**Розробка серверної частини застосунку**

2 слайд

В основі клієнт-серверної архітектури лежать два компоненти: клієнт і сервер.

Клієнт – комп'ютер на стороні користувача, який відправляє запит до сервера для надання інформації або виконання певних дій.

Сервер – більш потужний комп'ютер або обладнання, призначене для вирішення певних завдань з виконання програмних кодів, виконання сервісних функцій за запитом клієнтів, надання користувачам доступу до певних ресурсів, зберігання інформації і баз даних.

Модель такої системи полягає в тому, що клієнт відправляє запит на сервер, де він обробляється, і готовий результат відправляється клієнтові. Сервер може обслуговувати кілька клієнтів одночасно. Якщо одночасно приходить більше одного запиту, то вони встановлюються в чергу і виконуються сервером послідовно. Іноді запити можуть мати пріоритети. Запити з більш високими пріоритетами повинні виконуватися раніше.

3 слайд

**Функції, які реалізуються на сервері:**

зберігання, доступ, захист і резервне копіювання даних;

обробка клієнтського запиту;

відправлення результату (відповіді) клієнту.

**Функції, які реалізуються на стороні клієнта:**

надання користувальницького інтерфейсу;

формулювання запиту до сервера і його відправка;

отримання результатів запиту і відправка додаткових команд (запитів на додавання, оновлення або видалення даних).

Архітектура клієнт-сервер визначає принципи спілкування між комп'ютерами, а правила і взаємодії визначені в протоколі.

Мережевий протокол – це набір правил, за якими відбувається взаємодія між комп'ютерами в мережі.

4 слайд

Розробка клієнт-серверних систем суттєво відрізняється за процесом і стилем від традиційних методів розробки інформаційних систем. Наприклад, підхід до розробки систем, орієнтований на централізоване середовище мейнфреймів і заснований на традиційних мовах програмування, навряд чи буде ефективним у клієнт-серверному середовищі, яке характеризується різноманітністю апаратного та програмного забезпечення.

Крім того, сучасні кінцеві користувачі мають вищі вимоги та частіше володіють більш глибокими знаннями про комп’ютерні технології, ніж користувачі в минулому, коли персональні комп’ютери лише почали поширюватися. Тому керівники, відповідальні за розробку та впровадження таких систем, повинні мати знання про новітні технології, що базуються на мультиплатформенних рішеннях, різних графічних інтерфейсах користувача (GUI), численних мережевих протоколах тощо.

(На [графіку](https://www.informit.com/articles/article.aspx?p=169547&seqNum=4) показано, як розвивалися різні технології серверної частини, починаючи з файлових серверів (1982), далі – серверів баз даних, групового програмного забезпечення (Groupware) і моніторів транзакцій (TP monitors) у 90-х, аж до розподілених об'єктів (Distributed Objects) у 1995 році та багаторівневої компонентної архітектури (N-tier componentware) після 2000 року.)

5 слайд

Інструменти розробки клієнт-серверних застосунків

У сучасному швидкозмінному середовищі вибір правильних інструментів для розробки клієнт-серверних застосунків є одним із найважливіших рішень. Як правило, керівники схильні обирати інструменти, що мають високі шанси на довгострокове використання. Однак вибір засобу для проєктування чи розробки застосунку також повинен ґрунтуватися на вимогах до системи. Після визначення цих вимог слід визначити характеристики інструменту, який відповідатиме потребам розробки.

6 слайд

Серед ключових можливостей інструментів для клієнт-серверної розробки можна виділити:

♦ Розробку на основі графічного інтерфейсу (GUI).

♦ Конструктор інтерфейсу, що підтримує різні середовища (Windows, OS/2, Motif, Macintosh тощо).

♦ Об’єктно-орієнтовану розробку з єдиним централізованим сховищем даних і застосунків.

♦ Підтримку різних типів баз даних (файлових, ієрархічних, мережевих, реляційних).

