*HỆ THỐNG KÝ ĐIỆN TỬ - ESIGN*

*KIẾN TRÚC PHẦN MỀM*

|  |  |
| --- | --- |
| Ngày: | 22-12-2020 |
| Phiên bản: | 1.0.0 |

**LỊCH SỬ THAY ĐỔI**

|  |  |  |  |
| --- | --- | --- | --- |
| Phiên bản | Người thực hiện | Ngày sửa đổi | Lý do thay đổi |
| 1.0.0 | Phạm Toàn | 22.12.2020 | Tạo mới |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**LỊCH SỬ PHÊ DUYỆT**

|  |  |  |  |
| --- | --- | --- | --- |
| Phiên bản | Người phê duyệt | Ngày phê duyệt | Ghi chú |
| 1.0.0 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**MỤC LỤC**

[DANH MỤC CÁC KÝ HIỆU 3](#_Toc58412667)

[I. Kiến trúc tổng quan 4](#_Toc58412668)

[II. Kiến trúc dữ liệu 7](#_Toc58412669)

[III. Kiến trúc triển khai 10](#_Toc58412670)

[PHỤ LỤC I: TỔNG QUAN SẢN PHẨM 15](#_Toc58412671)

[1. Giới thiệu sản phẩm 15](#_Toc58412672)

[2. Đối tượng sử dụng sản phẩm 16](#_Toc58412673)

[3. Tương tác sản phẩm 16](#_Toc58412674)

[PHỤ LỤC II: TIÊU CHUẨN CHẤT LƯỢNG 18](#_Toc58412675)

[1. Hiệu năng 18](#_Toc58412676)

[2. Tính khả dụng 18](#_Toc58412677)

[3. Tính an toàn, bảo mật 19](#_Toc58412678)

[4. Tính sửa chữa và bảo trì 26](#_Toc58412679)

[5. Tính kiểm thử 26](#_Toc58412680)

[6. Tính co giãn 26](#_Toc58412681)

[7. Tính tái sử dụng 27](#_Toc58412682)

[8. Tính hỗ trợ 27](#_Toc58412683)

[9. Tính tích hợp 27](#_Toc58412684)

[10. Tính khôi phục thảm họa 28](#_Toc58412685)

[PHỤ LỤC III: CÁC YẾU TỐ ẢNH HƯỞNG 29](#_Toc58412686)

[1. Yếu tố về thời gian và ngân sách 29](#_Toc58412687)

[2. Yếu tố về công nghệ 29](#_Toc58412688)

[3. Yếu tố về nhân sự 29](#_Toc58412689)

[PHỤ LỤC IV: CÁC NGUYÊN TẮC THIẾT KẾ 30](#_Toc58412690)

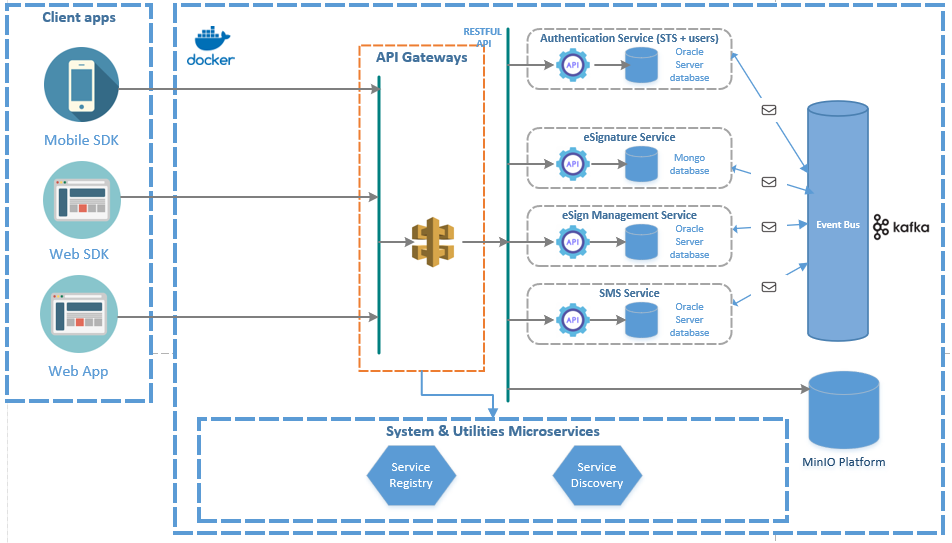
[1. Nguyên tắc phát triển 30](#_Toc58412691)

[2. Nguyên tắc tổ chức 30](#_Toc58412692)

# DANH MỤC CÁC KÝ HIỆU

|  |  |  |
| --- | --- | --- |
| **STT** | **Thuật ngữ** | **Diễn giải** |
| 1 | MicroService | Một trường phái kiến trúc phần mềm, nơi ứng dụng được chia nhỏ thành nhiều phần, chạy độc lập, giao tiếp qua mạng. |
| 2 | Monolithic | Một trường phái kiến trúc phần mềm, nơi ứng dụng được tổ chức thành một thể duy nhất. |
| 3 | Domain Driven Design | Là một cách tiếp cận trong xây dựng xây phần mềm, nơi phần mềm được tổ chức thành các mảng nghiệp vụ. |
| 4 | HTTP | Giao thức sử dụng cho các dịch vụ Web. |
| 5 | HTTPS | Giao thức HTTP dưới dạng bảo mật. |
| 6 | HTTP Action | Các thao tác dựa trên nền tảng giao thức HTTP: GET/POST/PUT/DELETE. |
| 7 | API | Giao diện lập trình ứng dụng, là cửa ngõ để các ứng dụng giao tiếp với nhau. |
| 8 | Restful API | Là API nhưng dựa trên các phương thức cơ bản của HTTP (GET/POST/PUT/DELETE). |
| 9 | Go live | Hành động đưa ứng dụng lên môi trường thật. |
| 10 | Auto Scale | Đại diện cho việc ứng dụng có thể tự động mở rộng năng lực xử lý dựa vào tải của hệ thống. |
| 11 | API Gateway | Hệ thống quản lý, hỗ trợ phát hành API cho các đối tượng sử dụng. |
| 12 | SOA | Kiến trúc hướng dịch vụ. |
| 13 | K8S | Một nền tảng mã nguồn mở tự động hoá việc quản lý, scaling và triển khai ứng dụng dưới dạng container hay còn gọi là Container orchestration engine |
| 14 | CI/CD | Thuật ngữ đại diện cho việc tích hợp và triển khai liên tục. |
| 15 | JMeter | Phần mềm hỗ trợ kiểm thử tải hệ thống. |
| 16 | eSign | Hệ thống ký điện tử |

# Kiến trúc tổng quan



Hình 1 - Mô hình tương tác sản phẩm

Thành phần cơ bản, dùng chung cho các hệ thống Microservice:

- API Gateway: API Gateway chịu trách nhiệm định tuyến các request, tổng hợp và chuyển đổi giao thức. Tất cả các request từ client sẽ đi qua API Gateway trước tiên. Sau đó định tuyến các request tới các microservice thích hợp. API Gateway giống như một tổng đài để điều phối các request đến từ trình duyệt (dạng HTTP REST request hay request đến ­­URL một trang web). Nếu chúng ta không sử dụng API Gateway thì clients sẽ phải gửi request trực tiếp tới service cụ thể nào đó.

