Решетуха Я.О. ПI-31/14

**BACK END**

В програмній інженерії терміни «front end» та «back end» розрізняють за принципом розділення відповідальності між рівнем представлення та рівнем доступу до даних відповідно.

Front end — це інтерфейс для взаємодії між користувачем і back end. Front end та back end можуть бути розподілені між однією або кількома системами.

Бекенд development - це процес програмування сайту і наповнення його функціоналом. Створення ядра сайту, розробка платформи сайту, наповнення його основним функціоналом і створення адміністративної зони - це і є бекенда розробка.

В програмній архітектурі може бути багато рівнів між апаратним забезпеченням та кінцевим користувачем. Кожен з цих рівнів може мати як front end, так і back end. Front — це абстракція, спрощення базового компоненту через надання користувачу зручного (user-friendly) інтерфейсу.

В проектуванні програмного забезпечення, наприклад, архітектура Модель-Вид-Контролер надає front end та back end для бази даних, компонентів обробки користувачів і даних. Розділення програмних систем на front end та back end спрощує розробку і розділяє підтримку. Емпіричне правило полягає в тому, що front (чи «клієнтська») сторона — це будь-який компонент, яким керує користувач. Код серверної сторони (чи «back end»-у) знаходиться на сервері. Плутанина виникає, коли хтось повинен зробити front-end зміни файлів на боці серверу. Більшість HTML-дизайнерів, наприклад, не повинні бути на сервері під час розробки HTML; і навпаки, серверні інженери, за визначенням, не повинні бути ніде окрім серверу. Потрібні обидва розробники, щоб в кінцевому результаті створити функціональний, інтерактивний веб-сайт.

Для більшості компʼютерних підсистем, графічні файлові менеджери є «front end»-ом для компʼютерних файлових систем, і оболонкою для операційної системи. Користувач діє від імені «front end»-у, а back end запускає програми операційної системи у відповідь.

Використання інтерфейсу командного рядка (CLI) потребує знання спеціальної термінології і памʼятання команд, а графічний інтерфейс користувача (GUI) виступає замість нього «front end»-ом як стільничне середовище. В Unix-середовищі, ncurses — це простий псевдографічний front end для інтерфейсу командного рядка. На рівні інтерфейсу командного рядка Unix найбільше байтових потоко-орієнтованих (тобто таких, що використовують stdin/stdout/stderr як інтерфейс) програм діють як фільтри — автономні програми, які також можуть слугувати «front end»-ами і «back end»-ами для інших програм. (Вони функціонують завдяки передачі даних між собою через конвеєр, в основному для обробки тексту; наприклад: $ grep word ~user/dir/infile | sort | tee ~user/dir/outfile)

В компʼютерній мережі front end може належати до будь-якого апаратного забезпечення, що оптимізує чи захищає мережевий трафік. Така річ називається апаратним front end додатком, бо вона знаходиться перед зовнішньою мережею чи на її межі. Мережевий трафік проходить через апаратний front end перед входом до мережі.

В системах керування вмістом (CMS) терміни front end та back end можуть належати до тієї частини CMS, яка відображається кінцевому користувачу і до адміністративної частини відповідно.

В компіляторах front end транслює компʼютерний програмний сирцевий код в проміжне представлення, а back end працює з проміжним представленням, щоб генерувати машинний код. Back end зазвичай оптимізований для генерації коду, який працює швидше. Front-end/back-end відмінності можуть розділитись на парсерну частину і back end, який генерує і оптимізує код. Деякі проекти, такі як GCC, надають вибір між кількома «front end»-ами (синтаксичний аналіз різних сирцевих мов) чи «back end»-ами (генерація коду для процесорів різного призначання).

В синтезі мовлення, front end належить до частини системи синтезу, яка конвертує вхідний текст у символьно-фонетичний образ, а back end конвертує символьно-фонетичний образ безпосередньо в звуки.

В контексті WWW-програм, посередник — це сервіс, який функціонує одночасно як сервер на «front end»-і та як клієнт на «back end»-і.

**обов'язки сучасних back-end розробників**

Писати код вже не досить:

Давайте розглянемо типовий день розробника на Java. Його робота зводиться до таких завдань:

- Зайти на віддалений сервер через консоль і налаштувати одну з компонент, підправити скрипт, сконфігуровровать один з допоміжних сервісів.

- Система ведення проектів; спілкуватися з колегами на предмет наявності завдань, які необхідно зараз вирішити.