♦ Доступ до даних незалежно від моделі даних (через SQL або нативні методи навігації).

♦ Прозорий доступ до кількох баз даних одночасно.

♦ Підтримку повного життєвого циклу розробки системи (SDLC) – від планування до впровадження та супроводу.

♦ Можливості командної розробки.

♦ Інтеграцію зі сторонніми інструментами розробки (CASE-системи, бібліотеки тощо).

♦ Підтримку швидкої розробки прототипів і методологій Rapid Application Development (RAD).

♦ Сумісність із різними платформами (операційними системами, апаратним забезпеченням, графічними інтерфейсами).

♦ Підтримку проміжного програмного забезпечення (middleware), зокрема ODBC, IDAPI, APPC тощо.

♦ Підтримку різних мережевих протоколів (TCP/IP, IPX/SPX, NetBIOS тощо).

Не існує універсального інструменту для розробки клієнт-серверних застосунків, який підтримував би всі можливі графічні інтерфейси, операційні системи, проміжне ПЗ та бази даних. Тому керівникам важливо обирати інструменти, які відповідають вимогам до розробки, наявним людським ресурсам і апаратній інфраструктурі. У більшості випадків для реалізації системи знадобиться використання кількох інструментів, щоб задовольнити всі або більшість вимог. Вибір засобів розробки – лише один із кроків. Не менш важливо переконатися, що система відповідає своїм завданням на рівні клієнта, сервера та мережевої взаємодії.

7 слайд

Фази розробки клієнт-серверних систем

Перед початком проєктування та розробки необхідно розробити маркетинговий план. Метою цього плану є залучення підтримки кінцевих користувачів і керівництва для впровадження клієнт-серверного середовища.

Хоча не існує єдиного підходу до цього процесу, загальна ідея полягає в тому, щоб концептуалізувати клієнт-серверну систему з точки зору її масштабу, оптимізації ресурсів та управлінських переваг. Іншими словами, план передбачає комплексний підхід і взаємодію всіх відділів організації.

Існує 6 основних фаз розробки клієнт-серверної системи.

8 слайд

Інфраструктура інформаційної системи — самостійне вивчення

Мета полягає в тому, щоб визначити фактичний стан наявних комп'ютерних ресурсів. Самостійне вивчення повинно включати принаймні таке:

Інвентаризація програмного та апаратного забезпечення.

Деталізований та описовий список критичних додатків.

Деталізований інвентаризаційний список людських ресурсів (персонал та навички).

Деталізований список проблем та можливостей.

Визначення інфраструктури клієнт/сервер

Результат першої фази, у поєднанні з метою комп'ютерної інфраструктури компанії, є вхідними даними для розробки основного проєкту інфраструктури клієнт/сервер. У цьому плані розглядаються основні проблеми апаратного та програмного забезпечення для клієнта, сервера та мережевих платформ

Вибір вікна можливостей

Наступний етап — це вибір правильної системи, на основі якої буде побудований пілотний проєкт клієнт/сервер. Після визначення пілотного проєкту, потрібно дуже ретельно його сформулювати, зосередившись на проблемі, наявних ресурсах та наборі чітко визначених і реалістичних цілей. Проєкт описується у бізнесових термінах, а не технологічним жаргоном. При визначенні системи важливо ретельно спланувати витрати. Необхідно збалансувати витрати з ефективними перевагами системи. Також потрібно переконатися, що вибрана пілотна реалізація принесе негайні та відчутні результати. Наприклад, система, розробка якої займає два роки, а отримання відчутних результатів ще три — не є прийнятною.

Зобов'язання керівництва

Зобов'язання на всіх рівнях є необхідним при впровадженні нових технологій, які впливають на всю організацію. Керівництво повинно забезпечити наявність необхідних ресурсів (люди, апаратне забезпечення, програмне забезпечення, кошти, інфраструктура) для системи. Зазвичай призначають особу, яка буде виступати як керівник змін, допомагаючи полегшити процес змін ролей співробітників у організації.