- Authorzation Service: Cung cấp cơ chế oAuth2, là nơi quản lý user, role.

- Service register:

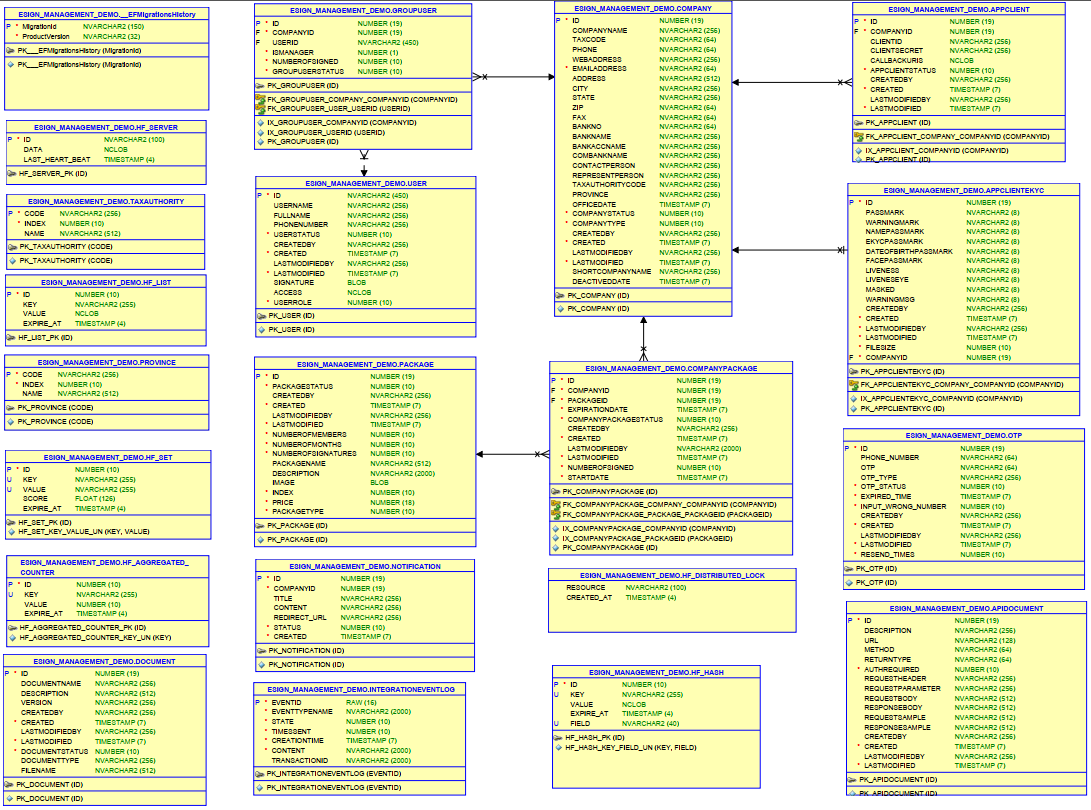
* Service Registry: Service Registry giữ các thực thể microservices và địa chỉ của chúng. Thực thể microservices được đăng kí với service registery khi bắt đầu chạy và hủy đăng kí khi tắt. Người dùng có thể tìm các services đang tồn tại và địa chỉ của chúng qua service registry.
* Service Discovery: Để tìm các microservices đang tồn tại và địa điểm của chúng, chúng ta cần một quy trình truy tìm dịch vụ. Có hai mô hình để triển khai Service Discoversy là Client-side Discovery và Server-side Discovery. Hiện tại hệ thống đang sử dụng mô hình Server-side Discovery và dùng nginx làm Ingress (cân bằng tải mềm).
* ELK: Log tập trung toàn bộ hệ thống, sử dụng các dịch vụ: Logstash, Elaticsearch, Kibana  
  Hệ thống Ký điện tử được xây dựng bởi các thành phần chính sau:
* Web Portal: Là ứng dụng chạy trên trình duyệt, cung cấp các tính năng giúp khách hàng tạo và quản lý tài khoản, ký điện tử, tra cứu thông tin tài liệu. Thành phần này sử dụng .Net Core để xây dựng ứng dụng. Nhóm lập trình Web chịu trách nhiệm phát triển thành phần này.
* Mobile SDK: Là bộ SDK sử dụng cho ứng dụng iOS hay Android, cho phép khách hàng xem và ký điện tử trực tiếp trên ứng dụng di động, bằng hình thức ký điện tử OTP hoặc OTP kết hợp eKYC. Nhóm lập trình Mobile chịu trách nhiệm phát triển thành phần này.
* Web SDK: Là bộ SDK sử dụng cho ứng dụng web, khách hàng xem và ký điện tử trực tiếp trên ứng dụng di động, bằng hình thức ký điện tử OTP hoặc OTP kết hợp eKYC. Nhóm lập trình Web chịu trách nhiệm phát triển thành phần này.
* Authentication Service: Đây là thành phần hỗ trợ quản lý người dùng, gồm các tính năng xác thực và phân quyền. Sử dụng cơ chế OAuth2, một phương thức chứng thực kiểu ủy quyền. Client xác thực với server cấp quyền (authorization server) và nhận một token gọi là "Access token". Access token không chứa bất kì thông tin gì về client. Nó chỉ là một tấm vé tham chiếu đến thông tin người dùng mà server cấp quyền có thể truy xuất đến. Do đó, đây cũng được gọi là token kiểu tham chiếu "by-reference token" và an toàn để sử dụng trên mạng lưới mở và internet.
* eSignature Service: Đây là thành phần hỗ trợ xử lý luồng ký điện tử, được xây dựng trên công nghệ Python Flask kết hợp ký số lên dữ liệu. Nhóm lập trình API chịu trách nhiệm phát triển thành phần này.
* eSign Management Service: Đây là thành phần quản lý khách hàng, gói cước, luồng ký điện tử. Thành phần này sử dụng .Net Core để xây dựng. Nhóm lập trình API chịu trách nhiệm phát triển thành phần này.
* SMS Service: Đây là thành phần tiện ích hỗ trợ việc gửi SMS thông báo cho khách hàng theo các mẫu đã định trước. Đồng thời cho phép lưu lại lịch sử SMS đã gửi. Nhóm lập trình API chịu trách nhiệm phát triển thành phần này.
* Oracle Database: Đây là thành phần lưu trữ dữ liệu sử dụng hệ quản trị cơ sở dữ liệu SQL. Nhóm hệ thống chịu trách nhiệm triển khai thành phần này.
* Mongo Database: Đây là thành phần lưu trữ dữ liệu sử dụng hệ quản trị cơ sở dữ liệu NoSQL. Nhóm hệ thống chịu trách nhiệm triển khai thành phần này.
* MinIO Platform: đây là thành phần lưu trữ dữ liệu dạng object. Thành phần này được xây dựng sử dụng nền tảng MinIO, đây là nền tảng có sẵn, nhóm hệ thống chịu trách nhiệm triển khai thành phần này. Nền tảng này cung cấp SDK cho các hệ thống khác tương tác.
* Khi lượng truy cập vào hệ thống tăng tất cả các thành phần đều có khả năng mở rộng theo chiều ngang (scaling out) về mặt vật lý và có thể chia ra các vùng để tăng khả năng đáp ứng của hệ thống. Chi tiết xem ở mục Kiến trúc triển khai.

