**DMS**

Logistics là quá trình lên kế hoạch, áp dụng và kiểm soát các luồng dịch chuyển của hàng hóa hay thông tin liên quan tới nguyên nhiên liệu vật tư (đầu vào) và sản phẩm cuối cùng (đầu ra) từ thời điểm xuất phát tới điểm tiêu thụ.

Quản trị chuỗi cung ứng (Supply Chain Management) bao gồm tất cả những hoạt động quản trị logistics cũng như những hoạt động sản xuất và thúc đẩy sự phối hợp về quy trình và hoạt động của các bộ phận marketing, kinh doanh, thiết kế sản phẩm, tài chính, công nghệ thông tin. (<https://www.container-transportation.com/logistics-la-gi.html>)

Khái niệm chuỗi cung ứng rộng hơn, bao gồm cả logistics và quá trình sản xuất. Ngoài ra chuỗi cung ứng chú trọng hơn đến hoạt động mua hàng (procedurement) trong khi logistics giải quyết các vấn đề chiến lược, phối hợp giữa marketing và sản xuất.

Hệ thống quản lý phân phối (Distribution Management System) là phần mềm quản lý chuỗi cung ứng hàng hóa của doanh nghiệp, giúp họ quản lý các hoạt động phân phối hàng hóa ra thị trường, kiểm soát các kênh phân phối, quản lý nhân viên, người bán hàng, kiểm soát hàng hóa trong kho, hàng tồn kho, kế hoạch vận chuyển hàng hóa đến địa chỉ mua hàng,…

Ví dụ, Masan Group là một công ty lớn trong lĩnh vực kinh tế tư nhân Việt Nam, tập trung hoạt động trong ngành hàng tiêu dùng và tài nguyên của Việt Nam. Masan quản lý các nền tảng kinh doanh có quy mô lớn nhằm phát triển và khai thác các tiềm năng trong lĩnh vực tiêu dùng và tài nguyên.

Còn rất nhiều những tập đoàn bán lẻ khác có cùng cơ chế quản lý kinh doanh như Masan, mỗi tập đoàn như vậy đều cần có một hệ thống riêng để quản lý chuỗi cung ứng của mình.

Vì vậy DMS là một trong số những phần mềm quản lý doanh nghiệp có tính ứng dụng cao, phù hợp với mọi doanh nghiệp sản xuất và phân phối. Đối với các doanh nghiệp lớn (ví dụ Masan, VinGroup) có đội ngũ nhân viên bán hàng đông đảo và các kênh phân phối phức tạp, phần mềm DMS càng quan trọng và là công cụ không thể thiếu.

Làm thế nào để nắm được nhanh nhất xu thế, biến động của thị trường?

Làm thế nào để kiểm soát phân phối tốt, duy trì tồn kho ở mức tối ưu, tiết kiệm thời gian?

Tự động hóa bán hàng, tăng hiệu quả bán hàng cho đội ngũ nhân viên bán hàng?

Với hệ thống quản lý kênh phân phối (DMS), các doanh nghiệp lớn với vài trăm, hoặc vài nghìn nhân viên bán hàng , hàng chục nghìn điểm bán sẽ dễ dàng làm được việc này.

Lợi ích của phần mềm DMS:

Là công cụ *tự động hóa bán hàng*, giúp nhân viên bán hàng (salesman) tiết kiệm thời gian, tăng chất lượng chăm sóc khách hàng, tối ưu doanh thu. Mọi thông tin nhân viên bán hàng cần khi viếng thăm một điểm bán lẻ (rental outlet) bao gồm thông tin về khách hàng (customer), lịch sử mua hàng, các báo cáo bán hàng, thông tin sản phẩm, chương trình khuyến mãi, … có sẵn trong ứng dụng di động DMS để nhân viên có thể dễ dàng theo dõi tại điểm bán.

*Quản lý hiệu quả làm việc của NVBH*, nhà quản lý có thể nắm được lộ trình viếng thăm khách hàng của nhân viên trên bản đồ lịch sử checkin. Thông qua đó đánh giá được NVBH có đang tích cực ngoài thị trường hay không, khách hàng có được chăm sóc tốt hay không.

Đặc biệt, qua hệ thống, nhà quản trị sẽ biết được những cửa hàng nào đang liên tục phát sinh doanh số, những cửa hàng nào lâu rồi chưa phát sinh đơn hàng mới. Từ đó có thể sắp xếp, phân chia hợp lý các NVBH vào các tuyến bán hàng, thay đổi tần suất viếng thăm khách hàng cho phù hợp với thực tế, tránh lãng phí nguồn lực đồng thời có thể chăm sóc được kỹ hơn các cửa hàng trọng tâm.

Cập nhật thị trường là điểm nổi bật của phần mềm DMS. Giúp các doanh nghiệp phân phối có thể kiểm soát được hàng tồn tại từng điểm bán lẻ, đại lý nhằm đưa ra kế hoạch sản xuất, điều phối hàng hóa phù hợp. (<https://mobiwork.vn/phan-mem-dms-la-gi/>)

**Công nghệ**

ReactJS

Ngày nay ReactJS đã trở nên rất phổ biến bởi những tính năng linh hoạt và đơn giản với hơn 1.300.000 developer và hơn 94.000 trang web đang sử dụng ReactJS (<https://topdev.vn/blog/reactjs-nhung-dieu-ban-can-phai-biet/>).

ReactJS là một thư viện JavaScript mã nguồn mở được thiết kế bởi Facebook để tạo ra những ứng dụng web hấp dẫn, nhanh và hiệu quả với source code tối thiểu. Mục đích cốt lõi của ReactJS không chỉ khiến cho trang web phải thật mượt mà còn phải nhanh, khả năng mở rộng cao và đơn giản. Trên website chính thức của React tổng quan rằng: ReactJS – “A JavaScript for library for building user interface”, tức là React sinh ra để phục vụ tầng View, tập trung vào xây dựng giao diện.

