інструменту командного рядка. Fasterer пропонує деякі покращення швидкості для окремих компонентів системи;
- Bullet – допоможе збільшити продуктивність програми за рахунок зменшення кількості запитів, що виконуються. Також допомагає виявити та позбутися N+1 запитів і невикористовуваного нетерплячого завантаження;
- SimpleCov - покриття коду Ruby з потужною бібліотекою конфігурації та автоматичним об'єднанням покриття в тестових наборах. SimpleCov – це інструмент аналізу покриття коду для Ruby. Він використовує вбудовану в Ruby бібліотеку Coverage для збору даних про покриття коду, але значно спрощує обробку результатів, надаючи чистий API для фільтрації, групування, об'єднання, форматування та відображення цих результатів, надаючи вам повний набір для аналізу покриття коду, який можна налаштувати за допомогою всього пари рядків коду.

Pry-rails - це потужна альтернатива стандартній оболонці IRB для Ruby. У ній є підсвічування синтаксису, гнучка архітектура плагінів, виклик під час виконання, перегляд вихідних джерел та документації. Дуже корисний інструмент для кожного розробника, який дозволяє якісно виконувати налагодження коду.

Draper - додає об'єктно-орієнтований шар логіки подання у ваші програми Rails. Без Draper ця функціональність могла б заплутатися у процедурних помічниках або додати громіздкість у ваші моделі. За допомогою декораторів Draper ви можете обернути свої моделі логікою, пов'язаною з уявленням, щоб організувати – і протестувати – цей шар вашого додатка набагато ефективніший.

HTML, або HyperText Transfer Protocol, дозволяє користувачам Інтернету створювати та структурувати розділи, абзаци та посилання за допомогою елементів, тегів та атрибутів. Однак варто зазначити, що HTML

не вважається мовою програмування, оскільки вона не може створювати динамічні функції.

HTML має безліч варіантів використання, а саме:

– веб розробка - розробники використовують HTML-код для проектування того, як браузер відображає елементи веб-сторінки, такі як текст, гіперпосилання та медіа файли;
– навігація в Інтернеті - користувачі можуть легко переміщатися та вставляти посилання між пов'язаними сторінками та веб-сайтами, оскільки HTML активно використовується для вставки гіперпосилань;
– веб-документація - HTML дозволяє організовувати та форматувати документи, подібно до Microsoft Word.

Варто також зазначити, що HTML тепер вважається офіційним веб-стандартом. Консорціум World Wide Web Consortium (W3C) підтримує та розвиває специфікації HTML, а також регулярно оновлює їх. Тому під час розробки веб додатку HTML є невід’ємною частиною.

ERB (Embedded Ruby) - це функція Ruby, яка дозволяє зручно генерувати будь-який текст у будь-якій кількості шаблонів [12]. Самі шаблони поєднують у собі звичайний текст і код Ruby для підстановки змінних та управління потоком, що робить їх простими у написанні та супроводі.

Хоча ERB найчастіше використовується для створення веб-сторінок, він також застосовується для створення XML-документів, RSS-каналів, вихідного коду та інших форм структурованих текстових файлів. Він може бути надзвичайно корисним, коли вам потрібно створити файли, що включають безліч повторень стандартного шаблону, наприклад набори модульних тестів. Основний компонент ERB – це бібліотека, яку ви можете викликати у своїх додатках Ruby та задачах Rake.

Ця бібліотека приймає будь-який рядок як шаблон і не накладає жодних обмежень на джерело шаблону. Ви можете визначити шаблон повністю у своєму коді або зберігати його у зовнішньому місці та завантажувати при необхідності. Це означає, що ви можете зберігати шаблони у файлах, базах даних SQL або в будь-якому іншому сховищі, яке ви хочете використовувати.

Дистрибутиви Ruby також включають утиліту командного рядка, яка дозволяє обробляти шаблони, що зберігаються у файлах, без написання додаткового коду. Логічно ця утиліта називається erb. ERB є частиною стандартної бібліотеки Ruby. Для її використання не потрібно інсталювати жодне інше програмне забезпечення.

Каскадні таблиці стилів (CSS) - це мова таблиць стилів, що використовується для опису представлення документа, написаного мовою розмітки, такою як HTML.

Назва "каскадний" походить від зазначеної схеми пріоритетів визначення того, яке правило стилю застосовується, якщо певному елементу відповідає більше одного правила. Ця каскадна схема пріоритетів передбачувана.

CSS має простий синтаксис та використовує ряд англійських ключових слів для вказівки імен різних властивостей стилю.

Таблиця стилів складається зі списку правил. Кожне правило чи набір правил складається з одного або кількох селекторів та блоку декларації.

CSS призначений для розділення уявлення та змісту, включаючи макет, кольори та шрифти. Такий поділ може покращити доступність контенту; забезпечити більшу гнучкість і контроль щодо характеристик подання; дозволити декільком веб-сторінкам спільно використовувати форматування шляхом вказівки відповідного CSS в окремому файлі .css, що зменшує складність та повторення структурного контенту; та дозволити кешувати

файл .css для підвищення швидкості завантаження сторінки між сторінками, які спільно використовують цей файл та його форматування.

Розділення форматування та вмісту також дозволяє представити одну й ту саму сторінку розмітки в різних стилях для різних методів візуалізації, наприклад, на екрані, друку, голосом (через мовний браузер або програму читання з екрана) і на тактильних пристроях на основі шрифту Брайля. У CSS також є правила альтернативного форматування, якщо контент доступний на мобільному пристрої.

Sass - це стабільна і потужна мова розширення CSS професійного рівня у світі [22]. Sass має такі переваги:

- сумісність із CSS - Sass повністю сумісний з усіма версіями CSS. Ми серйозно ставимося до цієї сумісності, щоб ви могли легко використовувати будь-які доступні бібліотеки CSS;
- багатий функціонал - Sass може похвалитися більшою кількістю функцій та можливостей, ніж будь-яка інша мова розширення CSS. Команда Sass Core Team нескінченно працює над тим, щоб не тільки йти в ногу з часом, але залишатися попереду;
- зрілість - Sass активно підтримується вже понад 15 років його командою Core Team;
- велика спільнота - Sass активно підтримується та розвивається консорціумом з кількох технологічних компаній та сотень розробників;
- фреймворки - Існує безліч фреймворків, створених з використанням Sass;
- схвалено галуззю - знову і знову індустрія вибирає Sass як основну мову розширення CSS.