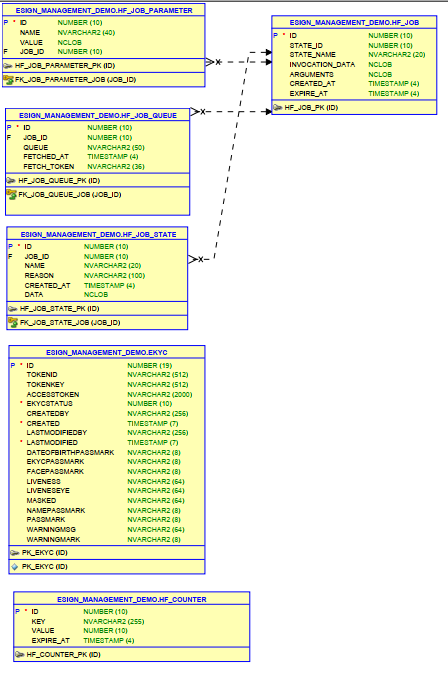
Các thành phần độc lập giao tiếp với nhau thông qua RESTful API và Apache Kafka (giao tiếp bất đồng bộ thông qua message).

Tất cả các Microservice nghiệp vụ của hệ thống đều là Stateless Microservice và triển khai trên mô hình k8s nên khi lượng truy cập sử dụng tăng thì hệ thống sẽ tự động giãn nở (scaling-out) theo cấu hình mà người quản trị hệ thống thiết lập. Các thành phần khác của hệ thống (API Gateway, Services Registry, Kafka, EFK, Distributed caching, …) đều có khả năng mở rộng khi cần.

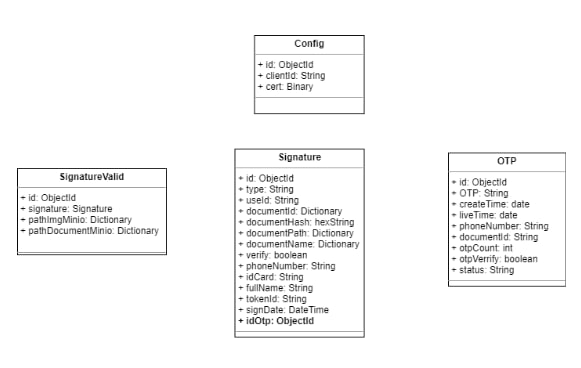
# Kiến trúc dữ liệu

**eSign Management Database (OracleDB)**

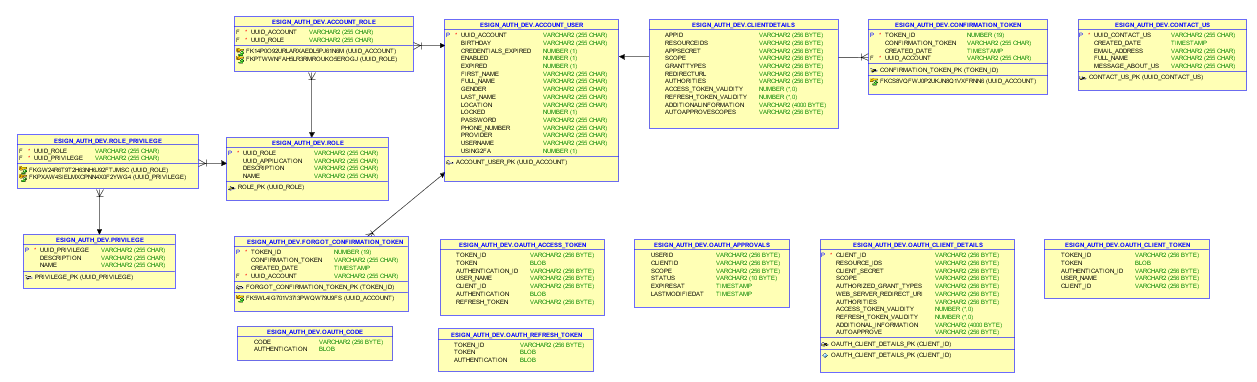




**eSign Service (MongoDB)**



**Authentication Service (OraceDB)**



# Kiến trúc triển khai

Triển khai và vận hành là khâu quan trọng trong vòng đời phần mềm, với Microservice để cung cấp đầy đủ thông tin cho đội triển khai và vận hành, tham khảo tài liệu chi tiết là Microservice Deployment and Operation Guide. Còn trong phạm vi tài liệu này chỉ điểm qua để cung cấp góc nhìn và cách tiếp cận của Microservice về vấn đề này.