Tư tưởng của ReactJS là xây dựng lên các component có tính tái sử dụng, dễ dàng cho việc chia nhỏ vấn đề, kiểm thử, giúp chúng ta dễ dàng quản lý, mở rộng hệ thống. Đặc tính của ReactJS là luôn giữ các component ở trạng thái stateless nhiều nhất có thể, khiến ta dễ dàng quản lý nó. Bản thân các component này không có trạng thái, nó nhận đầu vào từ bên ngoài và chỉ hiển thị ra dựa vào các đầu vào đó, điều này cũng lý giải tính tái sử dụng (reuse) và tiện lợi trong kiểm thử (testing) của ReactJS.

**Virtual DOM**

(DOM image) <https://viblo.asia/p/hieu-sao-ve-virtual-dom-trong-reactjs-bWrZngDblxw>

Trước tiên, DOM là viết tắt của Document Object Model là một chuẩn được định nghĩa bởi W3C dùng để truy xuất và thao tác trên code Html bằng các ngôn ngữ lập trình thông dịch (scripting language) như JavaScript.

Trong hình vẽ có thể thấy tất cả các thẻ Html sẽ được quản lý trong các đối tượng *document*, thẻ cao nhất là thẻ html, tiếp đến là phân nhánh body và head, trong head có thẻ style, title, … trong body chưa các thẻ html. Như vậy thông qua DOM, JavaScript có thể thay đổi tất cả các phần tử Html, các thuộc tính Html, Css, loại bỏ hoặc thêm các thành phần và thuộc tính Html mới, tạo ra các sự kiện khi tương tác,… Tức là bạn có thể thay đổi cả trang web với DOM.

Khi DOM thay đổi, trình duyệt phải tính toán lại Css và dựng lại trang web, điều này sẽ tốn thời gian, nhất là với những ứng dụng Single Page Application, việc sửa đổi DOM là liên tục không ngừng nghỉ. Hay khi xử lý các sự kiện (event) như click, submit, … DOM sẽ tìm tất cả các node liên quan đến sự kiện và cập nhật nếu thấy nó cần thiết. Vậy thì có cần thiết khi phải tìm tất cả các node liên quan không? Hay sẽ hiệu quả hơn khi chỉ tìm node nào cần cập nhật.

Virtual DOM xuất hiện để giải quyết những vấn đề này. Virtual DOM gắn với ReactJS, thay vì xử lý DOM Tree thủ công, chúng định nghĩa các component trông giống DOM (vì vậy mà cú pháp JSX nhìn rất giống Html), còn ReactJS sẽ thực hiện công việc ở tầng thấp hơn. Tổng quát thì Virtual DOM là một định dạng dữ liệu JavaScript nhẹ dùng để thể hiện nội dung của DOM tại một thời điểm nhất định nào đó. Nó có tất cả các thuộc tính giống như DOM nhưng không có khả năng tương tác lên màn hình như DOM.

Sự đặc biệt của Virtual DOM nằm ở cơ chế Snapshots và Diffing. Khi cần cập nhật phần tử giao diện, React sẽ lấy một snapshot của Virtual DOM (có thể hiểu là bản ghi trạng thái ngay lúc đó), sử dụng snapshot này để so sánh với một Virtual DOM trước khi thực hiện thay đổi.

(Virtual dom image)

Quá trình tiếp theo, React sẽ sử dụng thuật toán Diffing để so sánh và đối chiếu để biết được cập nhật ở đâu sau đó cập nhật nó mà bỏ qua những thành phần không liên quan. Chỉ những thay đổi này được cập nhật trên DOM và sẽ làm màn hình thay đổi.

**Single Page Application**

Với ReactJS, ta dễ dàng tạo ra một Single Page Application (SPA). Khác với những ứng dụng web truyền thống Single Page Application có một trang gốc và trong trang gốc đó, chúng ta có thể tải nhiều trang con (tương ứng với các thành phần của trang gốc) mà không gây bất kì ảnh hưởng gì đến trang gốc. Trong khi các ứng dụng web truyền thống phải tải lại toàn bộ trang khi chúng ta tương tác với trang web thì Single Page Application chỉ load phần trang cần thiết. Các thành phần chung như header, footer, menu, side bar,… thường xuất hiện ở nhiều trang của ứng dụng sẽ được Single Page Application load một lần duy nhất ở trang gốc.

(Single Page Application image)

Do vậy Single Page Application mang lại nhiều ưu điểm. Thứ nhất việc render html ở server sẽ cực kì tốn tài nguyên nếu trang web có nhiều người dùng, với Single Page Application điều này chỉ xảy ra lần đầu tiên khi người dùng truy cập trang chủ, còn sau đó việc render sẽ do client đảm nhiệm.

Thứ hai Single Page Application tách biệt front-end và back-end, SPA giao tiếp với server chủ yếu qua Json REST Api giúp cho dữ liệu gửi và trả giữa client và server giảm đến mức tối thiểu. Việc phát triển, kiểm thử cũng có thể độc lập giữa front-end và back-end.

Thứ ba, trong suốt quá trình sử dụng, chỉ có dữ liệu là được truyền qua lại giữa client và server, còn các tài nguyên tĩnh (Html, Css, Script, …) chỉ được tải một lần duy nhất, vì vậy sẽ giảm thiểu băng thông cho server.

Thứ tư, Single Page Application giúp tăng trải nghiệm người dùng, là một ứng dụng web nhưng người dùng tương tác giống như một ứng dụng cho Desktop vậy.