Рекомендації щодо впровадження

Рекомендації щодо впровадження повинні включати, принаймні:

Використовувати "відкриті" інструменти або інструменти на основі стандартів.

Сприяти безперервному навчанню в галузі апаратного та програмного забезпечення, інструментів і принципів розробки.

Шукати постачальників і консультантів для надання спеціалізованого навчання та впровадження проектів, апаратного забезпечення та прикладного програмного забезпечення.

Огляд та оцінка

Необхідно переконатися, що система відповідає критеріям, визначеним на третій фазі. Треба постійно вимірювати продуктивність системи, оскільки зростання навантаження на систему може збільшити мережевий трафік і уповільнити мережу. Необхідно ретельно моделювати продуктивність мережі, щоб забезпечити належну роботу системи під великим навантаженням користувачів. Моделювання продуктивності повинно здійснюватися на серверній частині, клієнтській частині та на мережевому рівні.

9 слайд

Оцінка продуктивності та моделювання навантажень — це критично важливий етап. Але щоб система стабільно працювала та ефективно масштабувалася, важливо дотримуватись певних стандартів у розробці серверної частини. . Ці стандарти визначають правила та методи, за якими різні системи — бази даних, сервери, клієнтські програми — взаємодіють між собою. Вони гарантують, що всі компоненти працюють узгоджено, дозволяють інтегрувати нові технології без ризику несумісності та підвищують рівень безпеки.

10 слайд

Дотримання стандартів у серверних технологіях забезпечує:

**Інтеграція з існуючими ІТ-інвестиціями**: Стандарти дозволяють без зусиль додавати нові технології та продукти до вже існуючих систем.

**Незалежність продуктів**: Можливість комбінувати найкращі продукти без залежності один від одного (бази даних, інструменти розробки, програмне забезпечення для підключення).

**Модульність і гнучкість**: Стандарти дозволяють створювати модульні програми, які не залежать від змін у мережі чи інтерфейсі.

**Забезпечення безпеки**: Стандарти сприяють покращенню загальної безпеки системи.

**Зменшення навантаження на обслуговування**: Стандарти допомагають знизити витрати на обслуговування та адміністрування систем.

**Оптимізація продуктивності**: Швидке виконання попередньо скомпільованого коду завдяки стандартам.

11 слайд

Дотримання стандартів — це не лише про внутрішню структуру коду, а й про комунікацію між клієнтом і сервером. Взаємодія між ними відбувається через HTTP-протокол, який визначає, як клієнт надсилає запити та отримує відповіді. Основними методами HTTP є:

**Запити GET** використовуються для отримання даних із сервера. Коли клієнт надсилає запит GET, він просить сервер надіслати певний ресурс, наприклад веб-сторінку чи зображення. Запит також може містити такі параметри, як терміни пошуку, які визначають, які дані клієнт хоче отримати. Запити GET часто використовуються, коли користувач натискає посилання або вводить URL-адресу у своєму браузері.

Користувач хоче шукати продукт на веб-сайті електронної комерції. Вони вводять пошуковий термін у вікно пошуку та натискають «Пошук». Клієнт надсилає запит GET на сервер із пошуковими термінами як параметрами. Сервер повертає список продуктів, які відповідають умовам пошуку.

12 слайд

**Запити POST** використовуються для надсилання даних на сервер. Коли клієнт надсилає запит POST, він включає дані в тіло запиту, які сервер може використовувати для виконання дії або оновлення бази даних. Запити POST часто використовуються, коли користувач надсилає форму або надсилає повідомлення.

Користувач хоче створити новий обліковий запис на платформі соціальних мереж. Вони заповнюють реєстраційну форму та натискають «Надіслати». Клієнт надсилає на сервер запит POST з інформацією про користувача в тілі запиту. Сервер створює новий обліковий запис і надсилає відповідь клієнту з підтвердженням створення.

