# Câu hỏi sau phỏng vấn:

1.Em đã làm những dự án nào ở GMO trong những dự án này?( dự án đã ghi trong cv).

2.RequestMapping và RestController là gì.

3. em dùng có dùng Procedures không khi nào thì dùng. em có thể mô tả 1 lần sử dụng không. lợi ích của việc sử dụng

4 giải pháp khi làm việc với dữ liệu lớn.

5. em làm Unit chưa. Unit gồm những gì.

6. Dependency injection là gì? Nêu ví dụ luôn trong dự án đã làm.

1. Có biết về OOP

2. Miêu tả 4 tính chất về OOP, ví dụ về kế thừa trong java, một class có thể kế thừa nhiều class không.

3. Hiểu gì về spring boot

4. Đang làm module nào trong dự án hiện tại

5. Thấy chức năng nào là khó nhất trong các dự án đã làm

6. Có biết viết tài liệu bằng tiếng anh không, đối tượng khách hàng là ai?

7. Arrow Function?

8. Phân biệt @RequestMapping và @GetMapping

9. Thymeleaf có sử dụng được trong các file xml không?

10. Tài liệu DD có những thành phần gì? UT làm những gì?

11. Có sử dụng thư viện cho thiết kế font end không? Có gặp lỗi gì khi sử dụng bootstrap? Có sử dụng datatable không? Sort trong datatable, Filter trong datatable?

12. @Autowired là gì.

13. Các module của Spring framework mà em biết

[**Câu hỏi sau phỏng vấn:**](#_1vasc2jw78ly) **1**

[**Giới thiệu**](#_n0unffvakyhh) **3**

[**Câu hỏi về backend:**](#_qii9ujm162sb) **4**

[Giải thích sự khác nhau giữa interface và abstract](#_kp4zg9d0c88o) 4

[Giải thích về những thành phần mới trong Java 8](#_gdsv56md2xpf) 4

[Giải thích Lambda và những ưu điểm](#_piiv9o1gmp11) 4

[Giải thích về Stream API](#_g9c8d37cewfs) 5

[Giải thích sự khác nhau giữa @Controller, @RestController và @AdvanceController](#_vf7jakkc24et) 5

[Giải thích cơ chế vận hành của @Autowired](#_26ow68t9kbcc) 5

[Giải thích về WebSocket](#_fjjz1usxvbp5) 5

[Giải thích về Generic, Annotation và cơ chế Reflection](#_weea52i2c8pc) 6

[Câu hỏi tình huống: Có nên xóa các code cũ khi bảo trì không?](#_f9fjrit59z1) 6

[Câu hỏi tình huống: Có nên để hai hoặc nhiều chức năng ngắn của chương trình vào chung 1 class không](#_akyct3x5vv2n) 6

[Giải thích khi nào nên sử dụng interface, khi nào thì dùng Abstract](#_nkh2wih6owu3) 6

[Những khó khăn gặp phải khi xử lý dữ liệu lớn](#_jjuytdjvgdhn) 6

[Hỏi về design pattern: những pattern nào hay dùng và cơ chế hoạt động của nó](#_yvyuhx21y605) 7

[Nêu trường hợp ví dụ sử dụng generic](#_2su0sommwri4) 7

[Giải thích về Socket](#_tqkxryr8ev3m) 7

[Giải thích về Thread và multithread Java](#_ptnw3b6aasj4) 7

[Giải thích về dependency Injection](#_87t8ab95w39z) 8

[Giải thích về inverse of control](#_wvdycghxvt19) 8

[Giải thích về transaction](#_phq0clsq76ki) 8

[Giải thích khi nào nên sử dụng interface, khi nào thì dùng Abstract](#_wcc1zitdlfau) 8

[Giải thích về Binary Search Tree trong java](#_6pcmtxl348yo) 8

[Collection trong java](#_7ajbbgpals7l) 9

[Design pattern là gì? , đã làm việc với những design pattern nào?](#_pk9i36vrz380) 9

[Lý do nên sử dụng Design Patterns](#_wtdmf46xgf9l) 9

[spring security](#_nnwqbzke0p9s) 9

[spring data jpa](#_3v9vp89qcrik) 10

[spring boot](#_2y98t3x8agpr) 10

[Giải thích về UNION](#_caqxj1bj2tcg) 10

[Phân biệt JOIN, INNER JOIN, RIGHT JOIN, LEFT JOIN.](#_pmanu4q3rjv1) 10

[Cách sử dụng của Procedure](#_3fhi94q5x74l) 11

[Cách debug trong Postgres](#_ms2owvh92hjx) 11

[Giải thích về câu lệnh lồng, và những chú ý cần lưu ý khi sử dụng câu lệnh lồng.](#_iitjnmspf8x0) 11

[Lệnh LIMIT trong SQL](#_pvejcejy9tc7) 11

[giải thích về ORDER BY](#_382it9bfhnih) 11

[**Câu hỏi về font-end:**](#_1uk8evf38zla) **12**

[Giải thích cách vận hành của ReactJs và vòng đời trong ReactJS](#_9zu4k2bid0b6) 12

