**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**



**XÂY DỰNG HỆ THỐNG GIÁO DỤC THEO HỆ TÍN CHỈ - SỬ DỤNG KIẾN TRÚC PHÂN TÁN MICROSERVICE**

***Nhóm 24 - Sinh viên thực hiện***

1. Nguyễn Thanh Sơn – 20106421
2. Phạm Thanh Sơn – 20106801
3. Đào Thị Ngọc Bích – 20058761

**MỤC LỤC**

[1. Giới thiệu Hệ thống Quản lý giáo dục theo tín chỉ 3](#_Toc167054663)

[2. Kiến trúc sử dụng: 4](#_Toc167054664)

[2.1. Kiến trúc phân tán Microservice 4](#_Toc167054665)

[2.2. Lý do sử dụng 6](#_Toc167054666)

[2.2.1. Lý do chọn kiến trúc phân tán Microservice: 6](#_Toc167054667)

[2.2.2. Lý do chia thành các Server và Database như hình trên: 6](#_Toc167054668)

[2.2.3. Ưu và nhược điểm của kiến trúc phân tán Microservice 9](#_Toc167054669)

[2.3. Các quy tắc khi áp dụng Microservice vào hệ thống 10](#_Toc167054670)

[2.3.1. Giảm thiểu tối đa các field không cần thiết cho mỗi service 10](#_Toc167054671)

[2.3.2. Đảm bảo chắc chắn dữ liệu gốc được sao chép tới các service khác 11](#_Toc167054672)

[2.4.3. Sử dụng mô hình phân tầng để tổ chức code 11](#_Toc167054673)

[2.4. Chức năng chính của hệ thống 11](#_Toc167054674)

[2.4.1. UserAuthenticationService 11](#_Toc167054675)

[2.4.2. DangKyHocPhanService: 11](#_Toc167054676)

[2.4.3. LichHocService: 11](#_Toc167054677)

[2.4.4. KetQuaHocTap Service: Quản lý kết quả học tập 12](#_Toc167054678)

[3. Các lược đồ Diagram 12](#_Toc167054679)

[3.1. Use case diagram 12](#_Toc167054680)

[3.2. Class diagram 13](#_Toc167054681)

[3.3. Component diagram 13](#_Toc167054682)

[3.4. Package diagram 14](#_Toc167054683)

[3.5. Deployment diagram 15](#_Toc167054684)

[4. Một số chức năng đã thực hiện 15](#_Toc167054685)

[5. Thực thi trên giao diện 19](#_Toc167054686)

# 1. Giới thiệu Hệ thống Quản lý giáo dục theo tín chỉ

Hệ thống Giáo dục theo tín chỉ quản lý tiến trình học, kết quả học tập của sinh viên. Cho phép sinh viên quyết định đăng ký thời điểm học, tiến trình nhanh hay chậm bằng cách cho sinh viên chọn lớp học được giáo vụ mở sẵn.

Cách sử dụng hệ thống: Giáo vụ sẽ thêm dữ liệu của sinh viên vào hệ thống theo ngành và khóa. Hệ thống trả về cho mỗi thông tin sinh viên một Mã số sinh viên(MSSV). Sinh viên sử dụng mã đó và và mật khẩu mặc định “1111” để đăng nhập hệ thống.

Vào ngày mở đăng ký, khi đăng nhập vào, sinh viên có thể xem được những môn học mình có thể đăng ký (chương trình khung cho mỗi ngành và khóa được giáo vụ tạo từ trước). Một học phần có thể có nhiều học phần tiên quyết, có thể là học phần tự chọn. Sinh viên có thể đăng ký tối đa 30 tín chỉ trong 1 học kỳ, và các lịch học không được trùng nhau. Sau khi đăng ký, sinh viên có thể xem lịch học của mình. Môn học có thể có chỉ có lý thuyết hoặc có thực hành. Mỗi tín chỉ lý thuyết tương ứng 5 buổi học, mỗi tính chỉ thực hành tương ứng 10 buổi học. Hệ thống cũng hỗ trợ giáo vụ quản lý phòng loại lý thuyết hay thực hành. Giảng viên cũng có thể xem lịch dạy của mình.