**React Router**

React Router là thư viện định tuyến (routing) chuẩn của React, nó giúp giao diện của ứng dụng đồng bộ với URL trên trình duyệt. React Router cho phép định tuyến luồng dữ liệu (data-flow) trong ứng dụng web một cách rõ ràng.

Với React Router, việc xây dựng Single Page Application trở nên vô cùng dễ dàng.

(react router image)

Ảnh trên thể hiện cấu hình cơ bản của React Router với đường dẫn (/path) và Route và giao diện tương ứng.

**Redux**

Một ứng dụng web sẽ nhận dữ liệu từ phía máy chủ (back-end), hay nhận những thao tác của người dùng (input, click, submit, …), những thứ này chúng ta gọi đó là trạng thái (state) của ứng dụng. Nếu biết được trạng thái của ứng dụng tại một thời điểm nào đó, chúng ta sẽ biết vào thời điểm đó ứng dụng đã nhận dữ liệu nào, những thao tác nào đã được người dùng truyền lên.

Ví dụ: Khi chúng ta click vào nút Back / Forward trên trình duyệt thì mỗi trang là một trạng thái của ứng dụng.

Như đã trình bày ở trên, ReactJS xây dựng lên các Single Page Application, tức chỉ render một trang, và tất cả các thành phần của ứng dụng sẽ được lưu trữ trong đó. Vì thế, nếu ứng dụng phức tạp lên theo thời gian, các component sẽ nhiều lên, và việc quản lý các state của chúng cũng ngày một lớn dần. Giao diện ứng dụng (UI) cũng trở nên phức tạp vì chúng ta cần quản lý các công việc active Routes, selected tabs, spinners, pagination, …vv

Ví dụ trong ReactJS để truyền dữ liệu giữa các component anh em, một state phải tồn tại (live) trong một component cha, một phương thức (method) để update chính state này được cung cung cấp bởi component cha, từ đây sẽ truyền xuống props của các component con. Do vậy nếu một state phải được chia sẻ giữa các component cách khá xa nhau trong một tree component thì state này sẽ phải được truyền từ một component đến một component khác cho đến khi nó đến được nơi mà nó được gọi.

(redux explain image)

Như trong hình vẽ trên, giả sử nếu có một sự kiện ở node d3 kích hoạt muốn thay đổi state d4 thì luồng dữ liệu sẽ được truyền từ node d3 trở về node gốc là a, sau đó từ node a lại truyền data đến các node con. Thứ tự truyền: d3 – c2 – b1 – a – b2 – c4 – d4. Tương tự nếu muốn thay đổi state ở c3 thì thứ tự truyền là: d3 – c2 – b1 – a – b2 – c3.

Điều này làm cho bộ phận quản lý state trong ứng dụng trở nên phức tạp và bừa bộn, do vậy ta cần một công cụ quản lý trạng thái (state management tool) như Redux. Giải pháp Redux đưa ra như sau:

(redux solve image)

Quay lại ví dụ ở trên thì ta cần map sự kiện từ node d3 về store của Redux rồi ở node d4, c3 cần connect với store và cập nhật dữ liệu thay đổi. (<https://insights.innovatube.com/redux-th%E1%BA%ADt-l%C3%A0-%C4%91%C6%A1n-gi%E1%BA%A3n-ph%E1%BA%A7n-1-76a3fa2c31ab>)

**Nguyên lý vận hành – Redux**

Cách Redux hoạt động khá đơn giản. Redux có một store lưu trữ toàn bộ state của ứng dụng. Mỗi component có thể truy cập trực tiếp đến state được lưu trữ thay vì phải truyền từ component này qua component khác.

(redux workflow gif)

Redux có 3 thành phần là Actions, Store và Reducers.

Actions đơn giản là các events, mô tả những gì xảy ra như là cách mà chúng ta gửi dữ liệu từ ứng dụng đến Redux store, dữ liệu có thể đến từ sự tương tác của user và ứng dụng, API calls hoặc khi submit một form,…

Tuy nhiên action lại không chỉ rõ phần state nào thay đổi, việc này do Reducer đảm nhiệm. Reducer nhận vào một state cũ và action được gửi lên sau đó trả về một state mới.

|  |
| --- |
| (previousState, action) => newState |

Những states này được lưu như những objects và chúng định rõ cách state của một ứng dụng thay đổi trong việc phản hồi một action gửi đến store.

Store là nơi lưu lại các state của ứng dụng và nó là duy nhất trong bất kì một ứng dụng Redux nào.

**Middleware**

Một ứng dụng thực tế đòi hỏi có những thao tác xử lý cần thời gian để phản hồi (các thao tác bất đồng bộ lấy dữ liệu từ api hay các thao tác đọc ghi file hay đọc cookie từ trình duyệt, …), các thao tác như vậy gọi là **side effect**. Để giải quyết được các side effect này, trong Redux ta cần thực hiện nó ở middleware.

Middleware là những đoạn mã trung gian nằm giữa các request và response. Nó nhận các request, thi hành các mệnh lệnh tương ứng trên request đó. Sau khi hoàn thành nó trả về hoặc chuyển kết quả cho một Middleware khác.

(middleware example image)

Ở Redux, Middleware giải quyết vấn đề hơi khác nhưng có định nghĩa tương tự. Đó là middleware cho phép chúng ta can thiệp vào giữa thời điểm dispatch một action và thời điểm action đó đến được reducer. Chúng ta có thể thấy sự thay đổi của luồng hoạt động khi có sử dụng middleware ở hình dưới.

