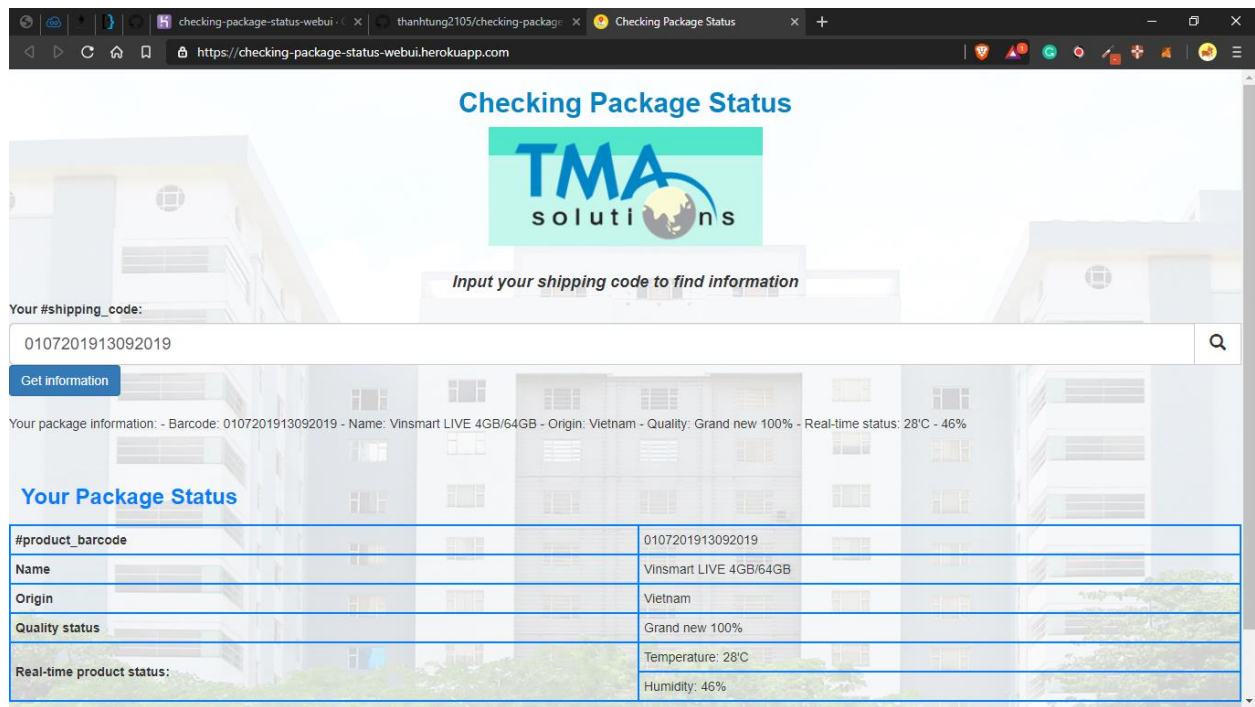


Phần giao diện (WebUI) mẫu đã được deploy tại trang web của **Heroku**:

<https://checking-package-status-webui.herokuapp.com/>



| Your Package Status | |
|---------------------------|------------------------|
| #product_barcode | 0107201913092019 |
| Name | Vinsmart LIVE 4GB/64GB |
| Origin | Vietnam |
| Quality status | Grand new 100% |
| Real-time product status: | Temperature: 28°C |
| | Humidity: 46% |

Hiện tại, phần tra cứu được tạo với một ví dụ về việc input barcode để tra cứu test, với barcode mẫu là: *0107201913092019*

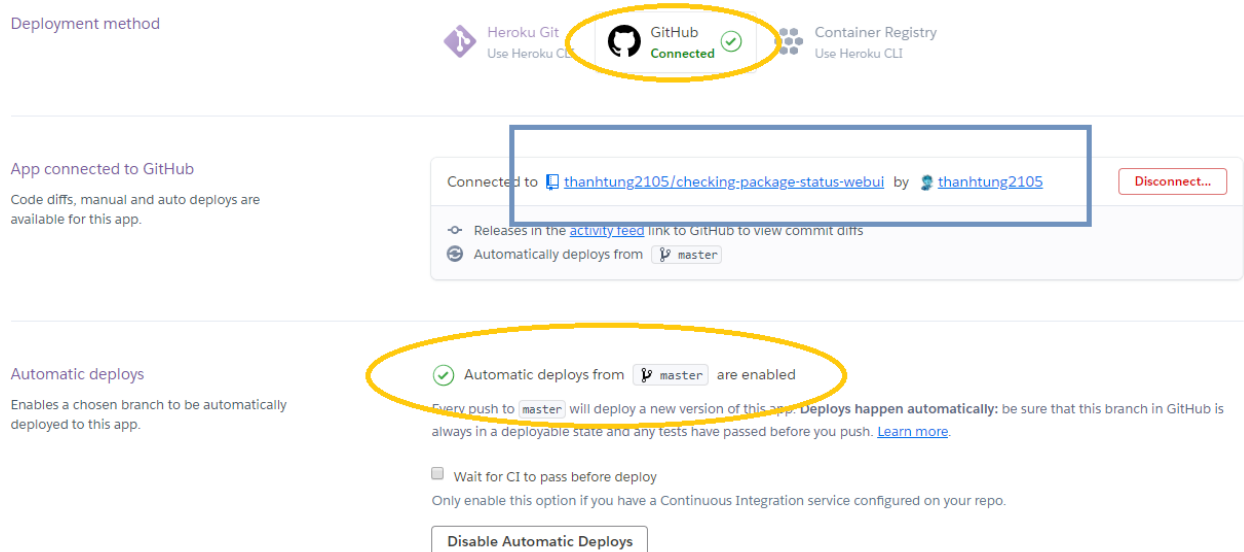
Về việc sử dụng google map API để truy xuất địa điểm, google map API đã được import vào source code của file NodeJS Applications. Nhưng do API được lấy là từ tài khoản miễn phí, nên phần bản đồ chỉ hiện thị khi dưới hình thức Dev, còn khi public trên một trang nó sẽ bị mất đi. Có thể test phần bản đồ trên một App Dev Web online sau:

https://jsfiddle.net/pttung_batch26/g7h6om92/

Phần ứng dụng truy xuất (WebUI) **bắt buộc** được viết bằng NodeJS Application vì chỉ có **NodeJS** mới có tập lệnh để thêm vào các hành động truy xuất của **Web3**, từ đó ta có thể import thông tin **Web3js** để lấy lên những thông tin từ Blockchain. Heroku sẽ là cầu nối tạo chức năng liên

kết trực tiếp giữa WebUI với **Rinkeby TestNet** để lấy dữ liệu của các giao dịch thông qua các hàm từ thư viện của **Web3js**.

Đối với phần **NodeJS** này, source code của phần **NodeJS Applications** được up lên Github, sau đó dùng **Heroku** tạo một Ứng dụng, rồi liên kết trực tiếp Heroku với Github tạo sẵn có source code của **NodeJS App** rồi kích hoạt chức năng **Automatics Deploys**, để ta chỉ cần sửa source từ Github và trang web mà Heroku cung cấp sẽ tự thay đổi dựa vào việc lấy thông tin từ link Github kèm sẵn.



Phần này, phần **source code** của **NodeJS Applications** sẽ được chúng em kèm theo sẵn file để CTY dễ dàng tùy biến xử lý phát triển mà không cần phụ thuộc vào tài khoản **Heroku** cũng như **Github** của em tạo sẵn. Tạo tính chủ động cho việc tiếp tục phát triển sau này.

Do mặt *kiến thức hạn chế* về nền tảng Web cũng như viết App bằng **NodeJS**, nên chúng em chưa rõ được cơ chế để kiểm soát các file trong **NodeJS App**. Hiện tại source code trên đã có sẵn các phần thông tin và chức năng của một WebUI cho người dùng tương tác (khung sườn). Đồng thời cũng có thêm file readme để hướng dẫn cách sử dụng thư viện web3js để xử lý việc lấy dữ liệu từ Ethereum Blockchain khi được cung cấp đầy đủ thông tin về *địa chỉ Contract*, *địa chỉ tài khoản* và *privateKey*. Nhưng hiện tại về kiến trúc file của **NodeJS** khá xa lạ so với tầm hiểu biết của tụi em, nên tụi em chưa biết nên **import** phần web3js này vào file nào và xử lý nó như thế nào cả. Mong anh cùng mọi người có thể tìm hiểu hoặc tìm bạn nào chuyên môn về Web cũng như **NodeJS Applications** để dễ dàng hơn trong việc xử lý vấn đề này.

Tụi em cũng có tìm hiểu được một hướng giải quyết khác cho việc viết một **Ethereum Web Application sử dụng Web3.js, Node.js và Express** tại bài viết ở trang dưới đây:

https://www.polarsparc.com/xhtml/Ethereum-Web-App.html?fbclid=IwAR2pjicctHh4j_QvkRKzxdqsxIOD_Nxa49QxVPunb95cW_bv1E0b2LLnVgk

Sự khác biệt giữa việc dùng cách hiện tại và cách trên bài viết là:

- Ở cách hiện tại thì phần giao tiếp với Blockchain thông qua web3js được thực hiện **trực tiếp** trên trình duyệt người dùng. **Heroku** chỉ chịu trách nhiệm lưu trữ lại *html* và *js*.
- Còn ở cách theo bài viết thì việc giao tiếp với Blockchain thông qua web3js sẽ được thực hiện ở **server**, client chỉ hiển thị nội dung, hiện tại tụi em chưa tìm thấy sự khác biệt của 2 cách trên vì ứng dụng còn nhỏ, nhưng do hạn chế về mặt kiến thức web, nên tụi em sẽ chọn cách xử lý ở client-side.