Тема 2. Архітектура типових веб-застосувань

Поняття архітектури ПЗ

Архітектура програмного забезпечення (англ. software architecture) — це структура програми або обчислювальної системи, яка містить програмні компоненти, видимі зовні властивості цих компонентів, а також відносини між ними. Цей термін також відноситься до документування архітектури програмного забезпечення. Документування архітектури ПЗ спрощує процес комунікації між зацікавленими особами, дозволяє зафіксувати прийняті на ранніх етапах проектування рішення про високорівневий дизайн системи і дозволяє використовувати компоненти цього дизайну і шаблони повторно в інших проектах.

Компоненти інформаційної системи по виконуваних функціях можна розділити на три шари: шар представлення, шар бізнес-логіки і шар доступу до даних.

Шар представлення - все, що пов'язано зі взаємодією з користувачем: натиснення кнопок, рух миші, отрисовка зображення, виведення результатів пошуку і так далі

Бізнес логіка - правила, алгоритми реакції додатка на дії користувача або на внутрішні події, правила обробки даних.

Шар доступу до даних - зберігання, вибірка, модифікація і видалення даних, пов'язаних з вирішуваною додатком прикладним завданням

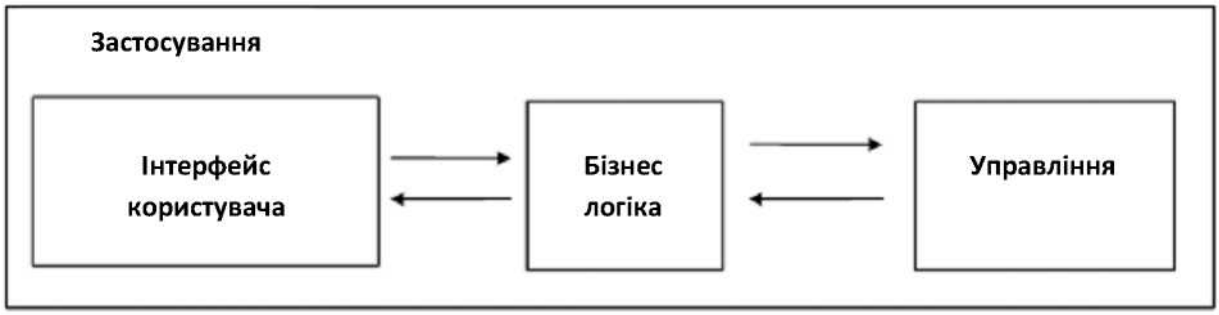

Рисунок 1.4 Компоненти інформаційної системи

Види архітектур сучасних програмних додатків

Історично першими з'явилися комп'ютерні системи з **централізованою архітектурою** додатків. При використанні цієї архітектури усе програмне забезпечення автоматизованої системи виконується централізовано на одному комп'ютері, що виконує одночасно багато завдань і що підтримує велику кількість користувачів. На цьому комп'ютері повністю здійснюється процес введення/ виведення інформації, а також її прикладна обробка. В якості центрального комп'ютера для такої системи може застосовуватися або велика ЕОМ(що називається також майнфреймом) або так звана МІНІ-ЕОМ. До подібного комплексу підключаються периферійні пристрої для введення/виведення інформації від кожного користувача.

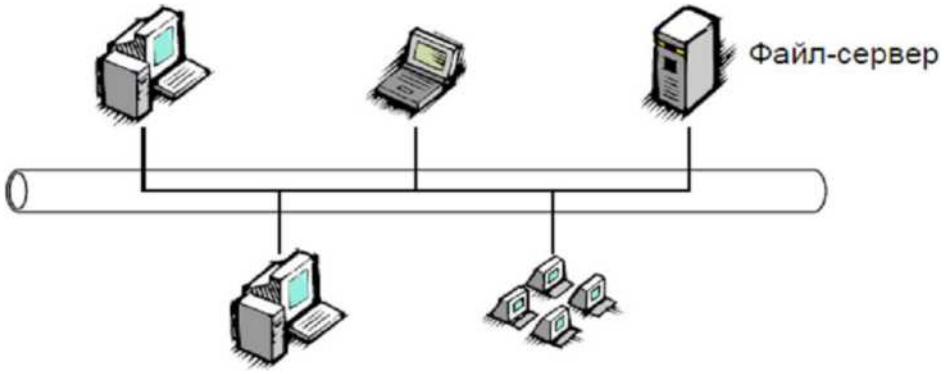

Рисунок 2. Файл-серверна архітектура

Наступний етап розвитку архітектури ПЗ **— файл-сервер —** це виділений сервер, призначений для виконання файлових операцій введення- виводу і зберігаючий файли будь-якого типу. Як правило, має великий об'єм дискового простору, реалізованому у формі RAID— масиву для забезпечення безперебійної роботи і підвищеної швидкості запису і читання даних.