13 слайд

**Запити PUT** використовуються для оновлення наявних даних на сервері. Коли клієнт надсилає запит PUT, він містить дані в тілі запиту, які вказують, як слід оновлювати існуючі дані. Запити PUT часто використовуються, коли користувач редагує наявний запис або оновлює інформацію свого профілю.

Користувач хоче оновити інформацію свого профілю на веб-сайті пошуку роботи. Вони редагують свою інформацію та натискають «Зберегти». Клієнт надсилає на сервер запит PUT з оновленою інформацією в тілі запиту. Сервер оновлює профіль користувача та надсилає відповідь клієнту з підтвердженням оновлення.

14 слайд

**Запити DELETE** використовуються для видалення даних із сервера. Коли клієнт надсилає запит DELETE, він просить сервер видалити певний ресурс, наприклад файл або запис бази даних. Запити DELETE часто використовуються, коли користувач видаляє повідомлення або вилучає товар зі свого кошика для покупок.

Користувач хоче видалити повідомлення зі своєї папки "Вхідні" в службі електронної пошти. Вони вибирають повідомлення та натискають «Видалити». Клієнт надсилає серверу запит DELETE з ідентифікатором повідомлення. Сервер видаляє повідомлення та надсилає відповідь клієнту з підтвердженням видалення.

Розуміння різних типів запитів у програмуванні на стороні сервера є важливим для створення веб-додатків, які можуть обробляти різні взаємодії користувача. Кожен із запитів GET, POST, PUT і DELETE служить унікальній меті та надсилає різну інформацію на сервер. Знання того, як ефективно використовувати ці запити, може допомогти розробникам створювати ефективніші та ефективніші веб-додатки.

15 слайд

Цикл запит-відповідь — це основна взаємодія між клієнтом і сервером у програмуванні на стороні сервера. Це передбачає надсилання клієнтом запиту на сервер, а сервер відповідає даними, які клієнт може потім показати користувачеві.

Цикл починається з надсилання клієнтом запиту на сервер. Запит зазвичай містить метод (наприклад, GET, POST, PUT або DELETE), який визначає тип запиту, а також будь-які параметри або дані, необхідні серверу для обробки запиту. Як тільки сервер отримує запит, він обробляє його та генерує відповідь.

Потім сервер надсилає відповідь клієнту. Відповідь містить код статусу, який вказує, чи був запит успішним чи ні, а також будь-які дані чи інформацію, які потрібні клієнту для виконання запиту користувача.

Коли клієнт отримує відповідь, він обробляє його та відображає дані користувачеві.

Цикл запит-відповідь є фундаментальним процесом програмування на стороні сервера. Це включає в себе клієнт, який надсилає запит на сервер, сервер обробляє запит і генерує відповідь, а клієнт відображає дані користувачеві. Розуміння того, як працює цей цикл, має вирішальне значення для створення веб-додатків, які можуть обробляти взаємодію користувачів і забезпечувати безперебійну роботу користувача.

Проте цей процес має і свої ризики. Під час кожного запиту клієнт може надсилати чутливі дані — паролі, платіжні реквізити. Саме тому наступний крок — це забезпечення безпеки під час обміну даними.

16 слайд

Безпека є важливою умовою програмування на стороні сервера, оскільки веб-програми можуть бути вразливими до різноманітних атак, у тому числі витоку даних, атак, міжсайтових сценаріїв тощо. Ці атаки можуть поставити під загрозу цілісність і конфіденційність даних, завдати шкоди користувачам і репутації програми.

Існує кілька заходів, які можна вжити для захисту веб-програм від атак. Одним із найважливіших кроків є реалізація механізмів автентифікації та авторизації, щоб гарантувати, що лише авторизовані користувачі можуть отримати доступ до програми та її даних. Це може включати використання безпечних систем входу, застосування політики паролів і використання багатофакторної автентифікації.

