ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення автоматизованих систем

Пояснювальна записка

до магістерської роботи на ступінь вищої освіти магістр

на тему: «**РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДПРИЄМСТВА З ВИКОРИСТАННЯМ JAVASCRIPT»**

Виконав: студент 6 курсу, групи ІСДМ–61

спеціальності 126 Інформаційні системи та технології

Усик М.Л.

Керівник Ткаленко О.М.

Рецензент_____

Нормоконтроль_ _____

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ

Кафедра Інженерії програмного забезпечення автоматизованих систем
Ступінь вищої освіти «Магістр»
Напрям підготовки 126 Інформаційних систем та технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
ІПЗАС
_____ К.П.Сторчак
“____” _____2023 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Усику Максиму Леонідовичу
(прізвище, ім’я, по батькові)

1.Тема роботи Розпобка інформаційної системи підприємства з використанням JavaScript
Керівник роботи к.т.н., доцент Ткаленко О.М.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “__” хх 2022 року №__
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи:
3.1 Інформаційна система підприємства.
3.2 Науково-технічна література по тематиці магістерської роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1 Дослідження поняття веб-розробки.
4.2 Класифікація інформаційних систем та області їх застосування
4.2 Дослідження основ реалізації інформаційної системи
4.3 Реалізація інформаційної системи підприємства.

5.Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
5.1 Мета роботи;

6. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Підбір науково-технічної літератури		Виконано
2.	Аналіз адміністративних систем та області їх застосування		Виконано
3.	Дослідження основ реалізації інформаційної системи		Виконано
4.	Приклад практичної реалізації інформаційної системи		Виконано
5.	Вступ, висновки, реферат, оформлення роботи		Виконано
6.	Розробка обов`язкових демонстраційних матеріалів		Виконано

Студент _____ Усик М.Л.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ткаленко О.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи 73 с., 11 рис., 81 джерело.

Об`єкт дослідження – розробка інформаційної системи підприємства.

Предмет дослідження – інформаційна система підприємства.

Мета роботи – розробка інформаційної системи з використанням JavaScript.

Методи дослідження – методи об’єктно-орієнтовного проектування, методи системного аналізу, методи оптимального управління.

У роботі проведено аналіз шляху розробки інформаційної системи підприємства для обліку працівників. Така система дозволить автоматизувати процеси, щодо ведення обліку людей, які працюють у компанії. Досліджено актуальні для сьогодення технології менеджменту, які дозволяють забезпечити ефективний менеджмент для процесу розробки інформаційної системи. Запропоновано один з оптимальних підходів до розробки інформаційної системи підприємства.

Проведено дослідження технологій розробки, таких як HTML, CSS, JavaScript, TypeScript, NestJS. Визначено найоптимальніший набір технологій і фреймворків, для створення веб додатку. Цей набір дозволяє реалізувати інформаційну систему, яку потребує підприємство.

Галузь використання – сучасні інформаційні системи.

JAVSCRIPT, HTML, CSS, TYPESCRIPT, ВЕБ-ДОДАТОК, ОБЛІК, ДАНІ, ІНФОРМАЦІЙНА СИСТЕМА, АВТОРИЗАЦІЯ, КОРИСТУВАЧ, ПРАЦІВНИК, БАЗА ДАНИХ.

ЗМІСТ

ВСТУП

Процес автоматизації процесів у підприємстві включає в себе багато одиниць компютерної техніки, кількість якої з кожною модернізацією завжди збільшується. Обробка великої кількості даних (BigData), потреба у швидкому пошуку необхідної інформації, друк звітів, який допомагає у прийнятті оптимальніх рішень, саме це дозволяє зростати кількості та попиту на комп'ютерну техніку, що підтримується інформаційно-обліковими системами.

Кожна компанія та підприємство, перш за все, має ціль - максимальна ефективність виробництва, і інформаційні системи дозволяють досягти цієї цілі.

Створення сучасних інформаційних систем — складна задача, яка вирішується за допомогою спеціальних вмінь, методик та набору інструментів.[1]

Наявність такої інформаційної системи значно робить роботу простішою для всіх працівників, хто використовує дану систему, автоматизовує процеси підприємства, які до недавна робилися вручну, що дозволяє зменшити вплив людського фактору, а отже, буде значно менше повсякденних помилок, що на великих виробництвах можуть обійтися дуже дорого. Створена інформаційна система окуповує себе дуже швидко за рахунок цих, і багатьох інших факторів.

Однак не всі компанії готові витрачати великі гроші на придбання таких систем, особливо, коли її фінансові прибутки не є стійкими. У такому випадку є сенс використовувати інформаційну систему, яка може частково автоматизувати роботу окремої групи працівників, що працюють із великою кількістю інформації. В такому разі створюється локальна інформаційна система із спеціально визначеними функціями та можливостями. Вагома перевага — така система буде набагато більш дешевою за ту, яку б створювали для автоматизації всієї компанії.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Прикладне програмне забезпечення

Прикладна програма (скор. додаток) — це комп'ютерна програма, призначена для вирішення певної задачі, що відрізняється від роботи самого комп'ютера[1], яке зазвичай використовується кінцевими користувачами.[2] Прикладами є текстові процесори, медіаплеєри та бухгалтерське програмне забезпечення. Іншими основними класифікаціями програмного забезпечення є системне програмне забезпечення, пов'язане з роботою комп'ютера, і службове програмне забезпечення («служби»).

Програми можуть бути вже встановлені на комп'ютері разом з його системним програмним забезпеченням або опубліковані окремо. Також можуть бути створені власні, з відкритим або закритим програмним кодом.[4] Термін «додаток» зазвичай використовується коли мають на увазі програми для мобільних телефонів, або планшетів.

В інформаційних технологіях додаток це прикладна програма або програмне забезпечення — це комп'ютерна програма, призначена для вирішення задач, пов'язаних з певною діяльністю. Відштовхуючись від того, які задачі вона має виконувати, програма може обробляти з текст, математичні операції, музику, графічні матеріали. Також є програми, які працюють з кількома з цих елементів. Одні набори програм заточені на певне одне завдання, як приклад — обробка тексту; інші можуть виконувати декілька завдань, обробка відео і музики разом.[5]

Створені користувачем програми працюють відповідно до самих потреб користувачів. Програми, створені користувачами, мають шаблони електронних таблиць, макроси текстового процесора, моделювання, графіку, аудіо та анімації. Фільтри електронної пошти – це теж програмне забезпечення для користувачів. Користувачі самотужки пишуть ці програми, і дуже часто не знають, яке воно насправді важливе.

Класифікація програмного забезпечення

Є багато способів класифікації програмного забезпечення.

З юридичної точки зору прикладне програмне забезпечення класифікується за способом «чорної скриньки», враховуючи права кінцевих користувачів.

За правами власності

Прикладне програмне забезпечення поділяють на два основні класи: програмне забезпечення із закритим серцевим кодом і програмне забезпечення з відкритим кодом. Також існують безкоштовні, та запатентовані програми.

Запатентовані програми знаходяться під авторським правом, а ліцензія на них надає обмежені права для використання.

Безкоштовні програми з відкритим серцевим кодом розповсюджуються вільно, і можуть змінюватися будь ким.

За мовою програмування

Програми можна також класифікувати за мовою програмування, якою вони були написані, а також щодо їх призначення та результатів роботи. Для створення програм на персональний комп'ютер використовують мови програмування C++, C#, Java, Python. Для створення веб додатків використовують JavaScript, HTML, CSS, TypeScript. Кожна мова має як свої плюси так і мінуси. А також відрізняється за призначенням. Java підходить для створення стабільних і безпечних програм. C# для роботи з графікою. Python для нейронних мереж. А JavaScript є монополістом у сфері веб розробки, і займає усю нішу цілком. Однак, JavaScript також іноді використовують для розробки програм на комп'ютер, або також для мобільної розробки.

За спеціалізацією

Прикладне програмне забезпечення також можна розділити на спеціалізоване або загальне.[12][13] Загальні програми є більш популярними і, так як вони мають загальні функції, наприклад, текстові процесори або бази даних. Спеціалізовані додатки – це вузько-нішеві продукти, що розроблені для певного типу галузі, бізнесу або відділу в організації. Такі програми стараються увібрати у себе усі, потрібні функції, у даній галузі, кожен можливий аспект, наприклад, виробничого чи банківського працівника, бухгалтерії чи обслуговування клієнтів.

Існує багато типів прикладного програмного забезпечення:[14]

- Пакет програм складається з кількох програм, що об’єднані разом. Зазвичай вони мають пов’язані функції, та інтерфейси користувача і можуть взаємодіяти один з одним, наприклад відкривати файли одне одного. Такі додатки зазвичай надаються у пакетах, напр. Microsoft Office, LibreOffice та iWork, які об’єднують текстовий процесор, електронну таблицю тощо; але існують пакети програм і для інших цілей, наприклад музика, відео і т. д.
- Корпоративне програмне забезпечення відповідає потребам цілої організації і може використовуватися у декількох відділах, часто у великому розподіленому середовищі. Приклади: системи планування ресурсів підприємства, системи управління відносинами з клієнтами (CRM), механізми реплікації даних та програмне забезпечення для управління ланцюгом поставок. Департаментальне програмне забезпечення — це підтип корпоративного програмного забезпечення, зосередженого на менших організаціях або групах у великій організації.
- Програмне забезпечення для інфраструктури підприємства має загальні можливості, необхідні для підтримки систем корпоративного програмного забезпечення. (бази даних, сервери електронної пошти та системи для керування мережами та безпекою.)
- Програмне забезпечення для перегляду файлів зазвичай використовується для перегляду вмісту без редагування, але також може мати функціонал, який дозволяє зміну цього файла. Зазвичай це програми для перегляду розважального і цифрового контенту. (Браузери, медіа-програвачі).
- Освітнє програмне забезпечення надає функції для перегляду навчальних матеріалів, тестування, ведення лекцій, контроль процесу навчання, електроний журнал оцінок. Використовується викладачами, студентами, учнями у школі.
- Програмне забезпечення для моделювання дозволяє створювати віртуальні моделі фізичних або абстрактних явищ у наукових цілях, або розважальних цілях.

1.2 Поняття веб-додатку

Веб-додаток — це прикладне програмне забезпечення, яке працює на веб-сервері, на відміну від комп'ютерних програм, що запускаються локально в операційній системі (ОС) пристрою. Веб-додатки використовують архітектуру «клієнт-сервер», де «клієнту» (користувачу) надається доступ до додатку через стороній веб-сервер. Користувач отримує доступ до веб-додатку через веб-брузер, використовуючи мережу Інтернет. Приклади: веб-пошта, онлайн банки, онлайн магазини.

Немає чіткої відмінності між динамічним веб-сайтом будь-якого виду та веб-додатком. Веб-додатки зазвичай мають великий набір функціоналу, схожий на той, який мають програми, розроблені для комп'ютерів. Але програмуються на HTML, CSS, JavaScript. Наразі ці технології є дуже добре розвинутими, і дозволяють створювати додатки будь якої степені складності.

Односторінкові веб додатки (Single page application) дуже схожі на програми, тому що вони не використовують типову веб-парадигму переміщення між сторінками з різними URL-адресами. Тому що використовується новий підхід, у якому окремі компоненти можна замінити або оновити без оновлення всієї веб-сторінки.

У попередніх моделях обчислень, таких як клієнт-сервер, навантаження програми розподілялося між кодом на сервері та кодом, встановленим локально на кожному клієнті. Іншими словами, програма має власну попередньо скомпільовану клієнтську програму як інтерфейс користувача, яку потрібно встановити окремо на ПК кожного користувача. Оновлення коду сервера програми також часто потребує оновлення коду клієнта, встановленого на робочій станції кожного користувача, що збільшує витрати на підтримку та знижує продуктивність. Крім того, як клієнтський, так і серверний компоненти програми часто тісно прив'язані до певної архітектури комп'ютера та операційної системи, і їх перенесення на інші програми часто є дуже дорогим, за винятком найбільших програм (сьогодні рідна мобільна програма також обмежена в деяких відношеннях або у всьому вищезазначеному).

Навпаки, веб-програми використовують веб-документи, написані в стандартних форматах, які підтримують різні веб-браузери, наприклад HTML і JavaScript. Веб-додатки можна розглядати як певний варіант клієнт-серверного програмного забезпечення, яке завантажується на клієнтську машину за допомогою стандартних процедур, таких як HTTP, коли здійснюється доступ до веб-сторінки. Веб-додаток клієнта може оновлюватися кожного разу, коли здійснюється доступ до веб-сторінки. Під час сеансу веб-браузер інтерпретує та відображає сторінки та діє як загальний клієнт для будь-якої веб-програми.

Інтерфейс

Спеціальні методи програми, такі як малювання на екрані, відтворення звуків і доступ до клавіатури та миші, можливі за допомогою JavaScript, CSS, а також Java, Flash і Silverlight. Багато служб об'єднали все це в більш звичний інтерфейс, який нагадує зовнішній вигляд операційної системи. Ці технології також підтримують загальні методи, такі як перетягування. Веб-розробники часто використовують сценарії на стороні клієнта, щоб додати функціонал, особливо для створення інтерактивного досвіду, який не потребує перезавантаження сторінки. Зовсім нещодавно були розроблені технології для координації сценаріїв на стороні клієнта з технологіями на стороні сервера, такими як ASP.NET, J2EE, Perl/Plack і PHP.

Ajax — техніка веб-розробки, що використовує комбінацію різних технологій, створює більш інтерактивний досвід.

Структура інформаційної системи

Програми зазвичай розбиваються на логічні одиниці, які називаються «рівнями», де кожному шару призначається певна роль. [4] Традиційні програми складаються лише з 1-рівневого підходу, який знаходиться на клієнтській машині, але веб-програми за своєю природою піддаються n-рівневому підходу. [4] Хоча можливо багато варіацій, найпоширенішою структурою є трирівнева процедура. [4] У найбільш поширеній формі ці три рівні послідовно називаються рівнем презентації, рівнем додатків і рівнем зберігання. Веб-браузер – це перший рівень (презентація), механізм, який використовує певну технологію динамічного веб-вмісту, як-от ASP, CGI, ColdFusion, Dart, JSP/Java, Node.js, PHP, Python або Ruby

on Rails, є середнім рівнем (логіка програми), а база даних — третій рівень (сховище). [4] Веб-браузери надсилають запити до середнього рівня, який обслуговує їх, надсилаючи запити та оновлюючи бази даних і створюючи інтерфейси користувача.

1.3 Основні принципи веб-розробки

Веб-сайт — це набір веб-сторінок і пов’язаного вмісту, ідентифікованих загальним доменним іменем і опублікованих принаймні на одному веб-сервері. Добре відомі приклади wikipedia.org, google.com і amazon.com.

Усі публічні веб-сайти разом складають Всесвітню павутину. Існують також приватні веб-сайти, які доступні лише в приватній мережі, як-от сайт компанії для її співробітників.

Веб-сайти часто присвячені певній темі чи меті, наприклад новинам, освіті, бізнесу, розвагам або соціальній мережі. Гіперпосилання між веб-сторінками скеровують навігацію сайтом, зазвичай починаючи з домашньої сторінки.

Користувачі можуть отримати доступ до веб-сайту з різних пристроїв, включаючи настільні ПК, ноутбуки, планшети та смартфони. Програми, які використовуються на цих пристроях, називаються веб-браузерами.

Статичний веб-сайт — це веб-сторінка, яка зберігається на сервері у форматі, який надсилається веб-переглядачу клієнта. Він в основному написаний мовою гіпертекстової розмітки (HTML); каскадні таблиці стилів (CSS) використовуються для керування зовнішнім виглядом HTML. Зображення часто використовуються як частина основного вмісту, щоб досягти бажаного вигляду та відчуття. Аудіо чи відео також можна вважати «статичним» вмістом, якщо воно відтворюється автоматично або загалом не є інтерактивним. Цей тип веб-сайтів зазвичай відображає однакову інформацію для всіх відвідувачів. Подібно до роздачі друкованих брошур клієнтам або замовникам, статичні веб-сайти мають тенденцію надавати послідовну стандартну інформацію протягом тривалого періоду часу. Хоча власник веб-сайту може регулярно оновлювати його, редагування тексту, фотографій та іншого вмісту виконується вручну та може

потребувати базових навичок дизайну веб-сайту та програмного забезпечення. Прості форми або маркетингові приклади веб-сайтів, наприклад класичні веб-сайти, веб-сайти з п’ятьма сторінками або веб-сайти з брошурами, часто є статичними веб-сайтами, оскільки вони надають користувачам попередньо визначену статичну інформацію. Це може включати інформацію про компанію та її продукти та послуги, що надаються за допомогою тексту, фотографій, анімації, аудіо/візуальних і навігаційних меню.

Статичні веб-сайти все ще можуть використовувати серверні компоненти (SSI) для зручності редагування, наприклад спільного використання спільного рядка меню на кількох сторінках. Оскільки поведінка веб-сайту для читача залишається статичною, він не вважається динамічним веб-сайтом.

Динамічний веб-сайт – це веб-сайт, який часто й автоматично змінюється або налаштовується. Динамічні сторінки на сервері генеруються «на льоту» за допомогою комп'ютерного коду, який генерує HTML (за зовнішній вигляд відповідає CSS, звідси і статичні файли). Існують різні структури програмування, такі як CGI, Java Servlets і Java Server Pages (JSP), Active Server Pages і ColdFusion (CFML), які можна використовувати для створення динамічних веб-систем і динамічних веб-сайтів. Поширені мови програмування, такі як Perl, PHP, Python і Ruby, надають різноманітні фреймворки веб-додатків і системи веб-шаблонів для полегшення та спрощення створення складних динамічних веб-сайтів.

