Store API.

Thứ tự:

* Import file .env
* Import file express
* CREATE INSTANCE App = EXPRESS()
* Nhúng notFoundMiddleWare và errorMiddleware
* Sử dụng json middleware
* Setting route để gọi đến api/v1/products
* Sử dụng notFound middleware, errorHandle middleware
* Setting port, cho server lắng nghe ở 1 port nào đó sau đó
* Setting connect db
* Xong bước đầu
* Setting đến route api/v1/products
* Setting controller với 2 route, 1 route là để sài 1 route để test
* Setting route để hướng đến controller.
* Nhúng file route vào file app.js
* Nhúng module vào để xử lý lỗi.

*Giải thích:* Cơ bản là: lúc trước chúng ta bao các đoạn xử lý vào try catch, nếu nó xuất hiện lỗi thì chúng ta sẽ quăng nó đến middleware tiếp theo để xử lý lỗi, nhưng trong project này, thư viện express-async-errors sẽ sử lý cho chúng ta. Lúc trước chúng ta cần sử dụng next, có thể sử dụng cùng với một đối tượng Error nào đó để khởi tạo lỗi và quăng đến middleware bằng lệnh next() để xử lý. Bây giờ chúng ta chỉ cần sử dụng câu lệnh:

// throw (Error('testing async error'));

Để có thể quẳng ra lỗi mà nó gặp phải. Ở đây, nó sẽ khởi tạo đối tượng Error và quăng ra lỗi này đến middleware để sử lý lỗi mà không cần gọi đến next.

Thực hiện tạo model cho schema. Trong đó có giá trị value thì setting enum để chọn một số công ty nhất định. Trong bài tiếp theo ta sẽ load động những giá trị này.

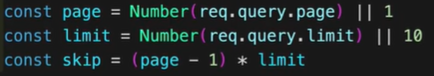
Trước khi tạo ra các function về filter thì ta cần có data trong csdl. Ta có 2 cách để có, cách 1 là thêm function add rồi add data vào, cách 2 là ta có thể pass toàn bộ data đó vào trực tiếp csdl luôn (viết file populate để làm việc đó cho mình).

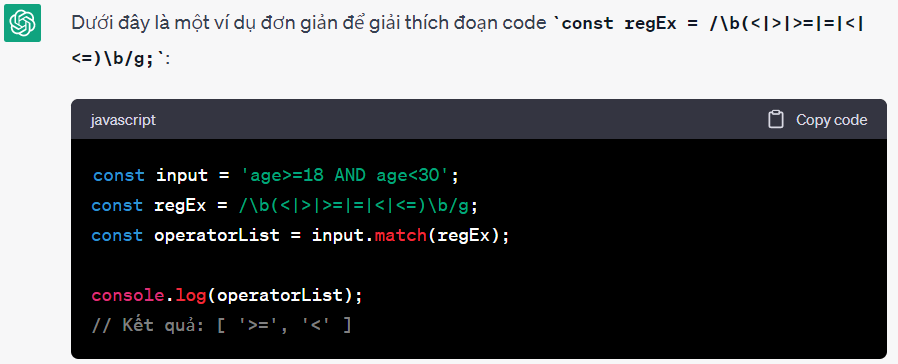
Thay vào đó, nếu muốn kết thúc quá trình với mã trạng thái là 0 (thành công), ta nên sử dụng **process.exit(0)**.

Trong khi đó, **process.exit(1)** sẽ kết thúc quá trình với mã trạng thái là 1 (lỗi).

Đối số truyền vào process.exit() thì chỉ là mã trả về của trạng thái khi thực thi xong các câu lệnh. Các mã trạng thái này chỉ nhằm để cho lập trình viên phân biệt được để đưa ra các xử lý cần thiết.