WHMCS – це Ruby обгортка для WHMCS, яка дозволяє дуже легко виконувати запити на API WHMCS використовуючи лаконічний синтаксис Ruby.

2.4. Опис структури системи та алгоритмів її функціонування

В першу чергу розглянемо логічну структуру програмного виробу з описом функцій його складових частин і зв'язку між ними на рис. 2.3:

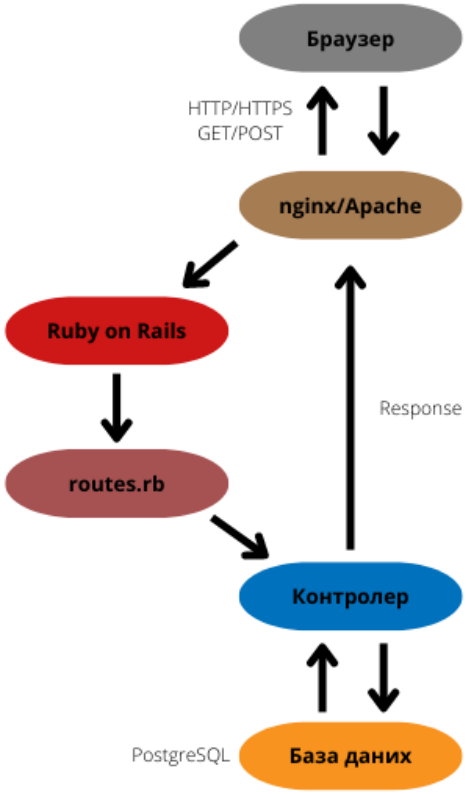

Рис. 2.3. Процес обробки сервером запиту від браузера.

Спочатку браузер надсилає HTTP-запит, щоб отримати контент сайту, у якому просить сервер надіслати дані для відображення сторінки. У цьому запиті міститься інформація про браузер, тимчасові файли, вимоги до з'єднання і так далі.

Завдання браузера - якомога докладніше пояснити серверу, яка саме інформація йому потрібна.

У спілкуванні браузера та сервера виділяють два типи запитів. GET-запит використовується для отримання даних із сервера - наприклад, відобразити картинку, текст або відео. POST-запит — використовується для надсилання даних із браузера на сервер, наприклад, коли користувач надсилає повідомлення, зображення або завантажує файл.

Наступним етапом є обробка отриманого запиту сервером - сервер отримав запит від браузера із докладним описом того, що йому потрібно. Тепер йому потрібно опрацювати цей запит. Цим завданням займається спеціальне серверне програмне забезпечення, наприклад, nginx або Apache. Найчастіше такі програми прийнято називати веб-серверами.

Веб-сервер, у свою чергу, перенаправляє запит на подальшу обробку до програми-обробника — Ruby on Rails. Програма уважно вивчає вміст запиту, наприклад, розуміє, в якому форматі потрібно надіслати відповідь і які саме файли потрібні. Програмне забезпечення визначає в який контролер передати запит завдяки файлу з роутами.

В залежності від отриманих браузером параметрів контролер може зробити запит до бази даних задля вибірки певних даних або просто знайти певну сторінку, яку потрібно показати користувачу.

Коли відповідь сформована, вона надсилається веб-сервером назад браузеру. У відповіді зазвичай міститься контент для відображення веб-сторінки, інформація про тип стиснення даних, способи кешування, файли cookie, які потрібно записати і так далі.

Браузер розпаковує отриману відповідь і поступово починає відображати отриманий контент на екрані користувача – цей процес називається рендерингом.

Спочатку браузер завантажує лише основну структуру HTML-сторінки. Потім послідовно перевіряє всі теги та надсилає додаткові GET-запити для отримання з сервера різних елементів – картинки, файли, скрипти, таблиці стилів тощо. Тому при завантаженні сторінки браузер і сервер продовжують обмінюватися між собою інформацією.

Паралельно з цим на комп'ютер зазвичай зберігаються статичні файли користувача - щоб при наступному відвідуванні не завантажувати їх заново і швидше відобразити користувачеві вміст сторінки.

Як тільки рендеринг завершено, користувачеві з'явиться повністю завантажена сторінка сайту.

Наступним кроком буде опис структури бази даних системи. Таблиці та їх зв’язки між собою зображено на рис. 2.4:

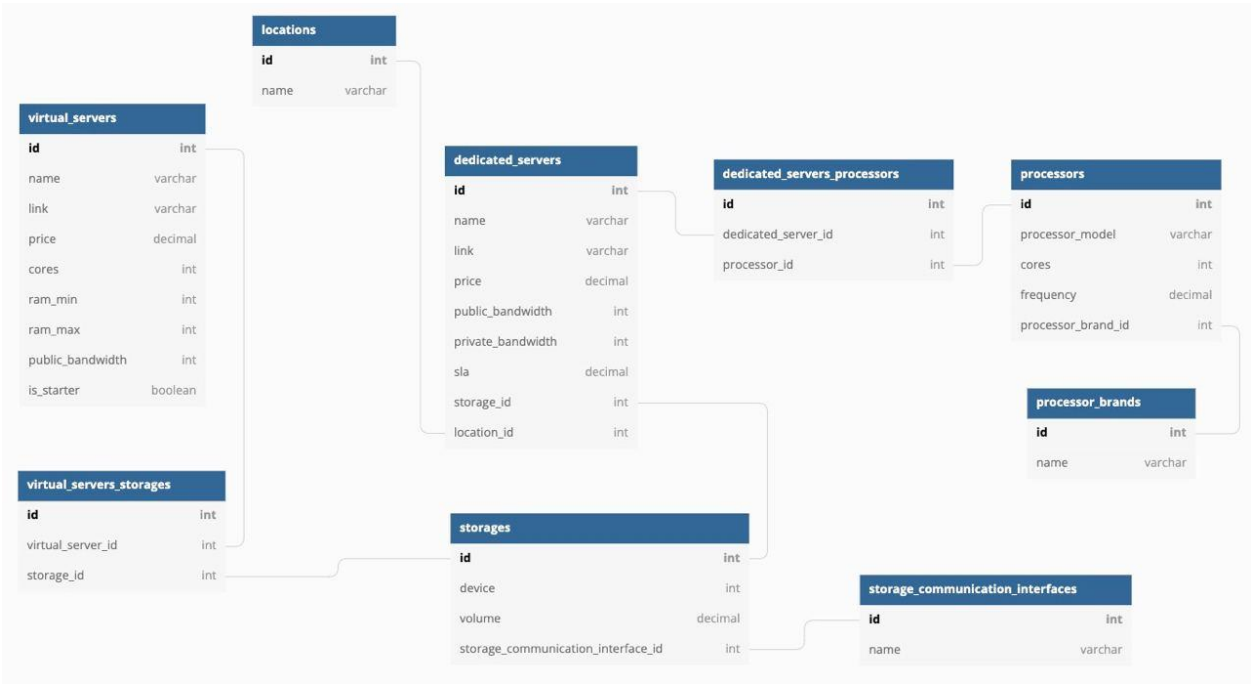

Рис. 2.4. Діаграма структури бази даних.