## Kiến trúc triển khai



**Hình 3 – Mô hình triển khai**

Mô hình triển khai được chia làm thành 04 thành phần chính. Thành phần triển khai ứng dụng bao gồm các microservice được triển khai trên nền tảng Kubernetes, Oracle database, Mongo database, hệ thống lưu trữ Object Storage, hệ thống Caching Redis,hệ thống Message Broker Kafka. Thành phần nền tảng đảm bảo hạ tầng ổn định bao gồm Network và Firewall.

Toàn bộ các microservice của hệ thống đều được triển khai trên các worker node của hệ thống Kubernetes và chạy ít nhất 02 pod và có khả năng co dãn để đảm bảo năng lực và tính sẵn sàng của các dịch vụ. Ngoài ra mỗi Pod đều có cơ chế Healthy Check để tự động khôi phục dịch vụ nếu có thể xảy ra. Hệ thống Kubernetes hỗ trợ cơ chế Rolling Update giúp tự động hóa nâng cấp phiên bản mà không gây Downtime. Hệ thống Kubernetes đã được tích hợp quy trình chuỗi tích hợp liên tục bao gồm Automation Test và lưu trữ phiên bản theo quy định chung của tổng công ty.

Toàn bộ các database bao gồm Oracle DB và mongo database được triển khai theo mô hình cluster nhằm tăng khả năng đáp ứng và tính sẵn sàng của dữ liệu và có khả năng chia tải cho các thành phần phụ.

Hệ thống lưu trữ Object Storage MinIO (Cluster) hỗ trợ giao thức S3 protocol theo chuẩn lưu trữ phổ biến hiện nay giúp tương thích tích hợp dễ dàng với các thành phần Microservice. Hệ thống lưu trữ cung cấp đầy đủ các tính năng lưu trữ hiển thị, khởi tạo đường dẫn download tạm thời, lưu trữ sẵn sàng cao cho các hợp đồng phát sinh ở hệ thống eContract.

Hệ thống middle service bao gồm Redis và Kafka xây dựng theo giải pháp Cluster với cơ chế Cache và Message Broker giữa các Microservice giúp đảm bảo dữ liệu trao đổi luôn nhanh và sẵn sàng tin cậy

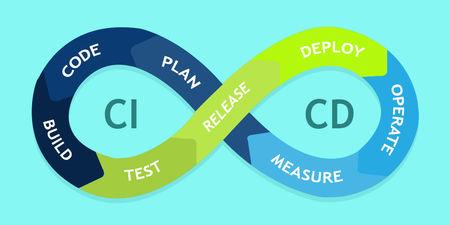
Hệ thống Network Firewall cung cấp trao đổi dữ liệu với băng thông cao (10Gbps) cùng hệ thống Firewall IDS/IPS giúp đảm bảo an toàn bảo mật phía ngoài Internet truy cập vào hệ thống Core eSign.

## Cách thức triển khai MICROSERVICE

Các Microservice hướng đến việc triển khai CI/CD với đầy đủ 3 giai đoạn:

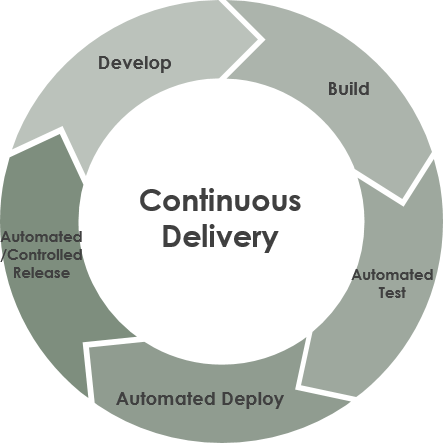
CI – Continuous Integration: Microservice được build và chạy kiểm thử một cách tự động.

CD – Continuous Delivery: Microservice được triển khai trên các môi trường test một cách tự động.



**Hình 12: Continuous Integration and Delivery**

CD – Continuous Deployment: Microservice có khả năng golive bất kỳ thời điểm nào và có khả năng golive nhiều lần trong ngày.



**Hình 13: Mô hình CD**



**Hình 14: Sơ đồ khối triển khai hệ thống Ký điện tử**

## Vận hành MICROSERVICE

Trong khâu vận hành Microservice thì có các mảng sau:

**Tracing**

Microservice đồng bộ tất cả log của từng Microservice lên hệ thống log tập trung là ELK. Các bản tin của cùng một giao dịch được liên kết với nhau thông qua một chuỗi ký tự duy nhất, do vậy khá dễ dàng trong việc truy vết giao dịch.

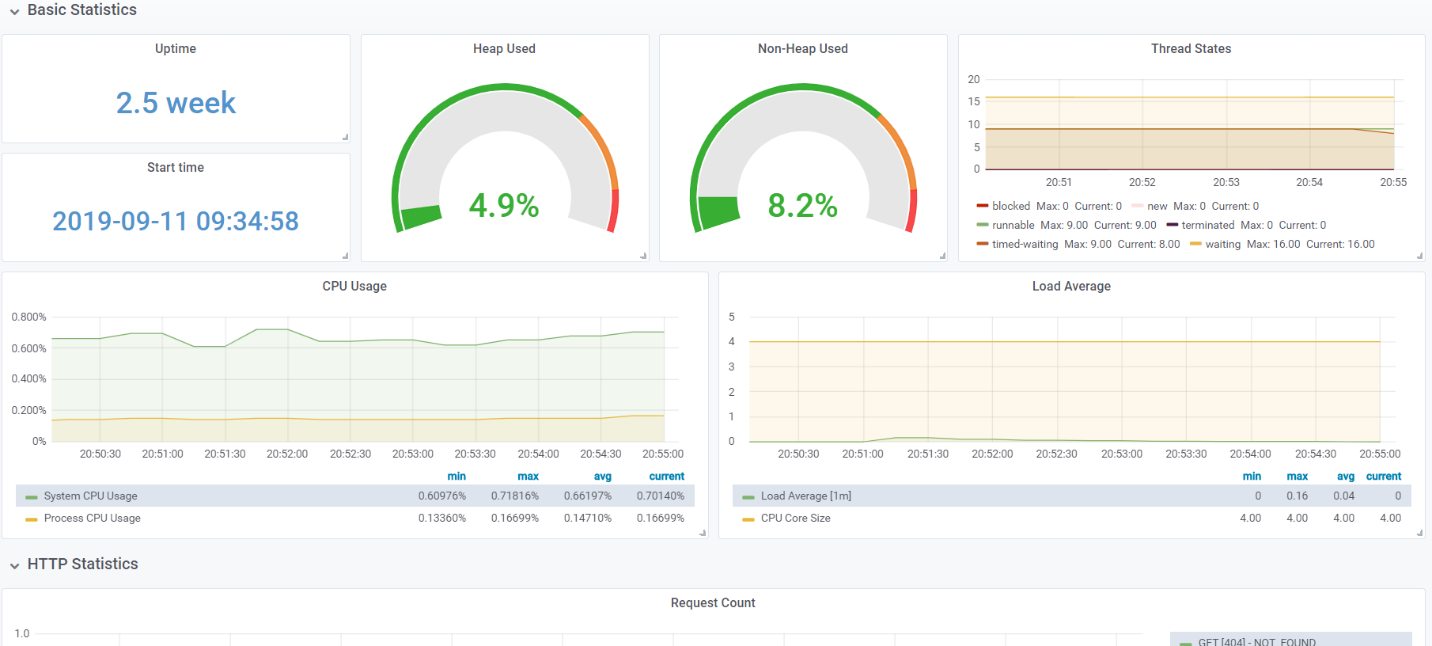
Bản tin truyền qua các Microservice khác nhau của cùng một giao dịch cũng được liên kết với nhau thông qua một chuỗi ký tự duy nhất, giúp việc truy vết giao dịch dễ dàng.