Сайт може відображати поточний статус діалогу між користувачами, відстежувати зміни умов або певним чином персоналізувати інформацію на основі індивідуальних запитів користувачів. Наприклад, коли запитується домашня сторінка веб-сайту новин, код, запущений на веб-сервері, може об’єднати збережений фрагмент HTML із новинами, отриманими через RSS із бази даних або іншого веб-сайту, щоб створити сторінку з найновішою інформацією. Динамічні сайти можуть бути інтерактивними, використовуючи форми HTML, зберігаючи та зчитуючи файли cookie браузера або створюючи ряд сторінок, які відображають історію минулих кліків. Іншим прикладом динамічного вмісту є веб-сайт роздрібної торгівлі з базою даних медіа-продуктів, який дозволяє користувачам вводити пошуковий запит, наприклад ключове слово "The Beatles".

У відповідь вміст веб-сторінки змінився на те, що було раніше, а потім відобразив список продуктів The Beatles, таких як компакт-диски, DVD-диски та книги. Динамічний HTML використовує код JavaScript, щоб повідомити веб-браузеру, як інтерактивно змінювати вміст сторінки. Один із способів змоделювати певний тип динамічного веб-сайту, уникаючи зниження продуктивності через запуск динамічного механізму для кожного користувача чи з'єднання, — це регулярна автоматична регенерація великої кількості статичних сторінок.

На ранніх сайтах був лише текст, потім з'явилися картинки. Потім почали використовуватися плагіни веб-браузера, щоб додати аудіо, відео та інтерактивність (наприклад, для багатозадачних Інтернет-програм, які відображають складність настільних програм, таких як текстові процесори). Прикладами таких плагінів є Microsoft Silverlight, Adobe Flash, Adobe Shockwave та аплети, написані мовою Java. HTML 5 містить інструменти для аудіо та відео без плагінів. JavaScript також вбудований у більшість сучасних веб-браузерів і дозволяє творцям веб-сайтів надсилати код у веб-браузер, який повідомляє, як інтерактивно змінювати вміст сторінки та спілкуватися з веб-серверами за потреби. Внутрішнє представлення вмісту в браузері називається Document Object Model (DOM), а технологія – Dynamic HTML.

WebGL (Web Graphics Library) — це сучасний API JavaScript для відтворення інтерактивної 3D-графіки без використання плагінів. Завдяки цьому інтерактивний вміст, такий як 3D-анімація, візуалізація та відеопояснення, можна представити користувачам у найбільш інтуїтивно зрозумілий спосіб. [5]

Тенденція веб-сайтів 2010 року під назвою «адаптивний дизайн» забезпечує користувачам оптимальний досвід перегляду, надаючи їм дизайн на основі розміру екрана пристрою. Ці сайти забезпечують багатий досвід користувача, змінюючи дизайн відповідно до пристрою чи платформи. [6]

Веб-сайти можна розділити на дві великі категорії – статичні та інтерактивні. Інтерактивний сайт — це частина спільноти веб-сайтів Web 2.0, яка дозволяє взаємодіяти між власником сайту та відвідувачами або користувачами сайту. Статичні сайти надають або збирають інформацію, але не дозволяють безпосередньо взаємодіяти з аудиторією чи користувачами. Деякі веб-сайти є

інформаційними або створеними любителями або для особистого використання чи розваги. Багато веб-сайтів мають на меті заробити гроші, використовуючи одну або кілька бізнес-моделей, зокрема:

- Публікації цікавого контенту і продаж контекстної реклами через прямі продажі або через рекламні мережі.
- Електронна комерція: придбання товарів або послуг безпосередньо через веб-сайт
- Реклама товарів чи послуг, доступних в організації
- Freemium: базовий вміст доступний безкоштовно, але преміум-контент вимагає оплати (наприклад, сайт WordPress, який є платформою з відкритим кодом для створення блогів або веб-сайтів).

Деякі веб-сайти можуть бути включені до однієї або кількох із цих категорій. Наприклад, веб-сайт компанії може рекламувати продукцію компанії, але він також може надавати інформаційні документи, такі як офіційні документи. Існує багато підкатегорій, крім перерахованих вище. Фан-сайт може бути присвятою власника певній знаменитості. Веб-сайти обмежені архітектурними обмеженнями, такими як обчислювальна потужність, призначена для веб-сайту. Дуже великі веб-сайти, такі як Facebook, Yahoo!, Microsoft і Google, використовують багато серверів і пристроїв балансування навантаження, наприклад комутатори Cisco Content Services, щоб розподілити навантаження відвідувачів між кількома комп’ютерами в різних місцях. На початку 2011 року Facebook використовував дев'ять центрів обробки даних із приблизно 63 000 серверів.

Веб-розробка — це робота, пов’язана з розробкою веб-сайтів для Інтернету (World Wide Web) або Інтранету (приватної мережі). [1] Веб-розробка може варіюватися від розробки простих окремих статичних сторінок звичайного тексту до складних Інтернет-програм (Інтернет-додатків), електронної комерції та соціальних мереж. Більш вичерпний перелік завдань, які зазвичай беруть участь у веб-розробці, може включати веб-інженерію, веб-дизайн, розробку веб-контенту, відносини з клієнтами, створення сценаріїв на стороні клієнт/сервер, налаштування безпеки веб-сервера та мережі та розвиток електронної комерції.

Серед веб-професіоналів «веб-розробка» зазвичай відноситься до основних недизайнерських аспектів створення веб-сайту: написання розмітки та кодування. [2] Веб-розробка може використовувати систему керування контентом (CMS), щоб зробити вміст простішим і доступнішим за допомогою базових технічних інструментів.

Для великих організацій і підприємств групи веб-розробників можуть складатися із сотень людей і використовувати стандартні методології, як-от waterfall/agile, під час розробки веб-сайтів. Меншим організаціям може знадобитися лише один штатний розробник або розробник за контрактом, або додаткові завдання на відповідні посади, наприклад, графічний дизайнер чи технік з інформаційних систем. Веб-розробка може бути результатом спільної роботи відділів, а не доменом певного відділу. Існує три типи професій веб-розробників: інтерфейсний розробник, інтерфейсний розробник і повний розробник. Веб-розробники відповідають за поведінку та візуальні елементи, які працюють у браузері користувача, тоді як інтерфейсні розробники обслуговують сервер.

З часу комерціалізації Інтернету галузь веб-розробок розвивається. Зростання цієї галузі відбувається завдяки компаніям, які прагнуть використовувати свої веб-сайти для реклами та продажу товарів і послуг споживачам. [3]

Існує багато інструментів з відкритим кодом для веб-розробки, таких як BerkeleyDB, GlassFish, стек LAMP (Linux, Apache, MySQL, PHP) і Perl/Plack. Це дає можливість мінімізувати витрати на навчання веб-розробці. Іншим фактором, що стимулює розвиток галузі, є поява легкого у використанні WYSIWYG програмного забезпечення веб-розробки, такого як Adobe Dreamweaver, BlueGriffon і Microsoft Visual Studio. Використання такого програмного забезпечення вимагає знання мови гіпертекстової розмітки (HTML) або мов програмування, але основи можна швидко вивчити та застосувати.

Зростаюча кількість інструментів і методів допомагає розробникам створювати більш динамічні та інтерактивні веб-сайти. Крім того, веб-розробники тепер допомагають розробляти програми як веб-сервіси, які традиційно були доступні лише як програми для настільних ПК. Це відкриває багато можливостей

для поширення інформації та децентралізації ЗМІ. Прикладом є розвиток хмарних сервісів, таких як Adobe Creative Cloud, Dropbox і Google Drive. Ці веб-сервіси дозволяють користувачам взаємодіяти з програмами з кількох місць, а не прив'язуватися до певної робочої станції відповідно до середовища їхньої програми.

Прикладом великих змін у комунікаціях і торгівлі, викликаних веб-розробкою, є електронна комерція. Сайти онлайн-аукціонів, такі як eBay, змінили спосіб, у який споживачі знаходять і купують товари та послуги. Інтернет-магазини, такі як Amazon.com і Buy.com (серед багатьох інших), провели революцію у сфері покупок і торгівлі для багатьох споживачів. Іншим прикладом трансформаційної комунікації, спричиненої веб-розробкою, є ведення блогів. Такі веб-програми, як WordPress і Movable Type, створили середовище для ведення блогів для різних веб-сайтів. Широке використання систем керування контентом із відкритим вихідним кодом і корпоративних систем керування контентом посилило вплив веб-розробки на онлайн-взаємодію та спілкування.

Розвиток веб-сайтів також вплинув на особисте спілкування та маркетинг. Веб-сайт більше не є просто інструментом для роботи чи бізнесу, а ширшою комунікаційною та соціальною мережею. Такі сайти, як Facebook і Twitter, забезпечують платформу для спілкування користувачів і більш особистий та інтерактивний спосіб для організацій залучати громадськість.

Веб-розробка враховує багато аспектів безпеки, наприклад перевірку помилок під час введення даних через форми, фільтрацію вихідних даних і шифрування. Шкідливі практики, такі як впровадження SQL, можуть виконуватися ненавмисно користувачами, але зазвичай вони мають лише базові знання веб-розробки. Сценарії можна використовувати для використання веб-сайтів для надання несанкціонованого доступу зловмисникам, які намагаються зібрати таку інформацію, як адреси електронної пошти, паролі та безпечний вміст, наприклад номери кредитних карток.

Деякі з них залежать від серверного середовища, де працює мова сценаріїв, наприклад ASP, JSP, PHP, Python, Perl або Ruby, і тому не обов'язково підтримуються розробником. Однак, щоб запобігти таким експлойтам,

рекомендується ретельне тестування веб-додатків перед публічним випуском. Якщо на веб-сайті є будь-яка контактна форма, вона повинна містити поле captcha, щоб комп'ютерні програми не могли автоматично заповнювати форму та надсилати електронні листи.

Захист вашого веб-сервера від вторгнення широко відомий як захист портів сервера. Багато технологій використовуються для захисту інформації, що передається з одного місця в інше в Інтернеті. Наприклад, сертифікати TLS (або «сертифікати SSL») видаються центрами сертифікації для запобігання шахрайству в Інтернеті. Багато розробників регулярно використовують різні форми шифрування під час передачі та зберігання конфіденційної інформації. Базове розуміння питань безпеки інформаційних технологій часто є частиною знань веб-розробника.

Оскільки після тестування та випуску нових веб-програм усе ще виявляються нові вразливості системи безпеки, виправлення безпеки часто оновлюються для широко використовуваних програм. Робота веб-розробника часто полягає в тому, щоб постійно оновлювати додатки, коли випускаються патчі безпеки та виявляються нові проблеми.

1.4 Менеджмент

Менеджмент (або управління) — це управління організацією, незалежно від того, чи це бізнес, некомерційна організація чи державна установа. Це мистецтво і наука управління ресурсами бізнесу.

Управління включає діяльність з формулювання стратегії організації та координації зусиль її співробітників (або волонтерів) для досягнення її цілей шляхом використання наявних ресурсів, таких як фінансові, природні, технологічні та людські ресурси. «Ведення бізнесу» [1] і «змінний бізнес» — це дві концепції, які використовуються в менеджменті для розрізнення між продовженням надання товару чи послуги та його пристосуванням до мінливих

потреб клієнтів (див. Тенденції). Термін «менеджмент» також може стосуватися тих, хто керує організацією – менеджерів.

Деякі вивчають менеджмент у коледжах чи університетах; основні ступені менеджменту включають бакалавра комерції (B.Com.), бакалавра ділового адміністрування (BBA.), магістра ділового адміністрування (MBA.), магістра менеджменту (MSM або MIM), для державного сектору також існує державне управління (MPA). Особи, які прагнуть стати фахівцями з менеджменту або фахівцями, дослідниками менеджменту чи професорами, можуть отримати ступінь доктора менеджменту (DM), доктора бізнес-адміністрування (DBA) або доктора бізнес-адміністрування чи менеджменту. За останні кілька десятиліть відбувся рух до управління, заснованого на доказах. [2]

Більші організації зазвичай мають три рівні менеджерів у структурі піраміди [3]:

- Найвищі керівники, такі як члени ради директорів і головний виконавчий директор організації (CEO) або президент, встановлюють стратегічні цілі та політику організації та приймають рішення про те, як організація працює в цілому. Керівники вищої ланки, як правило, є професіоналами виконавчого рівня, які керують і звітують прямо чи опосередковано перед керівництвом середньої ланки.
- Керівний персонал середньої ланки, такий як керівники філій, регіональні менеджери, начальники відділів, керівники відділів тощо, керують персоналом передової лінії управління. Вони повідомляють лінійним керівникам стратегічні цілі та політику вищого керівництва.
- Лінійні керівники, такі як супервайзери та керівники першої лінії, контролюють роботу звичайних працівників (або волонтерів у деяких волонтерських організаціях) і забезпечують керівництво їх роботою. Лінійні керівники зазвичай виконують функції управління, які традиційно вважалися ядром менеджменту. Незважаючи на назву, вони зазвичай вважаються частиною робочої сили, а не частиною ієрархії управління організацією.

У невеликих організаціях менеджери можуть мати ширшу сферу діяльності і виконувати кілька або навіть усі ролі, звичайні для великих організацій.

Соціологи вивчають менеджмент як дисципліну, вивчаючи такі сфери, як соціальна організація, організаційна адаптація та організаційне лідерство. [4]

Рівні керування

Більшість організацій мають три рівні управління: керівництво першого рівня, керівництво середньої ланки та вище керівництво. Лінійні керівники є найнижчим рівнем управління і відповідають за управління роботою некерівного персоналу, який безпосередньо бере участь у виробництві або створенні продукції організації. Менеджерів першої ланки часто називають керівниками, але їх також можна назвати лінійними керівниками, офіс-менеджерами або навіть майстрами. Керівництво середньої ланки включає всі рівні управління від передової лінії організації до верхівки. Ці менеджери контролюють роботу керівників на передньому плані та можуть займати такі посади, як керівники відділів, керівники проектів, керівники заводів або керівники відділів. Вище керівництво несе відповідальність за прийняття загальноорганізаційних рішень і встановлення планів і цілей, які впливають на всю організацію. Ці особи зазвичай займають такі посади, як виконавчий віце-президент, президент, керуючий директор, головний операційний директор, головний виконавчий директор або голова правління.

Ці менеджери поділяються на рівні повноважень і виконують різні завдання. У багатьох організаціях існує піраміда менеджерів на кожному рівні. Нижче на кожному рівні детально описані різні обов’язки та можливі посади. [26]

Вище керівництво

Найвище або вище керівництво — це невелика група, що складається з ради директорів (включаючи невиконавчих директорів, виконавчих директорів і незалежних директорів), президента, віце-президентів, генерального директора та інших виконавчих членів рівня C. У різних організаціях є різні члени, які займають керівні посади в команді, яка може включати фінансового директора, технічного директора тощо. Вони відповідають за контроль і нагляд за діяльністю всієї організації. Вони задають «тон зверху» та визначають стратегічні плани, політику компанії та приймають рішення щодо загального напрямку організації.

Крім того, керівники вищої ланки відіграють важливу роль у мобілізації зовнішніх ресурсів. Керівництво підзвітне акціонерам, громадськості та урядовим установам, які наглядають за компаніями та подібними організаціями. Деякі члени вищого керівництва можуть бути публічним обличчям організації, і вони можуть виголошувати промови, представляючи нові стратегії або виступаючи на тему маркетингу.

Правління, як правило, в основному складається з невиконавчих директорів, які мають фідуціарні обов'язки перед акціонерами та які не беруть участі в щоденному управлінні організацією, хоча це залежить від типу (наприклад, державна чи приватна), розміру та культури організації. Теоретично ці директори несуть відповідальність за порушення цих обов'язків і зазвичай охоплені страхуванням відповідальності директорів і посадових осіб. За оцінками, генеральні директори компаній зі списку Fortune 500 витрачають 4,4 години на тиждень на виконання своїх обов'язків, заробляючи середню винагороду в 212 512 доларів США в 2010 році. Рада встановлює корпоративну стратегію, приймає важливі рішення, такі як великі придбання [27], а також наймає, оцінює та звільняє топ-менеджмент (головний виконавчий директор або генеральний директор). Керівники часто наймають на інші посади. Однак участь правління збільшилася у найманні на інші посади, наприклад, фінансового директора (CFO). [28] У 2013 році опитування понад 160 генеральних директорів і директорів державних і приватних компаній виявило, що головними слабкими сторонами генеральних директорів були «керівна здатність» і «участь у правлінні», а 10% переоцінених [29]. Рада також може мати певних працівників (наприклад, внутрішніх аудиторів), які підпорядковуються їй, або безпосередньо наймати незалежних підрядників; наприклад, рада (через аудиторський комітет) зазвичай вибирає аудиторів.

Корисні навички для керівників вищої ланки залежать від типу організації, але зазвичай включають[30] широке розуміння конкуренції, глобальної економіки та політики. Крім того, генеральний директор несе відповідальність за реалізацію та визначення (у межах правління) широкої політики організації. Виконавче керівництво виконує повсякденні деталі, включаючи: інструкції щодо

бюджетування відділу, процедури, графіки; призначення керівників середньої ланки, наприклад керівників відділів; координацію відділів; зв'язки із ЗМІ та урядом; спілкування з акціонерами.