З наведеної діаграми можна побачити, що база даних складається з дев’яти таблиць. Перша з них – locations, яка відповідає за зберігання регіонів, в яких розташовані дата центри серверів та має зв'язок з таблицею dedicated_servers один до багатьох. Таблиця містить наступні поля:

– id – integer, primary key, унікальний;
– name – varchar, унікальний.

storage_communication_interface – таблиця, що зберігає назви специфікацій протоколів доступу до твердотілих накопичувачів (наприклад: NVMe, SATA). Має зв'язок з таблицею storages один до багатьох та містить наступні поля:

– id – integer, primary key, унікальний;
– name – varchar, унікальний.

storages – таблиця, що містить у собі типи накопичувачів, їх об’єм. Пов’язана з таблицею dedicated_servers зв’язком один до багатьох та з таблицею virtual_servers зв’язком багато до багатьох через суміжну таблицю virtual_servers_storages. Включає наступні поля:

- id – integer, primary key, унікальний;
- device – integer (використовується як enum);
- volume – decimal;
- storage_communication_interface_id – integer, foreign key (для зв’язку з таблицею storage_communication_interface).

virtual_servers – таблиця, що містить дані про віртуальні сервери. Має зв'язок з таблицею storages багато до багатьох. Поля:

- id – integer, primary key, унікальний;
- name – varchar, унікальний;
- link – varchar;
- price – decimal;
- cores – integer;
- ram_min – integer;
- ram_max – integer;
- public_bandwidth – integer;
- is_starter – boolean.

processor_brands – таблиця, що містить назви брендів процесорів та пов’язана з таблицею processors зв’язком один до багатьох. Містить наступні поля:

- id – integer, primary key, унікальний;
- name – varchar, унікальний.

processors – таблиця, що містить загальні дані про процесори, такі як кількість ядер, модель процесору та частоту. Пов’язана з таблицею processor_brands зв’язком один до багатьох через foreign key processor_brand_id та з таблицею dedicated_servers зв’язком багато до

багатьох через суміжну таблицю dedicated_servers_processors. Містить такі поля:

- id – integer, primary key, унікальний;
- processor_model – varchar, унікальний;
- cores – integer;
- frequency – decimal;
- processor_brand_id – integer, foreign key (для зв’язку з таблицею processor_brand).

dedicated_servers – таблиця, що містить загальні дані про виділені сервери, такі як назва, посилання на білінгову панель, вартість та інші. Пов’язана з таблицею processors зв’язком багато до багатьох через суміжну таблицю dedicated_servers_processors, таблицею locations зв’язком один до багатьох через foreign key location_id, а також з таблицею storages зв’язком один до багатьох через foreign key storage_id. Має поля:

- id – integer, primary key, унікальний;
- name – varchar, унікальний;
- link – varchar;
- price – decimal;
- public_bandwidth – integer;
- private_bandwidth – integer;
- sla – decimal;
- storage_id – integer, foreign key (для зв’язку з таблицею storages);
- location_id – integer, foreign key (для зв’язку з таблицею locations).
-

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Після відкриття браузера та введення у адресний рядок ім’я сервера, на якому знаходиться веб-сайт, користувач потрапляє на головну сторінку сайту.

В якості різнотипних вхідних даних веб-сайту можуть виступати дані, що стосуються сортування та фільтрації продуктів та послуг на сторінках сайту, а саме:

- ціна – обирається користувачем завдяки спеціальним повзункам для фільтрації по ціні товару або послуги;
- бренд процесора – вводиться завдяки чекбоксам на панелі фільтрів та використовується для фільтрації серверів по використаному бренду процесора;
- кількість процесорів – обирається завдяки спеціальним повзункам на панелі фільтрації;
- кількість ядер та кількість потоків – обирається повзунками на панелі фільтрації;
- частота ядра – обирається повзунками на панелі фільтрації;
- пам'ять – обирається з використанням повзунків для фільтрації по кількості оперативної пам’яті у наданому сервері;
- сховище – обирається чекбоксами для фільтрації по інтерфейсним протоколам сховища, використовуваним на сервері;
- публічна мережа – обирається повзунками для фільтрації по швидкості публічної мережі, використовуваною на сервері;
- приватна мережа – обирається повзунками для фільтрації по швидкості приватної мережі, використовуваною на сервері;
- локація – обирається чекбоксами для фільтрації серверів по місцезнаходженню;
- вартість – використовується користувачем для сортування товарів та послуг по вартості: від більшої до меншої та від меншої до більшої.

Вихідними даними веб-сайту є вміст сторінок, а також сформований список товарів та послуг, залежно від наданих користувачем даних про фільтрацію або сортування.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

2.6.1. Використані технічні засоби

Для функціонування системи було використано віртуальний виділений сервер (VPS) з одноядерним процесором, 2 GB оперативної пам’яті, 20 GB SSD накопичувач з інтерфейсом SATA та швидкістю публічної мережі 100 Mbit/s.

2.6.2. Використані програмні засоби

Для функціонування системи було використано стек LAMP, до якого входять операційна система Linux з дистрибутивом Ubuntu версії 18.04, Apache – програмне забезпечення для розміщення веб-сервера, MySQL – система управління реляційними базами даних та PHP 7 – мова програмування, завдяки якій створюються веб-ресурси.

Також для взаємодії з білінговою панеллю було встановлено спеціальне програмне забезпечення WHMCS, яке використовує мову PHP для роботи з веб-сторінками та MySQL для збереження, отримання та редагування даних.