[Cách vận hành:](#_4yhsxmjbag5h) 12

[Life cycle: ReactJS - Tìm hiểu về Component Life Cycle (viblo.asia)](#_mdsybfvq1zzd) 12

[Giải thích về Redux](#_75dwvq8hlz94) 12

[Giải thích ngắn về cơ chế vận hành của Bootstrap: grid line, ....](#_ujpkr1pijz05) 13

[Mô tả nghiệp vụ của các dự án đã tham gia?.](#_tx40tf4xw649) 13

[Đã làm việc với xuất file có dung lượng lớn chưa? (công ty đang chưa có giải pháp để xử lý?)](#_i80yzaz3lycp) 13

[Design pattern là gì?. được chia làm những nhóm nào?. Đã làm việc với các loại loài.](#_urroscxviczp) 13

[Hiểu gì về lập trình OOP?.](#_ev4xgikafhu) 13

[Hiểu gì về nguyên lý SOLID?. Dùng nó ở chỗ nào trong dự án.(Giải thích từng nguyên lý).](#_mq8kajszgc0n) 13

[Dùng gì để quản lý source code.](#_72fowupf64dw) 14

[Làm thế nào để lấy code mới nhất?.](#_7emdynlexuk3) 14

[Git rebase là gì?](#_b5zgioc3y09g) 14

[Git cherry pick ?](#_5tz6kmids4qq) 14

[Cách xử lý conflict?](#_o4ceqkuo99r3) 14

[Với dữ liệu lớn để tìm kiếm nhanh thì làm thế nào? (Nếu dùng index thì cách đánh index thế nào?)](#_g9svq2cgkdjs) 14

[Trong cv có mô tả dùng JPA, Hibernate, MYBATIS, SPRING DATA JPA thì có điểm gì khác nhau khi dùng ?](#_ux9enhtq10tc) 14

[Về Unit Test thì viết code test hay chỉ test trên giao diện thôi?. Nếu viết code thì viết test ở những tầng nào(Controller, Repository,Service ?).](#_nv5sfvmn8uir) 14

[Viết testcase thì căn cứ vào đâu để viết? có tuân theo nguyên tắc nào không?](#_fotm1tt3gdoq) 14

[Cách quản lý task làm việc thế nào?](#_ym0kzp715ro9) 15

[Làm thế nào biết được mình đã code xong chức năng đấy?](#_pd3hluv6amct) 15

[Đã từng sử dụng biểu đồ gì để design detail](#_x7ualiyz2mb2) 15

[**Front-end:**](#_yhw299m5dquj) **15**

[Có biết về html css js không?. đã dùng js để validate chưa?](#_prgtods9404b) 15

[Có biết về bootstrap không?. grid, flex,...](#_ow0etend9myb) 15

[Phân biệt Spring boot và Spring MVC](#_1qany64t1vo3) 15

[Phân biệt Spring & Struts](#_6uo9ng3yp60a) 16

[https, http, phương thức nào của http lưu được cookie](#_nx1wtbuu2gti) 16

[Một vài câu hỏi liên quan như annotation, transaction...](#_ooifx5usjqvp) 16

[Kĩ thuật : multi thread, timer, socket...](#_5bqrn9nc040j) 16

[Sử dụng biểu thức Lambda?](#_fqwtr1b77syy) 16

[Callback function là gì? Tại sao lại sử dụng callback?](#_tj0x97na3lg8) 16

[Design pattern là gì? Em đã làm việc với những design pattern nào?](#_3v4jq61j9v10) 16

[Restful API?](#_f0w79p7jdzdc) 17

[Mô hình Microservice?](#_vid33fus5lsj) 17

[Sử dụng JPA thế nào? Làm thế nào để lấy dữ liệu từ db bằng JPA?](#_vhfuudg8qj15) 17

[Làm thế nào để lấy dữ liệu từ nhiều bảng khác nhau bằng jpa khi không có entity phù hợp?](#_2dc58s4gmcur) 17

[Sử dụng Thymeleaf như thế nào? Lấy ví dụ về cách truyền dữ liệu](#_2rcp5j2c2akb) 17

[Sử dụng api như thế nào? lấy ví dụ về cách truyền dữ liệu](#_hy2vpio8lpsp) 17

[Làm thế nào để tự động quét các user hết hạn và thông báo cho họ?](#_j2arw3bbqx4j) 17

[Giải thích về @Controller, @RestController](#_fphze24tph2a) 17

# 

# Giới thiệu

-nhấn mạnh dự án java

-Ho va ten

-Số năm kinh nghiệm 1 => 1,5 năm

-Đã tham gia mấy dự án.

-Dự án thứ 1 => 3 : làm về gì, trong bao lâu, nhiệm vụ trong dự án.

-Tổng kết :

+qua các dự án đã làm các phase (DD, CD, UT), làm GIT, Task (Jira hoặc Redmine), communication (Chatwork hoặc Slack..)