(middleware workflow without middleware gif)

Ta có thể tự viết một middleware hoặc có thể dùng những thư viện middleware được xây dựng sẵn. Hiện tại có một vài thư viện middleware cho Redux, ví dụ như redux-thunk, redux-saga, redux-observable, … mỗi thư viện có phương pháp giải quyết vấn đề side-effect riêng. Trong project này, em sử dụng redux-saga.

**Redux-saga**

Redux-saga là một thư viện hỗ trợ việc xử lý side-effect trong ứng dụng React/Redux (ví dụ như xử lý bất đồng bộ khi load dữ liệu,…) và làm cho các ứng dụng này trở nên đơn giản hơn. Bằng cách sử dụng Generator Function, redux-saga giúp ta viết code bất đồng bộ (async code) nhìn giống như là đồng bộ (synchronos).

**Generator Function** là function có khả năng hoãn lại quá trình thực thi mà vẫn giữ nguyên được ngữ cảnh của function. Khác với function bình thường là thực thi và trả về kết quả, thì Generator function có thể thực thi, tạm dừng trả về kết quả và thực thi tiếp (bằng cách sử dụng từ khóa Yield). Nếu như function bình thường khi được gọi sẽ thực thi hết tất cả các câu lệnh trong hàm thì Generator function có khả năng tạm ngưng trước khi hàm kết thúc và có thể tiếp tục chạy tại một thời điểm khác. Chính chức năng này giúp ta giải quyết được vấn đề bất đồng bộ, hàm sẽ dừng và đợi async chạy xong rồi tiếp tục thực thi.

**Nguyên lý hoạt động Redux-saga**

Redux-saga cung cấp các hàm helper effect, các hàm này sẽ trả về một effect object chứa các thông tin chỉ dẫn middleware của Redux có thể thực hiện tiếp các hành động khác. Các hàm helper effect sẽ được thực thi trong các generator function. Ví dụ một số helper effect trong Redux-saga:

takeEvery(): thực thi và trả về kết quả của một action được gọi

takeLastest(): nếu ta thực hiện một loạt các actions, nó sẽ chỉ thực thi và trả về kết quả của action cuối cùng.

put(): dispatch một action.

call(): gọi một function. Nếu nó trả về một promise, sẽ tạm dừng saga cho đến khi promise được giải quyết.

|  |
| --- |
| //Thực hiện function fetchPersonListSaga khi nhận được action FETCH\_PERSON\_LIST  yield takeEvery(FETCH\_PERSON\_LIST, fetchPersonListSaga);  //Thực hiện dispatch action pushSuccessNotification  yield put(pushSuccessNotification(sequence, "Saved successfu")); |

**Material UI**

**Material Design**

Material UI là một thư viện các React Component và được tích hợp thêm cả Google’s Material Design. Trước tiên, chúng ta sẽ tìm hiểu về nguyên lý Material Design.

Material Design là phong cách thiết kế áp dụng chủ yếu trong thiết kế ứng dụng Web, ứng dụng Mobile và đã trở thành một xu hướng phổ biến hiện nay. Đối với những Designer thiết kế UX/UI (giao diện / trải nghiệm người dùng), hay các lập trình viên Front-end thì thuật ngữ Material Design không còn xa lạ. Có rất nhiều ứng dụng nổi tiếng thiết kế theo phong cách Material Design như các ứng dụng của Google (Google+, Gmail, Google Maps, …), Evernote, ePay, …

(material design image)

Material Design là hình thức phát triển cao hơn của Flat Design (thiết kế phẳng), tuy nhiên thay vì cảm giác “phẳng lì” trên toàn bộ giao diện, Material Design là những lớp xếp chồng lên nhau, tạo chiều sâu, điểm nhấn hơn những thiết kế phẳng thông thường. Material Design chủ yếu tập trung vào những đường nét đơn giản, sử dụng những gam màu đậm, nổi bật, đồng thời cũng thường sử dụng những yếu tố đồ họa có cảm giác 3D, có hiệu ứng “nổi lên” (float) trên giao diện. Ngoài ra, thiết kế này còn bao gồm những chuyển động tự nhiên, tất cả những điều này đều nhằm mục đích mang lại cho người dùng trải nghiệm mới mẻ, thú vị và gần gũi hơn.

3 yếu tố căn bản của Material Design:

Thứ nhất: Không gian

Không gian dưới lớp kính màn hình thiết bị được mô phỏng như một không gian 3 chiều Oxyz với chiều sâu là trục Oz. Để tạo chiều sâu cho thiết kế, designer cần điều chỉnh ánh sáng một cách phù hợp.

(material design space image)

Thứ hai: Ánh sáng

Ánh sáng là yếu tố môi trường được sử dụng nhằm thể hiện tính 3 chiều của không gian. Hệ quả của ánh sáng là hiệu ứng đổ bóng (Drop Shadow), sẽ phân định vị trí các lớp Material trong không gian theo trục Oz. Có hai loại nguồn sáng được kết hợp

Nguồn sáng chiếu trực tiếp: nguồn sáng này rất quan trọng, giống như nguồn sáng đèn pin, nó mang lại hiệu ứng đổ bóng mạnh và sắc nét.

Ánh sáng môi trường: ánh sáng này nhẹ nhàng và không rõ nguồn, tạo viền bóng nhẹ xung quanh.

Thông thường, Material Design kết hợp cả hai nguồn sáng, mang đến hiệu ứng bóng tổng hợp, mô phỏng không gian thực tế.

(material design light image)

Thứ ba: Material

Là những mặt phẳng có độ dày đồng nhất 1dp (1 in ≈ 160 dp) và nằm song song với mặt phẳng Oxy.