Окрім цього, для встановлення основного проекту, було використано мову програмування Ruby версії 2.7.2, фреймворк Ruby on Rails версії 6 та систему управління базами даних PostgreSQL.

2.6.3. Виклик та завантаження програми

Для запуску програми на локальному комп’ютері потрібно переконатися, що маємо наступне встановлене програмне забезпечення:

- Git;
- Bundler;
- Ruby 2.7.2
- Ruby on Rails 6

– PostgreSQL

Якщо всі перелічені інструменти встановлені та налаштовані, то можна приступати до процесу встановлення проекту. Для цього знадобиться доступ для Git репозиторію, в якому знаходиться проект, тож, треба слідувати наступним крокам:

1) клонувати проект на свій комп'ютер використовуючи команду git clone http_link_to_project;
2) перейти до директорії з проектом завдяки команді cd path_to_project у своєму терміналі;
3) запустити команду bundle – ця команда встановлює всі залежності (бібліотеки) з файлу Gemfile;
4) викликати команду rails db:create – створює бази даних для development та test оточення;
5) викликати команду rails db:migrate – викликає усі існуючі файли міграції та створює усі необідні таблиці та поля у базі даних;
6) викликати команду rails db:seed – створює усі необхідні записи у базі даних для мінімальної працездатності системи;
7) викликати команду rails server – це запустить локальний сервер програми використовуючи localhost (127.0.0.1:3000).

Для запуску тестів та впевненості, що всі компоненти програми працюють добре, можна використати команду rspec – вона запустить усі тести, що знаходяться в директорії spec у корні проекту та покаже відсоток покриття додатку тестами.

Щоб переконатися, що код написаний слідуючи найкращим практикам та запустити статичні аналізатори коду, можна використати наступні команди:

– rubocop – перевіряє порушення стилів написання Ruby коду;
– fasterer – перевіряє різні місця у вашому Ruby коді, які можна пошвидшити.

2.6.4. Опис інтерфейсу користувача

Інтуїтивно зрозумілий та дружелюбний до користувача інтерфейс сайту – завжди добре. Це означає, що відвідувачі не залишать ресурс відразу ж як зайшли, а навпаки, оцінять його належним чином. Що, безперечно, позначиться і на іміджі компанії, зокрема, і на конверсії в цілому. Тож, спершу, розглянемо всі можливі варіанти доступу користувача до різних сторінок та продемонструємо дорожню карту сайту на рис. 2.5:

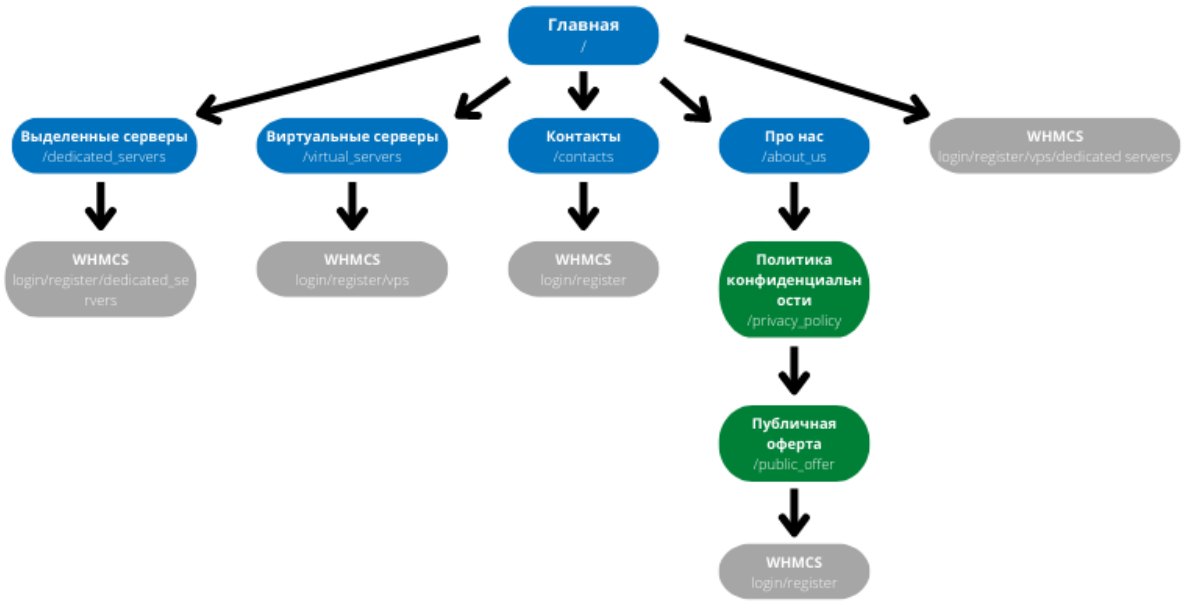

Рис. 2.5. Карта сайту

Перебуваючи на головній сторінці сайту, користувач може використовувати навігаційне меню в хедері. На головній сторінці можна ознайомитися зі списком послуг, які надаються потенційним клієнтам та знайти відповіді на часті запитання загального характеру (які методи оплати використовуються, де знаходяться сервери і тому подібне). Також на всіх сторінках знаходяться кнопки «Регистрация» и «Войти», що дозволяють користувачу перейти до білінгової панелі WHMCS та увійти в свій обліковий запис або зареєструватися. Окрім цього є можливість перейти на наступні сторінки:

– «Выделенные серверы»;
– «Виртуальные серверы»;
– «Контакты»;
– «Про нас».

На сторінці «Выделенные серверы» користувач може ознайомитися зі списком доступних для замовлення виділених серверів, прочитати характеристики, використати панель фільтрації та сортування. У кожній карточці товару є кнопка «Подробнее», що містить у собі посилання на білінгову панель для більш детального ознайомлення з послугою та оплати.

На сторінці «Выделенные серверы» можна ознайомитися з доступними тарифами віртуальних виділених серверів, перейти до білінгової панелі для оплати та більш детального ознайомлення з продуктом, а також знайти відповіді на часті питання стосовно VPS-хостингу.

На сторінці «Контакты» можна знайти інформацію щодо способів зв’язку з представниками або дізнатися контактну інформацію задля звернення до технічної підтримки.