+Các dự án đã từng làm chủ yếu back end, front end

# **Câu hỏi về backend:**

## Giải thích sự khác nhau giữa interface và abstract

| Abstract | Interface |
| --- | --- |
| - Đơn kế thừa  - Abstract class có các biến **final, non-final, static and non-static**.  - Abstract class **có thể cung cấp nội dung cài đặt cho phương thức của interface**.  - Abstract class có phương thức **abstract** (không có thân hàm) và phương thức **non-abstract** (có thân hàm).  - Abstract có thể có private, protected, public | - Đa kế thừa  - Interface chỉ có các biến **static và final**.  - Interface không có thân hàm  - Interface **không thể cung cấp nội dung cài đặt cho phương thức của abstract class**.  - Interface chỉ có phương thức **abstract**. Từ java 8, nó có thêm **các phương thức default và static**.  - Các biến và phương thức trong interface chỉ có public |

[*Link tham khảo*](https://viettuts.vn/java/su-khac-nhau-giua-abstract-class-va-interface)

## Giải thích về những thành phần mới trong Java 8

[*Link tham khảo*](https://www.youtube.com/watch?v=T1vipqu6mOM)

## Giải thích Lambda và những ưu điểm

* Giải thích: Lambda là một hàm không có tên, các tham số và phần thực thi có thể được phân biệt bởi dấu “->”.
* Lambda Expression là một hàm không có tên và không thuộc bất kỳ lớp nào, không có phạm vi truy cập (private, public hoặc protected), không khai báo kiểu trả về.
* Ưu điểm: Giảm số lượng dòng code, hỗ trợ thực thi code một cách tuần tự hoặc song song.

[*Link tham khảo*](https://www.youtube.com/watch?v=T1vipqu6mOM)

## Giải thích về Stream API

* Stream API giúp cho việc thao tác với tập hợp trở nên dễ dàng hơn thông qua các phương thức lambda. Các method trong stream api đều là các functional interface

[*Link tham khảo*](https://www.youtube.com/watch?v=cDJn3oo9Z3E)

## Giải thích sự khác nhau giữa @Controller, @RestController và @AdvanceController

**- @Controller**: **@Controller** là một annotation ở class. Là một chuyên môn hóa của **@Component**. Nó đánh dấu class là một class để xử lý request web. Nó thường được sử dụng để phục vụ các request từ UI. Mặc định thì nó trả về một chuỗi cho biết router nào cần redirect.

***-* @ResponseBody** annotation được dùng để thông báo với controller rằng Java Object trả về cho client sẽ tự động ánh xạ sang JSON

- **@RestController = @Controller + @ResponseBody**

**- @ControllerAdvice :** đánh dấu class xử lý các exception được ném ra

| **@Controller** | **@RestController** | **@ControllerAdvice** |
| --- | --- | --- |
| - Đánh dấu class là Spring MVC Controller  - Nhận request rồi return về một viewresolver và bắt buộc view đó phải tồn tại, và điều hướng, chuyển tiếp yêu cầu | - Đánh dấu là một Rest controller sử dụng trong các dịch vụ RESTful webservice  - Kết hợp **@ResponseBody** với **@Controller** và dữ liệu trả về là 1 object được mapping dạng json | Đánh dấu class xử lý các exception được ném ra |

## Giải thích cơ chế vận hành của @Autowired

**@Autowired** đánh dấu cho Spring biết rằng sẽ tự động inject bean tương ứng được IOC Container quản lý vào vị trí được đánh dấu. Các bean đó đã được spring nạp thông qua các annotation @Component, @Service @Component @Bean

## Giải thích về WebSocket

- Websocket là loại giao tiếp 2 chiều sử dụng giao thức TCP để truyền gửi dữ liệu giữa client với server

- Các sự kiện của websocket

1. onopen: Sự kiện xảy ra khi kết nối Socket được thành lập

2. onmessage: Sự kiện xảy ra khi client nhận dữ liệu từ server

3. onerror: Sự kiện xảy ra khi có bất kỳ lỗi nào trong giao tiếp

4. onclose: Sự kiện xảy ra khi kết nối được đóng

- Websocket hiện tại đang tồn tại 2 phương thức cơ bản như sau:

+ Send: Đây là phương thức send (data) dùng để gửi dữ liệu tới server

+ Close: Đây là phương thức Close () chỉ kết nối đang tồn tại.

[*Link tham khảo*](https://bizflycloud.vn/tin-tuc/websocket-la-gi-uu-nhuoc-diem-cua-websocket-khi-su-dung-lam-phuong-thuc-giao-tiep-trong-moi-truong-internet-20210122155209401.htm)

## Giải thích về Generic, Annotation và cơ chế Reflection

* Generic: sử dụng khi cần thao tác nhiều kiểu dữ liệu cùng 1 lúc hay dùng để build các hàm chung
* Generics là cách thức lập trình tổng quát cho phép một object hoạt động với nhiều kiểu dữ liệu khác nhau

[*Link tham khảo*](https://giasutinhoc.vn/lap-trinh/lap-trinh-java-co-ban/generic-trong-java-bai-6-2/)

* Annotations là một tính năng mới được thêm vào từ java 5. Nó được hiểu là một dạng chú thích hoặc metadata được chèn vào code java để đặc tả dữ liệu cho một đối tượng, giá trị nào đó. Nó có thể được xử lý tại thời điểm compile hoặc tại thời điểm runtime.