(material design material image)

Các mặt phẳng Material sắp xếp chồng lên nhau theo trục Oz. Thông qua việc thay đổi kích thước của bóng, ta sẽ dễ dàng mô tả vị trí tương đối của mỗi lớp so với lớp khác.

Để có một thiết kế ấn tượng với Material Design cần chú ý một vài hiệu ứng và chi tiết:

Hiệu ứng tự nhiên: ví dụ khi bạn nhấn chọn một thành phần, hiệu ứng sóng trên màn hình sẽ tỏa ra tự vị trí ngón tay bạn chứ không phải từ một hướng cố định.

Hiệu ứng bề mặt: khi chuyển trang, các thành phần phải chuyển động một cách tự nhiên và liên tục chứ không biến mất

Có thứ tự: những thành phần ở sau sẽ xuất hiện trước, thành phần lớn hơn sẽ xuất hiện trước, thành phần quan trọng hơn sẽ xuất hiện trước.

Thống nhất: chuyển động của những Material phải thống nhất từ cùng một hướng, tạo sự đồng đều cho tổng thiết kế.

**Material UI**

Như đã trình bày ở trên, Material UI là một thư viện các React component tích hợp thêm Google’s Material Design. Material UI cung cấp khá đầy đủ các component để có thể tạo ra một trang web một cách nhanh chóng hơn mà không phải ngồi chỉnh Css từng tí một.

Ví dụ để tạo ra các nút bấm như hình dưới, ta chỉ việc sử dụng Button component mà Material UI cung cấp.

(material ui button image)

Việc thêm màu sắc là vô cùng đơn giản với những thuộc tính được định nghĩa sẵn, màu sắc là chuẩn theo thiết kế Material Design.

Với Material UI, chúng ta còn dễ dàng chia bố cục và responsive trang web. Grid component sẽ chia màn hình theo bố cục 12 cột, 5 loại màn hình theo kích cỡ (xs, sm, md, lg, xl)

(responsive web image)

Cùng một nội dung nhưng khi được hiển thị trên các màn hình khác nhau sẽ hiển thị theo cách khác nhau, đảm bảo sự thuận tiện nhất cho người dùng.

Thuật ngữ “Responsive Design” ám chỉ cách thiết kế trang web hiển thị tương thích với mọi kích thước thiết bị, tức là bố cục trang web sẽ tự đáp ứng theo hành vi người dùng và môi trường hiển thị. Môi trường này chính là kích thước của trình duyệt, kích thước hoặc hướng xoay thiết bị. Thiết kế Responsive không chỉ giúp cho người dùng có một trải nghiệm thú vị hơn khi truy cập website, mà còn giúp chủ sở hữu dễ dàng quản lý các trang web của mình hơn.

Ngoài ra Material UI cũng có sẵn kho Icon khổng lồ trên đầy đủ các lĩnh vực giúp chúng ta dễ dàng chọn ra icon đẹp và phù hợp nhất với mỗi nội dung trên trang web.

**Redis**

Redis là viết tắt của Remote Dictionary Server (máy chủ điều khiển từ xa), lưu trữ dữ liệu dưới dạng KEY-VALUE trong bộ nhớ, có mã nguồn mở và có tốc độ truy cập nhanh để dùng làm cơ sở dữ liệu, bộ nhớ đệm (cache), trình chuyển tiếp tin nhắn và danh sách tác vụ chờ xử lý.

Redis hiện cung cấp thời gian phản hồi ở tốc độ chưa đến một mili giây, giúp thực hiện hàng triệu yêu cầu mỗi giây cho các ứng dụng thời gian thực trong lĩnh vực Trò chơi, Quảng cáo, Dịch vụ tài chính, Chăm sóc sức khoe, IoT, … Cụ thể, Redis thường được chọn sử dụng cho các hoạt động lưu trữ bộ nhớ đệm, quản lý phiên, trò chơi, bảng xếp hạng, phân tích thời gian thực, dữ liệu không gian địa lý, ứng dụng đặt xe, trò chuyện / nhắn tin, phát trực tiếp nội dung đa phương tiện, …

Redis là một cơ sở dữ liệu NoSQL. NoSQL là một dạng cơ sở dữ liệu phi quan hệ, sử dụng nhiều loại mô hình dữ liệu đa dạng để truy cập và quản lý dữ liệu trong bộ nhớ và tìm kiếm, NoSQL được tối ưu hóa dành riêng cho các ứng dụng yêu cầu mô hình dữ liệu linh hoạt có lượng dữ liệu lớn và độ trễ thấp, đạt được bằng cách giảm bớt một số hạn chế về tính nhất quán của dữ liệu.

(redis image)

Cách thức hoạt động của Redis

Toàn bộ dữ liệu của Redis nằm trong bộ nhớ (RAM), khác với những loại cơ sở dữ liệu thông thường khác lưu dữ liệu trên ổ đĩa hoặc ổ SSD. Phần lớn các tác vụ trên cơ sở dữ liệu truyền thống đều yêu cầu truy cập qua lại tới ổ đĩa, do vậy sẽ tốn thời gian tìm kiếm, dữ liệu của Redis nằm trên RAM nên sẽ không mất thời gian này. Do đó Redis có thể hỗ trợ thêm khá nhiều tác vụ và có thời gian phản hồi nhanh hơn. Hiệu suất của Redis rất tốt với các tác vụ đọc ghi thông thường mất chưa đầy một mili giây và hỗ trợ hàng triệu tác vụ mỗi giây.

Khác với các cơ sở dữ liệu quan hệ như MySQL hay PostgreSQL, Redis không có bảng. Redis lưu dữ liệu dưới dạng KEY-VALUE và hỗ trợ nhiều cấu trúc dữ liệu cơ bản như hash, list, set, sorted set, string, … Bên cạnh đó Redis có 2 background threads chuyên làm nhiệm vụ định kì ghi dữ liệu trên đĩa cứng, cơ chế backup này giúp cho Redis có độ bảo mật và sửa lỗi cao.