* Chỉnh sửa lại controller của product
* Thêm phương thức find vào trong hàm, có thể cung cấp các thuộc tính cần tìm để tìm những object phù hợp với mong đợi. Thì thông thường partern của nó là mình sẽ lấy các property từ tham số query. Sau đó, kiểm tra từ property một và nếu tồn tại property đó thì ta tiến hành gán cho object mang nhiệm vụ đại diện để tìm kiếm.
* Đối với tìm kiếm các properties số hoặc đặc biệt thì ta sử dụng các operator: (1) operator {$regex: công thức} tìm kiếm các trường theo công thức chung có sẵn. Trong trường hợp tìm kiếm một thuộc tính không đầy đủ. Ví dụ tìm kiếm tên mà chỉ ghi họ hoặc tìm kiếm tên mà chỉ ghi tên cuối thì kết quả nó vẫn hiển thị phù hợp.
* Ngoài ra, chúng ta có thể sắp xếp các kết quả để nó có thể hiện thị phù hợp. Ví dụ sắp xếp theo tên để hiển thị từ a-z hay z-a. Sắp xếp theo giá để nó có thể hiện thị giá cao giá thấp, v.v. Có thể sắp xếp dựa theo một hay nhiều thuộc tính.
* Tất cả những cái trên cơ bản là ta định nghĩa cái phương thức find() của mongodb, sau khi phương thức find được định nghĩa thì nó sẽ trả ra các giá trị cần thiết. Khi phương thức find() được thực thi thì ta sẽ có được một hoặc một list object. Ta có thể sử dụng thêm các hàm limit(), sort(), select để có thể customize object hoặc list object được trả về đó.
  + Sort(): truyền vào fileds để có thể sắp xếp các giá trị từ lớn tới bé hoặc bé tới lớn, tùy vào dev
  + Select(): sau khi tìm được được các object thì mình có thể chiếu để lấy các thuộc tính cần thiết cho function
  + Limit(): sử dụng để quy định số item hiện ra
  + Skip(): quy định số item có thể skip để hiện ra. Ví dụ hiện ra item1, item2, item3, item5. Khi skip 1 thì nó hiện ra item2, item3, item5.
* Setup cơ chế phân trang:

 đoạn code phân trang.

* 

Sử dụng biểu thức chính quy để tìm các toán tử trong request.

* Khi mình truyền vào params thì mình sẽ truyền vào ở dạng chuỗi có các toán tử. Mình sẽ tìm cách phân tách các toán tử này ra, từ đó xử lý các logic để đưa về dạng so sáng, từ đó có thể làm các so sánh cho các thuộc tính numeric.
* Bình thường, nếu truyền tham số (object) với cú pháp: {field: {operator: value}} thì mongodb sẽ giúp mình tìm ra object hoặc một list object có value ứng với field phù hợp.

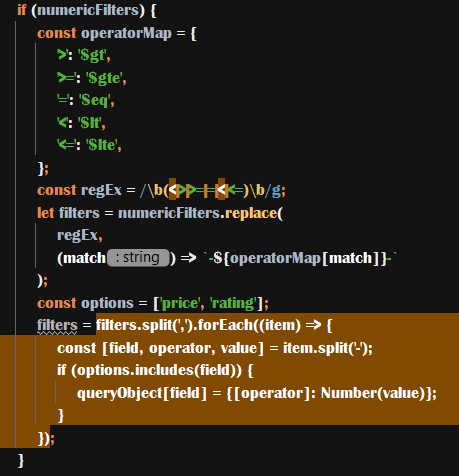
Ví dụ:



* Giải thích chi tiết code:
  + Đầu tiên: tạo một object `operatorMap` để thiết lập các key value để ánh xạ các giá trị toán tử cần thiết
  + Sử dụng biểu thức chính quy `regEx` để thiết lập các toán tử nhằm đối chiếu với các toán tử được truyền vào tham số trong chuỗi
  + Thực hiện biến đổi chuỗi string `numericFilters` được truyền vào dựa vào các toán tử trong biểu thức chính quy `regEx` trở thành dạng được MongoDB support thông qua đối tượng KeyValue `operatorMap`

Ví dụ: từ chuỗi: `price>40,rating>4` thực hiện xong các bước đó có thể chuyển đổi thành: `price-$gt-40,rating-$gt-4`

* + Tiếp theo, thực hiện phân tách chuỗi thông qua dấu phẩy. Nếu thực hiện thành công theo như ví dụ trên. Ta có thể biến đổi nó thành dạng: [‘price-$gt-40’, ‘rating-$gt-4’]
  + Tiếp theo, thực hiện phân tách chuỗi thông qua dấu gạch ngang. Và lần lượt lưu chuỗi đó thành 3 thuộc tính riêng biệt trong mảng. Sử dụng kiến thức của: destructuring assignment để gán các giá trị từ một mảng hoặc chuỗi thành các biến riêng lẻ.
  + Tiếp theo, kiểm tra biến chứa các thuộc tính, nếu chuỗi mới được tách ra nằm trong jacs trị của thuộc tính đó thì tiến hành tạo object và lưu lại đối tượng theo dạng {field: {operator: value}} format theo mongoDB và có thể tìm kiếm.



Reference:

<https://hn.algolia.com/api>

<https://mongoosejs.com/docs/guide.html>

<https://www.mongodb.com/docs/manual/core/document/>