[*Link tham khảo*](https://codelearn.io/sharing/annotation-trong-java-la-gi)

Reflection sử dụng khi có 1 object nhưng không biết nó thuộc class nào, có thể khai phá những thông số, thuộc tính của class

## Câu hỏi tình huống: Có nên xóa các code cũ khi bảo trì không?

* Không nên xóa, xem code cũ sẽ hiểu về cách xử lí, dựa vào đó để tìm lỗi và tối ưu code hoặc khi thay đổi có 1 version mới nhưng không đáp ứng được như version cũ thì có thể quay lại tránh những lỗi phát sinh

## Câu hỏi tình huống: Có nên để hai hoặc nhiều chức năng ngắn của chương trình vào chung 1 class không

* Không nên vì mỗi class sẽ đảm nhiệm 1 chức năng duy nhất để dễ bảo trì và phát triển

## Giải thích khi nào nên sử dụng interface, khi nào thì dùng Abstract

* Không chú trọng vào cấu trúc của đối tượng mà chú trọng vào hành vi của đối tượng thì dùng interface, khi các lớp cần liên hệ với nhau thì dùng abstract

1. Khi nào sử dụng List khi nào sử dụng ArrayList

Khi đã xác định được dạng dữ liệu rồi thì sẽ sử dụng ArrayList, còn nếu chưa thì sẽ sử dụng List vì List là 1 interface nên sẽ tự tìm được kiểu phù hợp với dữ liệu

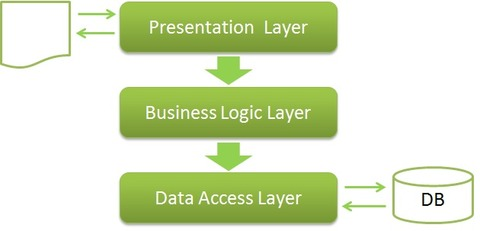
## Những khó khăn gặp phải khi xử lý dữ liệu lớn

Khó khăn: Tốc độ xử lý chậm, quá nhiều truy vấn khiến hệ thống quá tải có thể sập hệ thống.

Giải pháp:

* Chia nhỏ dữ liệu ra theo từng mục riêng biệt
* Phân vùng dữ liệu cần xử lý
* Backup dữ liệu quá cũ ra và xóa trên cơ sở dữ liệu
* Đặt chỉ mục index cho trường dữ liệu bảng cần xử lý

## Hỏi về design pattern: những pattern nào hay dùng và cơ chế hoạt động của nó

* **Singleton**: khởi tạo đối tượng 1 lần duy nhất và cung cấp 1 cách toàn cục
* **MVC Design Pattern**: Model-View-Controller, view là cái hiển thị cho người dùng thao tác, khi có yêu cầu hoặc dữ liệu từ view về thì controller sẽ nhận xử lý rồi dựa theo model để trả về cho view.
* **3 Layer**:
* 

Presentation Layer (GUI) : Lớp này có nhiệm vụ chính giao tiếp với người dùng.

Business Logic Layer (BLL) : xử lý logic, nghiệp vụ

Data Access Layer (DAL) : Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu

* **Template Method Pattern**: Định nghĩa một bộ khung của một thuật toán trong một chức năng, chuyển giao việc thực hiện nó cho các lớp con

## Nêu trường hợp ví dụ sử dụng generic

Khi muốn xây dựng các hàm chung có thể áp dụng cho nhiều kiểu dữ liệu ví dụ như khi muốn mapping từ json sang object thay vì mỗi lần mapping lại viết lại mấy dòng code để chuyển đổi thì có thể tạo 1 phương thức có đầu vào là json còn đầu ra là đối tượng có kiểu Generic

## Giải thích về Socket

Socket là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên internet. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều

Dùng mô hình TCP/IP

TCP: giao thức truyền nhận dữ liệu

IP: xác định địa chỉ và định tuyến từng gói để đảm bảo nó đến đúng nơi.

## Giải thích về Thread và multithread Java

* Thread là đơn luồng( tiến trình con , đơn vị nhỏ nhất của tiến trình)
* multithread là đa luồng , luồng xử lý gần như đồng thời , tiến trình tùy theo mức độ ưu tiên mà nhường nhau chạy

## Giải thích về dependency Injection

* Để cho 1 object có thể nhận được object khác truyền vào nó thông qua các Abstraction giúp làm giảm sự phụ thuộc giữa các object và các module với nhau, kiểu vẫn có ràng buộc nhưng theo cách lỏng lẻo có thể can thiệp

[*Link tham khảo*](https://www.youtube.com/watch?v=WHT0_Co5RSM)

* Giúp xóa bỏ sự phụ thuộc hard-code và làm cho ứng dụng của bạn dễ mở rộng và maintain(duy trì) hơn.
* ví dụ :

MySQLDAO (class MySQLDAO chuyên thực hiện truy vấn với cơ sở dữ liệu MySQL của ứng dụng)