Середній менеджмент

До його складу входять генеральний директор, начальник філії та начальник відділу. Вони підзвітні вищому керівництву за виконання функцій свого відділу. Вони більше часу приділяють організаційно-управлінським функціям. Їхню роль можна підкреслити як впровадження організаційних планів відповідно до політики компанії та цілей вищого керівництва, вони визначають і обговорюють інформацію та політику від вищого керівництва до нижчого керівництва, і, що найважливіше, вони мотивують і направляють нижче керівництво. для кращої продуктивності.

Менеджери середнього рівня — це менеджери середнього рівня організації, які підпорядковані вищому керівництву, але вище, ніж члени керівництва найвищого рівня. Операційний менеджер може бути добре продуманим керівником середньої ланки або класифікуватися як некерівна особа, відповідальна за певну політику організації. Продуктивність середнього рівня має вирішальне значення для будь-якої організації, оскільки вона доповнює розрив між верхнім і нижнім рівнями.

Їх функції включають:

1. Розробка та впровадження ефективних групової і міжгрупової роботи та інформаційних систем.
2. Визначення та відстеження показників ефективності на рівні групи.
3. Діагностування та вирішення внутрішньогрупових та міжгрупових проблем.
4. Розробка та впровадження системи винагород, які підтримують кооперативну поведінку. Вони також приймають рішення та діляться ідеями з вищим керівництвом.

Лінійний менеджмент

До лінійних керівників належать супервайзери, керівники відділів, керівники та керівники груп. Вони зосереджені на контролі та керівництві штатними працівниками. Зазвичай вони відповідають за призначення завдань

працівникам, керівництво та нагляд за повсякденною діяльністю працівників, забезпечення якості та кількості продуктів та/або послуг, внесення пропозицій та пропозицій щодо роботи працівників та вирішення проблем, з якими працівники не можуть вирішувати питання керівництву середньої ланки або іншому управлінському персоналу. Менеджери першого рівня або «передові» також подають приклад співробітникам. На деяких роботах лінійні керівники також можуть виконувати деякі з тих самих завдань, що й співробітники, принаймні деякий час. Наприклад, у деяких ресторанах лінійні менеджери також обслуговують клієнтів у дуже зайнятий час доби. Як правило, лінійні керівники вважаються частиною робочої сили, а не частиною належного керівництва організації, незважаючи на виконання традиційних функцій управління.

Менеджери першої лінії зазвичай забезпечують:

- Навчання для нових співробітників
- Основний нагляд
- Мотивацію
- Відгуки про ефективність і керівництво

Деякі топ-менеджери можуть також забезпечувати планування кар'єри для співробітників, які прагнуть просуватися в організації.

1.5 Поняття інформаційної системи

Інформаційна система (ІС) — це формальна соціально-технічна організаційна система, призначена для збору, обробки, зберігання та розповсюдження інформації. [1] З соціотехнічної точки зору інформаційні системи складаються з чотирьох компонентів: завдань, людей, ролей і технології. [2] Інформаційну систему можна визначити як інтеграцію компонентів збору, зберігання та обробки даних, дані яких використовуються для надання інформації, полегшення отримання знань і полегшення прийняття рішень у цифрових продуктах [3].

Комп’ютеризована інформаційна система – це система, що складається з людей і комп’ютерів, які обробляють або інтерпретують інформацію. [4][5][6][7]

Термін також іноді використовується просто для позначення комп'ютерної системи, на якій встановлено програмне забезпечення.

«Інформаційні системи» також є академічним полем дослідження систем, конкретно посилаючись на інформацію та додаткові мережі комп'ютерного обладнання та програмного забезпечення, які люди та організації використовують для збору, фільтрації, обробки, створення та розповсюдження даних. [8] Основна увага приділяється інформаційним системам із чітко визначеними межами, користувачами, процесорами, сховищем, входами, виходами та комунікаційними мережами, як описано вище. [9]

У багатьох організаціях відділ або відділ, відповідальний за інформаційні системи та обробку даних, називається «Інформаційні служби». [10][11][12][13]

Будь-яка інформаційна система розроблена для підтримки операцій, управління та прийняття рішень. [14][15] Інформаційні системи — це інформаційно-комунікаційні технології (ІКТ), які використовуються організаціями, і способи взаємодії людей із цією технологією для підтримки бізнес-процесів. [16]

Деякі автори чітко розрізняють інформаційні системи, комп'ютерні системи та бізнес-процеси. Інформаційні системи часто включають компоненти ІКТ, але не пов'язані виключно з ІКТ, а натомість зосереджені на кінцевому використанні інформаційних технологій. Інформаційні системи також відрізняються від бізнес-процесів. Інформаційні системи допомагають контролювати ефективність бізнес-процесів. [17]

Alter [18][19] продемонстрував переваги розгляду інформаційних систем як особливого типу робочих систем. Робоча система - це система, в якій людина або машина використовує ресурси для виконання процесів і дій для виробництва конкретного продукту або послуги для клієнта. Інформаційна система – це працююча система, діяльність якої спрямована на захоплення, передачу, зберігання, пошук, обробку та відображення інформації. [20]

Таким чином, інформаційні системи взаємопов'язані з системами даних з одного боку та з системами діяльності з іншого [21]. Інформаційна система — це форма комунікаційної системи, в якій дані представлені й обробляються як форма

соціальної пам’яті. Інформаційні системи також можна розглядати як напівформальну мову, яка підтримує прийняття рішень і дії людини.Інформаційна система є основним напрямком досліджень організаційної інформатики [22].

Відділи інформаційних технологій у великих організаціях часто мають значний вплив на розвиток, використання та застосування інформаційних технологій у бізнесі. Існує багато методів і процесів, які можна використовувати для розробки та використання інформаційних систем. Багато розробників використовують методи системної інженерії, такі як життєвий цикл розробки систем (SDLC), для систематичної поетапної розробки інформаційних систем. Фазами життєвого циклу розробки системи є планування, аналіз системи та вимоги, проектування, розробка, інтеграція та тестування, впровадження та експлуатація та технічне обслуговування. Останні дослідження спрямовані на досягнення [29] та вимірювання [30] постійного колективного розвитку таких систем в організаціях за допомогою всіх. Інформаційні системи можуть бути розроблені власними силами (всередині організації) або сторонніми. Цього можна досягти за допомогою аутсорсингу окремих компонентів або всієї системи. [31] Окремим випадком є географічний розподіл команд розробників (офшори, глобальні інформаційні системи).

Види інформаційної системи

«Класичний» погляд на інформаційні системи в підручниках [27] у 1980-х роках — це системна піраміда, що відображає організаційну ієрархію, як правило, система обробки транзакцій у нижній частині піраміди, за якою йдуть інформаційні системи управління та системи підтримки прийняття рішень. , і закінчується виконавчою інформаційною системою вгорі. Хоча модель піраміди була корисною з моменту її першого формулювання, було розроблено багато нових технологій і з’явилися нові класи інформаційних систем, деякі з яких більше не відповідають початковій моделі піраміди.

Деякі приклади таких систем:

- система підтримки прийняття рішень
- соціальні інформаційні системи
- система управління процесом

- інформаційна система управління
- системи підприємства
- сховища даних
- планування ресурсів підприємства
- обчислювальна платформа
- експертні системи
- пошукові системи
- геоінформаційна система
- мультимедійна інформаційна система
- автоматизація офісу.

Інформаційна система (на основі комп’ютера) — це, по суті, ІС, яка використовує комп’ютерні технології для виконання деяких або всіх запланованих завдань. Основними компонентами комп’ютерної інформаційної системи є:

Апаратне забезпечення – це такі пристрої, як монітори, процесори, принтери та клавіатури, які працюють разом для отримання, обробки та відображення даних та інформації.

- Програмне забезпечення - це програми, які дозволяють апаратному забезпеченню обробляти дані.
- Бази даних - це набір пов'язаних файлів або таблиць, що містять пов'язані дані.
- Мережі - це сполучна система, яка дозволяє різним комп'ютерам розподіляти ресурси.
- Процедури – це команди для об’єднання вищевказаних компонентів для обробки інформації та отримання бажаного результату.

Перші чотири компоненти (апаратне забезпечення, програмне забезпечення, база даних і мережа) складають те, що називається платформою інформаційних технологій. Персонал з інформаційних технологій може потім використовувати ці компоненти для створення інформаційних систем для контролю заходів безпеки, управління ризиками та даними. Ці види діяльності називаються послугами інформаційних технологій. [28]

Деякі інформаційні системи підтримують частини організації, інші підтримують цілі організації, а треті підтримують групи організацій. Пам’ятайте, що кожен відділ або функціональна область в організації має власну колекцію програм або інформаційних систем. Ці функціональні зональні інформаційні системи (FAIS) є будівельними блоками більш загальної ІС, а саме систем бізнес-аналітики та інформаційних панелей. Як випливає з назви, кожен FAIS підтримує певну функцію в організації, наприклад: бухгалтерський облік, фінансові системи, системи управління виробництвом (POM), системи маркетингу та людські ресурси. У сфері фінансів і бухгалтерського обліку менеджери використовують IT-системи для прогнозування доходів і ділової активності, визначення найкращих джерел і використання коштів, а також проведення аудитів, щоб переконатися, що організації є надійними, а всі фінансові звіти та документи точні. Іншими типами організаційних інформаційних систем є FAIS, системи обробки транзакцій, планування ресурсів підприємства, системи автоматизації офісу, інформаційні системи управління, системи підтримки прийняття рішень, експертні системи, панелі керування, системи управління ланцюгами поставок та системи електронної комерції. Інформаційна панель — це особлива форма ІС, яка підтримує всіх керівників організації. Вони забезпечують швидкий доступ до своєчасної інформації, а також прямий доступ до структурованої інформації у формі звітів. Експертні системи намагаються відтворити роботу експертів, застосовуючи здатність міркувати, знання та досвід у певній області.

За визначенням Лангефорса [32] комп’ютеризована інформаційна система – це технічно реалізоване середовище для:

- запису, зберігання та відтворення мовних виразів,
- для того, щоб робити висновки з таких виразів.

Географічні інформаційні системи, земельні інформаційні системи та системи інформації про стихійні лиха є прикладами нових інформаційних систем, але в цілому їх можна вважати системами просторової інформації. Розробка системи здійснюється поетапно, включаючи:

- Розпізнавання та уточнення проблеми
- Збір інформації

- Специфікація вимог до нової системи
- Проектування системи
- Побудова системи
- Реалізація системи
- Огляд і обслуговування.[33]

2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Архітектура і модель життєвого циклу інформаційної системи

Модель «клієнт-сервер» — це структура розподіленої програми, яка розподіляє завдання або робочі навантаження між постачальниками ресурсів або послуг (званими серверами) і запитувачами послуг (званими клієнтами). Клієнти та сервери зазвичай спілкуються через комп’ютерну мережу на різному обладнанні, але клієнт і сервер можуть бути в одній системі. Хост сервера запускає одну або кілька серверних програм, які спільно використовують ресурси з клієнтами. Клієнти зазвичай не надають спільний доступ до власних ресурсів, а натомість запитують вміст або послуги на сервері. Таким чином, клієнт ініціює сеанс зв'язку з сервером в очікуванні вхідних запитів. Прикладами комп’ютерних програм, які використовують модель клієнт-сервер, є електронна пошта, мережевий друк і Всесвітня павутина.

Характеристики клієнт-сервер описують зв'язок між взаємодіючими програмами в додатку. Серверний компонент надає функціональні можливості або послуги одному або кільком клієнтам, які ініціюють такі запити на обслуговування. Сервери класифікуються відповідно до послуг, які вони надають. Наприклад, веб-сервер розміщує веб-сторінки, а файловий сервер — комп’ютерні сторінки. Спільними ресурсами може бути будь-що: від серверного програмного забезпечення та електронних компонентів до програм і даних, до процесорів і пристроїв зберігання даних. Спільні ресурси сервера є послугою.

Те, чи є комп’ютер клієнтом, сервером або тим і іншим, визначається характером програми, яка потребує функції обслуговування. Наприклад, програмне забезпечення комп’ютера та файлового сервера може працювати на одному комп’ютері, щоб обслуговувати різні дані клієнтам, які роблять різні типи запитів. Клієнтське програмне забезпечення також може взаємодіяти з серверним

програмним забезпеченням на тому самому комп’ютері. Зв’язок між серверами (наприклад, для синхронізації даних) іноді називають сервер-сервер.

Загалом кажучи, послуга є абстракцією обчислювальних ресурсів, і клієнту не потрібно піклуватися про те, як сервер поводиться під час процесу запиту та відповіді. Клієнт повинен розуміти відповідь лише на основі відомого протоколу програми, тобто вмісту та формату даних, які запитуються для служби.

Клієнти та сервери обмінюються повідомленнями за шаблоном обміну повідомленнями запит-відповідь. Клієнт надсилає запит, а сервер повертає відповідь. Це повідомлення є прикладом міжпроцесорного зв'язку. Щоб комп’ютери могли спілкуватися, вони повинні мати спільну мову та правила, щоб і клієнт, і сервер знали, чого очікувати. Мова та правила спілкування визначаються протоколом спілкування. Усі клієнт-серверні протоколи працюють на прикладному рівні. Протокол прикладного рівня визначає базову діалогову схему. Для подальшої стандартизації обміну даними сервери можуть реалізувати інтерфейси прикладного програмування (API). API — це рівень абстракції для доступу до послуг. Це полегшує аналіз, обмежуючи комунікації певними форматами контенту. Абстрагуючи доступ, він полегшує міжплатформений обмін даними.

Сервер може отримувати запити від багатьох різних клієнтів за короткий проміжок часу. Комп’ютери можуть обробляти обмежену кількість завдань у будь-який час і покладатися на системи планування для визначення пріоритетів вхідних запитів клієнтів для їх задоволення. Щоб запобігти зловживанням і збільшити доступність, програмне забезпечення сервера може обмежувати доступність клієнта. Атаки на відмову в обслуговуванні мають на меті використати зобов’язання сервера обробляти запити, перевантажуючи сервер надто частими запитами. Якщо конфіденційну інформацію потрібно передавати між клієнтом і сервером, слід використовувати шифрування.

Коли клієнт банку використовує веб-браузер (клієнт) для доступу до послуг онлайн-банкінгу, клієнт ініціює запит до веб-сервера банку. Облікові дані клієнта можна зберігати в базі даних, при цьому веб-сервер звертається до сервера бази даних як клієнт. Сервер додатків використовує бізнес-логіку банку для

інтерпретації повернутих даних і надання вихідних даних веб-серверу. Нарешті, веб-сервер повертає відображені результати веб-браузеру клієнта.

На кожному кроці цієї послідовності обміну повідомленнями клієнт-сервер комп'ютер обробляє запит і повертає дані. Це шаблон повідомлення запит-відповідь. Коли всі запити задовольняються, послідовність завершується, і веб-браузер передає дані клієнту.

Цей приклад ілюструє шаблон проектування, застосовний до моделі клієнт-сервер.

Модель клієнт-сервер не вимагає, щоб хост-сервер мав більше ресурсів, ніж клієнт-хост. Замість цього він дозволяє будь-якому комп'ютеру загального призначення розширювати свої можливості шляхом спільного використання ресурсів інших хостів. Однак централізовані обчислення розподіляють велику кількість ресурсів між кількома комп'ютерами. Чим більше обчислень переноситься з клієнтського хоста на центральний комп'ютер, тим простіший клієнтський хост. Він значною мірою покладається на мережеві ресурси (сервери та інфраструктуру) для обчислень і зберігання. Бездисковий вузол навіть завантажує власну операційну систему з мережі, тоді як комп'ютерний термінал взагалі не має операційної системи і має лише серверний інтерфейс введення-виведення. Клієнти, такі як персональні комп'ютери, багаті на ресурси і не покладаються на сервери для виконання основних функцій.

З 1980-х до кінця 1990-х років, коли ціна та можливості мікрокомп'ютерів падали, багато організацій перемістили свої обчислення з централізованих серверів, таких як мейнфрейми та міні-комп'ютери, на товсті клієнти. Це робить управління комп'ютерними ресурсами більш персональним, але управління інформаційними технологіями стає більш складним. У 2000-х роках веб-програми були вже достатньо розвинутими, щоб конкурувати з прикладним програмним забезпеченням, розробленим для певної мікроархітектури. Поява цієї розвинутої, більш доступної пам'яті та сервіс-орієнтованої архітектури стала одним із факторів, які призвели до тенденції хмарних обчислень у 2010-х роках.

На додатках клієнт-серверної моделі, розподілені обчислювальні програми часто використовують однорангову (P2P) архітектуру.

У моделі клієнт-сервер сервер, як правило, призначений для роботи як централізованої системи так і обслуговування багатьох клієнтів. Вартість обчислювальної потужності, пам’яті та пам’яті сервера слід коригувати відповідно до очікуваного навантаження. Системи балансування навантаження та відновлення після відмови часто використовуються для масштабування серверів за межі однієї фізичної машини.