Іншим важливим заходом безпеки є шифрування конфіденційних даних, таких як паролі та дані кредитних карток. Це можна зробити за допомогою галузевих стандартних алгоритмів шифрування, які шифрують дані під час передачі та зберігання.

Існує також кілька найкращих практик, яких можна дотримуватися, щоб запобігти поширеним вразливостям безпеки. Наприклад, розробники повинні очистити введені користувачем дані, щоб запобігти атакам, перевірити введені користувачем дані, щоб переконатися, що вони відповідають необхідному формату, і використовувати підготовлені оператори або параметризовані запити, щоб запобігти атакам SQL.

Міжсайтовий сценарій (XSS) — це ще одна поширена вразливість, якій можна запобігти, запровадивши перевірку вхідних даних і кодування вихідних даних. Це передбачає перевірку введеного користувачем зловмисного коду та кодування будь-якого виводу, щоб запобігти його інтерпретації як код.

На додаток до цих заходів розробники також можуть використовувати фреймворки та бібліотеки, орієнтовані на безпеку, щоб забезпечити безпеку своїх веб-додатків. Ці структури забезпечують набір вказівок і інструментів, які можна використовувати для виявлення та запобігання поширеним вразливостям безпеки.

Безпека є важливою умовою програмування на стороні сервера, і розробники повинні вживати профілактичних заходів для захисту веб-додатків від атак. Це може включати впровадження механізмів автентифікації та авторизації, шифрування конфіденційних даних, дезінфекцію введених користувачами та використання фреймворків і бібліотек, орієнтованих на безпеку. Дотримуючись найкращих практик безпеки, розробники можуть гарантувати, що їхні веб-програми захищені від атак, а також забезпечать безпечну та надійну роботу для своїх користувачів.

***Слайд 17***

Серверні мови та фреймворки.

1. PHP - це популярна мова серверного програмування, яка використовується для створення динамічних веб-додатків. Вона має відкритий вихідний код і проста у вивченні, що робить її популярним вибором серед веб-розробників. PHP часто використовується в поєднанні з веб-фреймворками, такими як Laravel, Symfony або CodeIgniter. Ці фреймворки надають структуру для організації коду і спрощують багато поширених завдань, таких як доступ до бази даних і автентифікація користувачів.

Приклад: WordPress, популярна система управління контентом, створена за допомогою PHP і фреймворку WordPress.

***Слайд 18***

2. Python - це універсальна мова програмування, яка популярна в багатьох сферах, включаючи серверне програмування. Її часто використовують у поєднанні з веб-фреймворками, такими як Django, Flask або Pyramid. Ці фреймворки надають інструменти для створення веб-додатків, включаючи підтримку доступу до баз даних, автентифікації користувачів і маршрутизації.

Django - це популярний веб-фреймворк для Python, який надає інструменти для швидкого та легкого створення веб-додатків. Django включає вбудовану підтримку доступу до баз даних, автентифікації користувачів та маршрутизації, а також багато інших функцій, які полегшують створення веб-додатків.

Приклад: Instagram, популярний додаток для обміну фотографіями, створений за допомогою Python та фреймворку Django.

***Слайд 19***

3. Ruby - це динамічна, об'єктно-орієнтована мова програмування, яка часто використовується для веб-розробки. Вона часто використовується разом з фреймворком Ruby on Rails, який забезпечує структуру для створення веб-додатків. Ruby on Rails має багато вбудованих функцій, таких як доступ до баз даних і маршрутизація, які полегшують швидке створення веб-додатків.

Приклад: GitHub, популярний репозиторій коду та платформа для спільної роботи, створений з використанням Ruby on Rails.