Các ứng dụng phổ biến

Lưu trữ bộ nhớ đệm (caching)

Sử dụng bộ nhớ đệm để giảm độ trễ khi truy cập dữ liệu, tăng năng suất và giảm tải cho cơ sở dữ liệu và ứng dụng của bạn. Redis có thể phục vụ những dữ liệu thường xuyên được yêu cầu với thời gian phản hồi chưa đến một mili giây và cho phép dễ dàng thay đổi quy mô nhằm đáp ứng mức tải cao hơn mà không cần tốn kém chi phí vào back-end.

Bảng xếp hạng game

Redis là giải pháp hay được các nhà phát triển game dùng để xây dựng bảng xếp hạng theo thời gian thực (real-time leaderboard). Sử dụng cấu trúc dữ liệu Sorted Set của Redis, cấu trúc dữ liệu này đảm bảo tính duy nhất của các thành phần trong khi vẫn duy trì danh sách được sắp xếp theo điểm số của người dùng. Danh sách cập nhật mỗi khi điểm số người dùng thay đổi.

(realtime leaderboard image)

Lưu trữ phiên (session)

Các nhà phát triển ứng dụng thường sử dụng Redis để lưu trữ, quản lý phiên cho các ứng dụng quy mô Internet. Quản lý dữ liệu phiên chẳng hạn như hồ sơ người dùng, thông tin xác thực đăng nhập (token), trạng thái phiên, …

Trò chuyện, nhắn tin, hàng chờ xử lý tác vụ

Redis hỗ trợ Pub/Sub (là cấu trúc gửi – nhận tin nhắn mà người gửi và người nhận không biết nhau) với nhiều cấu trúc dữ liệu như list, sorted set, hash. Điều này cho phép Redis hỗ trợ những chat rooms hiệu năng cao, luồng tin nhắn theo thời gian thực.

So sánh Redis và một loại cơ sở dữ liệu quan hệ thông thường (MySQL)

|  |  |  |
| --- | --- | --- |
| Cấu trúc cơ sở dữ liệu | Lưu trữ dạng KEY-VALUE | Lưu trữ dạng bảng |
| Lưu dữ liệu trong RAM, là máy chủ cấu trúc dữ liệu vì các key có thể chứa string, hash, list, set và sorted set | MySQL cung cấp một máy chủ cơ sở dữ liệu quan hệ SQL rất nhanh, đa luồng, đa người dùng, mạnh mẽ |
| Ưu điểm | -Dễ cài đặt, sử dụng, deploy, maintain, …  -Lưu trữ dữ liệu trong bộ nhớ nên cho hiệu năng cao và tốc độ nhanh  -Mã nguồn mở, ổn định, chi phí hiệu quả  -Khả năng mở rộng cao, hỗ trợ sao lưu vào đĩa cứng  -Cấu trúc dữ liệu đa dạng | -Cơ sở dữ liệu quan hệ mã nguồn mở được sử dụng rộng rãi  -Dễ sử dụng, khả năng tương thích cao, hỗ trợ đa nền tảng, cộng đồng phát triển mạnh mẽ  -Hỗ trợ index và full-text searching |
| Trường hợp nên sử dụng | -Có dữ liệu dạng KEY-VALUE  -Cần lưu cache  -Cần hiệu năng cao  -Kích thước dữ liệu ổn định | -Cần cơ sở dữ liệu quan hệ  -Khi có các hoạt động phân tán  -Cần bảo mật cao và hoạt động đơn giản  -Không sử dụng khi dữ liệu lớn dần và không thể cache hết lên bộ nhớ |
| Nhược điểm | -Redis sử dụng RAM nên khi lượng file cache lớn sẽ dẫn đến thiếu RAM cho server  -Không thể truy vấn trực tiếp các object | -Nhiều vấn đề về lưu trữ procedures và triggers  -Xử lý transaction không hiệu quả |

**Postgresql**

**Microservice**

**Nginx**

**Azure**

**Chương**

Khảo sát thực tế về sự cần thiết của một hệ thống quản lý phân phối

Khi chưa sử dụng giải pháp DMS, có rất nhiều lỗ hổng mà các doanh nghiệp phân phối phải đối mặt như sau đây:

Tồn kho

Không kiểm soát được hàng tồn kho của nhà phân phối, cửa hàng để tối ưu hóa chuỗi cung ứng và các chiến lược tiêu thụ hàng. Công ty sẽ không có sự chủ động trong việc cung ứng và tiêu thụ hàng trên thị trường, đặc biết là loại mặt hàng có thời hạn sử dụng ngắn, sẽ không kiểm soát được hàng đang ở đâu trong chuỗi cung ứng của mình.

Bao phủ

Việc quản lý và tối ưu tuyến bán hàng theo từng ngành hàng, khu vực, đội bán hàng để gia tăng độ bao phủ cũng là vấn đề.

Doanh số

Doanh số luôn là vấn đề đau đầu của các nhà quản lý, họ sẽ không có đầy đủ thông tin để triển khai các chương trình bán hàng một cách có hiệu quả.

Hiệu suất

Làm thế nào để biết đội ngũ bán hàng của công ty có hoạt động với hiệu suất 100% hoặc hơn không? Làm sao để thiết kế các KPI và kiểm soát hiệu quả giúp họ có động lực bán hàng, tăng hiệu suất làm việc? Đội ngũ nhân viên bán hàng có được theo sát và huấn luyện các kỹ năng bán hàng hay không?