Сторінка «Про нас» інформує користувача о загальних відомостях про представника послуг з аренди виділених та віртуальних серверів, отримати посилання соціальні мережі проекту. Окрім цього на цій сторінці є посилання на сторінки для ознайомлення з політикою конфіденційності та публічною офертою.

Для зручного редагування характеристик віртуальних та виділених серверів на сайті присутня панель адміністратора, яка надає можливість додавати, видаляти та редагувати будь-які дані стосовно серверів та використовувати зручний пошук або фільтрацію.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми – 780;
2. Коефіцієнт складності програми (1,25…2,0) – 1,6;
3. Коефіцієнт корекції програми в ході її розробки (0,05...0,1) - 0,1;
4. Годинна заробітна плата програміста - 100 грн/год;
5. Коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі (1,2...1,5) - 1,2;
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (від 3-х до 5 років) – 0,8;
7. Вартість машино-години ЕОМ - 16 грн/год.

Нормування праці в процесі створення програмного забезпечення істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \qquad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 60 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_{д}$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q - передбачуване число операторів програми (q = 780);

C - коефіцієнт складності програми (C = 1,6);

p - коефіцієнт корекції програми в ході її розробки (p = 0,1).

Звідси умовне число операторів в програмі:

$$Q = 780 \cdot 1{,}6 \cdot (1 + 0{,}1) = 1372{,}8.$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}\text{, людино-годин,} \tag{3.3}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 0,8.

Приймемо збільшення витрат праці в наслідок недостатнього опису завдання не більше 50% (B = 1.2). З урахуванням коефіцієнта кваліфікації k = 0,8, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (1372{,}8 \cdot 1{,}2) / (80 \cdot 0{,}8) = 25{,}74 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20...25)\cdot k}$$, людино-годин, (3.4)

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), отримаємо:

$$t_a = 1372{,}8 / (21 \cdot 0{,}8) = 52{,}3$$ людино-годин.

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)\cdot k}$$, людино-годин, (3.5)

$$t_n = 1372{,}8 / (23 \cdot 0{,}8) = 47{,}75$$ людино-годин.

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}$$, людино-годин, (3.6)

$$t_{отл} = 1372{,}8 / 4 \cdot 0{,}8 = 274{,}56$$ людино-годин.

– за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5 \cdot t_{отл}$$, людино-годин, (3.7)

$$t^k_{отл} = 1{,}5 \cdot 274{,}56 = 411{,}84$$ людино-годин.

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{дp} \cdot t_{до}$$, людино-годин, (3.8)

де $t_{дp}$ – трудомісткість підготовки матеріалів і рукопису:

$$t_{дp} = \frac{Q}{(15..)\cdot k}$$, людино-годин, (3.9)

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{дp}$$, людино-годин. (3.10)

Підставляючи відповідні значення, отримаємо:

$$t_{\partial p} = 1372{,}8 \,/\, (17 \cdot 0{,}8) = 64{,}6 \text{ людино-годин.}$$

$$t_{\partial o} = 0{,}75 \cdot 64{,}6 = 48{,}45 \text{ людино-годин.}$$

$$t_{\partial} = 64{,}6 + 48{,}45 = 113{,}05 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 60 + 25{,}74 + 52{,}3 + 47{,}75 + 274{,}56 + 113{,}05 = 573{,}4 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $K_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрати машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{\text{ПО}} = З_{\text{ЗП}} \cdot З_{\text{МВ}} \text{ , грн.} \qquad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{\text{ЗП}} = t \cdot C_{\text{ПР}} \text{ , грн,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$C_{\text{ПР}}$ – середня годинна заробітна плата програміста становить 100 грн / год, отримуємо:

$$З_{\text{ЗП}} = 573{,}4 \cdot 100 = 57340 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{\text{мв}} = t_{отл} \cdot C_{\text{мч}} \text{ , грн,} \qquad (3.13)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{\text{мч}}$ – вартість машино-години ЕОМ, грн/год (16 грн/год).

Підставивши в формулу (3.13) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 274{,}56 \cdot 16 = 4392{,}96 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$K_{ПО} = 57340 + 4392{,}96 = 61732{,}96 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс,} \qquad (3.14)$$

де B_k – число виконавців (дорівнює 2);

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні $F_p = 176$).

Звідси витрати на створення програмного продукту:

$$T = 573{,}4 \,/\, 2 \cdot 176 = 1{,}63 \text{ міс.}$$

ВИСНОВКИ

Програмне забезпечення призначене для реалізації інформаційної системи для реселінгу хостингу серверів засобами фреймворку Ruby on Rails. Метою кваліфікаційної роботи було створення сайту з реселінгу виділених та віртуальних серверів, що зможе викликати інтерес та зацікавленість у людей, які мають свій власний бізнес та мають бажання інтегрувати його до Інтернету, адже сайт формує авторитет компанії, залучає нових клієнтів, розширяє ринок збуту, допомагає оперативно отримати дані про ціни товарів або послуг в зручному вигляді. Це значно економить великі гроші на покупку дорогого обладнання, утримання власного дата-центру та догляд за ним. В разі виникнення проблем, їх вирішенням буде займатися безпосередньо провайдер, що також економить гроші на утриманні штату працівників.

Також на даний сайт можуть звернути увагу не тільки прихильники бізнесу, але й школярі або студенти, які, завдяки широкому вибору тарифних планів, можуть орендувати сервер та використовувати його потужність для тестування власних проектів або інших потреб.

Сайт зроблений за допомогою таких мов web-програмування: Ruby, JavaScript, а також Ruby on Rails, HTML, CSS, SASS, jQuery, PostgreSQL.

Під час виконання даної дипломної роботи були виконані наступні задачі:

1) Дослідження та аналіз на тему «Реселінг хостингу».
2) Здійснено аналіз існуючих хостингових компаній з метою виявлення їх властивостей, потенційних можливостей та недоліків.
3) Розроблено архітектуру бази даних.
4) Розроблено алгоритм та реалізацію програми.

Створений Web-додаток має наступні властивості:

1) Надає можливість ознайомитися зі списком послуг та товарів, представлених на сайті.

2) Надає можливість сортувати список послуг та товарів.
3) Дає можливість додавати, видаляти та редагувати будь-які дані, які стосуються представлених на сайті послуг завдяки адміністративній панелі ActiveAdmin;
4) Встановлює зв'язок з білінговою панеллю WHMCS для зручного перегляду статистики, зміни характеристик виділеного або віртуального серверу та здійснення оплати послуг.