**Monitoring**

Tất cả Microservice , cũng như các server vệ tinh đều được giám sát real-time qua một ứng dụng tập trung đó là Grafana/Prometheus Server. Qua đây người vận hành có thể nắm được các thông số thiết yếu của một Microservice , từ đó có thể biết được ứng dụng có đang hoạt động bình thường hay không.

Có nhiều thông số thuộc các mảng khác nhau được giám sát qua đây:

* Hạ tầng: CPU/RAM/Disk/Network
* Middleware: JVM Heap/Thread/GC/Class/Tomcat Thread/Tomcat Error/...
* Microservice : Uptime/Request/Response/Database/Error/...



**Hình 15: Giám sát Microservice qua Grafana**

Cảnh báo là một phần quan trọng trong khâu vận hành, các Microservice trang bị cho mình hệ thống cảnh báo qua Grafana Server. Mọi thông số giám sát được thì đều cảnh báo được, do vậy Microservice có các cảnh báo ở đầy đủ các mức như phần giám sát: Hạ tầng, Middleware, Microservice .

Cảnh báo hỗ trợ gửi thông qua các hình thức khác nhau: Email, SMS, Chat tool (nếu có), Jira (nếu tích hợp).

# PHỤ LỤC I: TỔNG QUAN SẢN PHẨM

## Giới thiệu sản phẩm

Giải pháp phần mềm Ký điện tử của VNPT mà chúng tôi tạm đặt tên là KÝ ĐIỆN TỬ, là hệ thống phần mềm thay thế chữ ký viết tay của cá nhân hay doanh nghiệp, giúp chứng thực tác giả đã ký vào dữ liệu đó.

Phần mềm Ký điện tử được thiết kế theo kiến trúc microservice, chạy trên nền tảng container cho phép tích hợp hoàn toàn đầy đủ trên hệ thống IDG của Công ty VNPT IT. Ngoài ra, nội dung hợp đồng sau khi ký kết và có hiệu lực thi hành sẽ được lưu trên hệ thống VNPT đảm bảo tính an toàn, minh bạch, không thể sửa đổi và nhất là có thể truy xuất dễ dàng bất cứ đâu.

Phần mềm Ký điện tử nhằm đáp ứng yêu cầu về pháp luật của nhà nước và quy định của Tập đoàn VNPT, Tổng công ty Vinaphone.

Trong đó bao gồm:

* Quản lý khách hàng
* Quản lý dịch vụ tích hợp
* Quản lý gói cước
* Quản lý dịch vụ sử dụng
* Quản lý dịch vụ EKYC
* Quản lý tài liệu tích hợp, SDK
* Quản lý tài khoản
* Báo cáo thống kê
* Cấu hình hệ thống
* Ký điện tử
* Lịch sử ký
* Quản lý tổ chức
* Quản lý tài khoản thành viên
* Quản lý gói dịch vụ
* Thông tin API Key
* Báo cáo sản lượng

Một số chức năng khác liên quan tới quản trị hệ thống

## Đối tượng sử dụng sản phẩm

* Hỗ trợ các cá nhân, tổ chức thực hiện ký điện tử và xác minh các tài liệu một cách minh bạch, nhanh chóng, thuận tiện.

## Tương tác sản phẩm



Hình 6 - Mô hình tương tác sản phẩm

# PHỤ LỤC II: TIÊU CHUẨN CHẤT LƯỢNG

Hệ thống cần thiết kế để đảm bảo các chất lượng như sau:

## Hiệu năng

|  |  |  |
| --- | --- | --- |
| **Mã** | **Yêu cầu** | **Đặc tả** |
| 1 | Yêu cầu về thời gian | Hệ thống đáp ứng các yêu cầu:   * Thời gian cho phép để hiện thị đầy đủ trang thông tin/cổng thông tin điện tử tĩnh là 5 (s); * Thời gian cho phép để gửi kết quả tìm kiếm thông tin là 10 (s) * Thời gian Xem chi tiết cấu hình là 7 (s) * Thời gian Xem tài liệu tích hợp là 7 (s) * Thời gian truy cập màn hình ký điện tử 7 (s) * Thời gian đăng nhập + truy cập dashboard 10 (s) * Thao tác ký điện tử (gồm upfile, nhập mã otp, thêm chữ ký, hoàn thành ký tài liệu) 1 file là 5 (s) mỗi chức năng (tùy thuộc vào dung lượng file up lên) * Thời gian Xem lịch sử ký điện tử 7 (s) * Thời gian cho phép để hệ thống phản hồi lại thông tin đã tiếp nhận yêu cầu xử lý từ phía người sử dụng là 5 (s) |
| 2 | Hiệu năng | Đảm bảo khả năng 150 người sử dụng dịch vụ truy cập đồng thời |