- Шанувати інструкцію до недавно введеному фреймворку, розібратися і виконати з його допомогою одне із завдань.

- Виправити помилку, недавно знайдену тестувальниками.

- Відповісти на серію листів або повідомлень в листуванні з замовником. Зрозуміло, англійською. При необхідності зателефонувати і обговорити питання усно.

- Обговорити з колегами подальші плани на проекті, проблеми та ідеї. Крім запланованих зустрічей це може відбуватися під час обіду або в невеликому перерві, коли ви всі вільні.

- Поспілкуватися з новачками, які ще вникають в проект (якщо ви - провідний розробник). Надати свою допомогу, якщо потрібно.

- Спілкуватися з іншими командами, якщо над проектом працює більше однієї команди. Час від часу може виникати необхідність обговорити питання, наприклад, з front-end розробниками або «ораклістамі», інженерами баз даних. Можливо, знадобитися звертатися до них за порадою.

- Стежити за станом серверів: контролю версій, безперервної інтеграції, базою даних, тестового, робочого і бойового серверів. Це мінімум, а можливо, їх буде набагато більше, це залежить від складності проекту.

- Посидіти самому або в парі і подумати над архітектурою системи або якоїсь однієї компонентою.

Тут я перерахував не всі активності, а тільки самі поширення і базові.

Чи не все вищеописане робить кожен розробник в команді, і не кожен день. Як правило, ці завдання розподіляються за ролями. Але все-таки потрібно тримати ці активності в розумі і в міру необхідності приділяти час тієї чи іншої.

Бекенд робить обробку інформації користувачів отриманої з front-офісу, і повертає front-end'у результат у зрозумілій йому формі.

Бекенд програмування - це веб програмування, метою якого є реалізація серверної сторони сайту, інтеграція бази даних і зв'язок її з призначеної для користувача (front-end) стороною. Розробка бекенда сайту так само включає настройку і установку на сервер необхідного програмного забезпечення.

Простіше кажучи, фронтенд передає інформацію і команди від користувача в бекенда, а той в свою чергу виробляє їх обробку. Або якщо вже зовсім просто, то - Front-end створюється для користувача сайту, а Back-end для його адміністратора.

**Платформи для створення сайтів: CMS, фреймворки і SaaS-рішення**

Найбільш популярні види платформ для розробки сайту

Найбільш поширений варіант - це розробка на CMS-платформах.

Система управління сайтом (Content Management System) - це програмний продукт, який служить для розробки деяких стандартних різновидів сайтів. Майже всі CMS модульні, а модулі багатьох з них зібрані в комплекти (або редакції), призначені для тих чи інших видів сайтів. Є коробкові CMS для простих сайтів, для каталогів, для інтернет-магазинів, для блогів, для новинних порталів і для інших видів сайтів.

Велика частина невеликих і середніх сайтів будується саме на CMS-платформах, так як це найбільш доцільний з економічної точки зору підхід: вимоги до таких сайтів з технічної точки зору невисокі, а вбудовані в CMS модулі зазвичай задовольняють висунутим до них бізнес-вимогам.

Ми, наприклад, розробляємо комерційні сайти на UMI.CMS і 1C-Bitrix. Це вітчизняні комерційні CMS-платформи, які вже багато років використовуються для створення сайтів і добре зарекомендували себе.

Найгнучкіша і найбільш потужна платформа - фреймворк.

Фреймворк - це програмний продукт, який служить основою для сайту, але зазвичай не містить в собі готових програмних модулів для реалізації конкретних бізнес-процесів. Висловлюючись технічною мовою, фреймворк - це більш низкоуровневое рішення, ніж CMS. Розробники, при створенні сайту на фреймворку, створюють не тільки публічну частину сайту, але і проектують базу даних, розробляють алгоритми для модулів системи, а також створюють адміністративний інтерфейс для управління проектом. Необхідність серйозних витрат на програмування робить розробку більш дорогий, але і результат виходить більш індивідуальним.

Цей вид платформ використовують майже всі великі веб-проекти (на CMS побудована лише дуже мала частина дійсно серйозних проектів), а також переважна більшість веб-додатків і веб-сервісів (коробкових рішень для унікальних бізнес-процесів просто не існує, а використання не дуже відповідних CMS в якості основи для кастомізації дуже ускладнює розробку).

Ми, наприклад, розробляємо серйозні веб-проекти на фреймворку Ruby on Rails. Даний фреймворк дозволяє створювати дійсно складні сайти, бізнес-додатки і веб-сервіси. Швидкість роботи і стійкість до навантажень у таких рішень на порядок вище (в порівнянні з CMS), а сопровождаемость і вартість володіння при цьому не сильно відрізняються.