Створена програма надає користувачу можливість ознайомитись з необхідним матеріалом і застосувати його на практиці**.**

Вартість даного програмного забезпечення становить 61732,96 грн. і не вимагає додаткових витрат при впровадженні та експлуатації програми. Очікуваний час розробки становить приблизно 1,63 місяці. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

config/routes.rb // файл з роутами

```
Rails.application.routes.draw do
  devise_for :admin_users, ActiveAdmin::Devise.config
  ActiveAdmin.routes(self)

  root 'main#index'
  resources :virtual_servers, only: :index
  resources :dedicated_servers, only: :index
  resources :contacts, only: :index
  resource :about_us, only: :show
  resource :privacy_policy, only: :show
  resource :public_offer, only: :show
end
```

app/views/layouts/_header.html.erb // html частина з хедером

```
<div class="header">
    <div class="conteiner">
        <nav class="header--nav">
            <ul class="header-list">
                <li class="top--list">
                    <%= link_to('Выделенные серверы', dedicated_servers_path, class: 'top--list__item') %>
                </li>
                <li class="top--list">
                    <%= link_to('Виртуальные серверы', virtual_servers_path, class: 'top--list__item') %>
                </li>
                <li class="top--list">
                    <a class="top--list__item" href="/instructions.html">База знаний</a>
                </li>
                <li class="top--list">
                    <%= link_to('Контакты', contacts_path, class: 'top--list__item') %>
                </li>
                <li class="top--list">
```

```
            <%= link_to('Про нас', about_us_path, class: 'top--list__item') %>
          </li>
        </ul>
      </nav>
      <div class="line"></div>
      <div class="header--bottom">
        <%= link_to root_path, class: 'logo' do %>
          <%= image_tag('cloud.svg', alt: 'logo') %>
          <p>Limial</p>
        <% end %>
        <div class="reg--log">
          <button class="btn reg">Регистрация</button>
          <button class="btn log">Войти</button>
        </div>
      </div>
    </div>
</div>
```

app/views/layouts/_header.html.erb // html частина з футером

```
<footer>
  <div class="conteiner">
    <div class="footer-list">
      <ul class="footer-main-info">
        <li class="logo">
          <%= link_to root_path, class: 'logo' do %>
            <%= image_tag('cloud.svg', alt: 'logo') %>
            <p>Limial</p>
          <% end %>
        </li>
        <li>
          <p class="slogan">
            Надёжный хостинг для ваших любимых проектов.
          </p>
        </li>
        <li class="footer-main-info__contacts">
          <%= image_tag('mail.svg', alt: 'mail') %>
```

```
        <a href="mailto:sales@limial.com">sales@limial.com</a>
    </li>
    <li class="footer-main-info__contacts">
        <%= image_tag('phone.svg', alt: 'tel') %>
        <a href="tel:+380665767991">+ 38(066)576-7991</a>
    </li>
</ul>
<ul class="footer-help">
    <li>
        <p>
            Тех. Поддержка
        </p>
    </li>
    <li>
        <a href="#">Поддержка</a>
    </li>
    <li>
        <a href="/app/instructions.html">База знаний</a>
    </li>
    <li>
        <a href="mailto:abuse@limial.com">Сообщить о жалобе (abuse@limial.com)</a>
    </li>
</ul>
<ul class="footer-company">
    <li>
        <p>
            Компания
        </p>
    </li>
    <li>
        <%= link_to('О нас', about_us_path) %>
    </li>
    <li>
        <%= link_to('Контакты', contacts_path) %>
    </li>
```

```
            <li>
              <a href="#">Документы</a>
            </li>
            <li>
              <a href="#">Способы оплаты</a>
            </li>
          </ul>
          <ul class="footer-social">
            <li>
              <p>
                Социальные сети
              </p>
            </li>
            <ul class="social-media">
              <li>
                <a href="#"><%= image_tag('facebook.svg', alt: 'facebook') %></a>
              </li>
              <li>
                <a href="#"><%= image_tag('telegram.svg', alt: 'telegram') %></a>
              </li>
              <li>
                <a href="#"><%= image_tag('twitter.svg', alt: 'twitter') %></a>
              </li>
            </ul>
          </ul>
        </div>
    </div>
</footer>
```

app/controllers/main_controller.rb // контролер для головної сторінки сайту

```
class MainController < ApplicationController
  def index; end
end
```