Sau khi học xong, giảng viên sẽ gửi bảng điểm lớp học cho giáo vụ nhập điểm. Kết quả học phần có 5 cột điểm với môn lý thuyết, tương ứng các cột thường kỳ 1, thường kỳ 2, thường kỳ 3, giữa kỳ, cuối kỳ; 8 cột với môn có thực hành thường kỳ 1, thường kỳ 2, thường kỳ 3, giữa kỳ, thực hành 1, thực hành 2, thực hành 3, cuối kỳ. Điểm phải >=1 và <=10. Sau khi nhập điểm, sinh viên có thể vào xem kết quả học tập của mình. Bao gồm: điểm trung bình hệ số, xếp loại hệ chữ, kết quả đạt/rớt môn. Điểm trung bình >= 3 và cuối kỳ >= 4 sẽ qua môn, ngược lại.

Sau khi hoàn thành chương trình học, sinh viên sẽ đăng ký xét tốt nghiệp. Hệ thống sẽ giúp giáo vụ kiểm tra sinh viên đã hoàn thành chương trình trong niên giám chưa, sau đó đánh dấu là sinh viên đã tốt nghiệp. Kết quả học tập của sinh viên vẫn được lưu trên hệ thống trong vòng 20 năm.

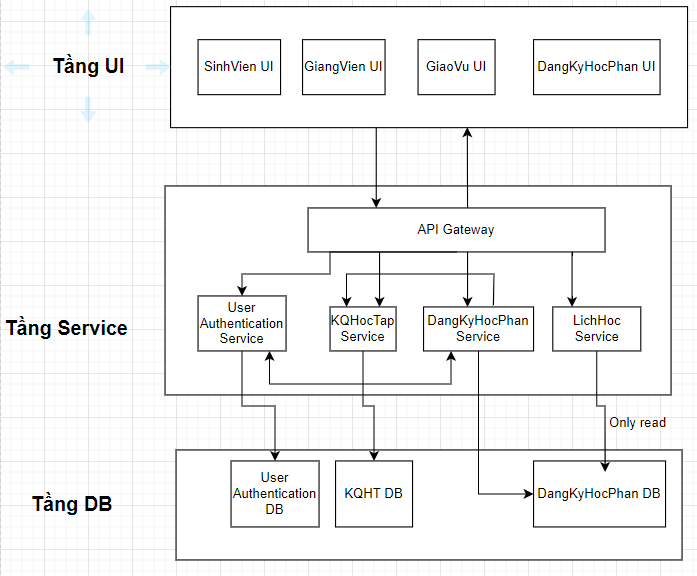
# 2. Kiến trúc sử dụng:

* Kiến trúc phân tán Microservice.
* Khuyến khích sử dụng cơ sở dữ liệu có cấu trúc cho DangKyHocPhanDB và cơ sở dữ liệu không cấu trúc cho UserAuthenticationDB và KetQuaHocTapDB.

## 2.1. Kiến trúc phân tán Microservice

**Microservice** là một loại hệ thống phân tán. Trong đó, các chức năng của hệ thống lớn được tổ chức thành các thành phần nhỏ hơn gọi là service. Các server được triển khai độc lập (independent deployability), có thể có cơ sở dữ liệu riêng và kết nối với nhau qua mạng. Chúng làm việc cùng nhau tạo thành một hệ thống.

**Hệ thống** được chia thành các Service và Database như sau:



Request từ tầng UI sẽ được gửi tới API Gateway, kiểm tra chứng thực rồi điều phối xuống tầng Service. Sử dụng Eureka để quản lý các service. Dùng RestTemplate để các service gọi nhau. Trong đó:

* DangKyHocPhanService gọi UserAuthenticationService để tạo bản sao Nganh.
* UserAuthenticationService gọi DangKyHocPhanService để tạo bản sao SinhVien và GiangVien
* DangKyHocPhanService gọi KetQuaHocTapService để tạo bản sao HocPhan