## Tính khả dụng

|  |  |  |
| --- | --- | --- |
| **Mã** | **Yêu cầu** | **Đặc tả** |
| 1 | Phù hợp với nhu cầu | Hệ thống phù hợp với nhu cầu người sử dụng |
| 2 | Khả năng giúp người sử dụng học được | Khả năng hệ thống hỗ trợ khả năng học cách sử dụng đối với nhóm các đối tượng sử dụng khác nhau nhằm thực hiện được các tác vụ trong một ngữ cảnh cụ thể, kết quả đáp ứng được yêu cầu của đối tượng sử dụng. |
| 3 | Giao diện người sử dụng | Có khả năng hiển thị, hoạt động chính xác, đầy đủ trên hầu hết các trình duyệt phổ biến phiên bản mới nhất (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari,...).  Hỗ trợ các nền tảng di động phổ biến: Android và iOS |
| 4 | Khả năng truy cập | Hệ thống dễ truy cập từ môi trường Internet |

## Tính an toàn, bảo mật

* Database phải được bảo mật bằng cơ chế xác thực, phân quyền mạnh.
* Bảo mật mức người sử dụng: Chương trình sử dụng xác thực Oauth quản lý xác thực đảm bảo tính bảo mật cho tất cả các tài khoản (username và password) hoạt động bình thường, mỗi tài khoản được phân quyền theo vai trò.
* Sử dụng SSL cho đường truyền.
* Đảm bảo an ninh hệ thống: Đảm bảo tính an ninh, chống xâm nhập trái phép, chống tấn công của virus, spyware.

