Technological choices

Summary :

Eliot Courtel
February 2020

wellcheck.fr/documentation

All the architecture is powered using docker and docker-compose (see document from february 2020 for more informations).

Wellcheck make us face some technological problems and choice which leads to a potential solution that we now use.

Problems:

- UX Reactivity
- Easy & understable backend code for non-technical staff
- Deal with large Amount of data
- Needs to be easly Manageable
- Have to be testable by any person
- Generate trustable documents

To recap our system we now have:

- A main web server (Ngnix) working with let's encrypt certification
- An admin interface (html5 | css3 | PHP7 - vanilla | js - vanilla) # admin.wellcheck.fr
- A document handler (html5 | css3 | PHP7 - vanilla) # doc.wellcheck.fr
- An error handler (html5 | css3) # wellcheck.fr/error/
- A webapp (html5 | css3 | js - VueJs) # dashboard.wellcheck.fr
- A presentation web (html5 | css3) # wellcheck.fr
- An API (Python3.7 - Bottle & Custom Framework) # api.wellcheck.fr
- A Mobile Application (JS - ReactNative)
- A Log Database (Elasticsearch6)
- An User Database (MySQL5)
- Databases tools (PhpMyAdmin | Kibana)
- An Access log Analyser (Goaccess)
- A test data generator (Python3.7)

UX Reactivity, was fixed by using either **light** or **native framework** (Vuejs | ReactNative)
The **backend code** is readeable thanks to **Python3** an interpreted, multi-paradigm and multiplatform programming language
We can **deal with a large amount of data** thanks to **ElasticSearch** a distributed and multi-entity search engine
The whole process is **manageable** thanks to **Phpmyadmin**, **Kibana** and **Goaccess** which each provide a service in the management of the application
The webapp is **testable** by using **test account** and **test device**
The application is designed to **push document hash** into a **smartcontract** (etherum blockchain)