## 2.2. Lý do sử dụng

### 2.2.1. Lý do chọn kiến trúc phân tán Microservice:

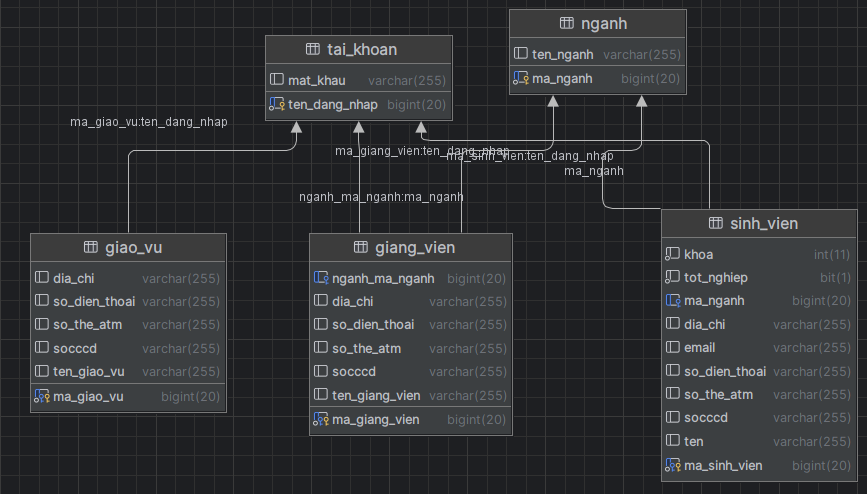
Thứ nhất, vào những ngày thường, hệ thống có lượt truy cập ít và thưa thớt. Nhưng vào các ngày mở đăng ký học phần, số lượng sinh viên truy cập vào hệ thống sẽ nhiều và liên tục. Khi có quá nhiều yêu cầu được gửi đến mà máy chủ vật lý của trường không thể xử lý kịp sẽ gây ra hiện tượng quá tải (overload), dẫn đến việc hệ thống trở nên chậm chạp, hoặc thậm chí là không phản hồi. Điều này sẽ ảnh hưởng tới các dịch vụ khác.

Thứ hai, các hệ thống trường học thường có nhu cầu mở thêm nhiều dịch vụ khác như: tạo các bài kiểm tra trong lớp học phần, tra cứu và mượn sách thư viện theo ngành học/môn học,... nên có nhu cầu chia sẽ dữ liệu. Nhưng nếu chia sẽ một khối lớn dữ liệu, có thể sẽ gây nặng nề cho dịch vụ muốn thêm vào.

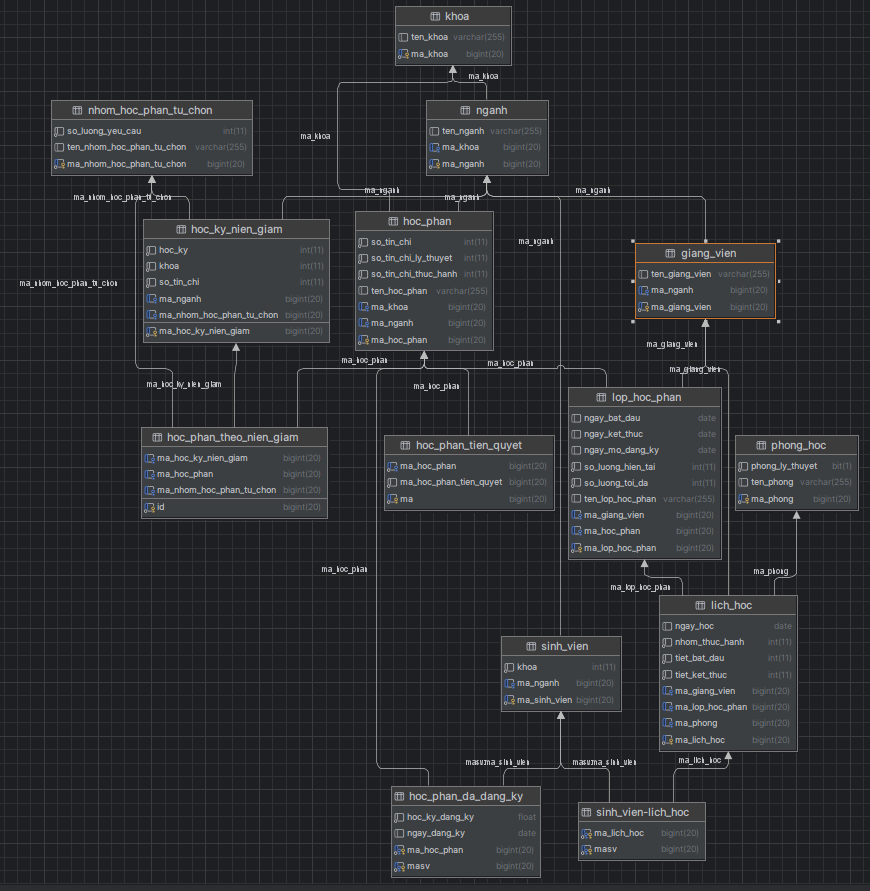
Microservice đáp ứng được 2 khó khăn trên: *tính độc lập và dễ mở rộng*.