Балансування навантаження визначається як упорядкований і ефективний розподіл мережевого або службового трафіку між декількома серверами в серверній. Кожен балансувальник навантаження знаходиться між клієнтським пристроєм і сервером, приймає вхідні запити та розподіляє їх на будь-який доступний сервер, який може їх виконати. У одноранговій мережі два або більше комп’ютерів (однорангова мережа) об’єднують свої ресурси та взаємодіють у децентралізованій системі. Однорангові вузли — це рівні або еквівалентні вузли в неієрархічній мережі. На відміну від клієнтів у мережі клієнт-сервер або черзі клієнт-клієнт, одноранговий зв’язок є прямим. У одноранговій мережі одноранговий алгоритм балансує навантаження, і навіть вузли зі скромними ресурсами можуть допомогти розподілити навантаження. Якщо сайт недоступний, він залишається доступним, поки інші партнери діляться його ресурсами. В ідеалі одноранговій мережі не потрібно досягати високої доступності, оскільки інші додаткові вузли можуть компенсувати будь-які неактивні ресурси; протокол перенаправляє запити в міру зміни доступності та перевантаження.

І клієнт-сервер, і головний-підлеглий вважаються підкатегоріями розподілених однорангових систем.

Загальний прототип системи показано на рис.2.1.

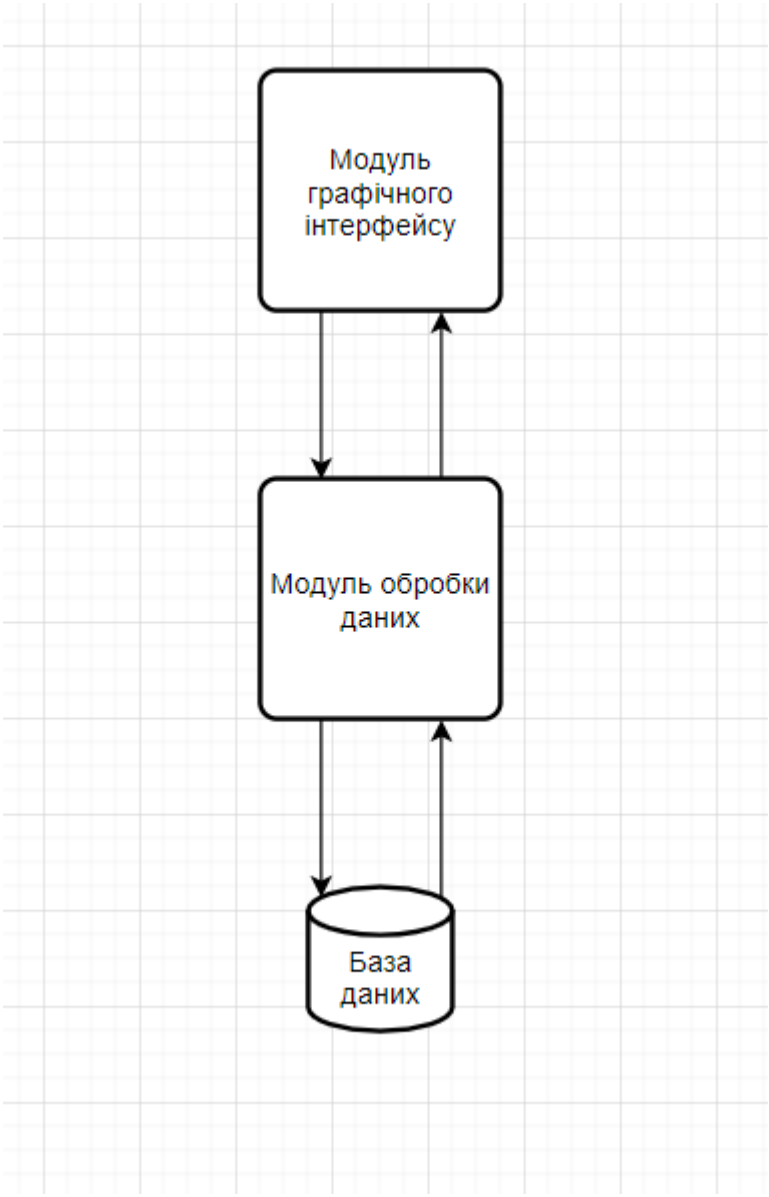

Рисунок 2.1 — Прототип системи

Водоспадна модель – це поділ проектної діяльності на лінійно послідовні етапи, тобто вони переходять з одного в інший, де кожен етап залежить від результатів попереднього етапу та відповідає спеціалізації завдань. [1] Такий підхід характерний для деяких напрямків інженерного проектування. У розробці програмного забезпечення [1] це, як правило, один із менш ітеративних і менш гнучких методів, оскільки прогрес тече в основному в одному напрямку («вниз», як у водоспаді), через концепцію, ініціацію, аналіз, дизайн, збірку, тестування та інші етапи, розгортання та обслуговування. [2]

Водоспадна модель розробки бере свій початок у промисловості та будівництві, де високоструктуроване фізичне середовище означало, що зміни дизайну стали дуже дорогими на ранніх стадіях процесу розробки. Коли розробка програмного забезпечення вперше була прийнята, не було прийнятого підходу, заснованого на знаннях. [3]

Перша відома демонстрація, що описує використання цих етапів у розробці програмного забезпечення, була проведена Феліксом Торресом і Гербертом Д. Беннінгтоном 29 червня 1956 року на симпозіумі з передових методів програмування для цифрових комп'ютерів. [4] Ця презентація присвячена розробці програмного забезпечення для SAGE. У 1983 році статтю було перевидано у передмові Беннінгтона, який пояснив, що етапи були організовані спеціально відповідно до спеціалізації завдання, зазначивши, що процес насправді не був суворо зверху вниз, а спирався на прототипи. [3]

Хоча термін «водоспад» не використовується в статті, перша формальна детальна діаграма процесу (пізніше названа «моделлю водоспаду») часто цитується як стаття 1970 року Вінстона Ройса. [5][6][7] Однак він також вважав це глибоко помилковим, оскільки тестування проводилося лише в кінці процесу, який він описав як «ризикований і схильний до невдач». [5] Інша частина його статті представляє п'ять кроків, які, на його думку, є необхідними для «усунення більшості ризиків розвитку, пов'язаних із підходом незмінного водоспаду». [5]

П'ять поетапних кроків Ройса (включно з написанням повної документації на різних етапах розробки) ніколи не прижилися, але його діаграма процесу з недоліками стала відправною точкою для опису підходу водоспаду. [8]

Ймовірно, найдавніше використання терміну «водоспад» було в статті 1976 року Белла і Теєра. [9]

У 1985 році Міністерство оборони США кодифікувало цей підхід у DOD-STD-2167A, а їхні Стандарти для підрядників із розробки програмного забезпечення стверджували, що «Підрядники повинні запроваджувати цикл розробки програмного забезпечення, який складається з таких шести етапів: аналіз вимог до програмного забезпечення, Попередній проект, детальний дизайн, кодування та модульне тестування, інтеграція та тестування».[10]

Модель

У оригінальній моделі водоспаду Ройса такі фази дотримуються по порядку:

1. Вимоги до системи та програмного забезпечення: вказано в документі вимог до продукту
2. Аналіз: у результаті створюються моделі, схеми та бізнес-правила
3. Дизайн: в результаті створюється архітектура програмного забезпечення
4. Кодування: розробка, перевірка та інтеграція програмного забезпечення
5. Тестування: систематичне виявлення та усунення дефектів
6. Операції: встановлення, міграція, підтримка та технічне обслуговування повних систем

Таким чином, модель водоспаду стверджує, що фазу слід вводити, лише якщо попередню фазу було переглянуто та перевірено.

Однак різні модифіковані моделі водоспаду, включно з остаточною моделлю Ройса, можуть включати незначні або значні зміни.[5] Ці зміни включають повернення до попереднього циклу після виявлення дефекту або повне повернення до фази проектування, якщо наступна фаза вважається недостатньою.

Витрачання часу на ранній стадії циклу виробництва програмного забезпечення може зменшити витрати пізніше. Наприклад, проблему, виявлену на початку процесу (наприклад, специфікація вимог), дешевше виправити (у 50-200 разів), ніж ту саму помилку, знайдену пізніше в процесі. [11]

У звичайній практиці метод водоспаду створює графік проекту, в якому 20-40% часу витрачається на перші дві фази, 30-40% витрачається на кодування, а решта витрачається на тестування та впровадження. Реальна організаційна структура проекту має бути чіткою. Більшість великих і середніх проектів включатимуть детальний набір процедур і засобів контролю для управління кожним процесом у проекті. [12]

Ще одним аргументом на користь моделі водоспаду є її акцент на документах (таких як документи вимог і проектні документи), а також на вихідному коді. У менш ретельно розроблених і задокументованих підходах, якщо члени команди залишають проект до завершення проекту, знання втрачаються, і проект може мати труднощі з відновленням після втрати. Якщо існує повний

робочий проектний документ (як передбачається Big Design Up Front і Waterfall Model), нові члени команди, навіть зовсім нові команди, повинні мати можливість ознайомитися, прочитавши документ. [13]

Модель водоспаду забезпечує структурований підхід; саму модель легко зрозуміти, оскільки вона проходить лінійно через окремі, легко зрозумілі та інтерпретовані етапи; вона також забезпечує легко ідентифіковані віхи в процесі розробки. Можливо, з цієї причини модель водоспаду використовується в багатьох підручниках і курсах програмної інженерії як початковий приклад моделі розробки. [14]

Критика

Клієнти можуть не знати точно, які їхні потреби, доки не побачать робоче програмне забезпечення, тому їхні потреби змінюються, що призводить до перепроектування, перепроектування та повторного тестування та збільшення витрат. [15]

Розробники можуть не знати про майбутні труднощі під час розробки нового програмного продукту або функції, і в цьому випадку краще переглянути дизайн, ніж дотримуватися дизайну, який не враховує будь-які нещодавно виявлені обмеження, вимоги чи проблеми. [16]

Організації можуть спробувати усунути відсутність конкретних потреб клієнтів, найнявши системного аналітика для вивчення існуючих ручних систем і аналізу того, що вони роблять і як їх можна замінити. На практиці, однак, важко чітко розділити системний аналіз і програмування. [17] Це пояснюється тим, що впровадження будь-якої значної системи майже неминуче виявить проблеми та крайові випадки, які системний аналітик не врахував.

Для вирішення проблем, визначених чистою моделлю водоспаду, були представлені модифіковані моделі водоспаду, такі як «Сашимі (водоспад етапів, що перекриваються), водоспад підпроекту та водоспад зменшення ризику». [11]

Деякі організації, такі як Міністерство оборони США, наразі підтримують каскадний підхід, починаючи з MIL-STD-498, опублікованого в 1994 році, який заохочує еволюційне отримання та ітераційну та поступову розробку. [18]

Хоча прихильники гнучкої розробки програмного забезпечення стверджують, що водоспадна модель є неефективним процесом розробки програмного забезпечення, деякі скептики вважають, що водоспадна модель є хибним аргументом, який використовується лише для просування альтернативних методів розробки. [19]

Фази Rational Unified Process (RUP) визнають потреби програмування фаз, щоб підтримувати проекти в потрібному напрямку, але заохочують ітерації всередині фаз (особливо в межах дисциплін). Фази RUP часто називають «водоспадами».

Модифіковані моделі водоспаду

У відповідь на проблеми, виявлені з "чистою" моделлю водоспаду, було введено кілька модифікованих моделей водоспаду. Ці моделі можуть зустріти частину або всю критику "чистої" моделі водоспаду.

До них належать те, що Стів МакКоннелл називає «покращеними водоспадами» [11]: «модель сашимі» Пітера ДеГрейса (водоспад із етапами, що перекриваються), водоспад із підпроектами та водоспад із зменшенням ризику. Існують інші комбінації моделей розробки програмного забезпечення, такі як «модель інкрементного водоспаду» [20].

Остаточна модель Ройса

Остаточна модель Вінстона В. Ройса, яку він мав намір вдосконалити на основі оригінальної «моделі водоспаду», показала, що зворотній зв’язок може (і часто відбувається) надходити від тестування коду до дизайну (як тестовий код, який виявляє недоліки дизайну), і від дизайну назад до специфікація вимог (оскільки проблеми з дизайном можуть вимагати вирішення конфліктів або інших незадовільних/нерозроблених вимог). У тій самій статті Ройс також виступає за розширену документацію, «роблячи це двічі, якщо можливо» та максимізуючи залучення клієнтів (схоже на екстремальне програмування).

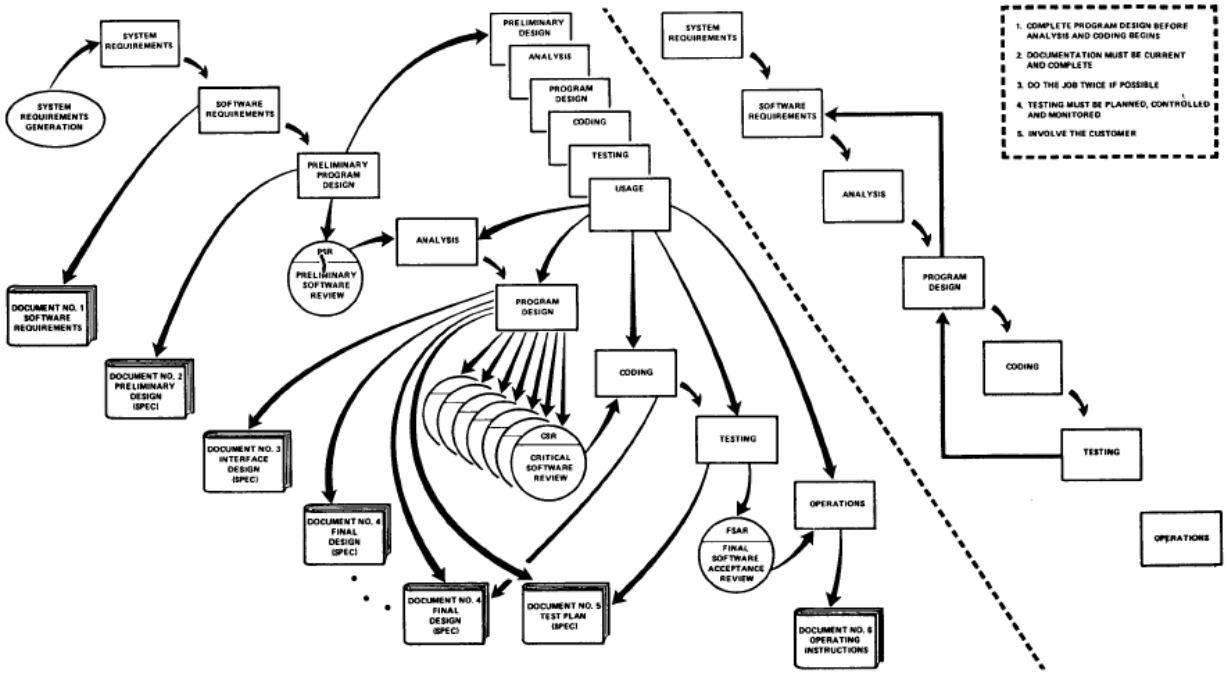

Рисунок 2.2 — Схема життєвого циклу

Примітки Royce до остаточної моделі:

1. Завершити проектування програми перед початком аналізу та розробки
2. Документація має бути актуальною та повною
3. Виконайте роботу двічі, якщо можливо
4. Тестування необхідно планувати та контролювати
5. Залучайте клієнта

2.2 Огляд мови програмування

JavaScript, який часто скорочують до JS, є мовою програмування, яка поряд з HTML і CSS є однією з основних технологій Всесвітньої павутини. Усі основні веб-браузери мають спеціальний механізм JavaScript для виконання коду на пристрої користувача.

JavaScript — це мова високого рівня, яка відповідає стандарту ECMAScript. [10] Він має динамічну типізацію, об’єктну орієнтацію на основі прототипу та першокласні функції. Це мультипарадигма, що підтримує керований подіями,

функціональний та імперативний стилі програмування. Він має інтерфейси прикладного програмування (API) для роботи з текстом, датами, регулярними виразами, стандартними структурами даних і об'єктною моделлю документа (DOM).

Стандарт ECMAScript не включає засоби введення/виведення (I/O), такі як мережеві засоби, засоби зберігання чи графіка. Насправді веб-браузер або інша система виконання надає JavaScript API для введення-виведення.

Механізми JavaScript спочатку використовувалися лише у веб-браузерах, але зараз є основним компонентом деяких серверів і різноманітних програм. Найпопулярнішим середовищем виконання для цієї мети є Node.js.

Незважаючи на те, що Java і JavaScript схожі за назвою, синтаксисом і відповідними стандартними бібліотеками, ці дві мови дуже відрізняються одна від одної та сильно відрізняються за дизайном.