|  |  |  |
| --- | --- | --- |
| **Mã YC** | **Yêu cầu** | **Đặc tả** |
|  | **Thông tin định danh** |  |
| 1 | Tên đăng nhập | Tên đăng nhập phải là duy nhất, chỉ bao gồm tập các ký tự là chữ cái, chữ số, dấu gạch dưới. |
| 2 | Thiết lập chính sách mật khẩu mạnh đảm bảo | Thiết lập chính sách mật khẩu mạnh đảm bảo   * Có độ dài lớn hơn hoặc bằng 8 ký tự bao gồm ký tự hoa, thường, số và ký tự đặc biệt. * Mật khẩu mới không được trùng với mật khẩu đã đặt trước đó. * Mật khẩu không sử dụng từ ngữ trong từ điển ví dụ như “abc123” * Khi được cung cấp mật khẩu mới hoặc được reset mật khẩu cần phải đổi ngay mật khẩu theo quy tắc ở trên. |
| 3 | chức năng Reset/ quên mật khẩu (nếu có) | * Đối với chức năng Reset/ quên mật khẩu: * Đường dẫn Reset/quên mật khẩu được gửi qua email phải bị mất hiệu lực sau lần truy cập đầu tiên hoặc sau 24 giờ nếu không được truy cập. * Nếu chức năng Reset/phục hồi mật khẩu thực hiện gửi mật khẩu qua email thì mật khẩu phải được sinh ngẫu nhiên và phải tuân theo chính sách mật khẩu tại mục 2. |
| 4 | mã Hash của mật khẩu trong DB | Chỉ lưu dạng mã Hash của mật khẩu trong DB, sử dụng thuật toán Hash từ SHA-256 trở lên, thêm chuỗi Salt ngẫu nhiên vào mật khẩu trước khi thực hiện Hash. |
| 5 | **Xử lý xác thực** | * Không trả về thông báo chi tiết thông tin định danh cho trường hợp người dùng đăng ký thông tin định danh (Username, Email,...) đã tồn tại tại chức năng đăng ký, hoặc gửi sai thông tin định danh tại các chức năng đăng nhập, Reset/quên mật khẩu, đổi địa chỉ Email. * Bật cơ chế xác thực kết hợp với Captcha hoặc các hình thức tương đương khi đăng nhập sai quá 5 lần liên tiếp. * Chỉ sử dụng phương thức POST để Submit thông tin định danh, khuyến nghị sử dụng HTTPS cho đường truyền để tăng tính bảo mật. |
| 6 | Quản lý phiên đăng nhập | * Session phải được sinh ngẫu nhiên và độ dài tối thiểu là 128-bit. * Session phải được thiết lập thời gian Timeout, giá trị Timeout cần cân bằng giữa nhu cầu thương mại và yếu tố bảo mật. * Tạo mới Session sau khi đăng nhập thành công. * Xóa giá trị Session ID và các dữ liệu gắn với Session đó khi người dùng đăng xuất. * Cấu hình thuộc tính “Secure” đối với các ứng dụng sử dụng HTTPS và “HTTP-Only” cho trường Session Cookie. |
| 7 | Phân quyền | * Kiểm tra phân quyền dựa trên các đối tượng được lưu tại máy chủ dịch vụ (VD: tham số lưu trên Session Server, dữ liệu lưu trên DB,...). * Phân quyền tối thiểu, chỉ đáp ứng đủ chức năng và tài nguyên cho người dùng/ứng dụng. Ứng dụng giới hạn các nghiệp vụ với từng nhóm/ người dùng cơ bản. * Phía giao diện người dùng: Chỉ hiển thị các thành phần giao diện, đường dẫn, hàm,... tương ứng với quyền của người dùng. * Phía server: Kiểm tra quyền tác động của người dùng/ứng dụng trên các hàm và tài nguyên tương ứng trước khi thực hiện bất cứ tác vụ nào tới hệ thống. * Phải có tính năng xóa phiên làm việc hiện tại của người dùng hoặc các cơ chế tương đương đối với các trường hợp quyền người dùng bị thay đổi hoặc bị disable bởi người dùng có thẩm quyền. * Không đặt trang quản trị Public Internet, trong trường hợp bắt buộc phải đặt public cần giới hạn các IP được phép truy cập hoặc sử dụng cơ chế xác thực đa nhân tố (Multiple Authentications). |
|  | **Tương tác với Back-End.** |  |
| 8 | **Mã hóa các dữ liệu nhạy cảm** | * Mã hóa các dữ liệu nhạy cảm trước khi lưu trữ (thông tin tài khoản ngân hàng, Private key,...) |
| 9 | **SQL** | * Phải có biện pháp phòng chống SQL Injection. * Trong 1 số trường hợp không sử dụng giải pháp phòng chống SQL injection, cần thiết lập danh sách các ký tự đầu vào mong muốn. |
| 10 | **Tương tác với hệ điều hành (nếu có)** | * Sử dụng các API hỗ trợ việc thực thi câu lệnh hệ thống. * Không truyền trực tiếp dữ liệu người dùng tới hệ điều hành, trong trường hợp bắt buộc cần thiết lập danh sách các đầu vào mong muốn. |
| 11 | **Tương tác với File** | * Không truyền trực tiếp dữ liệu từ người dùng đến các hàm include File. * Lập danh sách các định dạng file được phép upload. Chỉ cho phép 1 số loại file cơ bản được upload như word, excel,…không cho phép upload file .exe, .bat và giới hạn kích thước của file. * Validate file hợp lệ bằng cách kiểm tra đồng thời File header và phần mở rộng của File. * Với các trường hợp không bắt buộc thì không lưu File upload trong thư mục Web, bỏ quyền thực thi trên thư mục Upload. * Khi cần tham chiếu tới các File tồn tại trên hệ thống cần thiết lập danh sách đầu vào mong muốn hoặc gán các giá trị định danh tương ứng cho các File thay vì truyền tên File. * Không trả về đường dẫn tuyệt đối của File được lưu trữ trong sản phẩm, dịch vụ. * Tất cả dữ liệu, tài nguyên hệ thống (Báo cáo, File upload, File cấu hình...) không được lưu trong thư mục cho phép truy cập trực tiếp không qua xác thực. |
| 12 | **Xử lý Back-End HTTP Request** | Khi tạo HTTP request phía server, các tham số GET, POST cho yêu cầu đó tránh tạo từ dữ liệu phía người dùng, hoặc phải được kiểm tra cẩn thận để chống ghi đè các tham số khác |
|  | **Tương tác với Front-End** |  |
| 13 | **Kiểm soát dữ liệu đầu vào** | * Việc kiểm tra dữ liệu đầu vào phải được thực hiện phía dịch vụ. Không trực tiếp xử lý dữ liệu do người dùng trực tiếp nhập vào, yêu cầu phải kiểm tra có hợp lệ hay không và thông báo cho người dùng. * Thực hiện việc kiểm tra dữ liệu từ tất cả các nguồn dữ liệu có tương tác với người dùng (Các tham số lấy từ GET/POST request, HTTP Headers, dữ liệu lấy từ DB, dữ liệu từ File upload,...). * Xác định rõ chuẩn định dạng encode của dữ liệu đầu vào, thực hiện validate dữ liệu sau khi đã Decode đầu vào về 1 định dạng chuẩn và nhất quán. * Validate kiểu dữ liệu, phạm vi và độ dài dữ liệu và định dạng dữ liệu. * Nếu dữ liệu đầu vào bắt buộc là các ký tự đặc biệt, cần thiếp lập danh sách các ký tự đầu vào mong muốn. Lọc dữ liệu đầu vào người dùng, loại bỏ các ký tự độc hại phổ biến <, > , ‘ , « , %, &, +, or, \\, \ … để chống các hình thức SQL injection, XSS |
| 14 | **Kiểm soát dữ liệu đầu ra** | 1. Phải chỉ rõ Character Encoding cho dữ liệu đầu ra. 2. Response body phải được encode theo ngữ cảnh sử dụng. Một số trường hợp phổ biến:  * Đầu ra là HTML, thực hiện HTML Encode các kí tự đặc biệt (<>’”&) từ các nguồn dữ liệu không an toàn (Các tham số lấy từ GET/POST request, HTTP Headers, dữ liệu lấy từ DB, dữ liệu từ File upload,...có thể điều khiển được bởi người dùng). * Đầu ra là Json, thực hiện Encode dữ liệu trả về dạng Object, không trả về dạng mảng.  1. Response header: lọc bỏ các kí tự đặc biệt (\n, \r) do dữ liệu người dùng truyền vào. 2. Cookie trả về cần giới hạn tối thiểu nhất các thuộc tính (Domain, Path, HTTP Only, Expire, Secure). Tránh lưu trữ các dữ liệu nhạy cảm trên Cookie, nếu cần lưu trữ các dữ liệu nhạy cảm thì phải thực hiện mã hóa các dữ liệu này với thuật toán đối xứng mạnh và key chỉ được lưu trên Server. 3. Hạn chế việc chuyển hướng, chuyển tiếp đến các URI khác. Nếu ứng dụng có chức năng này cần phải lập danh sách các URI được phép thực hiện chuyển hướng, chuyển tiếp. |
| 15 | **Kiểm soát ngoại lệ và ghi log ứng dụng.** | * Xử lý các ngoại lệ bằng Try-Catch và trả về các thông báo lỗi chung đã Custom (không show exception ra giao diện chương trình), thông báo lỗi trả về không được chứa các thông tin nhạy cảm của người dùng, hệ thống,... * Các thông tin lỗi, ngoại lệ này phải được log lại để phục vụ bảo trì, xác định nguyên nhân lỗi ứng dụng. * File log không được đặt cùng thư mục mã nguồn của hệ thống. * Không lưu thông tin nhậy cảm (như thông tin mật khẩu, Session ID.) trong file log. * Giới hạn người dùng cho phép truy cập file log. |
| 16 | **Sử dụng Framewwork, Lib (thành phần bên thức ba)** | * Loại bỏ các thành phần, Lib không cần thiết. * Sử dụng phiên bản khuyến nghị của Framework tại thời điểm phát triển ứng dụng. * Thường xuyên cập nhật các bản vá lỗi cho Framework. |
| **17** | **Các yêu cầu khác** | Theo Quyết định số 854/QĐ-VNPT-IT-CLGSP-ĐHIT ngày 30/10/2018 về việc ban hành quy định an toàn,an ninh sản phẩm phần mềm trước khi cung cấp dịch vụ, 89/QĐ-VNPT-IT&VAS ngày 25/01/2018 Quy định đảm bảo an toàn hệ thống thông tin trong hoạt động của VNPT |

## Tính sửa chữa và bảo trì

* Thiết kế theo kiến trúc microservice đảm bảo được các service độc lập tương đối
* Khả năng phát triển độc lập của các microservice
* Khả năng triển khai độc lập các microservice
* Có thể áp dụng các công nghệ phù hợp với các microservice khác nhau
* Hỗ trợ rolling update
* Backup dữ liệu: Chương trình phải được sao lưu dữ liệu định kỳ, hoặc theo yêu cầu, không làm mất mát dữ liệu.
* Backup chương trình: Chương trình phải được sao lưu định kỳ đảm bảo tránh mất mát.