Số ảo

Tình trạng số ảo khá phổ biến trong bán hàng và phân phối, các số liệu ảo về điểm bán, doanh số, khuyến mãi, tồn kho, … luôn là nỗi lo lắng của các nhà quản lý vì điều này ảnh hưởng lớn đến việc ra quyết định của họ.

Tích hợp

Trong mô hình phân phối hiện đại và đa kênh, các công ty cung cấp, phân phối hàng hóa ra thị trường cũng sẽ đối diện với vấn đề tiếp xúc và xử lý thông tin với nhiều hệ thống, hình thức dữ liệu khác nhau. Làm thế nào để thiết kế mô hình dữ liệu chung đồng nhất nhưng vẫn đảm bảo các quy trình, điểm đặc thù của mỗi kênh, mỗi hệ thống, đảm bảo tính xuyên suốt, đồng nhất và kịp thời của thông tin.

Còn sau khi áp dụng giải pháp quản lý hệ thống phân phối (DMS) thì các nhà quản lý có những lợi ích sau:

Dữ liệu thật

Với các quy trình và chức năng được thiết kế chặt chẽ và hỗ trợ hiệu quả các đối tượng sử dụng trong hệ thống sẽ giúp công ty có được bộ dữ liệu đầy đủ và thật từ dữ liệu thị trường, điểm bán, tuyến bán hàng đến tồn kho, sell-in, sell-out, khuyến mãi, trưng bày, … hạn chế và ngăn chặn tối đa các trường hợp thay đổi dữ liệu bán hàng. Dữ liệu thất này sẽ giúp công ty ra quyết định và triển khai các chiến lược bán hàng hiệu quả và chính xác hơn.

Chủ động và làm chủ thị trường

Doanh nghiệp hoàn toàn chủ động trong triển khai chiến lược bán hàng, chủ động khi có sự thay đổi, biến động về nhân sự trong hệ thống phân phối và dễ dàng thiết lập các nhà phân phối mới, khu vực mới cho các hệ thống bán hàng mới.

Gia tăng giá trị thương hiệu

Việc triển khai và quản lý hiệu quả các chương trình bán hàng giúp thu hút và thúc đẩy tiêu thụ từ người tiêu dùng làm gia tăng sự trung thành từ người tiêu dùng cũng như sự hợp tác từ nhà phân phối vì họ thấy được lợi ích thật sự.

**Kiot Viet**

Hiện nay trên thị trường có một số ít sản phẩm phần mềm quản lý hệ thống phân phối cho doanh nghiệp (DMS). Sử dụng phần mềm này thường là những người quản lý của doanh nghiệp, công ty, hãng phân phối. Do tính chất đặc thù như vậy mà những người tiếp cận và sử dụng phần mềm kiểu này này khá ít, và phần mềm quản lý hệ thống phân phối DMS cũng không phổ biến rộng rãi cho mọi đối tượng. Một số phần mềm DMS có thể kể đến như Adaline, BS Silver, SSE, Perfect Warehouse, GM Sales, KiotViet, Suno, Sapo, … Mỗi phần mềm sẽ cung cấp những tính năng đặc biết riêng, tính phí hoặc không tính phí, dùng offline hoặc online, từng nền tảng (Windows, Android, iOS) và ưu nhược điểm khác nhau. Các phần mềm offline như Adaline, BS Silver, Perfect Warehouse, … sẽ có ưu điểm là miễn phí, tốc độ nhanh, bảo mật cao. Tuy nhiên chỉ phù hợp với những cửa hàng bán lẻ, vừa và nhỏ, tính năng hạn chế, không có khả năng mở rộng. Sapo hay KiotViet là các phần mềm quản lý phân phối phổ biến hơn khi cung cấp cho người dùng nhiều tính năng với mức giá rẻ, giao diện làm việc trực tuyến, nhanh, dễ sử dụng, … Các phần mềm kiểu này phù hợp với những công ty, doanh nghiệp phân phối lớn hơn khi khả năng nắm bắt biến động thị trường là cần thiết để đưa ra những chiến lược kinh doanh kịp thời.

Ở Việt Nam hiện nay, KiotViet là phần mềm quản lý bán hàng phổ biến nhất với hơn 100,000 cửa hàng đang sử dụng và hơn 5,000 cửa hàng mới mỗi tháng. Đơn giản, dễ dùng, tiết kiệm chi phí và phù hợp với hơn 15 ngành hàng khác nhau.

KiotViet sẽ cung cấp cho người dùng một khung tính năng cố định ứng với từng ngành hàng khác nhau. Một phần mềm quản lý hoàn chỉnh có hai phần, một là trang quản lý, hai là trang bán hàng. Trang quản lý của KiotViet có các tính năng về quản lý hàng hóa (quản lý danh mục, thiết lập giá, …), quản lý giao dịch (lên hóa đơn, đặt hàng, xuất hàng, …), quản lý nhân viên, quản lý khách hàng và nhà cung cấp, xuất báo cáo (cuối ngày, theo tuần, theo tháng ,…). Trang bán hàng của KiotViet cung cấp một giao diện đơn giản tương tự như giao diện thanh toán bán hàng ở các cửa hàng bán lẻ hay siêu thị nhỏ. Đơn giản chỉ chọn sản phẩm, tính tiền, thanh toán và xuất hóa đơn cho khách.