### 2.2.2. Lý do chia thành các Server và Database như hình trên:

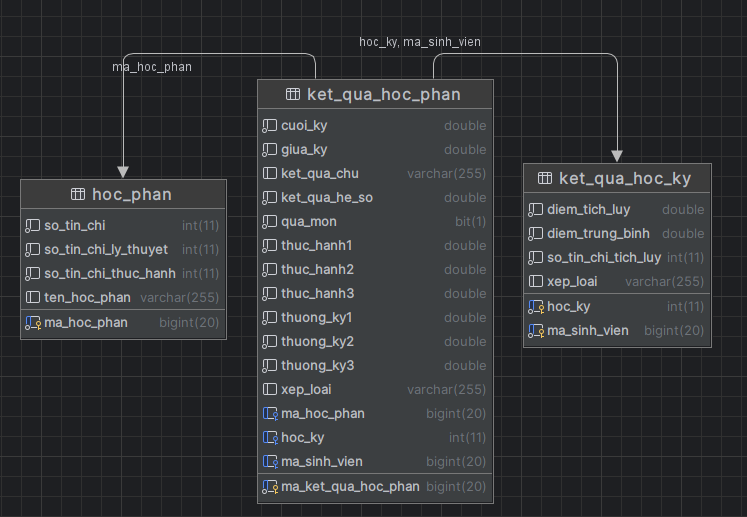
***Database được chia theo mối quan hệ giữa các thực thể để thực hiện 1 chức năng***. Sinh viên là đối tượng chính và đông nhất sử dụng hệ thống này, nên các chức năng chính của hệ thống là: Đăng ký học phần, xem lịch học, và xem kết quả học tập. Dựa theo các chức năng đó, có thể chia dữ liệu theo cấu trúc như sau:

**UserAuthenticationDB:** Quản lý thông tin User, thực hiện chức năng Đăng nhập. Đây cũng là nơi chứa các thông tin nhạy cảm (email, số điện thoại, địa chỉ, tài khoản ngân hàng) nên càng cần được tách rời với cơ sở dữ liệu chung và lưu ở nơi an toàn.  


**DangKyHocPhanDB**: Quản lý các môn học sinh viên có thể đăng ký theo ngành và khóa. Lưu trữ thông tin về các môn sinh viên đã đăng ký cũng như lịch học của sinh viên sau khi đăng ký thành công.



**KetQuaHocTapDB:** Quản lý kết quả học tập của sinh viên. Cho phép giáo vụ nhập điểm và sinh viên xem kết quả học tập.

****

\*Vì UserAuthenticationDB và KetQuaHocTapDB không sử dụng nhiều ràng buộc quan hệ, và có thông tin có thể bỏ trống nên khuyến khích dùng hệ cơ sở dữ liệu không cấu trúc.