Файл-серверні додатки — додатки, схожі по своїй структурі з локальними застосуваннями що використовують мережевий ресурс для зберігання даних у вигляді окремих файлів. Функції сервера у такому разі зазвичай обмежуються зберіганням даних(можливо також зберігання виконуваних файлів), а обробка даних відбувається виключно на стороні клієнта. Кількість клієнтів обмежена десятками зважаючи на неможливість одночасного доступу на запис до одного файлу. Проте клієнтів може бути в рази більше, якщо вони звертаються до файлів виключно в режимі читання.

Рисунок 3 Модель файлового сервера

Переваги:

- низька вартість розробки;
- висока швидкість розробки;
- невисока вартість оновлення і зміни ПЗ.

Недоліки:

- зростання числа клієнтів різко збільшує об'єм трафіку і навантаження на мережі передачі даних;
- високі витрати на модернізацію і супровід сервісів бізнес-логіки на кожній клієнтській робочій станції;
- низька надійність системи.

Архітектура клієнт-сервер

Архітектура є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з другого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не

мати жодного уявлення про існування інших клієнтів.

Загальноприйнятим є положення, що клієнти та сервери - це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми - і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним

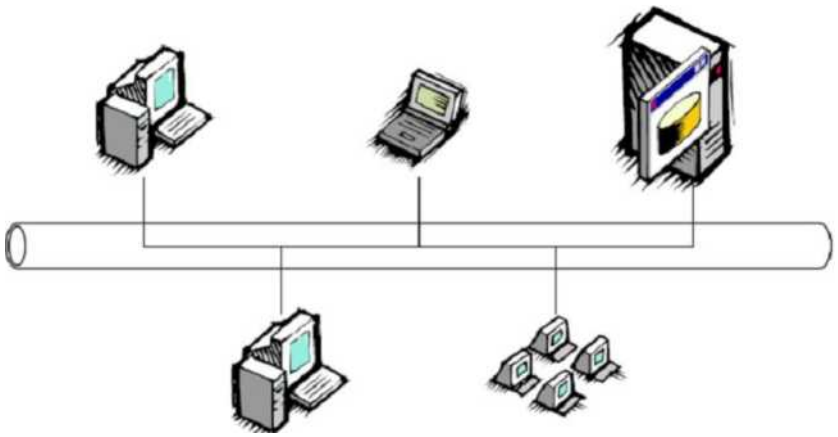

Рисунок 4 Архітектура клієнт-сервер

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна виокремити три рівні операцій:

- рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
- прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;
- рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів - клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

- модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;
- модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою

потужністю: кишенькові комп’ютери, мобільні телефони та ін.

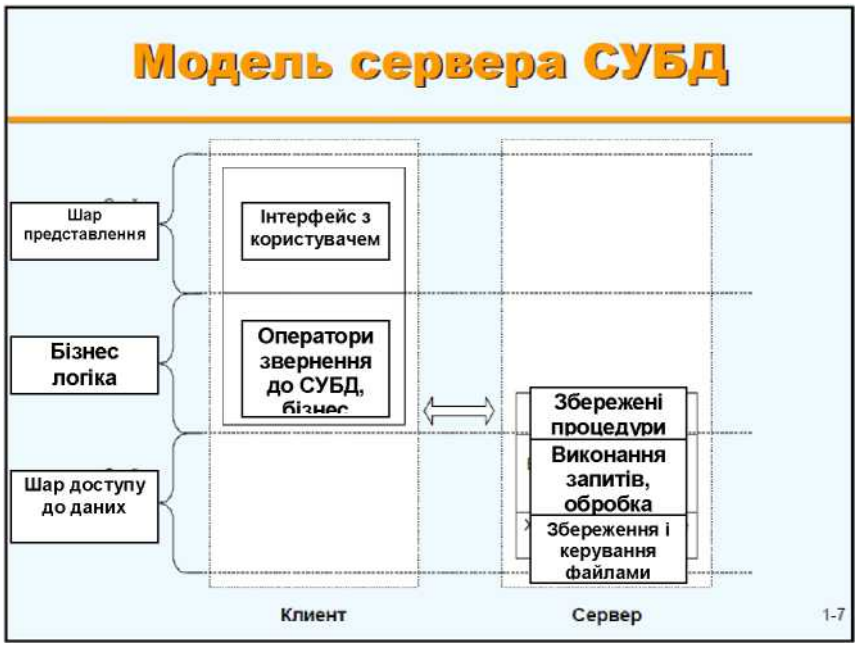

Рисунок 5 Архітектура клієнт-сервер

Плюси:

- Повна підтримка розрахованої на багато користувачів роботи
- Гарантія цілісності даних

Мінуси:

- Бізнес логіка додатків залишилася в клієнтському ПЗ. При будь-якій зміні алгоритмів, потрібно оновлювати призначене для користувача ПЗ на кожному клієнтові.
- Високі вимоги до пропускної спроможності комунікаційних каналів з сервером, що перешкоджає використання клієнтських станцій інакше як в локальній мережі.
- Слабкий захист даних від злому, особливо від недобросовісних користувачів системи.
- Висока складність адміністрування і налаштування робочих місць користувачів системи.
- Необхідність використати потужні ПК на клієнтських місцях.
- Висока складність розробки системи із-за необхідності виконувати бізнес-логіку і забезпечувати призначений для користувача інтерфейс в одній програмі

Трирівнева архітектура

Трирівнева архітектура, або триланкова архітектура (англ. three — tier або

англ. Multitier architecture) — архітектурна модель програмного комплексу, що припускає наявність в нім трьох компонентів: клієнтського застосування(що зазвичай називається «тонким клієнтом» або терміналом), сервера додатків, до якого підключено клієнтське застосування, і сервера бази даних, з яким працює сервер додатків.

- Клієнт — це інтерфейсний(зазвичай графічний) компонент, який представляє перший рівень, власне додаток для кінцевого користувача. Перший рівень не повинен мати прямих зв'язків з базою даних(за вимогами безпеки), бути навантаженим основною бізнес-логікою(за вимогами масштабованості) і зберігати стан додатка(за вимогами надійності). На перший рівень може бути винесена і зазвичай виноситься проста бізнес-логіка: інтерфейс авторизації, алгоритми шифрування, перевірка значень, що вводяться, на допустимість і відповідність формату, нескладні операції(сортування, угрупування, підрахунок значень) з даними, вже завантаженими на термінал.
- Сервер додатків розташовується на другому рівні. На другому рівні зосереджена большая частина бізнес-логіки. Поза ним залишаються фрагменти, що експортуються на термінали(см.вьше), а також занурені в третій рівень процедури, що зберігаються, і тригери.
- *Сервер бази даних* забезпечує зберігання даних і виноситься на третій рівень. Звичайно це стандартна реляційна або об'єктно-орієнтована СУБД. Якщо третій рівень є базою даних разом з процедурами, що зберігаються, тригерами і схемою, що описує додаток в термінах реляційної моделі, то другий рівень будується як програмний інтерфейс, що зв'язує клієнтські компоненти з прикладною логікою бази даних.

У простій конфігурації фізично сервер додатків може бути поєднаний з сервером бази даних на одному комп'ютері, до якого по мережі підключається один або декілька терміналів.

У «правильній»(з крапки зору безпеки, надійності, масштабування) конфігурації сервер бази даних знаходиться на виділеному комп'ютері(чи кластері), до якого по мережі підключені один або декілька серверів додатків, до яких, у свою чергу, по мережі підключаються термінали.

Переваги

В порівнянні з клієнт-серверною або файл-серверной архітектурою можна виділити наступні перевага трирівневої архітектури :

- масштабованість

- конфігурується — ізольованість рівнів один від одного дозволяє(при правильному розгортанні архітектури) швидко і простими засобами переконфігурувати систему при виникненні збоїв або при плановому обслуговуванні на одному з рівнів
- висока безпека
- висока надійність
- низькі вимоги до швидкості каналу(мережі) між терміналами і сервером додатків
- низькі вимоги до продуктивності і технічних характеристик терміналів, як наслідок зниження їх вартості. Терміналом може виступати не лише комп'ютер, але і, наприклад, мобільний телефон.