app/views/main/index.html.erb // html файл головної сторінки

```
<!-- START HERO -->
<div class="hero">
```

```
  <div class="main">
    <div class="conteiner">
      <div class="main__text">
        <h1 class="text__h1">
          Виртульные Выделенные Серверы нового <br>поколения
        </h1>
        <h2 class="text__h2">
          Создайте свой проект с помощью виртульных серверов нового поколения с NVMe дисками на борту
        </h2>
        <button class="btn more">
          Узнать больше
        </button>
      </div>
      <div class="main__img">
        <%= image_tag('server-hero.png', alt: 'image-server') %>
      </div>
    </div>
  </div>
</div>
<!-- END HERO -->
<!-- START OPTIONS -->
<div class="options">
  <div class="conteiner">
    <h2 class="options__h2">
      Опции
    </h2>
    <ul class="list-of-options">
      <li class="list-of-options__item">
        <%= image_tag('options-cloud.svg', class: 'options-img', alt: 'cloud') %>
        <div class="list-of-options__content">
          <h3 class="list-of-options__content__item">
            Панель cPanel
          </h3>
          <p>
            Полный контроль с любой точки мира
```

```
        </p>
      </div>
    </li>
    <li class="list-of-options__item">
      <%= image_tag('options-security-scan.svg', class: 'options-img', alt: 'scan') %>
      <div class="list-of-options__content">
        <h3 class="list-of-options__content__item">
          Балансировщик нагрузки
        </h3>
        <p>
          Максимальная надежность и гибкость
        </p>
      </div>
    </li>
    <li class="list-of-options__item">
      <%= image_tag('options-geo.svg', class: 'options-img', alt: 'geo-ip') %>
      <div class="list-of-options__content">
        <h3 class="list-of-options__content__item">
          Геолокация IP
        </h3>
        <p>
          Оптимизируйте собственное SEO
        </p>
      </div>
    </li>
    <li class="list-of-options__item">
      <%= image_tag('options-float.svg', class: 'options-img', alt: 'float-file') %>
      <div class="list-of-options__content">
        <h3 class="list-of-options__content__item">
          Плавающий IP
        </h3>
        <p>
          Работайте без перерывов
        </p>
      </div>
```

```
        </li>
      </ul>
    </div>
</div>
<!-- END OPTIONS -->


<!-- START SECURITY -->
<div class="security">
  <div class="conteiner">
    <div class="security__content">
      <div class="content-text">
        <h2>
          С лучшей безопасностью, не беспокойтесь ни о каких угрозах
        </h2>
        <h3>
          Мы предоставляем вашим услугам круглосуточную защиту от всех типов DDoS-атак без каких-либо ограничений по объему или продолжительности.
        </h3>
        <ul class="security-list">
          <li class="list__item">
            <%= image_tag('checklist.svg', class: 'checkout', alt: 'checkout') %>
            <p>Огромная сеть с пропускной способностью 4 ТБ/с</p>
          </li>
          <li class="list__item">
            <%= image_tag('checklist.svg', class: 'checkout', alt: 'checkout') %>
            <p>Вредоносный трафик блокируется за нескольких секунд</p>
          </li>
          <li class="list__item">
            <%= image_tag('checklist.svg', class: 'checkout', alt: 'checkout') %>
            <p>Легитимный Трафик не блокируется и достигает сервера</p>
          </li>
        </ul>
      </div>
      <div class="security-img">
        <%= image_tag('security-main-image.png', alt: 'lock') %>
      </div>
```

```
      </div>
    </div>
</div>

<!-- END SECURITY -->
<!-- START PROMO -->
<div class="promo">
    <div class="conteiner">
        <div class="promo-s">
            <h2 class="promo-text">
                Поднимите инфраструктуру на новый уровень с помощью Балансировщика нагрузки
            </h2>
            <div class="balancer">
                <%= image_tag('security.svg', class: 'balancer-img', alt: 'balancer') %>
                <p class="balancer-text">
                    Балансировщик нагрузки
                </p>
                <div class="balancer-price">
                    <p class="balancer-price1">
                        $3.24
                    </p>
                    <p class="balancer-price2"
/Месяц
                    </p>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- END PROMO -->
<!-- START WORLD MAP -->
<div class="map">
    <div class="conteiner">
        <div class="map-content">
            <h2 class="map-content__text">
```

```
            Серверы по всему миру
          </h2>
          <h3 class="map-content__text">
            Выбирайте удобное вам расположения сервера и уменьшайте задержки ваших проектов
          </h3>
          <div class="world-map">
            <%= image_tag('world-map.png', alt: 'map') %>
          </div>
        </div>
      </div>
</div>
<!-- END WORLD MAP -->
<!-- START QA -->
<div class="question">
    <div class="conteiner">
        <div class="question-content-text">
            <h2>
                Часто задаваемые вопросы
            </h2>
            <h3>
                Найдите ответы на частые вопросы
            </h3>
        </div>
        <div class="questiouns">
            <input class="hide" id="hd-1" type="checkbox">

            <label for="hd-1"><p>Какие преймущества использования Выделенных серверов?</p><%= image_tag('vniz.svg', alt: 'cursor-down') %></label>
            <div>
                Самое главное достоинство – полная управляемость и независимость от других пользователей. VPS решения предусматривают одновременное использование физического сервера многими пользователями, а в выделенном хостинге емкость жесткого диска, процессорная мощность и память находится в распоряжении всего одного пользователя.

                К недостаткам можно отнести относительно медленное масштабирование в сравнении с VPS. Исходя из этого выделенные серверы позволяют использовать максимальное количестов ресурсов и масксимальную надежность. Поэтому, выделеннные серверы - используют в высоконагруженных проектах.
            </div>
            <input class="hide" id="hd-2" type="checkbox">
```

<label for="hd-2"> Какие преймущества использования VPS? <%= image_tag('vniz.svg', alt: 'cursor-down') %></label>

Вы получите выделенные ресурсы, это означает, что вам не нужно делиться мощностью процессора, оперативной памятью или дисковым пространством. У вас есть определённое количество ресурсов, которыми вы распоряжаетесь по своему усмотрению.

Самое значительное преимущество VPS-хостинга – вы получаете root-доступ и полный контроль над сервером. Это означает, что вы можете установить всевозможные программы для вашей инфраструктуры и работать над ресурсоёмкими проектами. Когда ваш проект будет разрастаться вы можете в несколько кликов увеличить мощность сервера и обеспечить повышенную надежность работы проекта с помощью Балансировщика нагрузки. Поэтому наши VPS отплично подойдут для малых и средних проектов.

<input class="hide" id="hd-3" type="checkbox">

<label for="hd-3"> Где находятся ваши серверы? <%= image_tag('vniz.svg', alt: 'cursor-down') %></label>

Расположение дата-центров является основным фактором, который должен учитывать хостинг-провайдер. Для хорошого опыта работы с серверами, в любой точке мира, наши решения должны быть адаптируемыми и максимально приближенными к их пользователям. Предложение услуг в стране или рядом со страной где вы ведете бизнес, является гарантией доверия, безопасности и эффективности. Вот почему мы сотрудничаем с дата-центрами по всему миру и продолжаем каждый день расширять географический охват наших решений. Вы можете найти дата-центры с нашими серверами в следующих странах: Франция, Канада, США, Австралия, Германия, Польша, Великобритания и Сингапур.

<input class="hide" id="hd-4" type="checkbox">

<label for="hd-4"> Какие есть методы оплаты? <%= image_tag('vniz.svg', alt: 'cursor-down') %></label>

В данный момент мы поддерживаем такие методы оплаты - Masterpass, платежная карта, FacePay24, Visa Checkout, PrivatPay, Google Pay, Apple Pay.

<input class="hide" id="hd-5" type="checkbox">

<label for="hd-5"> Получаю ли я какую-нибудь поддержку? <%= image_tag('vniz.svg', alt: 'cursor-down') %></label>

Мы оказываем круглосуточную поддержку 365/7. Хостинг VPS и Выделенные серверы не администрируются нашими специалистами, однако наша команда поддержки клиентов сделает всё возможное, чтобы помочь с оплатой и общими техническими вопросами.