Trong ứng dụng quản lý phân phối của chúng em, ngoài những tính năng cơ bản, cần thiết như quản lý sản phẩm, quản lý hàng trong kho, quản lý giao dịch, quản lý nhân viên, quản lý phân quyền động thì ứng dụng một tính năng mới. Đó là ứng dụng cung cấp chức năng gợi ý, lên lịch trình viếng thăm cho các nhân viên bán hàng (salesman), lưu lại lịch sử check-in của nhân viên bán hàng. Nhân viên bán hàng là người hàng ngày sẽ thăm một số cửa hàng bán lẻ (rental outlet), kiểm tra lượng hàng hóa bán ra, từ đó lên hóa đơn để nhập hàng hóa mới về. Để các nhân viên bán hàng này hoạt động hiệu quả nhất, cần phân công họ vào các tuyến bán hàng phù hợp, để họ vừa có thể di chuyển một cách dễ dàng giữa các cửa hàng bán lẻ, vừa đến được nhiều cửa hàng nhất có thể. Chi tiết về tính năng mới này sẽ được trình bày trong chương sau của đồ án.

**Tổng quan chức năng**

**Biểu đồ usecase tổng quan**

(biểu đồ usecase tổng quan image)

Hai tác nhân chính trong biểu đồ usecase tổng quan là người quản lý (manager) và nhân viên bán hàng (salesman). Người quản lý ở đây là từ dùng chung, đại diên cho những người quản lý nghiệp vụ riêng (quản lý sản phẩm, quản lý hàng tồn kho, quản lý kho, quản lý giao dịch, quản lý tuyến bán hàng). Người quản lý sau khi đăng nhập vào hệ thống có thể thực hiện các nghiệp vụ quản lý như thêm / sửa / xóa sản phẩm, người dùng, … Nhân viên bán hàng sau khi đăng nhập có thể xem được lịch trình di chuyển của mình, lên hóa đơn mua hàng, check-in.

**Các biểu đồ usecase phân rã từng chức năng của hệ thống**

(account management image)

(facility management image)

(import management image)

(product management image)

(salesman image)

(salesroute management image)

Các biểu đồ hoạt động mô tả cách vận hành của từng chức năng

(account management image)

(add customer store image)

(add order image)

(add schedule image)

(phan quyen dong image)

(set price image)

Thiết kế cơ sở dữ liệu của ứng dụng

(order database image)

(product database image)

(salesroute database image)

(security database image)

**Bài toán phân cụm**

Trước tiên, ta cần có khái niệm về phân cụm (clustering). Clustering là cách nhóm một tập hợp các đối tượng (objects) dựa trên các đặc điểm, thuộc tính của chúng, tập hợp chúng lại theo từng nhóm dựa trên độ giống nhau (similarities).

Phân cụm là kỹ thuật quan trọng trong xử lý dữ liệu, nó thuộc lớp các phương pháp học không giám sát (Unsupervised Learning) trong học máy (Machine Learning). Giải thích dễ hiểu hơn thì phân cụm là các quy trình tìm cách nhóm các đối tượng đã cho vào các cụm (clusters), sao cho các đối tượng trong cùng một cụm tương tự nhau (similar) và các đối tượng khác cụm thì không tương tự nhau (dissimilar).

Mục đích của phân cụm là tìm ra bản chất bên trong các nhóm của dữ liệu và có thể áp dụng trong rất nhiều lĩnh vực

Trong marketing, xác định các nhóm khách hàng (khách hàng tiềm năng, khách hàng giá trị, phân loại và dự đoán hành vi khách hàng, …) sử dụng sản phẩm hay dịch vụ của công ty để giúp công ty có chiến lược kinh doanh hiệu quả hơn.

Trong sinh học, phân nhóm động vật và thực vật dựa vào các thuộc tính của chúng.

Trong bảo hiểm, tài chính: phân nhóm các đối tượng sử dụng bảo hiểm và các dịch vụ tài chính, dự đoán xu hướng của khách hàng, phát hiện gian lận tài chính, …

Trong phân tích dữ liệu web: phân loại tài liệu, phân loại người dùng web, …

**K-Means**

K-Means là thuật toán quan trọng và được sử dụng phổ biến trong kỹ thuật phân cụm. Tư tưởng chính của thuật toán K-Means là tìm cách phân nhóm các đối tượng đã cho vào K cụm (K là số cụm được xác định trước, K nguyên dương) sao cho khoảng các từ các đối tượng đến tâm nhóm (centroid) là nhỏ nhất.

Bài toán: Cho N điểm trên không gian 2 chiều. Làm thế nào để phân chia thành K nhóm mà các điểm trong một nhóm có khoảng cách gần trọng tâm của nhóm hơn so với khoảng cách đến trọng tâm của bất kì 1 nhóm nào khác.

Input

Cho N điểm. Mỗi điểm có dạng (x, y)

K là số nhóm (cụm) (K <= N)

Output

Danh sách K nhóm và các điểm của mỗi nhóm

Mô tả thuật toán K-Means

(k-means algoritms flow image)

Thuật toán K-Means được thực hiện qua các bước sau:

1 Chọn ngẫu nhiên K tâm (centroid) cho K cụm (cluster). Mỗi cụm được đại diện bằng các tâm của cụm

2 Tính khoảng cách giữa các điểm đến tâm (cx, cy) của từng nhóm. Dùng khoảng cách Euclidean

3 Nhóm các điểm vào nhóm có khoảng cách từ tâm đến điểm đó nhỏ nhất

4 Xác định lại tâm mới cho các nhóm

m là số phần tử của nhóm

5 Nếu cx, cy không đổi thì đó là tâm cần tìm. Nếu không thì làm lại bước 2.

(tong hop gif)

Áp dụng thuật toán K-Means vào gợi ý phân tuyến bán hàng cho người quản lý tuyến bán hàng. Đầu vào của bài toán là N điểm (tương ứng với N cửa hàng bán lẻ) và K cụm (do người quản lý thiết lập). Cần chia N điểm này cho K cụm một cách cân bằng và co cụm nhất.