Рисунок 6 Архітектура клієнт-сервер

Недоліки

Недоліки витікають з переваг. По порівнянню с клієнт-серверної або файл- серверной архітектурою можна виділити наступні недоліки трирівневої архітектури :

- більш висока складність створення додатків;
- складніше в розгортанні і адмініструванні;
- високі вимоги до продуктивності серверів додатків і сервера бази даних, а, означає, і висока вартість серверного устаткування;
- високі вимоги до швидкості каналу(мережі) між сервером бази даних і серверами додатків.

Розподілені інформаційні системи.

У літературі можна знайти різні визначення розподілених систем, причому жодне з них не є задовільним і не узгоджується з іншими. Для наших завдань вистачить досить вільної характеристики.

Розподілена система — це набір незалежних обчислювальних машин, що представляється їх користувачам єдиною об'єднаною системою.У цьому визначенні обмовляються два моменти. Перший відноситься до апаратури: усі машини автономні.

Другий торкається програмного забезпечення: користувачі думають, що мають справу з єдиною системою. Важливі обидва моменти. Пізніше в цій главі ми до них повернемося, але спочатку розглянемо деякі базові питання, що стосуються як апаратного, так і програмного забезпечення.

Характеристики розподілених систем :

1. Від користувачів приховані відмінності між комп'ютерами і способи зв'язку між ними. Те ж саме відноситься і до зовнішньої організації розподілених систем.

2. Користувачі і додатки однаково працюють в розподілених системах, незалежно від того, де і коли відбувається їх взаємодія.

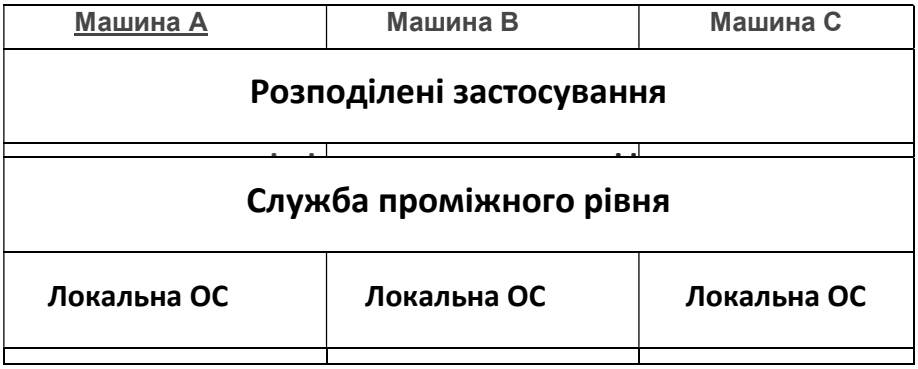

Рисунок 7 Модель розподіленої системи

Розподілені системи повинні також відносно легко піддаватися розширенню, або масштабуванню. Ця характеристика є прямим наслідком наявності незалежних комп'ютерів, але в той же час не вказує, яким чином ці комп'ютери насправді об'єднуються в єдину систему.

Розподілені системи зазвичай існують постійно, проте деякі їх частини можуть тимчасово виходити з ладу. Користувачі і додатки не повинні повідомлятися про те, що частини системи замінені або полагоджені або, що додані нові для підтримки додаткових користувачів.

Для того, щоб підтримати представлення системи в єдиному виді, організація розподілених систем часто включає додатковий рівень програмного забезпечення, що знаходиться між верхнім рівнем, на якому знаходяться користувачі і додатки, і нижнім рівнем, що складається з операційних систем.

Сервісно-орієнтована архітектура

Сервісно-орієнтована архітектура (англ. Service-oriented architecture, SOA) — архітектурний шаблон програмного забезпечення, модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабко пов'язаних замінних компонентів, оснащених стандартизованими інтерфейсами для взаємодії за стандартизованими протоколам.

Дуже часто становлення того чи іншого підходу супроводжується появою невірних або хибних трактувань, як це було, наприклад, з концепцією федеративного сховища даних. Не оминуло стороною це і сервіс-орієнтовану архітектуру. Так вважає представник компанії BEA Джерімі Уестерман (Jeremy Westerman). Саме тому в одній із своїх статей, присвячених SOA, він спеціально зупиняється на «проблемних місцях» і наводить докладні пояснення:

- SOA не є чимось новим: IT-відділи компаній успішно створювали і розгортали додатки, що підтримують сервіс-орієнтовану архітектуру, вже багато років — задовго до появи XML і Web-сервісів.
- SOA — це не технологія, а спосіб проектування і організації інформаційної архітектури та бізнес-функціональності.
- Купівля найновіших продуктів, що реалізують XML і Web-сервіси, не означає побудову додатків згідно з принципами SOA.