Перший веб-браузер із графічним інтерфейсом користувача Mosaic був випущений у 1993 році. Він також був доступний для нетехнічних людей, і він відіграв видатну роль у швидкому розвитку Всесвітньої павутини, що зароджувалася. [11] Провідний розробник Mosaic заснував Netscape Corporation, яка випустила вдосконалений браузер Netscape Navigator у 1994 році. Він швидко став найбільш використовуваним. [12][13]

У роки становлення Інтернету веб-сторінки могли бути лише статичними і не могли працювати динамічно після завантаження сторінки в браузер. Нова сцена веб-розробки хотіла усунути це обмеження, тому в 1995 році Netscape вирішила додати мову сценаріїв до Navigator. Для цього вони розглянули два підходи: вони співпрацювали з Sun Microsystems, щоб реалізувати мову програмування Java, і вони найняли Брендана Айха, щоб реалізувати мову Scheme. [6]

Керівництво Netscape швидко вирішило, що найкращим варіантом для Айха є розробка нової мови з синтаксисом, подібним до Java, але не надто відмінним від Scheme чи інших існуючих мов сценаріїв. [5][6] Хоча нова мова та її реалізація інтерпретатора називалися LiveScript, коли вона була вперше представлена як частина бета-версії Navigator у вересні 1995 року, назву змінили на JavaScript в офіційному випуску в грудні. [6][1][14]

Вибір назви JavaScript викликав плутанину, припускаючи, що вона безпосередньо пов’язана з Java. У той час почалася бульбашка доткомов, і Java була популярною новою мовою, тому Айх вважав, що назва JavaScript — це маркетинговий хід Netscape. [15]

Microsoft запустила Internet Explorer у 1995 році, спричинивши війну браузерів із Netscape. Що стосується JavaScript, Microsoft переробила інтерпретатор Navigator, щоб створити власний інтерпретатор під назвою JScript. [16]Вперше випущений у 1996 році, JScript спочатку підтримував розширення CSS і HTML. Кожна з цих реалізацій суттєво відрізняється від аналогів Navigator. [17][18] Через ці відмінності розробникам було важко змусити свої сайти добре працювати в обох браузерах, що призвело до років логотипу «перегляд найкраще в Netscape» і «перегляд в Internet Explorer» Широке використання найкращих результатів». [17][19]

У листопаді 1996 року Netscape представила JavaScript Ecma International як відправну точку для стандартної специфікації, якої могли дотримуватись усі постачальники браузерів. Це призвело до офіційної публікації першої специфікації мови ECMAScript у червні 1997 року.

Процес стандартизації тривав кілька років: ECMAScript 2 був опублікований у червні 1998 року, а ECMAScript 3 – у грудні 1999 року. Робота над ECMAScript 4 почалася в 2000 році. [16]

У той же час домінування Microsoft на ринку браузерів зростає. До початку 2000-х років частка ринку Internet Explorer становила 95% [20]. Це означає, що JScript стає стандартом де-факто для сценаріїв веб-клієнта.

Microsoft спочатку брала участь у процесі стандартизації та реалізувала деякі пропозиції у своїй мові JScript, але пізніше припинила співпрацю над роботою Ecma. Тому ECMAScript 4 було заархівовано.

Створення Node.js Райаном Далем у 2009 році призвело до значного збільшення використання JavaScript поза веб-браузерами. Node поєднує механізм V8, цикл подій і API введення/виведення, щоб забезпечити автономну систему виконання JavaScript. [27][28] Станом на 2018 рік мільйони розробників

використовували Node [29], а npm має найбільше модулів серед усіх менеджерів пакетів у світі [30].

Проект специфікації ECMAScript наразі публічно підтримується на GitHub із регулярними щорічними знімками, які використовуються для створення випусків. [31] Ознайомтеся з можливими змінами мови за допомогою комплексного процесу пропозиції. [32][33] Замість того, щоб перевіряти номер версії, розробники тепер перевіряють статус майбутніх функцій окремо. [31]

Поточна екосистема JavaScript має багато бібліотек і фреймворків, усталені методи програмування та інтенсивне використання JavaScript поза веб-браузерами. Крім того, з появою односторінкових додатків та інших веб-сайтів, наповнених JavaScript, було створено кілька транспіляторів, які допомагають у процесі розробки. [34]

TypeScript

TypeScript — це безкоштовна мова програмування з відкритим вихідним кодом, розроблена та підтримувана Microsoft. Це набір синтаксису JavaScript який додає необов’язкову статичну типізацію до мови. Він призначений для великомасштабної розробки додатків і компіляцію на JavaScript. [5] Оскільки це надмножина JavaScript, існуючі програми JavaScript також є дійсними програмами TypeScript.

TypeScript можна використовувати для розробки програм JavaScript для виконання на стороні клієнта та на стороні сервера (як у випадку з Node.js або Deno). Доступно кілька варіантів компіляції. Ви можете використовувати стандартний компілятор TypeScript [6] або викликати компілятор Babel, щоб перетворити TypeScript на JavaScript.

TypeScript підтримує файли, які можуть містити інформацію про тип для існуючих бібліотек JavaScript, так само як файли заголовків C++ можуть описувати структуру існуючих об’єктних файлів. Це дозволяє іншим програмам використовувати значення, визначені у файлі, як якщо б вони були статично введеними сутностями TypeScript. Такі популярні бібліотеки, як jQuery, MongoDB і D3.js, мають сторонні заголовки. Заголовки TypeScript також доступні для

модулів бібліотеки Node.js, що дозволяє розробляти програми Node.js у TypeScript. [7]

Сам компілятор TypeScript написаний на TypeScript і компілюється в JavaScript. Він ліцензований за ліцензією Apache 2.0. Андерс Хейлсберг — головний архітектор C#, творець Delphi та Turbo Pascal, працює над TypeScript. [8] [9][10][11]

Після двох років внутрішньої розробки в корпорації Майкрософт TypeScript був випущений для громадськості в жовтні 2012 року як версія 0.8. [12] [13] Невдовзі після першого публічного випуску Мігель де Іказа похвалив саму мову, але розкритикував відсутність у мові підтримки повноцінних IDE, крім Microsoft Visual Studio, яка була недоступна для Linux деякий час. [14][15] Станом на квітень 2021 року підтримку пропонують інші IDE та текстові редактори, зокрема Emacs, Vim, WebStorm, Atom[16] і Visual Studio Code від Microsoft. [17] TypeScript 0.9, випущений у 2013 році, додав підтримку дженеріків. [18]

TypeScript 1.0 було анонсовано на конференції розробників Microsoft Build у 2014 році. [19] Visual Studio 2013 Update 2 забезпечує вбудовану підтримку TypeScript. [20] У липні 2014 року команда розробників оголосила про подальші вдосконалення за допомогою нового компілятора TypeScript, який, як кажуть, у п’ять разів швидший. У той же час вихідний код, який спочатку розміщувався на CodePlex, було перенесено на GitHub.

22 вересня 2016 року був випущений TypeScript 2.0, який представив кілька функцій, включаючи можливість для програмістів додатково застосовувати нульову безпеку [22], яку іноді називають помилкою в мільярд доларів.

TypeScript 3.0 було випущено 30 липня 2018 року [23], приносячи багато нових функцій мови, таких як параметри залишків і кортежі у розширених виразах, параметри залишків типів кортежів, загальні параметри залишків тощо [24]

TypeScript 4.0 було випущено 20 серпня 2020 року. [25] Хоча у версії 4.0 не було внесено жодних критичних змін, вона додала мовні функції, такі як власні фабрики JSX і змінні типи кортежів. [25]

TypeScript виник через недоліки використання JavaScript для великомасштабної розробки додатків у Microsoft та її клієнтів. [26] Труднощі в роботі зі складним кодом JavaScript призводять до необхідності спеціальних інструментів для полегшення розробки компонентів мови. [27] Розробники TypeScript шукали рішення, яке б не порушувало сумісності зі стандартом і його кросплатформену підтримку. Знаючи, що поточна стандартна пропозиція ECMAScript обіцяє майбутню підтримку програмування на основі класів, TypeScript базується на цій пропозиції. Це призводить до появи компілятора JavaScript із набором синтаксичних мовних розширень, надмножини на основі пропозицій, яка перекладає TypeScript у звичайний JavaScript. У цьому сенсі функція класу TypeScript є очікуваною попередньою версією ECMAScript 2015. Унікальним аспектом, який не включено в пропозицію, але додано до TypeScript, є додаткова статична типізація (також відома як прогресивна типізація), яка дає змогу статичного аналізу мови для полегшення інструментів і підтримки IDE.

2.3 Фреймворк Nest.js

Nest (NestJS) — це платформа для створення ефективних, масштабованих серверних програм Node.js. Він використовує прогресивний JavaScript, будується на TypeScript і повністю підтримує його (досі дозволяє розробникам розробляти на простому JavaScript), а також поєднує елементи OOP (об’єктно-орієнтоване програмування), FP (функціональне програмування) і FRP (функціональне реактивне програмування).

Під капотом Nest використовує надійну структуру HTTP-сервера, наприклад Express (за замовчуванням), яку також можна налаштувати для використання Fastify за бажанням.

Nest забезпечує вищий рівень абстракції, ніж ці звичайні фреймворки Node.js (Express/Fastify), а також надає їхній API безпосередньо розробникам. Це звільняє розробників від використання безлічі сторонніх модулів, доступних для основної платформи.

В останні роки завдяки Node.js JavaScript став дуже популярним серед додатків у Інтернеті. Це призвело до створення таких чудових проектів, як Angular, React і Vue, які підвищують продуктивність розробників і дозволяють створювати швидкі зовнішні програми, які можна тестувати та масштабувати. Однак, незважаючи на те, що Node (і серверний JavaScript) мають багато чудових бібліотек, утиліт та інструментів, жоден із них не дозволяє ефективно вирішити головну проблему: архітектуру.

Nest пропонує готову архітектуру сервісів і модулів, яка дозволяє розробникам і командам створювати високоякісні, масштабовані, слабко пов'язані та прості в обслуговуванні програми. Архітектура значною мірою натхненна Angular.

2.4 Огляд бази даних

MongoDB — це кросплатформна документоорієнтована програма бази даних з відкритим кодом. MongoDB класифікується як програма бази даних NoSQL, яка використовує JSON-подібні документи та додаткові схеми. MongoDB розроблено MongoDB Inc. і ліцензується згідно з загальнодоступною ліцензією (SSPL).

Спеціальні запити

MongoDB підтримує пошук за полями, діапазонами та регулярними виразами. [30] Запити можуть повертати певні поля документа або вони можуть містити визначені користувачем функції JavaScript. Запити також можна налаштувати для повернення випадкової вибірки результатів заданого розміру.

Індексація

Поля в документах MongoDB можна індексувати за допомогою первинних і вторинних індексів.

Тиражування

MongoDB забезпечує високу доступність через набори реплік. [31] Набір реплік складається з двох або більше копій даних. Кожен член набору реплік може

діяти як основна або вторинна репліка в будь-який час. За замовчуванням усі записи та читання виконуються на первинній репліці. Вторинна репліка використовує вбудовану реплікацію для збереження копії первинних даних. Коли основна репліка виходить з ладу, набір реплік автоматично виконує процес вибору, щоб визначити, яка вторинна репліка повинна стати основною. Вторинні сервери можуть обслуговувати операції читання, але ці дані узгоджені лише за умовчанням.

Якщо репліковане розгортання MongoDB має лише одного вторинного члена, до набору необхідно додати окремий демон під назвою арбітр. Він одноосібно відповідає за вирішення виборчих питань нових первинних виборів. [32] Таким чином, ідеальне розподілене розгортання MongoDB вимагає принаймні трьох незалежних серверів, навіть якщо є лише один основний і один вторинний. [32]

Балансування навантаження

MongoDB масштабується горизонтально за допомогою сегментування. [33] Користувач вибирає шард-ключ, щоб визначити, як будуть розподілятися дані в колекції. Дані поділяються на діапазони (на основі ключів сегментів) і розподіляються між кількома сегментами. (Шард — це головний вузол з однією або декількома репліками.) Крім того, ключ сегмента можна хешувати, щоб відповідати сегменту, забезпечуючи рівномірний розподіл даних.

MongoDB може працювати на кількох серверах із балансуванням навантаження або реплікацією даних, щоб підтримувати роботу системи у разі апаратного збою.

Зберігання файлів

MongoDB можна використовувати як файлову систему під назвою GridFS із балансуванням навантаження та реплікацією даних на кількох машинах для зберігання файлів.

Ця функція називається Grid File System [34] і включена в драйвер MongoDB. MongoDB надає розробникам функціональність для роботи з файлами та вмістом. Доступ до GridFS можна отримати за допомогою утиліти mongofiles

або плагінів для Nginx[35] і lighttpd.[36]. GridFS ділить файли на частини та зберігає кожну частину як окремий документ. [37]

Агрегація

MongoDB надає три способи виконання агрегацій: конвеєри агрегації, функції зменшення карти та одноцільові методи агрегації. [38]

Map-reduce можна використовувати для пакетної обробки даних і операцій агрегації. Але згідно з документацією MongoDB, конвеєр агрегації забезпечує кращу продуктивність для більшості операцій агрегації. [39]

Агреговані структури дозволяють користувачам отримувати результати за допомогою пропозиції SQL GROUP BY. Агреговані оператори можуть бути об'єднані разом, щоб утворити канали – подібно до каналів Unix. Структура агрегації включає оператор $lookup, який може поєднувати документи з кількох колекцій, і статистичні оператори, такі як стандартне відхилення.

Виконання JavaScript на стороні сервера

JavaScript можна використовувати для запитів, агрегатних функцій (наприклад, MapReduce) і надсилати безпосередньо до бази даних для виконання.

Колекції

MongoDB підтримує колекції фіксованого розміру, відомі як обмежені колекції. Цей тип колекції зберігає порядок вставки та поводиться як циклічна черга, коли вона досягає заданого розміру.

Транзакції

MongoDB стверджувала, що підтримує багатодокументні транзакції ACID з моменту випуску версії 4.0 у червні 2018 року. [40] Це твердження було визнано помилковим.

Повідомлення про помилки та критика

Безпека

Дані з десятків тисяч установок MongoDB були викрадені через стандартну конфігурацію безпеки MongoDB, яка дозволяє будь-кому мати повний доступ до бази даних. Крім того, багато серверів MongoDB затримали з метою отримання викупу. [54][55]

Вересень 2017 р. Оновлено січень 2018 р. В офіційній відповіді Дейві Оттенхаймер, керівник відділу безпеки продуктів MongoDB, оголосив, що MongoDB вжила заходів для захисту від цих ризиків. [56]

Починаючи з випуску MongoDB 2.6, двійкові файли в офіційних пакетах MongoDB RPM і DEB за замовчуванням прив'язані до localhost. Починаючи з MongoDB 3.6, цю поведінку за замовчуванням було поширено на всі пакети MongoDB на всіх платформах. Таким чином, усі мережеві з'єднання з базою даних будуть заборонені, якщо адміністратор явно не налаштував це. [57]

Технічна критика

У певних ситуаціях збоїв, коли програма має доступ до двох окремих процесів MongoDB, але ці процеси не можуть отримати доступ один до одного, MongoDB може повертати застарілі дані. У цьому випадку MongoDB також може відкотити записані дані. [58] Цю проблему було вирішено з версії 3.4.0[59], випущеної в листопаді 2016 року. [60]

До версії 2.2 блокування було реалізовано на основі кожного сервера для кожного процесу. У версії 2.2 блокування реалізовано на рівні бази даних. [61] Починаючи з версії 3.0, [62] вводяться підключення механізмів зберігання, кожен з яких може реалізувати різні блокування. [62] У MongoDB 3.0 блокування реалізовано на рівні колекції механізму зберігання MMAPv1 [63], тоді як механізм зберігання даних WiredTiger використовує протокол паралелізму, який ефективно забезпечує блокування на рівні документа. [64] Навіть у версіях до 3.0 одним із способів підвищення паралельності було використання сегментування. [65] У деяких випадках читання та запис можуть спричинити блокування. Якщо MongoDB передбачає, що сторінка навряд чи буде в пам'яті, операція завершить блокування під час завантаження сторінки. У 2.2 використання блокування значно розширилося. [66]

До версії 3.3.11 MongoDB не міг виконувати сортування та обмежувався побайтовим порівнянням через memcmp, який не забезпечував правильного сортування для багатьох неанглійських мов при використанні кодувань Unicode. Питання вирішено 23 серпня 2016 року.

До MongoDB 4.0 індексні запити не були атомарними. Можливо, документи, оновлені під час запиту, були пропущені. [67] Введення проблеми читання знімка в MongoDB 4.0 усуває це явище. [68]

Незважаючи на те, що MongoDB стверджує в недатованій статті під назвою «MongoDB і Jepsen» [69], що їх база даних пройшла дослідницький тест розподіленої безпеки Jepsen, який він називає «найсуворішими тестами на безпеку, правильність і узгодженість даних у галузі», Jepsen опублікував статтю в Травень 2020 року, де йдеться про те, що MongoDB 3.6.4 фактично не пройшов тест, а новіша MongoDB 4.2.6 має більше проблем, зокрема ", де транзакція змінює порядок, щоб користувачі читання могли бачити результати майбутніх записів. [70][71] Джепсен зазначив у своєму звіті, що MongoDB не згадує ці висновки на сторінці MongoDB «MongoDB і Jepsen».

3 ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Проектування компонентного складу

Для ілюстрації структури програмного забезпечення слід використати діаграму компонентів (рис.3.1).