***Service được chia theo tính độc lập cho những chức năng chính và lưu lượng người dùng vào 1 thời điểm***. Lý do tách xem lịch học và đăng ký học phần thành 2 service riêng: nhằm tránh trường hợp không xem được lịch học vào ngày mở đăng ký học phần khi máy chủ không xử lý kịp các request gửi đến.

### 2.2.3. Ưu và nhược điểm của kiến trúc phân tán Microservice

* + 1. **Ưu điểm**

**- Tính sẵn dùng cao/Khả năng chịu lỗi (availability/reliability):** Nếu một dịch vụ bị lỗi, các dịch vụ khác vẫn có thể tiếp tục hoạt động bình thường.

**- Khả năng mở rộng/thay đổi dễ dàng (scalability):** Hệ thống được chia thành các đơn vị triển khai riêng biệt, nên khi cập nhật hay thêm mới một dịch vụ sẽ ít bị ảnh hưởng tới các dịch vụ khác.

- **Phân phối tải**: Hệ thống có thể phân phối tải làm giảm áp lực cho từng máy chủ.

**- Tính độc lập:** Các service có thể triển khai độc lập, ít bị phụ thuộc vào service khác. Giúp dễ quản lý dữ liệu. Hoặc mỗi service có thể sử dụngcác công nghệ khác nhau phù hợp với chức năng của nó

* + 1. **Nhược điểm**

- Hệ thống ưu tiên tính sẵn dùng, nên sẽ tạo ra các bản sao dữ liệu cần thiết cho các server, điều này dẫn đến việc dữ liệu bị trùng lặp, tốn tài nguyên cho việc lưu trữ.

- Không thể sử dụng transaction giữa các server với nhau. Nếu tính toán không kỹ lưỡng sẽ dẫn đến việc không đồng nhất dữ liệu, ảnh hưởng tới các chức năng chính của hệ thống.

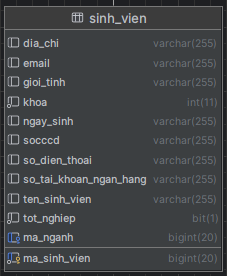
- Một số chức năng phải đi qua nhiều tầng hơn nên có thể sẽ chậm hơn khi dùng kiến trúc nguyên khối.

## 2.3. Các quy tắc khi áp dụng Microservice vào hệ thống

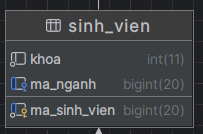
### 2.3.1. Giảm thiểu tối đa các field không cần thiết cho mỗi service

Các bản sao entity ở service khác chỉ sử dụng các field cần thiết cho chức năng của nó. Ví dụ:

Bản gốc SinhVien của UserAuthenticationService:



Bản sao SinhVien của DangKyHocPhanService:



### 2.3.2. Đảm bảo chắc chắn dữ liệu gốc được sao chép tới các service khác

Mã Id của dữ liệu gốc được sinh ra ngẫu nhiên. Sau khi lưu được dữ liệu đó, service chứa dữ liệu gốc phải đảm bảo bản sao (chứa Id được sinh ra) phải được lưu tới các service khác. Nếu request thất bại thì phải thực hiện lại bởi dữ liệu được sao chép có thể là preference key của service được sao chép dữ liệu

### 2.4.3. Sử dụng mô hình phân tầng để tổ chức code

Vì các chức năng của hệ thống sử dụng rất nhiều logic, điều kiện và ràng buộc nên code phải được tổ chức có cấu trúc rõ ràng. Sử dụng class DTO để nhận và gửi dữ liệu. Controller ở Presentation Layer nhận dữ liệu từ bên ngoài, gửi xuống Business Layer để xử lý logic, rồi đưa xuống Data Access Layer để lưu dữ liệu.