Muốn truy vấn tới cơ sở dữ liệu postgre.Phải xóa khai báo MySQLDAO trong ứng dụng và thay bằng PostgreDAO, sau đó muốn dùng lại MySQLDAO thì lại phải xóa PostgreDAO

Giải pháp tạo 1 interface AbstractDAO và cho các class DAO kia thừa kế AbstractDAO

Việc thay thế AbstractDAO bằng MySQLDAO/PostgreDAO được gọi là injection

Constructor Injection: Các dependency sẽ được truyền vào (inject vào) 1 class thông qua constructor của class đó. Đây là cách thông dụng nhất. (ví dụ trên mình dùng theo cách này)

Setter Injection: Các dependency sẽ được truyền vào 1 class thông qua các hàm Setter/Gette

## Giải thích về inverse of control

* Đảo ngược điều khiển, nó giúp làm thay đổi luồng điều khiển của chương trình một cách linh hoạt
* thường dùng với DI

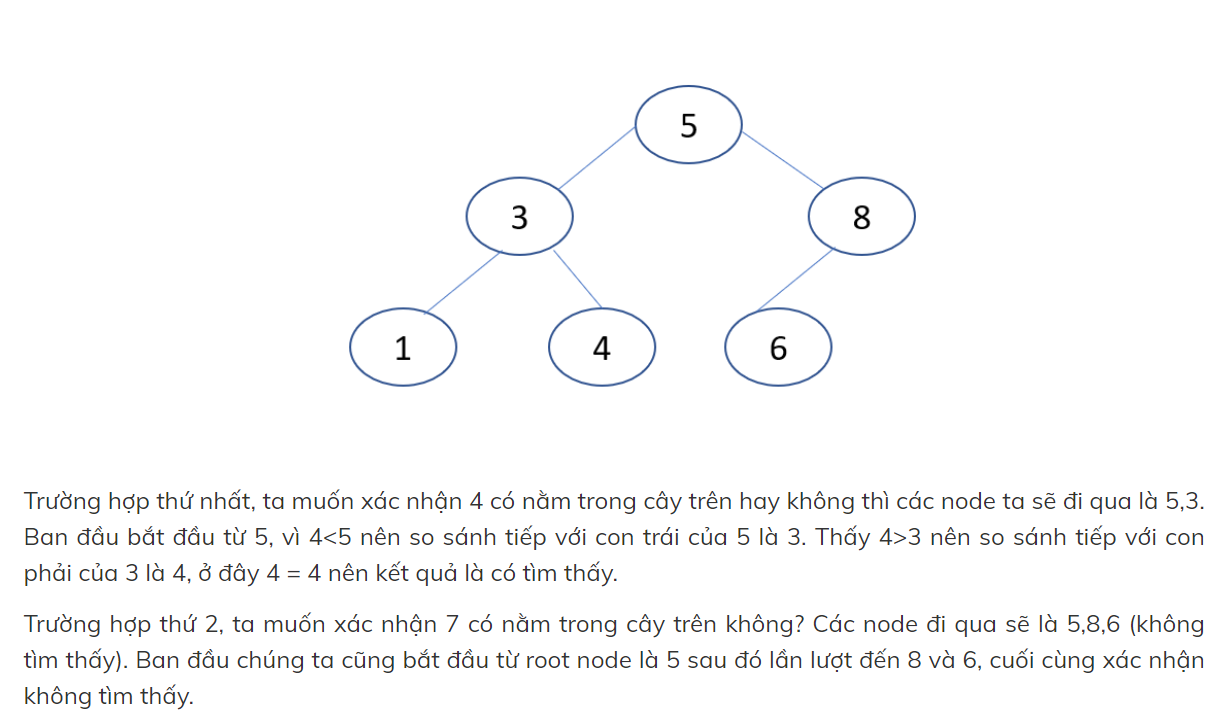
## Giải thích về transaction

* bao gồm 1 loạt các hành động được phải được thực hiện thành công cùng nhau, nếu 1 hành động thất bại thì tất cả các hành động trong loạt hành động đó sẽ trở về trạng thái ban đầu

## Giải thích khi nào nên sử dụng interface, khi nào thì dùng Abstract

* Không chú trọng vào cấu trúc của đối tượng mà chú trọng vào hành vi của đối tượng thì dùng interface, khi các lớp cần liên hệ với nhau thì dùng abstract

## Giải thích về Binary Search Tree trong java

* (tìm kiếm node x) duyệt từ node cha , duyệt tiếp xuống node con key có giá trị > = x
* 

## Collection trong java

* Java Collection cung cấp nhiều interface (Set, List, Queue, Deque vv) và các lớp (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet vv).
* Thư viện Interface chứa các chức năng xử lý, sắp xếp chuỗi

## Design pattern là gì? , đã làm việc với những design pattern nào?

-Design Patterns là một kỹ thuật trong lập trình hướng đối tượng, nó cung cấp các "mẫu thiết kế", giải pháp để giải quyết các vấn đề chung, thường gặp trong lập trình. Design Patterns giúp bạn giải quyết vấn đề một cách tối ưu nhất, cung cấp cho bạn các giải pháp trong lập trình OOP

### Lý do nên sử dụng Design Patterns

-Design pattern giúp cho dự án của chúng ta dễ bảo trì, nâng cấp và mở rộng. Hơn nữa do design pattern đã được các nhà nghiên cứu đúc kết ra nên khi sử dụng design pattern thì chúng ta sẽ hạn chế được các lỗi tiềm ẩn. Và cuối cùng là khi sử dụng design pattern thì sẽ giúp code của chúng ta sẽ dễ đọc hơn. Điều này rất có lợi khi làm việc nhóm.