## Tính kiểm thử

* Các chức năng đều phải được kiểm thử và pass qua kiểm thử mới được phép triển khai
* Một số module, chức năng chính cần thực hiện unit test > 95%
* Thiết kế giao diện, api theo chuẩn để có thể kiểm thử tự động bằng các công cụ như Katalon, Jmeter, Selenium,...

## Tính co giãn

* Triển khai trên nền tảng Kubernetes.
* Hệ thống hỗ trợ co dãn theo cả 2 chiều dọc và ngang.
* Việc co dãn hệ thống được thực hiện một cách tự động thông qua các hệ thống hỗ trợ.
* Thời gian co dãn: < 1 phút.
* Khi mức tải của các microservice > 85% trong vòng 30 phút sẽ thực hiện mở rộng.
* Khi mức tả của các microservice < 30% liên tục trong vòng 30 phút sẽ thực hiện co tài nguyên.

## Tính tái sử dụng

* Việc xây dựng các microservice độc lập tương đối nên sẽ tái sử dụng được các microservice cho một số hệ thống, cụ thể:
* Các microservice về xác thực, phân quyền
* Các microservice về API Gateway
* Các microservice nghiệp vụ như Log, SMS, ...

## Tính hỗ trợ

* Tất cả các lỗi hệ thống đều được xử lý tập trung và cho phép điều chỉnh các thông báo và ghi lại toàn bộ log lỗi.
* Hệ thống có các thành phần thu thập, giám sát tài nguyên của hệ thống sử dụng Zabbix, Prometheus

## Tính tích hợp

* Hệ thống cung cấp khả năng tích hợp với các hệ thống bên ngoài thông qua Restful API.
* Hệ thống cung cấp các SDK phục vụ tích hợp với mobile trên các nền tảng Android, iOS.
* Hệ thống cung cấp SDK phục vụ tích hợp với nền tảng web.
* Hệ thống cung cấp bộ tài liệu hướng dẫn tích hợp cho các ngôn ngữ lập trình.
* Hệ thống cung cấp tài liệu API phục vụ tích hợp.

## Tính khôi phục thảm họa

* Hệ thống có các thành phần backup dữ liệu và cấu hình của hệ thống sẵn sàng cho sự cố.
* Thời gian khôi phục hệ thống phụ thuộc vào mức độ nghiêm trọng của sự cố.

# PHỤ LỤC III: CÁC YẾU TỐ ẢNH HƯỞNG

Kiến trúc phần mềm được thiết kế ở trên bị ảnh hưởng bởi các yếu tố sau:

## Yếu tố về thời gian và ngân sách

* Thời gian làm dự án: Theo roadmap của sản phẩm.
* Ngân sách làm dự án – sản phẩm: Không .

## Yếu tố về công nghệ

* Theo định hướng của Công ty, kiến trúc của hệ thống xây dựng theo kiến trúc Microservice. Kiến trúc Microservice: ứng dụng được chia làm nhiều cấu phần, ít phụ thuộc lẫn nhau, các cấu phần này giao tiếp với nhau thông qua mạng. Để triển khai Microservice thì có nhiều cách khác nhau, có sự tham gia của nhiều phần mềm khác nhau, từ đó có nhiều kiến trúc triển khai khác nhau
* Hệ thống hướng tới đóng gói và chạy chương trình trên docker container, triển khai theo mô hình của Kubernetes.

## Yếu tố về nhân sự

* PO/PM: 01
* Scrum Master: 1
* DevOps: 2
* Kỹ sư lập trình: 5
* Kỹ sư tích hợp hệ thống: 02
* Kỹ sư lập trình Android: 02
* Kỹ sư lập trình IOS: 02
* Phân tích nghiệp vụ: 01
* Kiểm thử: 02

# PHỤ LỤC IV: CÁC NGUYÊN TẮC THIẾT KẾ

## Nguyên tắc phát triển

* Tuân thủ theo code convention của Java, .NET, Python, … theo từng service cụ thể.
* Thiết kế API tuân thủ theo HTTP Verb

(https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design)

* Tài liệu mô tả API tuân thủ theo Open API Specification

(http://spec.openapis.org/oas/v3.0.3)

* Source code khi đẩy lên hệ thống cần được thực hiện rà quét qua sonaqube.
* Source code tổ chức và lưu trữ tập trung trên hệ thống scm của Công ty.
* Tài liệu thiết kế, giải pháp được lưu trữ tập trung trên kms của Công ty.
* Sản phẩm phát triển áp dụng DevOpts.

## Nguyên tắc tổ chức

* Nguyên tắc về giao tiếp giữa các thành phần trong cấu trúc mã nguồn đối với toàn bộ solution:
* Các project, microservice được tổ chức theo dạng cây theo đúng mục đích.
  + Platform: tổ chức các microservice của toàn bộ hệ thống
  + Utils: Tổ chức các thư viện, tiện ích lập trình sử dụng cho toàn bộ các module
  + Portal: Tổ chức các module của portal cung cấp dịch vụ bao gồm self-service portal và admin portal
  + SDK: Tổ chức các module SDK giao tiếp với các hệ thống khác
* Nguyên tắc về giao tiếp giữa các thành phần trong cấu trúc mã nguồn đối với từng project:
* Được tổ chức theo các package và theo mô hình 3-layer bao gồm các gói chính:
  + Presentation layer: api interface, api factories, api implement
  + Bussiness layer: api service
  + Data Access layer: repositories
* Nguyên tắc về trạng thái của các thành phần trong phần mềm: các microservice trong hệ thống đều thiết kế theo state-less.
* Nguyên tắc về thiết kế lớp: toàn bộ class trong phần mềm được thiết kế tuân thủ theo nguyên lý SOLID.
* Nguyên tắc về nơi cài đặt xử lý nghiệp vụ: Toàn bộ phần xử lý nghiệp vụ sẽ được đặt trong phần bussiness layer.
* Nguyên tắc truy vấn dữ liệu: Toàn bộ các truy vấn dữ liệu phải sử dụng thông qua model. Đảm bảo không sử dụng cộng chuỗi để tranh Sql Injection.
* Nguyên tắc thiết kế CSDL:
  + Thiết kế cơ sở dữ liệu theo hướng attribute để đáp ứng được nhiều loại yêu cầu khác nhau.
  + Sử dụng các kỹ thuật indexing, partition để cải thiện hiệu năng của hệ thống.
  + Mỗi microservice sẽ có một cơ sở dữ liệu độc lập.
  + Chấp nhận lưu trữ thừa dữ liệu để phục vụ tối ưu truy vấn.