## 2.4. Chức năng chính của hệ thống

### 2.4.1. UserAuthenticationService

* Thêm sinh viên
* Thêm giảng viên
* Đăng nhập

### 2.4.2. DangKyHocPhanService:

* Cho phép sinh viên xem danh sách học phần có thể đăng ký
* Cho phép sinh viên đăng ký lớp học phần

### 2.4.3. LichHocService:

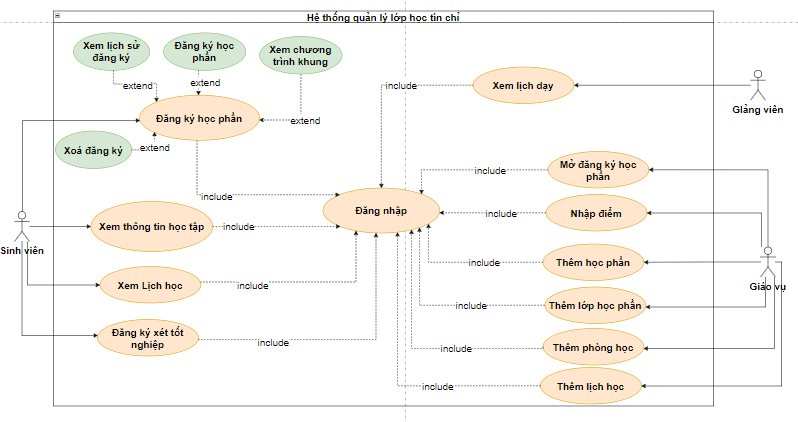
* Cho phép sinh viên xem lịch học
* Cho phép giảng viên xem lịch dạy

### 2.4.4. KetQuaHocTap Service: Quản lý kết quả học tập

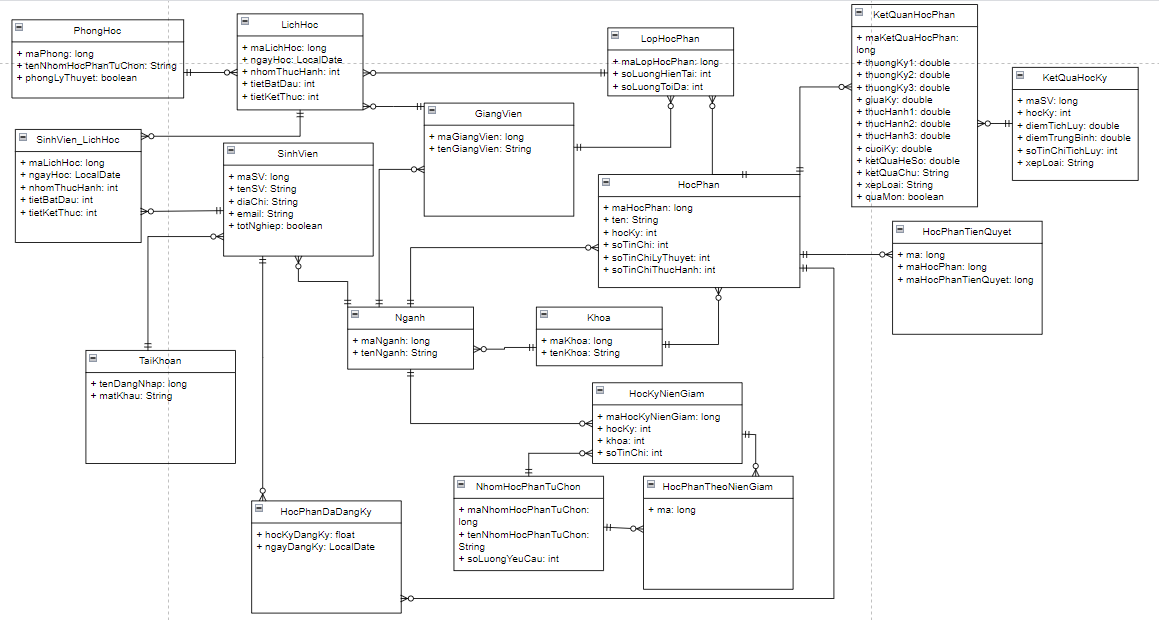
* Cho phép giáo vụ nhập điểm cho lớp học phần
* Cho phép sinh viên xem kết quả học tập của mình
* Cho phép giáo vụ xét tốt nghiệp cho sinh viên