* singleton
* Factory Method
* Abstract Factory
* prototype
* Builder

## spring security

* Spring security là 1 framework thuộc hệ thống Spring, dành riêng cho việc thiết lập bảo mật của ứng dụng bao gồm authentication và authorization
* Authentication: quá trình xác minh user, dựa vào thông tin đăng nhập mà user cung cấp. Ví dụ khi login, bạn nhập username và password, nó giúp hệ thống nhận ra bạn là ai.
* Authorization: Quá trình xác định xem user có quyền thực hiện những chức năng nào của hệ thống (đọc/sửa/xóa data), sau khi user đã authenticated thành công.

## spring data jpa

* Spring Boot JPA là một bản ghi chi tiết của Java để quản lý dữ liệu quan hệ trong các ứng dụng Java. Nó cho phép chúng ta truy cập và lưu trữ dữ liệu giữa các object/class Java và database quan hệ. JPA tuân theo Object-Relation Mapping (ORM). Nó là một tập hợp các interface. Nó cũng cung cấp một API EntityManager runtime để xử lý các câu query và giao dịch trên các object dựa trên database. Nó sử dụng ngôn ngữ truy vấn hướng đối tượng độc lập nền tảng JPQL (Java Persistent Query Language).
* [Giới thiệu về Spring JPA (techmaster.vn)](https://techmaster.vn/posts/36255/gioi-thieu-ve-spring-jpa)

## spring boot

* là mọt framwork của spring giúp tiết kiệm thời gian code , tập trung vào phát triển nghiệp vụ
* ❑Spring Boot là một dự án bổ sung của Spring, nhằm đơn giản hóa

việc phát triển ứng dụng Spring.

* ❑Dễ dàng hơn trong việc thiết lập và phát triển ứng dụng.
* ❑Nguyên tắc “Cấu hình mặc định” giảm thiểu viết mã cấu hình
* ❑Cho phép nhúng Web Server vào ứng dụng.
* ❑Có thể chạy ứng dụng web chạy bằng cơ chế dòng lệnh hoặc xuất ra

file war để triển khai lên Web Server.

* ❑Dễ dàng tương tác với hệ sinh thái công nghệ của Spring (JDBC, ORM,

Security…)

* Câu hỏi về CSDL:

## Giải thích về UNION

* Dùng để nối kết quả của hai câu lệnh SELECT lại với nhau mà không cần trả về bất kỳ hàng trùng lặp nào.
* Để sử dụng UNION mỗi câu lệnh select cần phải có:
* Cùng một số cột được chọn
* Cùng một số biểu thức cột
* Cùng kiểu dữ liệu
* Có cùng một trật tự

Không cần có cùng chiều dài

[Mệnh đề UNION trong SQL - học sql cơ bản đến nâng cao - VietTuts](https://viettuts.vn/sql/menh-de-union-trong-sql)

## Phân biệt JOIN, INNER JOIN, RIGHT JOIN, LEFT JOIN.

* JOIN là viết tắt của INNER JOIN
* INNER JOIN
* Dùng để join hai bảng, trong đó giá trị dùng để join phải đều có ở cả hai bảng. Kết quả là giao của hai tập hợp dữ liệu, các bản ghi chỉ xuất hiện ở 1 trong hai bảng sẽ bị loại
* RIGHT JOIN
* Lấy toàn bộ dữ liệu ở bảng bên phải với những dòng dữ liệu tương ứng ở bảng bên trái. Nếu không có dữ liệu tương ứng ,bảng bên trái sẽ trả về giá trị null
* LEFT JOIN
* Lấy toàn bộ dữ liệu ở bảng bên trái với những dòng dữ liệu tương ứng ở bảng bên phải. Nếu không có dữ liệu tương ứng, bảng bên phải sẽ trả về giá trị null

[Phân biệt và cách sử dụng các loại JOIN trong MSSQL Server – HaiNH's Blog (hainh2k3.com)](https://hainh2k3.com/phan-biet-va-cach-su-dung-cac-loai-join-trong-mssql-server/)

## Cách sử dụng của Procedure

[PROCEDURE (Thủ tục) trong SQL Server - QuanTriMang.com](https://quantrimang.com/procedure-thu-tuc-trong-sql-server-159768)

## Cách debug trong Postgres

* Đặt breakpoint debug như trong java

## Giải thích về câu lệnh lồng, và những chú ý cần lưu ý khi sử dụng câu lệnh lồng.

- Truy vấn lồng nhau là một truy vấn nằm trong một câu truy vấn khác.Coi câu truy vấn con như là một tham số để chuyền vào câu truy vấn mẹ