Джерімі Уестерман дає наступне визначення SOA: це парадигма, призначена для проектування, розробки та управління дискретних одиниць логіки (сервісів) в обчислювальному середовищі. Застосування цього підходу вимагає від розробників проектування додатків як набору сервісів, навіть якщо переваги такого рішення відразу неочевидні. Розробники повинні вийти за межі своїх додатків і подумати, як скористатися вже існуючими сервісами, або вивчити, як їх сервіси можуть бути використані їх колегами.

SOA підштовхує до використання альтернативних технологій і підходів (таких як обмін повідомленнями) для побудови додатків за допомогою зв'язування сервісів, а не за допомогою написання нового програмного коду. У цьому випадку, при належному проектуванні, застосування повідомлень

дозволяє компаніям своєчасно реагувати на зміну ринкових умов — настроювати процес обміну повідомленнями, а не розробляти нові програми. Ще до недавніх

пір термін «сервіс-орієнтована архітектура» був синонімом «Web-сервіс». SOA — виклик Web-сервісів за допомогою засобів і мов управління бізнес-процесами. SOA — це термін, який з'явився для опису виконуваних компонентів — таких як Web-сервіси — які можуть викликатися іншими програмами, які виступають у якості клієнтів або споживачів цих сервісів. Ці сервіси можуть бути повністю сучасними — або навіть застарілими — прикладними програмами, які можна активізувати як чорний ящик. Від розробника не потрібно знати, як працює програма, необхідно лише розуміти, які вхідні та вихідні дані потрібні, і як викликати ці програми для виконання. У найзагальнішому вигляді SOA припускає наявність трьох основних учасників: постачальника сервісу, споживача сервісу та реєстру сервісів. Взаємодія учасників виглядає досить просто: постачальник сервісу реєструє свої сервіси в реєстрі, а споживач звертається до реєстру із запитом.

Для використання сервісу необхідно дотримуватися угоди про інтерфейс для звернення до сервісу — інтерфейс повинен не залежати від платформи. SOA реалізує масштабованість сервісів — можливість додавання сервісів, а також їх модернізацію. Постачальник сервісу і його споживач виявляються непов'язаними — вони спілкуються за допомогою повідомлень. Оскільки інтерфейс повинен не залежати від платформи, то і технологія, використовувана для визначення повідомлень, також повинна не залежати від платформи. Тому, як правило, повідомлення є XML-документами, які відповідають XML-схемі.

Дійсно, відкриті стандарти, що описують XML і Web-сервіси, дозволяють застосовувати SOA до всіх технологій і додатків, встановлених в компанії. Як відомо, Web-сервіси базуються на широко поширених і відкритих протоколах: HTTP, XML, UDDI, WSDL і SOAP. Саме ці стандарти реалізують основні вимоги SOA — по-перше, сервіс повинен піддаватися динамічному виявленню і викликом (UDDI, WSDL і SOAP), по-друге, повинен використовуватися незалежний від платформи інтерфейс (XML). Нарешті, HTTP забезпечує функціональну сумісність.

Переваги використання SOA

Перш ніж, перерахувати переваги використання SOA, буде доречним нагадати, що переваги бувають різні: стратегічні і тактичні. SOA має ряд переваг як стратегічних, так і тактичних.

Стратегічна цінність SOA:

- Скорочення часу реалізації проектів, або «часу виходу на ринок».
- Підвищення продуктивності.
- Більш швидка і менш дорога інтеграція додатків і інтеграція B2B — зупинимося більш детально на цьому пункті.

Тактичні переваги SOA:

- Простіша розробка і впровадження додатків.
- Використання поточних інвестицій.
- Зменшення ризику, пов'язаного з впровадженням проектів в області автоматизації послуг і процесів.
- Можливість безперервного поліпшення наданої послуги.
- Скорочення числа звернень за технічною підтримкою.
- Підвищення показника повернення інвестицій (ROI).
- Перспективи

Контрольні питання

1. Що таке архітектура програмного забезпечення
2. З чого складається інформаційна система
3. Які є види сучасного ПЗ?
4. Порівняйте файл-серверну та клієнт-серверну архітектуру.
5. Переваги і недоліки трирівневої архітектури.
6. Принцип роботи розподілених систем.
7. SOA, її суть і особливості.