***Слайд 20***

4. Node.js - це популярна серверна платформа, яка дозволяє розробникам створювати масштабовані, високопродуктивні веб-додатки за допомогою JavaScript. Node.js побудований на основі JavaScript-рушія V8, який також використовується у веб-браузері Google Chrome. Node.js часто використовується разом з такими фреймворками, як Express або Koa, які надають інструменти для створення веб-додатків.

Приклад: LinkedIn, популярний професійний мережевий сайт, використовує Node.js для роботи своїх серверних додатків.

Існує багато популярних серверних мов програмування та фреймворків, з яких розробники можуть вибирати для створення веб-додатків. Кожна мова та фреймворк має свої сильні сторони та переваги, і вибір правильної мови залежить від конкретних потреб проекту. Розуміючи різні доступні варіанти, розробники можуть вибрати найкращі інструменти для роботи і створювати високоякісні веб-додатки, які відповідають потребам користувачів.

***Слайд 21***

Інтеграція з базами даних.

Серверне програмування відіграє важливу роль в інтеграції баз даних з веб-додатками. Використовуючи мови та фреймворки серверного програмування, розробники можуть створювати потужні та масштабовані додатки, які можуть зберігати та отримувати дані з бази даних.

Існує багато різних типів баз даних, які можна використовувати у веб-додатках, але деякі з найбільш поширених - MySQL, MongoDB і PostgreSQL. Кожна з цих баз даних має свої сильні сторони та переваги, і вибір правильної бази даних залежить від конкретних потреб програми.

***Слайд 22***

1. MySQL - це популярна система управління реляційними базами даних з відкритим вихідним кодом, яка широко використовується у веб-додатках. Вона відома своєю швидкістю, надійністю та простотою використання. MySQL часто використовується з мовою програмування PHP і сумісна з багатьма різними веб-фреймворками (Laravel, Symfony).

***Слайд 23***

2. MongoDB - це популярна база даних документів NoSQL, яка відрізняється масштабованістю та гнучкістю. Її часто використовують у веб-додатках, які потребують складних структур даних, таких як соціальні мережі або сайти електронної комерції. MongoDB часто використовується з мовою програмування Node.js і сумісна з багатьма різними веб-фреймворками, включаючи Express і Koa.

***Слайд 24***

3. PostgreSQL - ще одна популярна система управління реляційними базами даних з відкритим вихідним кодом, яка відома своєю стабільністю, надійністю та розширеними можливостями. Її часто використовують у веб-додатках, які вимагають високого рівня цілісності даних, наприклад, у фінансових або медичних програмах. PostgreSQL часто використовується з мовою програмування Python і сумісна з багатьма різними веб-фреймворками, включаючи Django і Flask.

***Слайд 25***

Серверні мови програмування та фреймворки надають різноманітні інструменти та бібліотеки для доступу до баз даних та керування ними. Ці інструменти дозволяють розробникам писати код, який взаємодіє з базою даних, наприклад, вставляти або оновлювати дані, а також робити запити до даних для отримання певної інформації. Інтегруючи бази даних з веб-додатками, розробники можуть створювати потужні та динамічні додатки, які можуть зберігати та отримувати великі обсяги даних, що є критично важливим для багатьох сучасних веб-додатків.

Отже, програмування на стороні сервера є ключовим аспектом сучасної веб-розробки, а розуміння моделі клієнт-сервер має важливе значення для розробки надійних і безпечних веб-додатків. Ми обговорили різні типи запитів, які клієнт може робити до сервера, цикл запитів-відповідей, популярні серверні мови та фреймворки, інтеграцію з базами даних та міркування безпеки.

Важливо зазначити, що існує набагато більше аспектів серверного програмування, ніж ми розглянули, і в цій галузі постійно з'являються нові розробки та технології. Однак, розуміючи основи клієнт-серверної моделі та різні інструменти і методи, що використовуються в серверному програмуванні, розробники можуть створювати надійні і безпечні веб-додатки, які відповідають потребам своїх користувачів.