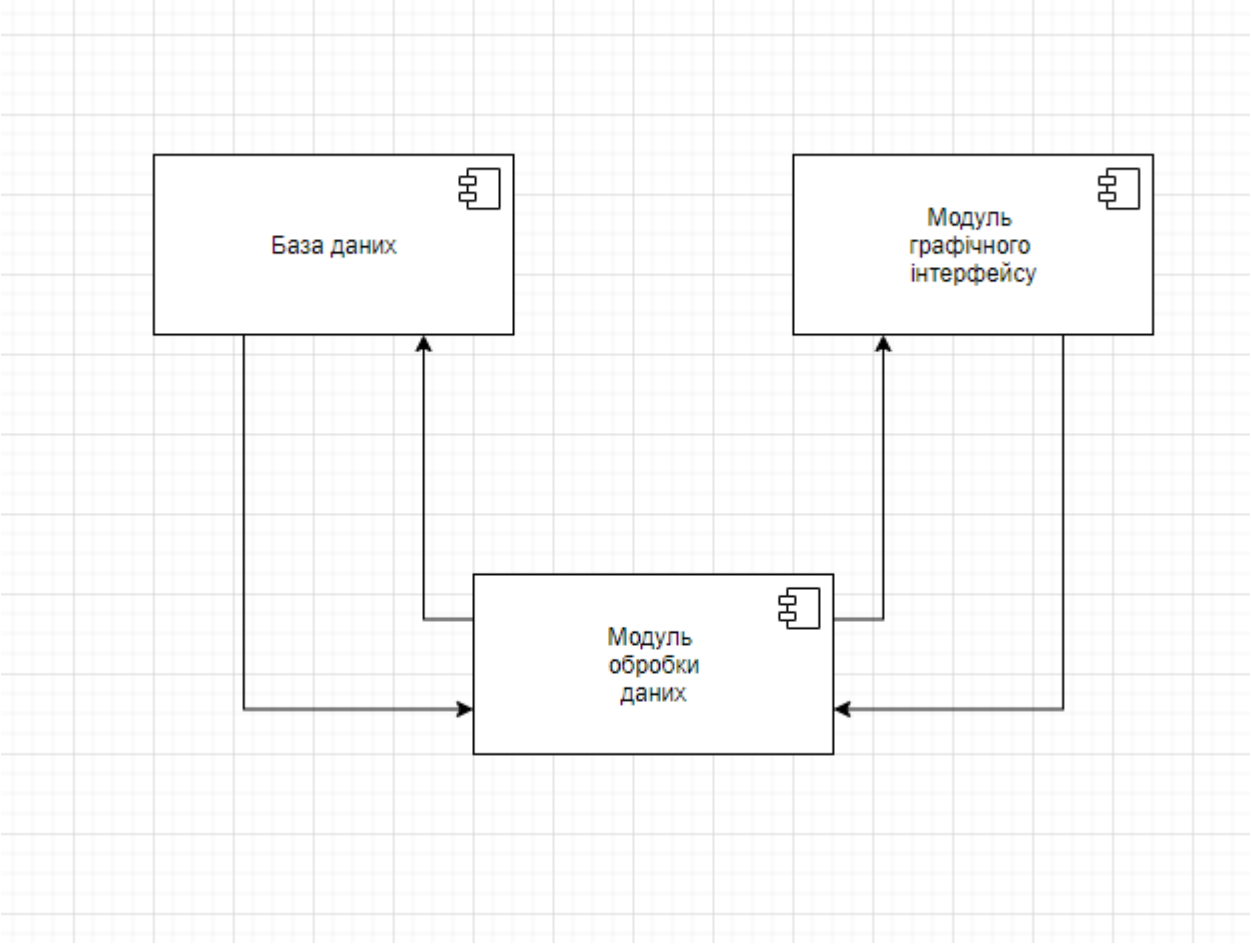

Рисунок 3.1 — Діаграма компонентів системи

3.2 Проектування розгортання інформаційної системи

Для опису технічного забезпечення доцільно використати діаграму розгортання, яка зображена на рис.3.2.

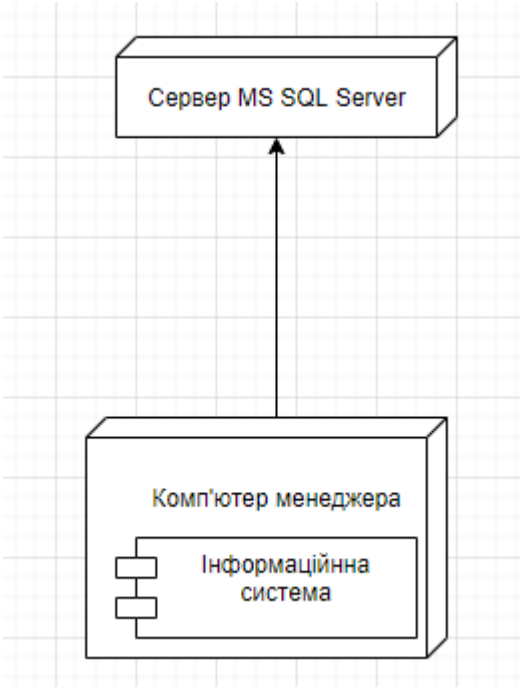

Рисунок 3.2 — Діаграма розгортання

3.3 Проектування внутрішньої будови інформаційної системи

У розробці програмного забезпечення діаграма класів на уніфікованій мові моделювання (UML) — це статична структурна діаграма, яка описує структуру системи, показуючи її класи, їхні атрибути, операції (або методи) і зв’язки між об’єктами.

Діаграми класів є основними будівельними блоками об’єктно-орієнтованого моделювання. Використовуються для загального концептуального моделювання структури програми та для детального перетворення моделі в програмний код. Діаграми класів також можна використовувати для моделювання даних. Класи на діаграмі класів представляють основні елементи, інтерактивні та програмовані класи в програмі.

На схемі класи представлені вікнами, які містять три відділення:

- У верхньому відділенні міститься назва класу. Надруковано жирним шрифтом і відцентровано, а перша літера написана великими літерами.
- Середній відсік містить атрибути класу. Вони вирівняні за лівим краєм, а перша буква мала.
- У нижньому відділенні містяться операції, які може виконувати клас. Вони також вирівняні за лівим краєм, а перша буква - мала.

Під час проектування системи багато класів ідентифікуються та групуються в діаграму класів, щоб допомогти визначити статичні зв’язки між ними. У детальному моделюванні класи концептуального проектування часто діляться на підкласи.

Залежності - це семантичні відносини між залежними та незалежними елементами моделі. Він існує між двома елементами, якщо зміна, визначена одним елементом (сервером або метою), викликає зміну в іншому елементі (клієнті або джерелі). Ця асоціація є односпрямованою. Залежність показана пунктирною лінією з порожнистою стрілкою, що вказує від клієнта до постачальника.

Для подальшого опису поведінки системи ці діаграми класів можуть бути доповнені діаграмами стану.

Асоціація представляє сімейство посилань. Бінарні асоціації (обидва кінці) зазвичай представлені у вигляді рядків. Асоціації можуть пов'язувати будь-яку кількість класів. Асоціація з трьома зв'язками називається потрійною асоціацією. Асоціації можна називати, а кінці асоціацій можна прикрашати іменами ролей, індикаторами власності, множинністю, видимістю та іншими властивостями.

Існує чотири різні типи асоціацій: двонаправлені, однонаправлені, агрегатні (за участю складених агрегатів) і рефлексивні. Найбільш поширеними є двонаправлені та однонаправлені асоціації.

Наприклад, клас польоту двонаправлено пов’язаний з класом літака. Асоціація представляє статичний зв’язок, спільний між об’єктами двох класів.

Агрегація є різновидом зв’язку «має»; агрегація є більш конкретною, ніж асоціація. Це асоціація, яка представляє частину цілого або частину зв’язку. Як показано, у професора є клас для викладання. Як тип асоціації, агрегати можуть

мати назви та ті самі атрибути, що й асоціації. Однак агрегація не може містити більше двох класів; вона має бути двійковою асоціацією. Крім того, навряд чи існує будь-яка різниця між агрегацією та асоціацією під час впровадження, а зв’язки агрегації можуть бути повністю виключені з діаграми. [7]

Агрегація може відбуватися, коли клас є колекцією або контейнером інших класів, але класи, що містяться, не мають сильної залежності від контейнера. Коли контейнер знищено, вміст контейнера все ще існує.

В UML він представлений графічно як розкритий ромб на класі. Агрегат — це семантично розширений об’єкт, який розглядається як одиниця в багатьох операціях, навіть якщо він фізично складається з кількох менших об’єктів.

Приклад: бібліотека та студенти. Тут студенти можуть існувати без бібліотеки, а зв’язок між студентами та бібліотекою агрегується.

Це означає, що один із двох споріднених класів (підкласів) вважається спеціалізацією іншого (супертипу), а суперклас є узагальненням підкласу. На практиці це означає, що будь-який екземпляр підтипу також є екземпляром суперкласу. Типове дерево узагальнень такої форми зустрічається в систематиці: люди є підкласом людиноподібних мавп, людиноподібні мавпи є підкласом ссавців тощо. Цей зв’язок найпростіше зрозуміти за допомогою фрази «А є Б» (людина — ссавець, ссавець — тварина).

Узагальнене графічне представлення UML — це відкритий трикутник у кінці суперкласу String (або дерева String), що зв’язує його з одним або кількома підтипами.

Відносини узагальнення також відомі як успадкування або відношення.

Суперклас (базовий клас) у зв’язку узагальнення також відомий як «батьківський клас», суперклас, базовий клас або базовий тип.

Підтип у зв’язку спеціалізації також називають «дочірнім», підкласом, похідним класом, похідним типом, успадкованим класом або успадкованим типом.

Зауважте, що цей зв’язок не те саме, що біологічне походження: ці терміни використовуються дуже часто, але можуть викликати плутанину.

А - це тип В

Наприклад, «дуб — порода дерева», «автомобіль — вид транспорту»

Узагальнення можна показати лише в діаграмах класів і діаграмах використання.

У UML-моделюванні відношення реалізації - це відношення між двома елементами моделі, в яких один елемент моделі (клієнт) реалізує (впроваджує або виконує) поведінку, визначену іншим елементом моделі (постачальником).

Графічне представлення реалізації в UML — це відкритий трикутник на кінці пунктирного інтерфейсу (або дерева ліній), що з'єднує його з одним або декількома розробниками. Проста стрілка використовується в кінці пунктирного інтерфейсу, підключеного до користувача. У діаграмах компонентів використовується угода про схему «куля-розетка» (розробники показують кульку або льодяник, а користувачі — розетку). Реалізації можуть бути показані лише на діаграмах класів або компонентів. Реалізація — це зв'язок між класами, інтерфейсами, компонентами та пакетами, який з'єднує клієнтські елементи з елементами провайдера. Взаємозв'язок реалізації між класом/компонентом та інтерфейсом вказує на те, що клас/компонент реалізує операції, надані інтерфейсом.

Залежність — це слабша форма зв'язку, яка вказує на те, що клас залежить від іншого класу, оскільки він використовує його в певний момент часу. Клас залежить від іншого класу, якщо незалежний клас є змінною параметра або локальною змінною методу залежного класу. Це відрізняється від асоціації, де властивості залежного класу є екземплярами незалежного класу. Іноді зв'язок між двома класами слабкий. Вони взагалі не реалізовані зі змінними-членами. Замість цього вони можуть бути реалізовані як аргументи функцій-членів.

Цей зв'язок часто описують як «А має Б» (у мами-кішки є кошенята, а у кошенят — мама-кішка).

UML-представлення асоціації — це лінія, що з'єднує два пов'язані класи. На кожному кінці рядка є діакритичні знаки. Наприклад, ми можемо використовувати стрілку, щоб вказати, що вістря видно з хвоста стріли. Ми можемо вказати властивість, помістивши кулю, вказавши роль, яку відіграє елемент на цьому кінці (з іншого кінця), вказавши назву ролі та множину екземплярів цієї сутності.

Класи сутностей моделюють довготривалу інформацію, яку обробляє система, а іноді й поведінку, пов’язану з цією інформацією. Їх не слід ідентифікувати як таблиці бази даних або інші сховища даних.

Вони намальовані як кола з короткою лінією, прикріпленою до нижньої частини кола. Крім того, вони можуть бути намальовані як звичайні класи, зі стереотипом "Entity" над назвою класу.

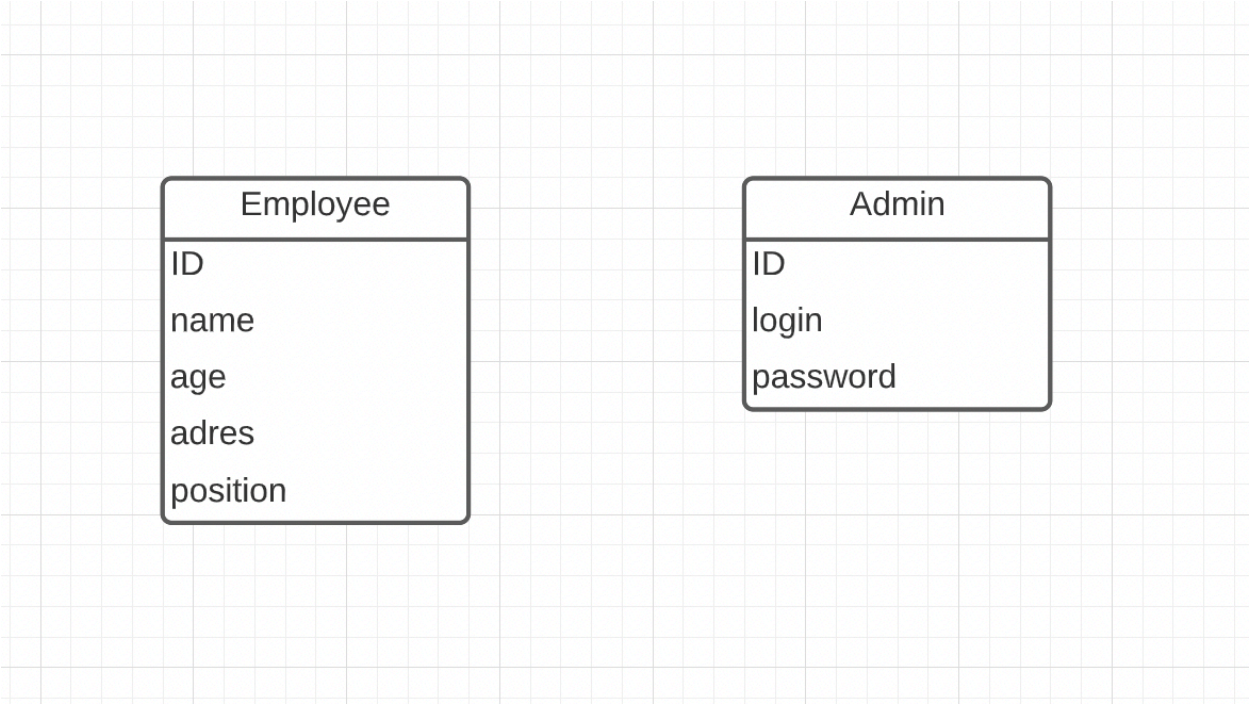

Рисунок 3.3 — Діаграма класів

3.4 Тестування

Тестування програмного забезпечення — це акт перевірки помилок і поведінки програмного забезпечення, що тестується, шляхом перевірки. Тестування програмного забезпечення також може забезпечити об’єктивне, незалежне уявлення про програмне забезпечення, дозволяючи компаніям оцінити та зрозуміти ризики впровадження програмного забезпечення. Методи тестування включають, але не обов’язково обмежуються:

- аналіз вимог до продукту щодо повноти та правильності в різних контекстах, таких як галузева перспектива, бізнес-перспектива, доцільність

та життєздатність впровадження, зручність використання, продуктивність, безпека, міркування інфраструктури тощо.

- перегляд архітектури продукту та загального дизайну продукту
- робота з розробниками продуктів над удосконаленням техніки кодування, шаблонів проектування, тестів, які можна написати як частину коду на основі різних методів, таких як граничні умови тощо.
- виконання програми або програми з метою перевірки поведінки
- перевірка інфраструктури розгортання та пов'язаних скриптів та автоматизації
- брати участь у виробничій діяльності, використовуючи методи моніторингу та спостереження

Тестування програмного забезпечення може надати користувачам або власникам об'єктивну, незалежну інформацію про якість програмного забезпечення та ризик його несправності.

Помилки

Помилки програмного забезпечення виникають через такий процес: Програмісти роблять помилки у коді, які викликають помилки у програмі (дефекти, баги) у вихідному коді програмного забезпечення. Якщо виникне ця помилка, у деяких випадках система видаватиме неправильні результати, що призведе до збою.

Не всі невдачі обов'язково призводять до невдач. Наприклад, помилки в мертвому коді ніколи не спричинять збій. Невиявлені збої можуть спричинити збої під час зміни середовища. Приклади таких змін середовища включають програмне забезпечення, що працює на нових комп'ютерних апаратних платформах, зміни у вихідних даних або взаємодію з іншим програмним забезпеченням. Один збій може спричинити широкий спектр симптомів збою.

Не всі помилки програмного забезпечення викликані помилками кодування. Загальним джерелом дорогих дефектів є прогалини у вимогах, невідомі вимоги, через які розробники додатків пропускають помилки. Прогалини у вимогах часто можуть бути нефункціональними вимогами, такими як тестування, масштабованість, ремонтопридатність, продуктивність та інші. Безпека

Статичні, динамічні та пасивні випробування

Існує багато підходів до тестування програмного забезпечення. Покрокові керівництва, покрокові керівництва або перевірки відомі як статичне тестування, тоді як виконання програмного коду з використанням заданого набору тестових випадків відоме як динамічне тестування.