<input class="hide" id="hd-6" type="checkbox">

<label for="hd-6"> Какое время безпрерывной работы продукта гарантировано? <%= image_tag('vniz.svg', alt: 'cursor-down') %></label>

Настоящее Соглашение об уровне обслуживания виртуального частного сервера (SLA) представляет собой политику, регулирующую использование перечисленных услуг Limial

и гарантирует время безперырвной работы сервера:

VPS - 99.9%

Rise - 99.9%

Best Value - 99.9%

Advance - 99.95%

Infrastructure - 99.95%

Game - 99.9%

Storage - 99.95%

HG - 99.95%

High Grade - 99.99%

Scale - 99.99%

<input class="hide anim" id="hd-7" type="checkbox">

<label for="hd-7"> Есть ли возврат средств? <%= image_tag('vniz.svg', alt: 'cursor-down') %></label>

Вы всегда можете оформить возврат средств и Вам будет возвращена первоначально уплаченная сумма в течение 30 дней после запроса на снятие средств, но дни, в которые использовалась услуга, вычитаются из суммы возмещения на пропорциональной основе.

<input class="hide" id="hd-8" type="checkbox">

<label for="hd-8"> Что запрещено размещать на сервере? <%= image_tag('vniz.svg', alt: 'cursor-down') %></label>

На серверах запрещено размещать информацию, которая противоречит законодательству Украины, материалы порнографического содержания, не лицензионное программное обеспечение.

На серверах компании не должно размещаться программное обеспечение, которое может быть использовано для нарушения работоспособности других серверов, или рассылки спама.

Более подробно можно узнать в договоре публичной оферты.

<!-- END QA -->

app/controllers/virtual_servers_controller.rb // контролер для сторінки з віртуальними серверами

class VirtualServersController < ApplicationController

```
  def index
    @virtual_servers = VirtualServersQuery.new(params).call.decorate
    @starter_virtual_server = VirtualServer.where(is_starter: true)
  end
end
```

app/queries/virtual_servers_query.rb // клас з описаною логікою для вибірки віртуальних серверів

```
class VirtualServersQuery
  MAX_VIRTUAL_SERVERS_PER_PAGE = 4
  DEFAULT_SORT_OPTION = :price_asc
  SORT_OPTIONS = {
    price_asc: ->(relation) { relation.order(price: :asc) }
  }.freeze

  def initialize(params, relation = VirtualServer.all)
    @params = params
    @relation = relation
  end

  def call
    SORT_OPTIONS[sort_by].call(scope)
  end

  private

  attr_reader :relation, :params

  def sort_by
    return DEFAULT_SORT_OPTION if params[:sort_by].blank? ||
SORT_OPTIONS.keys.exclude?(params[:sort_by].to_sym)

    params[:sort_by].to_sym
  end

  def scope
    relation.where(is_starter: false).limit(MAX_VIRTUAL_SERVERS_PER_PAGE)
  end
```

```
end
```

app/decorators/virtual_server_decorator.rb // клас з допоміжними функціями для сутності VirtualServer

```
class VirtualServerDecorator < Draper::Decorator
  delegate_all

  def upcase_name
    object.name.upcase
  end
end
```

app/controllers/dedicated_servers_controller.rb // контролер для сторінки з виділеними серверами

```
class DedicatedServersController < ApplicationController
    def index
        @dedicated_servers = DedicatedServersQuery.new(params).call.decorate
    end
end
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_Ільющенко.doc	Пояснювальна записка до дипломного проекту. Документ Word
Диплом_Ільющенко.pdf	Пояснювальна записка до дипломного проекту у форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і відкомпільовану програму
Презентація	
Презентація_Ільющенко.pptx	Презентація дипломного проекту

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке віддалений сервер для бізнесу. URL: https://iline.pro/blog/chto-takoe-udalennyj-server-dlya-biznesa/.
2. Google Cloud: What is a virtual server? URL: https://cloud.google.com/learn/what-is-a-virtual-server.
3. Best Server Virtualization Software. URL: https://www.g2.com/categories/server-virtualization.
4. History and Future of Dedicated Servers. URL: https://www.liquidweb.com/blog/history-future-dedicated-servers/.
5. Донецький національний технічний університет, 2016, 40 с. URL: https://studfile.net/preview/5679245/.
6. Владислава Рикова. Хостинг для сайту: основи правильного вибору провайдера. URL: https://vlada-rykova.com/ua/hosting-dlya-sajta/.
7. Реселінг хостингу: переваги та недоліки. URL: https://www.0382.ua/list/217863.
8. Desire Athow. What are the different types of web hosting? URL: https://www.techradar.com/web-hosting/what-are-the-different-types-of-web-hosting.
9. Світлана Щегель. Технічні характеристики сервера для хостингу сайту, 2021. URL: https://seranking.com/ru/blog/harakteristiki-servera-dlya-hostinga/.
10. Що таке реселінг хостингу. URL: https://vc.ru/u/383134-oleg-adminvps/329893-chto-takoe-reselling-hostinga.
11. Вигідний реселінг хостингу. URL: https://hostpro.ua/ua/reseller-tarify-ot-hostpro.html.
12. Stuart Ellis. An Introduction to ERB Templating. URL: http://www.stuartellis.eu/articles/erb/.

13. Роберт Мартін. Идеальный программист. Как стать профессионалом разработки ПО. Харків: Издательский дом Питер, 2012. 89 с.
14. Чернівецький національний університет ім. Ю. Федьковича, 2016, 3 с. URL: https://studfile.net/preview/5466693/page:3/.
15. Роберт Мартін. Идеальный программист. Как стать профессионалом разработки ПО. Харків: Издательский дом Питер, 2012. 124 с.
16. Віддалений сервер, 2019. URL: https://stekspb.ru/blog/udalenniy-server/.
17. About Ruby. URL: https://www.ruby-lang.org/en/about/.
18. Роберт Мартін. Идеальный программист. Как стать профессионалом разработки ПО. Харків: Издательский дом Питер, 2012. 207 с.
19. MVC - модель-представлення-контролер. URL: https://web-creator.ru/articles/mvc.
20. Tomas Habilton. Integration Testing: What is, Types, Top Down & Bottom Up Example, 2022. URL: https://www.guru99.com/integration-testing.html.
21. Pj Stevens. Understanding the Differences Between BDD & TDD, 2019. URL: https://cucumber.io/blog/bdd/bdd-vs-tdd/.
22. CSS with superpowers. URL: https://sass-lang.com/.