# 3. Các lược đồ Diagram

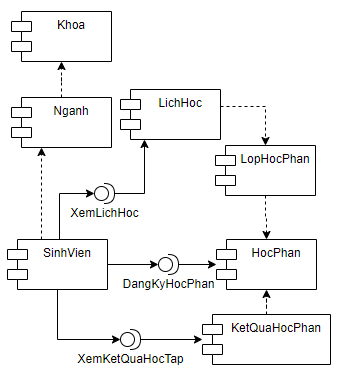
## 3.1. Use case diagram

****

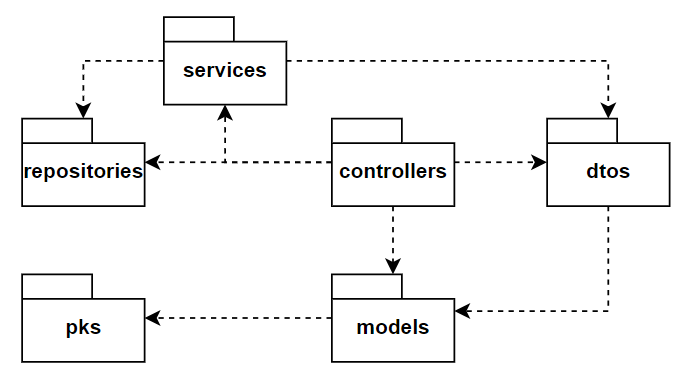
## 3.2. Class diagram



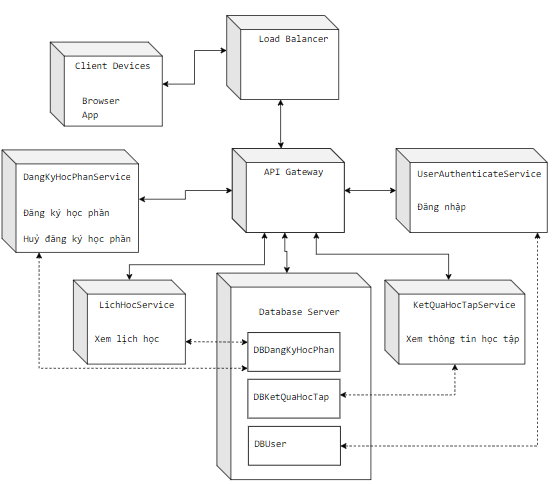
## 3.3. Component diagram

****

## 3.4. Package diagram

****

## 3.5. Deployment diagram

****

# 4. Một số chức năng đã thực hiện

**\* Nhập điểm cho lớp học phần:**

**- Path:**

**/**KetQuaHocTapService/KetQuaHocTap/nhapDiemChoLopHocPhan

**- RequestBody:**

{

"maHocPhan": 1,

"hocKy":1,

"listKetQuaSinhVien": [

{

"maSinhVien": 1,

"listDiem": [9,9,9,9,9]

},

{

"maSinhVien": 2,

"listDiem": [8,8,8,8,8]

}

]

}

**- Các bộ lọc:**

* Kiểm tra điểm nhập vào có hợp lệ không. Điểm phải <=10 hoặc >=1. Nếu dữ liệu không hợp lệ, hệ thống trả về danh sách các mã sinh viên có điểm không hợp lệ.
* Điểm phải là 5 cột với môn lý thuyết, và 8 cột với môn thực hành

**- Kết quả đạt được:**

* ***Lưu được điểm cho sinh viên*** tương ứng với môn học. Nếu sinh viên có kết quả đã tồn tại thì sẽ sửa lại thành kết quả mới
* ***Tính điểm, xếp loại*** theo tùy loại môn lý thuyết hay thực hành:

Môn lý thuyết: **((TK1+TK2+TK3)/3\*2*+*GK\*3+CK\*5)/10**

Môn thực hành: **(((TK1+TK2+TK3)/3\*2*+*GK\*3+CK\*5)\*số tín chỉ lý thuyết**

**+ (TH1+TH2+TH3)/3\*số tín chỉ thực hành)/tổng số tín chỉ**

* ***Tạo kết quả học kỳ***: Nếu sinh viên chưa có kết quả môn học trong học đó, hệ thống sẽ tạo một bản ghi dữ liệu KetQuaHocKy, đã có kết quả trong học kỳ đó thì sẽ sửa lại bản ghi KetQuaHocKy. Như vậy *mỗi lần sinh viên muốn xem điểm, hệ thống không cần tính toán lại kết quả của học kỳ đó*.