-Lưu ý: truy vấn lồng sẽ làm giảm tốc độ truy vấn

**Lưu ý khi sử dụng:**

* Truy vấn con phải được bao trong dấu ngoặc đơn ()
* Các truy vấn con, mà trả về nhiều hơn một hàng, có thể chỉ được sử dụng với các toán tử như: IN.
* Một truy vấn con không thể bị bao quanh một cách trực tiếp trong một hàm tập hợp
* Toán tử [**BETWEEN**](http://s) không thể được sử dụng với một truy vấn con;

[*Link tham khảo*](https://shareprogramming.net/truy-van-long-truy-van-con-trong-sql/)

## Lệnh LIMIT trong SQL

* Sử dụng LIMIT để giới hạn kết quả trả về lấy về một số dòng đầu tiên hoặc tập hợp nhỏ liên tục của kết quả trong SQL

SELECT column list

FROM table\_name

LIMIT số-dòng-cần-lấy;

SELECT column list

FROM table\_name

LIMIT vị-trí-bắt-đầu, số-dòng-cần-lấy;

## giải thích về ORDER BY

* Được dùng xắp xếp bản ghi trong bộ kết quả trả vể. Mệnh đề này chỉ nằm trong câu lệnh select

SELECT “biểu thức”

FROM “bảng”

[WHERE “điều kiện”]

ORDER BY “biểu thức” [ ASC | DESC ];

# **Câu hỏi về font-end:**

## Giải thích cách vận hành của ReactJs và vòng đời trong ReactJS

### Cách vận hành:

https://www.hostinger.vn/huong-dan/react-la-gi-va-no-hoat-dong-nhu-the-nao#:~:text=v%C3%A0%20t%C6%B0%C6%A1ng%20t%C3%A1c.-,React%20ho%E1%BA%A1t%20%C4%91%E1%BB%99ng%20nh%C6%B0%20th%E1%BA%BF%20n%C3%A0o%3F,-B%E1%BA%A1n%20c%C3%B3%20th%E1%BB%83

### Life cycle: [ReactJS - Tìm hiểu về Component Life Cycle (viblo.asia)](https://viblo.asia/p/reactjs-tim-hieu-ve-component-life-cycle-naQZRw4Plvx)

life cycle

component

mount

init state (componentWillMount)

-> render view

run useEffect (componentDidMount)

update state or not (shouldComponentUpdate)

if update state

-> calculate function with param state (componentWillReceiveProps)

update state (componentWillUpdate)

-> rerender view

unmount (componentWillUnmount)

-> rerender view

## Giải thích về Redux

Redux được định nghĩa là một **predictable state management tool dành riêng cho các ứng dụng của Javascript**. Redux se hỗ trợ viết ứng dụng hoạt động một cách nhất quán, chạy trong môi trường khác nhau và dễ hơn khi test.

## Giải thích ngắn về cơ chế vận hành của Bootstrap: grid line, ....

=> Một số câu hỏi khác:

## Mô tả nghiệp vụ của các dự án đã tham gia?.

## Đã làm việc với xuất file có dung lượng lớn chưa? (công ty đang chưa có giải pháp để xử lý?)

-[Hướng dẫn xuất dữ liệu lớn ra file excel với thư viện Apache POI - GP Coder (Lập trình Java)](https://gpcoder.com/3168-huong-dan-xuat-du-lieu-lon-ra-file-excel-voi-thu-vien-apache-poi/)

## Design pattern là gì?. được chia làm những nhóm nào?. Đã làm việc với các loại loài.

* Là các giải pháp tổng thể đã được tối ưu hóa, được tái sử dụng cho các vấn đề phổ biến trong thiết kế phần mềm. Design pattern giúp viết code rõ ràng hơn, dễ đọc , dễ hiểu hơn, dễ tái sử dụng hơn.
* Được chia ra làm 3 loại chính:
* Creational
* Structural
* Behavioral

## Hiểu gì về lập trình OOP?.

* **Lập trình hướng đối tượng (OOP)** là một kỹ thuật lập trình cho phép [**lập trình viên**](https://topdev.vn/blog/lap-trinh-vien/) tạo ra các đối tượng trong code trừu tượng hóa các đối tượng.
* [Lập trình hướng đối tượng là gì? | Object Oriented Programming - OOP (topdev.vn)](https://topdev.vn/blog/oop-la-gi/)

## Hiểu gì về nguyên lý SOLID?. Dùng nó ở chỗ nào trong dự án.(Giải thích từng nguyên lý).

* Là tập hợp của 5 nguyên lí sau đây:
* [**S**ingle responsibility principle](https://toidicodedao.com/2016/05/03/series-solid-cho-thanh-nien-code-cung-single-responsibility-principle/)

Một class chỉ nên giữ 1 trách nhiệm duy nhất

(Chỉ có thể sửa đổi class với 1 lý do duy nhất)

* [Open/closed principle](https://toidicodedao.com/2016/05/10/series-solid-cho-thanh-nien-code-cung-openclosed-principle/)

Có thể thoải mái mở rộng 1 class, nhưng không được sửa đổi bên trong class đó

(open for extension but closed for modification).