Статичне тестування часто неявне, наприклад коректура, як статичний аналіз програм, коли інструменти програмування/текстові редактори перевіряють структуру вихідного коду, а компілятори (прекомпілятори) перевіряють синтаксис і потік даних. Динамічне тестування відбувається під час запуску самої програми. Динамічне тестування може початися до того, як програма буде завершена на 100%, щоб перевірити різні частини коду та застосувати до окремих функцій або модулів. Типовими підходами для цього є використання заглушок/драйверів або запуск із середовища налагодження.

Статичне і динамічне тестування передбачає перевірку.

Пасивне тестування означає перевірку поведінки системи без будь-якої взаємодії з програмним продуктом. На відміну від активного тестування, тестувальники не надають жодних тестових даних, а натомість переглядають системні журнали та трасування. Вони шукають моделі та конкретну поведінку, щоб приймати рішення. Це пов’язано з офлайн-тестуванням і аналізом журналу.

Методи дослідження

Дослідницьке тестування — це методологія тестування програмного забезпечення, яка коротко описується як одночасне навчання, розробка тесту та виконання тесту. Джем Канер, який ввів цей термін у 1984 році, визначив дослідницьке тестування як «стиль тестування програмного забезпечення, який підкреслює особисту свободу та відповідальність окремого тестувальника, який постійно вдосконалює своє тестування, беручи до уваги пов’язане з тестуванням навчання, дизайн тесту та Якість роботи, виконання тестів та інтерпретація результатів тестів як взаємодопоміжні операції, що виконуються паралельно протягом усього проекту».

Метод «ящик».

Методи тестування програмного забезпечення традиційно поділяються на тестування білого ящика та тестування чорного ящика. Ці два підходи використовуються для опису точки зору тестувальника під час розробки тестового прикладу. Гібридний підхід, відомий як тестування сірого ящика, також можна застосувати до методів тестування програмного забезпечення. Це «довільне розрізнення» між тестуванням у чорній та білій скриньках дещо зникло, оскільки концепція «сірого ящика» побудови тестів на основі конкретних елементів дизайну набула популярності.

Таблиця 3.1 — Тестування додатку

1	Невірні данні при авторизації	При введенні невірних даних система повідомляє користувача про те, що такого користувача не існує, або дані введені невірно
2	Пусті поля при авторизації	При спробі авторизації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
3	Пусті поля при зміні даних працівника	При спробі змінити дані працівника з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.

4	Створення працівника з пустими полями	При спробі створення працівника з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.

Результати тестування показують, що система повністю функціональна і готова до використання в реальних умовах.

3.5 Презентація роботи інформаційної системи

Якщо ми зайдемо на головну сторінку нашої інформаційної системи нас зустрічає вікно авторизації (рис.3.4).

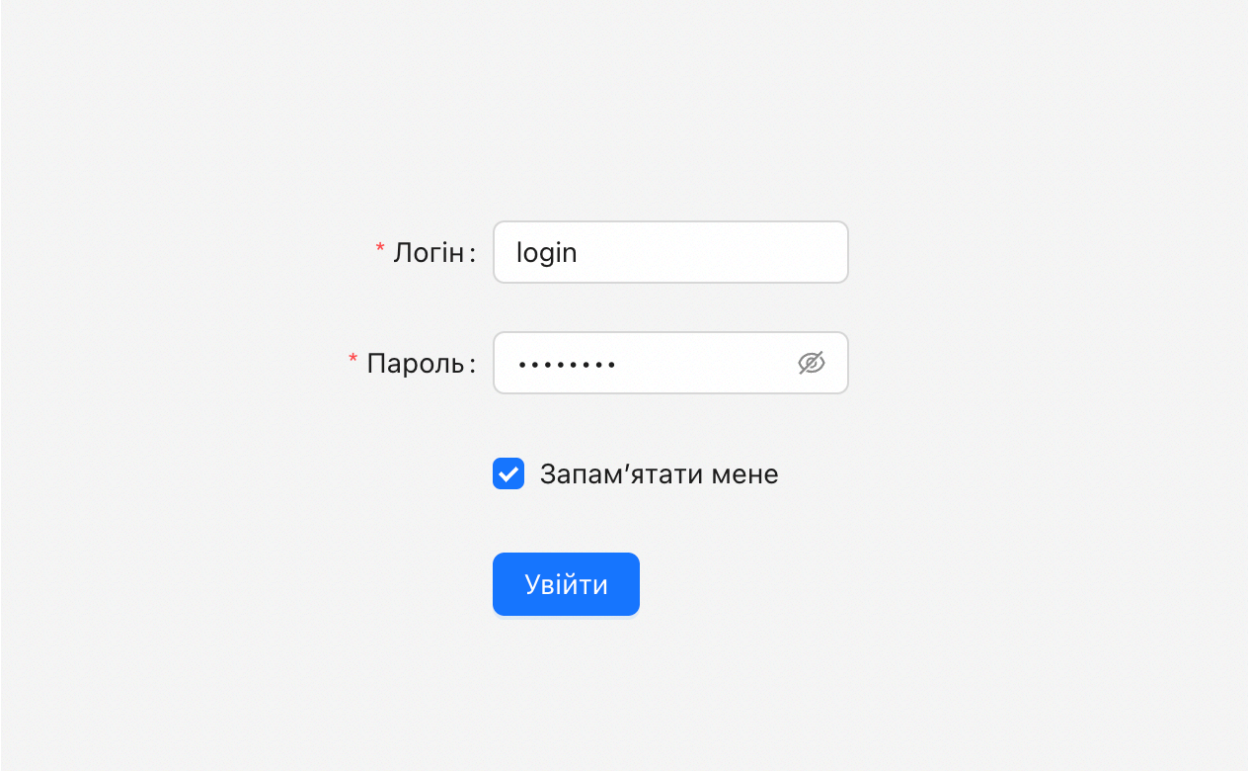

Рисунок 3.4 — Вікно авторизації

У цьому вікні ми маємо ввести дані для авторизації, щоб отримати доступ до інформаційної системи. Якщо ж ми спробуємо увійти з порожніми полями то це буде помилкою і система нас про це повідомить (рис.3.5).

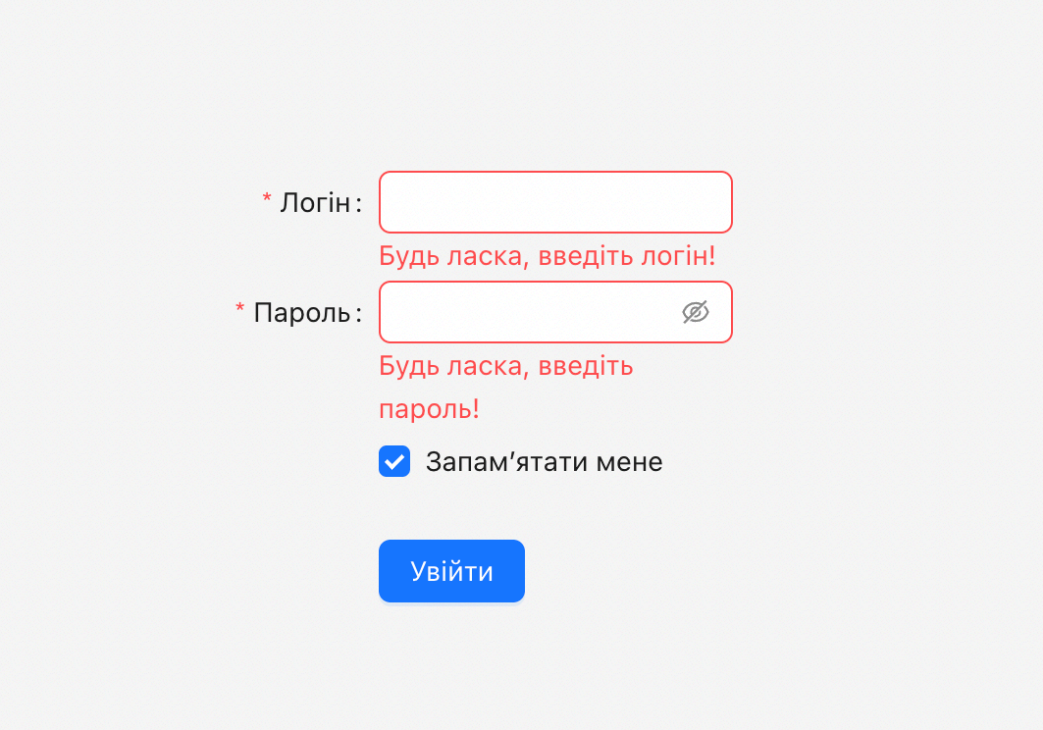

Рисунок 3.5 — Помилка авторизації

Після авторизації ми отримуємо доступ до нашої інформаційної системи. Тут ми бачимо список усіх працівників, що були внесені до системи (рис.3.6).

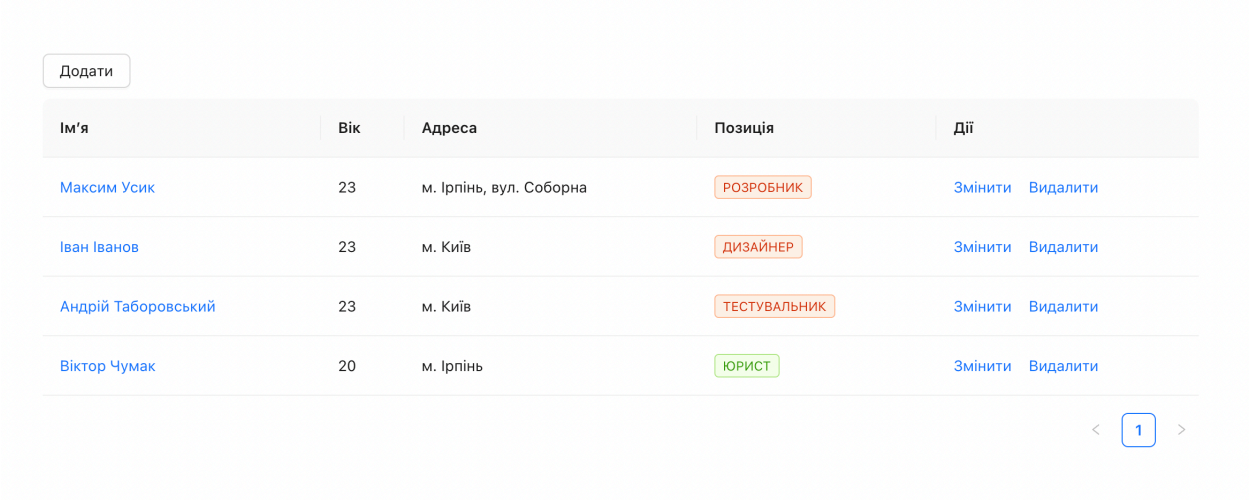

Рисунок 3.6 — Зовнішній вигляд системи

Це дані, які зберігаються у нашій базі даних. Ми можемо додавати нові записи, натиснувши на кнопку «Додати». У цьому випадку нам відкриється нове вікно, де ми маємо заповнити форму з особистими даними нової людини (рис.3.7). І потім натиснути «Створити». У такому випадку у нас буде створено новий запис у базі даних, який водображається у таблиці.

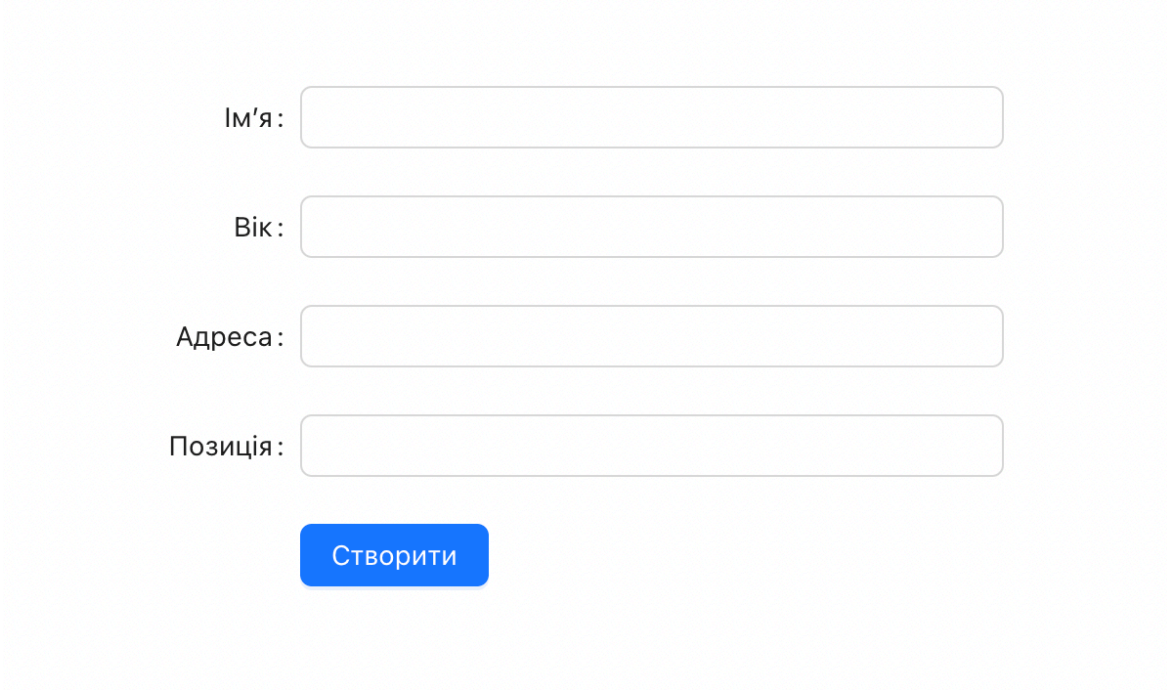

Рисунок 3.6 — Форма для додавання нового запису

Заповнимо форму особистими даними нової людини (рис.3.7).

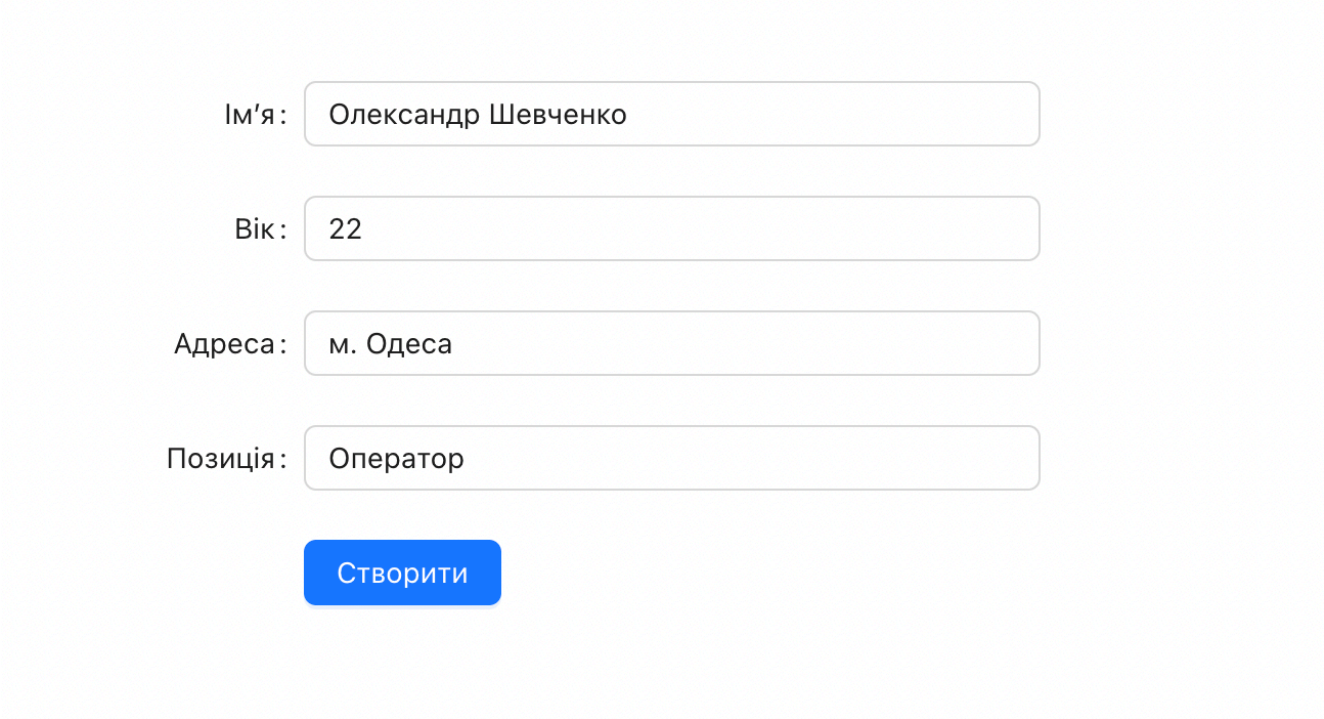

Рисунок 3.7 — Заповнена форма

Якщо ми натиснемо кнопку «Створити» то наш запис з'явиться у базі даних (рис.3.8).