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**\* Xem kết quả học tập của sinh viên**

**- Path: /**KetQuaHocTapService/KetQuaHocTap/getKetQuaSinhVien/{maSinhVien}

**/**KetQuaHocTapService/KetQuaHocTap/getKetQuaSinhVien/1

- **Response:**

{

"diemTrungBinh": 7.6,

"soTinChiTichLuy": 10,

"xepLoai": "Khá",

"ketQuaHocKySinhViens": [

{

"hocKy": 1,

"ketQuaHocPhanDetailRespons": [

{

"listDiem": [10.0, 10.0, 10.0, 10.0, 0.0, 0.0,0.0,10.0],

"ketQuaHeSo": 10.0,

"ketQuaChu": "A+",

"xepLoai": "Xuất sắc",

"quaMon": **true**,

"maHocPhan": 1,

"tenHocPhan": "Tieng viet thuc han"

},

{

"listDiem": [5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0],

"ketQuaHeSo": 5.0,

"ketQuaChu": "C",

"xepLoai": "Trung bình",

"quaMon": **true**,

"maHocPhan": 2,

"tenHocPhan": "OOP"

}

],

"diemTrungBinh": 7.5,

"soTinChiTichLuy": 6,

"xepLoai": "Khá"

},

{

"hocKy": 2,

"ketQuaHocPhanDetailRespons": [

{

"listDiem": [7.0,7.0,7.0,7.0,7.0,7.0,7.0,9.0],

"ketQuaHeSo": 7.75,

"ketQuaChu": "B",

"xepLoai": "Khá",

"quaMon": **true**,

"maHocPhan": 3,

"tenHocPhan": "WWW"

}

],

"diemTrungBinh": 7.75,

"soTinChiTichLuy": 4,

"xepLoai": "Khá"

}

]

}

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**\* Thêm học phần vào niên giám:**

**- Path:**

/DangKyHocPhanService/HocPhan/createHocKyNienGiam

**- Body:**

{

"maNganh": 1,

"khoa": 16,

"hocKy": 1,

"hocPhanTheoNienGiam": [1,2]

}

**- Điều kiện:** Mã ngành và mã học phần đã tồn tại

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**\* Xem niên giám:**

**- Path:**

/DangKyHocPhanService/HocPhan/getNienGiamByNganhAndKhoa?maNganh=1&khoa=16

**- Response:**

[

{

"maHocKyNienGiam": 1,

"hocKy": 1,

"danhSachHocPhan": [

{

"maHocPhan": 1,

"tenHocPhan": "Big data",

"soTinChiLythuyet": 3,

"soTinChiThuchanh": 1,

"batBuoc": **false**

},

{

"maHocPhan": 2,

"tenHocPhan": "Lap trinh huong doi tuong",

"soTinChiLythuyet": 2,

"soTinChiThuchanh": 1,

"batBuoc": **false**

}

]

},

{

"maHocKyNienGiam": 2,

"hocKy": 2,

"danhSachHocPhan": [

{

"maHocPhan": 4,

"tenHocPhan": "He co so du lieu",

"soTinChiLythuyet": 3,

"soTinChiThuchanh": 1,

"batBuoc": **false**

},

{

"maHocPhan": 5,

"tenHocPhan": "Mang may tinh",

"soTinChiLythuyet": 3,

"soTinChiThuchanh": 0,

"batBuoc": **false**

}

]

}

]

# 5. Thực thi trên giao diện

* + - 1. Xem điểm

A screenshot of a computer

Description automatically generated

* + - 1. Xem lịch học

A screenshot of a computer

Description automatically generated

* + - 1. Đăng ký học phần

A screenshot of a computer

Description automatically generated