CMS-платформи

CMS на ринку існує багато і вони дуже різні:

є безкоштовні, платi;

бувають коробкові і самописние;

велика частина CMS універсальні, тобто годяться для практично будь-яких комерційних сайтів;

деякі з CMS дуже спеціалізовані - вони підходять, наприклад, тільки для блогів або тільки для соціальних ЗМІ.

**Фреймворки**

Фреймворки - це платформа для складних сайтів або веб-додатків, яка оптимально підходить для створення сайтів і веб-додатків за індивідуальним проектом, які повинні працювати швидко і витримувати серйозні навантаження. Треба бути готовим до того, що при виборі фреймворка в якості платформи для розробки, доведеться досить багато часу і ресурсів витратити на розробку бізнес-логіки, це плата за гнучкість.

Яка платформа краще: CMS або фреймворк?

Відповідь на це питання дуже проста - дивлячись що саме треба розробити: якщо проект простий і вимоги до нього самі звичайні, то вибирається CMS, а якщо проект складний, то простіше його на фреймворку робити.

SaaS-платформи

Також зараз «набирає обертів» створення сайтів на SaaS-платформах. SaaS-платформа для створення сайтів - це можливість запустити досить простий веб-проект дуже швидко і вкрай дешево (причому ще і на умовах оренди). Рішення підходить для простих сайтів, тимчасових проектів і для перевірки бізнес-ідей. SaaS-платформи, як і CMS, бувають специфічними (тільки для інтернет-магазинів, наприклад) і універсальними (для всіх типових видів сайтів).

**Семантичне ядро ​​сайту**

Семантичне ядро ​​сайту - це упорядкований набір пошукових слів, їх морфологічних форм і словосполучень, які найбільш точно характеризують вид діяльності, товар або послугу, пропоновані сайтом. Семантичне ядро ​​має центральне ключове слово, як правило високочастотне, і всі інші ключові слова в ньому ранжуються у напрямку зниження частоти спільного використання з центральним запитом до загальної колекції документів. Таким чином, семантичне ядро ​​представляється у вигляді семантичного графа, де довжини його ребер обернено пропорційні частоті спільного згадки.

Ключові слова (пошукові запити) для семантичного ядра підбираються шляхом суворого аналізу послуг або товарів компанії, аналізу статистики пошукової системи, статистики сайту, конкурентів на ринку і сезонності ключових слів. Запити в семантичному ядрі повинні максимально відповідати уявленням цільових відвідувачів про сайт і про ту інформацію яка на ньому на їхню думку має бути присутня.

Семантичне ядро ​​сайту складають ті ключові слова, які виявила пошукова машина при скануванні сайту, однак, може серйозно відрізнятися від еталонного, на базі семантичного графа, що характеризує дану тематику. Це обумовлено особливостями бізнес моделі даного ресурсу.

**Мета створення семантичного ядра сайту**

Семантичне ядро утворює тематику сайту, яка буде враховуватися пошуковими системами.

Правильно сформований семантичне ядро є основою для оптимальної структури веб-ресурсу.

**Процедура складання семантичного ядра**

Перш чем збіраті семантичного ядро АдВордса, слід ретельно вивчити характер продажів и всю бізнес модель підприємства. Це дозволяє визначити, что Підходить в знайденому, а що ні. Семантичного ядро, зібране таким чином, виявилось найбільш наближеним до того ідеалу, до якого й сам Гугл винен прагнути. Існують сервіси, які роблять це автоматично, віділяючі з існуючого масиву найбільш цінні ключові фрази. Існує 3 види семантично ядер: комерційне, некомерційне, и об'єднане, в якi входять комерційні й некомерційні додатки одночасно. Поділ ядра на комерційне та некомерційне використовується в деяких просунутих технологіях виготовлення сателітів.

Системи «Google Analytics», «Яндекс.Вордстат» и подібні до них, якi надают інформацію про статистику ключовими слів, аналіз статистики основних ключовими слів, за Якими здійснюється Перехід на сайт, для которого складається семантичного ядро ​​(інформація доступна в «Google Analytics», «Яндекс.Метрика» и подібніх до них), моніторинг відімості конкурентів по просуваються для Розглянуто сайту запитам.

У роботі з ключовими словами слід враховувати їх регіональну приналежність, сезонність.