Рисунок 3.8 — Створений новий працівник

ВИСНОВКИ

В процесі виконання роботи було виконано наступні завдання:

- Було досліджено поняття менеджменту. У якості підхода до розробки було обрано методологію «водоспад». Він створює графік проекту, в якому 20-40% часу витрачається на перші дві фази, 30-40% витрачається на розробку, а решта витрачається на тестування та впровадження. Реальна організаційна структура проекту має бути чіткою.
- Дослідили поняття веб-розробки. Веб-додаток це програма яка працює на веб-сервері, на відміну від комп'ютерних програм, що запускаються локально в операційній системі пристрою. Веб-додатки використовують архітектуру «клієнт-сервер», де «клієнту» (користувачу) надається доступ до додатку через стороній веб-сервер. Користувач отримує доступ до веб-додатку через веб-брузер, використовуючи мережу Інтернет.
- Дослідили поняття інформаційної системи. Це формальна соціально-технічна організаційна система, призначена для збору, обробки, зберігання та розповсюдження інформації. Сучасні інформаційні системи дозволяють значно оптимізувати процеси у підприємствах.
- Вибрали мову програмування JavaScript для розробки клієнтської частини. Цей вибір був зроблений тому що це найпопулярніша мова, програмування у сфері веб-розробки. Вона має багато фреймворків, бібліотек, інструментів для розробки веб-додатків. Для серверної частини було обрано TypeScript і фреймворк NestJS.
- Вибрали базу даних MongoDB і впровадили її у інформаційну систему. Ця база даних була обрана через низку переваг: зрозуміла структура об'єкту, відсутність складних з'єднань join, розширені можливості запитів, легкість масштабування, реплікація і висока доступність, швидкі оновлення.
- Спроектували архітектуру системи, яка надає можливість розподіляти завдання або робочі навантаження між постачальниками ресурсів або послуг (серверами) і запитувачами послуг (клієнтами).

- Розробили інформаційну систему. Отримали робочий продукт, у вигляді системи з авторизацією, таблицею яка містить дані з бази даних, і дозволяє їх редагувати. Клієнтська частина використовує HTML, CSS, JavaScript, серверна - TypeScript.
- Провели тестування системи. За підсумками тестування системи є можливість авторизації, створення, зміни та видалення працівників, у системі. Відзначили її готовність до подальшого використання.

Завдяки чіткому виконанню поставлених на початку роботи завдань в результаті роботи було отримано повноцінну систему панелі адміністрування працівників, готову до використання в реальних умовах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Piccoli, Gabriele; Pigni, Federico (July 2018). Information systems for managers: with cases (4.0 ed.). Prospect Press. p. 28. ISBN 978-1-943153-50-3. Retrieved 25 November 2018.
2. O'Hara, Margaret; Watson, Richard; Cavan, Bruce (1999). "Managing the three levels of change". Information Systems Management. 16 (3): 64. doi:10.1201/1078/43197.16.3.19990601/31317.9. Retrieved 25 November 2018.
3. "Information Systems". 2020-11-12.
4. "information system". BusinessDictionary.com.
5. "Information Systems". Principia Cybernetica Web.
6. Vladimir Zwass (2016-02-10). "Information system". Britannica.
7. D'Atri A., De Marco M., Casalino N. (2008). "Interdisciplinary Aspects of Information Systems Studies", Physica-Verlag, Springer, Germany, pp. 1–416, doi:10.1007/978-3-7908-2010-2 ISBN 978-3-7908-2009-6
8. "Information Technology vs Information Systems: What's The Difference?". CityU of Seattle. 2020-01-16. Retrieved 2021-11-13.
9. Jessup, Leonard M.; Joseph S. Valacich (2008). Information Systems Today (3rd ed.). Pearson Publishing. Glossary p. 416
10. "What is Information Systems or Information Services (IS)?". Definition from Techopedia. Retrieved 6 March 2021.
11. "What is IS (information system or information services)?". WhatIs.com. Retrieved 6 March 2021.
12. "Information Services". Directory. Australian Government. 2 June 2017. Retrieved 6 March 2021.
13. "Information Services". Ramsey County. 12 September 2015. Retrieved 6 March 2021.
14. Bulgacs, Simon (2013). "The first phase of creating a standardised international innovative technological implementation framework/Software application".

International Journal of Business and Systems Research. 7 (3): 250. doi:10.1504/IJBSR.2013.055312. Retrieved 2015-11-02.

15. "SEI Report, "Glossary"". Archived from the original on September 3, 2007. Retrieved 2013-04-02.
16. Kroenke, D M. (2008). Experiencing MIS. Prentice-Hall, Upper Saddle River, NJ
17. O'Brien, J A. (2003). Introduction to information systems: essentials for the e-business enterprise. McGraw-Hill, Boston, MA
18. Alter, S. (2003) "18 Reasons Why IT-Reliant Work Systems Should Replace 'The IT Artifact' as the Core Subject Matter of the IS Field," Communications of the Association for Information Systems, 12(23), Oct., pp. 365–394, http://aisel.aisnet.org/cais/vol12/iss1/23/
19. Alter, S (2013). "Work System Theory: Overview of Core Concepts, Extensions, and Challenges for the Future". Journal of the Association for Information Systems. 14 (2): 72–121. doi:10.17705/1jais.00323.
20. Alter, S. (2006) The Work System Method: Connecting People, Processes, and IT for Business Results. Works System Press, CA
21. Bacon, C. James; Fitzgerald, Brian (2001-04-01). "A systemic framework for the field of information systems". ACM SIGMIS Database: The DATABASE for Advances in Information Systems. 32 (2): 46–67. doi:10.1145/506732.506738. ISSN 0095-0033. S2CID 15687595.
22. Beynon-Davies P. (2009). Business Information Systems. Palgrave, Basingstoke
23. Marc S. Silver, M. Lynne Markus, Cynthia Mathis Beath (Sep 1995). "The Information Technology Interactive Model: A Foundation for the MBA Core Course". MIS Quarterly: 361–390.
24. The Joint Task Force for Computing Curricula 2005. Computing Curricula 2005: The Overview Report (pdf) Archived 2014-10-21 at the Wayback Machine
25. Rockart et al. (1996) Eight imperatives for the new IT organization Sloan Management review.
26. Stair, Ralph (2020). Principles of Information Systems. George Reynolds (14th ed.). Mason, OH: Cengage. ISBN 978-0-357-11252-6. OCLC 1305839544.
27. Kroenke, D. M. (2015). MIS Essentials. Pearson Education

28. Laudon, K.C. and Laudon, J.P. Management Information Systems, Macmillan, 1988.
29. Rainer, R. Kelly Jr, and Casey G. Cegielski. Introduction to Information System: Support and Transforming Business Fourth Edition. New Jersey: John Wiley and Sons, Inc., 2012. Print.
30. Neumann, Gustaf; Sobernig, Stefan; Aram, Michael (February 2014). "Evolutionary Business Information Systems". Business and Information Systems Engineering. 6 (1): 33–36. doi:10.1007/s12599-013-0305-1. S2CID 15979292.
31. Aram, Michael; Neumann, Gustaf (2015-07-01). "Multilayered analysis of co-development of business information systems" (PDF). Journal of Internet Services and Applications. 6 (1). doi:10.1186/s13174-015-0030-8. S2CID 16502371.
32. Using MIS. Kroenke. 2009. ISBN 978-0-13-713029-0.
33. Börje Langefors (1973). Theoretical Analysis of Information Systems. Auerbach. ISBN 978-0-87769-151-8.
34. Computer Studies. Frederick Nyawaya. 2008. ISBN 978-9966-781-24-6.
35. "Computer and Logic Essentials – Units of study – Swinburne University of Technology – Melbourne, Australia".
36. "Building IT Systems – RMIT University".
37. "Systems Development – Units of study – Swinburne University of Technology – Melbourne, Australia".
38. Kelly, Sue; Gibson, Nicola; Holland, Christopher; Light, Ben (July 1999). "Focus Issue on Legacy Information Systems and Business Process Engineering: a Business Perspective of Legacy Information Systems". Communications of the AIS. 2 (7): 1–27.
39. Archibald, J.A. (May 1975). "Computer Science education for majors of other disciplines". AFIPS Joint Computer Conferences: 903–906. Computer science spreads out over several related disciplines, and shares with these disciplines certain sub-disciplines that traditionally have been located exclusively in the more conventional disciplines

40. Denning, Peter (July 1999). "Computer Science: The Discipline". Encyclopaedia of Computer Science (2000 Edition). The Domain of Computer Science: Even though computer science addresses both human-made and natural information processes, the main effort in the discipline has been directed toward human-made processes, especially information processing systems and machines
41. Coy, Wolfgang (June 2004). "Between the disciplines". ACM SIGCSE Bulletin. 36 (2): 7–10. doi:10.1145/1024338.1024340. ISSN 0097-8418. S2CID 10389644. Computer science may be in the core of these processes. The actual question is not to ignore disciplinary boundaries with its methodological differences but to open the disciplines for collaborative work. We must learn to build bridges, not to start in the gap between disciplines
42. Hoganson, Ken (December 2001). "Alternative curriculum models for integrating computer science and information systems analysis, recommendations, pitfalls, opportunities, accreditations, and trends". Journal of Computing Sciences in Colleges. 17 (2): 313–325. ISSN 1937-4771. ... Information Systems grew out of the need to bridge the gap between business management and computer science ...
43. Davis, Timothy; Geist, Robert; Matzko, Sarah; Westall, James (March 2004). τ ́εχνη: A First Step. Technical Symposium on Computer Science Education. pp. 125–129. ISBN 978-1-58113-798-9. In 1999, Clemson University established a (graduate) degree program that bridges the arts and the sciences... All students in the program are required to complete graduate level work in both the arts and computer science
44. Hoganson, Ken (December 2001). "Alternative curriculum models for integrating computer science and information systems analysis, recommendations, pitfalls, opportunities, accreditations, and trends". Journal of Computing Sciences in Colleges. 17 (2): 313–325. ISSN 1937-4771. The field of information systems as a separate discipline is relatively new and is undergoing continuous change as technology evolves and the field matures
45. Khazanchi, Deepak; Bjorn Erik Munkvold (Summer 2000). "Is information system a science? an inquiry into the nature of the information systems

discipline". ACM SIGMIS Database. 31 (3): 24–42. doi:10.1145/381823.381834. ISSN 0095-0033. S2CID 52847480. From this we have concluded that IS is a science, i.e., a scientific discipline in contrast to purportedly non-scientific fields

46. Denning, Peter (June 2007). "Ubiquity a new interview with Peter Denning on the great principles of computing". 2007 (June): 1. People from other fields are saying they have discovered information processes in their deepest structures and that collaboration with computing is essential to them.
47. "Computer science is the study of information" New Jersey Institute of Technology, Gutenberg Information Technologies Archived September 15, 2008, at the Wayback Machine
48. "Computer science is the study of computation." Computer Science Department, College of Saint Benedict Archived 2007-02-03 at the Wayback Machine, Saint John's University
49. "Computer Science is the study of all aspects of computer systems, from the theoretical foundations to the very practical aspects of managing large software projects." Massey University Archived 2006-06-19 at the Wayback Machine
50. Pearson Custom Publishing & West Chester University, Custom Program for Computer Information Systems, Pearson Custom Publishing, (2009) Glossary p. 694
51. Polack, Jennifer (December 2009). "Planning a CIS Education Within a CS Framework". Journal of Computing Sciences in Colleges. 25 (2): 100–106. ISSN 1937-4771.
52. Hayes, Helen; Onkar Sharma (February 2003). "A decade of experience with a common first year program for computer science, information systems and information technology majors". Journal of Computing Sciences in Colleges. 18 (3): 217–227. ISSN 1937-4771. In 1988, a degree program in Computer Information Systems (CIS) was launched with the objective of providing an option for students who were less inclined to become programmers and were more interested in learning to design, develop, and implement Information Systems, and solve business problems using the systems approach

53. CSTA Committee, Allen Tucker, et alia, A Model Curriculum for K-12 Computer Science (Final Report), (Association for Computing Machinery, Inc., 2006) Abstraction & p. 2
54. Freeman, Peter; Hart, David (August 2004). "A Science of Design for Software-Intensive Systems Computer science and engineering needs an intellectually rigorous, analytical, teachable design process to ensure development of systems we all can live with". Communications of the ACM. 47 (8): 19–21. doi:10.1145/1012037.1012054. ISSN 0001-0782. S2CID 14331332. Though the other components' connections to the software and their role in the overall design of the system are critical, the core consideration for a software-intensive system is the software itself, and other approaches to systematizing design have yet to solve the "software problem"—which won't be solved until software design is understood scientifically
55. Culnan, M. J. Mapping the Intellectual Structure of MIS, 1980–1985: A Co-Citation Analysis, MIS Quarterly, 1987, pp. 341–353.
56. Keen, P. G. W. MIS Research: Reference Disciplines and A Cumulative Tradition, in Proceedings of the First International Conference on Information Systems, E. McLean (ed.), Philadelphia, PA, 1980, pp. 9–18.
57. Lee, A. S. Architecture as A Reference Discipline for MIS, in Information Systems Research: Contemporary Approaches and Emergent Traditions, H.-E. Nisen, H. K. Klein, and R. A. Hirschheim (eds.), North-Holland, Amsterdam, 1991, pp. 573–592.
58. Mingers, J., and Stowell, F. (eds.). Information Systems: An Emerging Discipline?, McGraw- Hill, London, 1997.
59. John, W., and Joe, P. (2002) "Strategic Planning for Information System." 3rd Ed. West Sussex. John Wiley & Sons Ltd
60. "Scoping the Discipline of Information Systems" (PDF).
61. Basden, A. (2010) On Using Spheres of Meaning to Define and Dignify the IS Discipline. International Journal of Information Management, 30, 13–20. It employs the philosophy of the late Herman Dooyeweerd to differentiate distinct aspects or 'spheres of meaning'. The paper suggests that while computer science

finds the formative aspect, of shaping, structuring, processing, of central interest, and business and organizational fields find the economic and social aspects of central interest, the Information Systems field can find the lingual aspect of central interest while making links with the aspects of the neighbouring disciplines.

62. International Journal of Information Management, 30, 13–20.
63. "Information Systems". Sloan Career Cornerstone Center; Alfred P. Sloan Foundation. 2008. Retrieved June 2, 2008.
64. Galliers, R.D., Markus, M.L., & Newell, S. (Eds) (2006). Exploring Information Systems Research Approaches. New York, NY: Routledge.
65. Ciborra, C. (2002). The Labyrinths of Information: Challenging the Wisdom of Systems. Oxford, UK: Oxford University Press
66. Hevner; March; Park; Ram (2004). "Design Science in Information Systems Research". MIS Quarterly. 28 (1): 75–105. doi:10.2307/25148625. JSTOR 25148625. S2CID 13553735.
67. March, S.; Smith, G. (1995). "Design and natural science in Information Technology (IT)". Decision Support Systems. 15 (4): 251–266. doi:10.1016/0167-9236(94)00041-2.
68. Avgerou, C (2000). "Information systems: what sort of science is it?". Omega. 28 (5): 567–579. CiteSeerX 10.1.1.203.4718. doi:10.1016/s0305-0483(99)00072-9.
69. Benbasat, I.; Zmud, R. (2003). "The identity crisis within the IS discipline: defining and communicating the discipline's core properties". MIS Quarterly. 27 (2): 183–194. doi:10.2307/30036527. JSTOR 30036527. S2CID 6017797.
70. Agarwal, R.; Lucas, H. (2005). "The information systems identity crisis: focusing on high- visibility and high-impact research". MIS Quarterly. 29 (3): 381–398. doi:10.2307/25148689. JSTOR 25148689. S2CID 15537428.
71. El Sawy, O (2003). "The IS core –IX: The 3 faces of IS identity: connection, immersion, and fusion". Communications of the Association for Information Systems. 12: 588–598. doi:10.17705/1cais.01239.
72. Mansour, O., Ghazawneh, A. (2009) Research in Information Systems: Implications of the constant changing nature of IT capabilities in the social

computing era, in Molka-Danielsen, J. (Ed.): Proceedings of the 32nd Information Systems Research Seminar in Scandinavia, IRIS 32, Inclusive Design, Molde University College, Molde, Norway, August 9–12, 2009. ISBN 978-82-7962-120-1.

73. Orlikowski, W.; Iacono, C. (2001). "Research commentary: desperately seeking the "IT" in IT research—a call to theorizing about the IT artifact". Information Systems Research. 12 (2): 121–134. doi:10.1287/isre.12.2.121.9700. S2CID 10833059.
74. Kock, N.; Gray, P.; Hoving, R.; Klein, H.; Myers, M.; Rockart, J. (2002). "Information Systems Research Relevance Revisited: Subtle Accomplishment, Unfulfilled Promise, or Serial Hypocrisy?". Communications of the Association for Information Systems. 8 (23): 330–346. doi:10.17705/1CAIS.00823.
75. Casalino, N., Mazzone, G. (2008): Externalization of a banking information systems function. Features, regulatory and critical aspects, in Interdisciplinary Aspects of Information Systems Studies, D'Atri A., De Marco M., Casalino N. (Eds.), Physica-Verlag, Springer, Heidelberg, Germany, pp. 89–96, ISBN 978-3-7908-2009-6, doi:10.1007/978-3-7908-2010-2_12
76. Senior Scholars (2007). "AIS Senior Scholars Forum Subcommittee on Journals: A basket of six (or eight) A* journals in Information Systems" (PDF). Archived from the original (PDF) on October 3, 2007.
77. "AIS affiliated conferences". Archived from the original on 2012-02-15. Retrieved 2012-02-10.
78. "AIS chapter conferences". affiniscape.com. Archived from the original on 2012-02-27. Retrieved 2012-02-10.
79. "EDSIG Information Systems Educators".
80. "Association of Information Technology Professionals".
81. EDSIG, ISCAP and. "ISCAP Conferences – EDSIGCON & CONISAR".