* [Liskov substitution principle](https://toidicodedao.com/2016/05/17/series-solid-cho-thanh-nien-code-cung-liskov-substitution-principle/)

Trong một chương trình, các object của class con có thể thay thế class cha mà không làm thay đổi tính đúng đắn của chương trình

* [**I**nterface segregation principle](https://toidicodedao.com/2016/06/09/series-solid-cho-thanh-nien-code-cung-interface-segregation-principle/)

Thay vì dùng 1 interface lớn, ta nên tách thành nhiều interface nhỏ, với nhiều mục đích cụ thể

* [**D**ependency inversion principle](https://toidicodedao.com/2016/06/14/series-solid-cho-thanh-nien-code-cung-dependency-inversion-principle/)

1. Các module cấp cao không nên phụ thuộc vào các modules cấp thấp.

Cả 2 nên phụ thuộc vào abstraction.

2. Interface (abstraction) không nên phụ thuộc vào chi tiết, mà ngược lại.

( Các class giao tiếp với nhau thông qua interface,

không phải thông qua implementation.)

https://toidicodedao.com/2015/03/24/solid-la-gi-ap-dung-cac-nguyen-ly-solid-de-tro-thanh-lap-trinh-vien-code-cung/

## Dùng gì để quản lý source code.

* git

## Làm thế nào để lấy code mới nhất?.

* dùng pull để lấy code mới nhất

## Git rebase là gì?

* gộp nhánh code , giữ lại các commit của nhánh rebase

## Git cherry pick ?

* lấy commit

## Cách xử lý conflict?

* chỉnh sửa nội dung file dẫn tới xung đột
* thường xuyên commit và push
* pull trước khi làm việc và trước khi commit

## Với dữ liệu lớn để tìm kiếm nhanh thì làm thế nào? (Nếu dùng index thì cách đánh index thế nào?)

-đặt chỉ mục index cho trường dữ liệu bảng cần tìm có dứ liệu lớn.tạo chỉ mục bằng cách : CREATE INDEX ten\_index ON ten\_bang(tên cột desc or asc) tạo chị mục sẽ đánh index từ 0 đến n-1 ( trong đấy b là số bản ghi trong đb) Lưu ý : nó như tìm kiếm nhị phân;

## Trong cv có mô tả dùng JPA, Hibernate, MYBATIS, SPRING DATA JPA thì có điểm gì khác nhau khi dùng ?

* JPA là đặc tả cỏa ORM, chỉ chứa các interface không thực thi khai triển ORM
* Spring data JPA không cần xây dựng các lớp DAO chỉ cần tạo các lớp interface thừa kế từ lớp JPARepository và viết thêm các phương thức truy vấn, cấu hình kết nối với csdl thông qua file properties
* Hibernate Jpa kết nối với cơ sở dữ liệu thông qua file persistence.xml phải tạo các lớp DAO
* Mybasic cấu hình kết nối csdl và và mapper dữ liệu bảng trong csdl với class object trong java thông qua file xml.

-<https://viettuts.vn/mybatis/mybatis-example-mapper-bang-file-xml>

[Hướng dẫn sử dụng JPA và Hibernate bằng Spring Boot Data JPA - phần 1 (techmaster.vn)](https://techmaster.vn/posts/36274/huong-dan-su-dung-jpa-va-hibernate-bang-spring-boot-data-jpa-phan-1)

## Về Unit Test thì viết code test hay chỉ test trên giao diện thôi?. Nếu viết code thì viết test ở những tầng nào(Controller, Repository,Service ?).

-viết code test, test ở tầng service

## Viết testcase thì căn cứ vào đâu để viết? có tuân theo nguyên tắc nào không?

* Căn cứ vào đặc tả dự án. [Testcase và các nguyên tắc cần chú ý khi tạo Testcase (hangnt.net)](https://hangnt.net/testcase-va-cac-nguyen-tac-can-chu-y-khi-tao-testcase/)
* Nguyên tắc : 1. cần nắm rõ chức năng được phân công như là : kiểu dữ liệu, định dạng cần có, giá trị mặc định, mã hóa dữ liệu ……..

2. nên xây dựng 1 checklist tự review

3. viết test case nên sử dụng cú pháp ngắn gọn, đơn giản nhưng đủ ý

4. không đoán phán đoán yêu cầu, chỗ nào không hiểu thì hỏi PM, QA,leader….

## Cách quản lý task làm việc thế nào?

* Chia độ ưu tiên
* ước lượng thời gian

## Làm thế nào biết được mình đã code xong chức năng đấy?

- khi hoàn thành đúng các tiêu chí hoàn thành trước lúc code ( vi du : phải comment code theo mẫu nào đó, xuất file java doc ...)

* Rà soát lại code xem có phản ánh hết nội dung của đặc tả không.

Những tính năng mới của java ?. Trong các dự án có hay dùng không?

- gồm có : lamda, Stream API, Date & time api và default và static interface

- trong dự án có dùng đến lamda.

## Đã từng sử dụng biểu đồ gì để design detail

* mô tả luồng xử lý ( flowchart)
* [Flowchart là gì? Quy định ký hiệu và cách vẽ biểu đồ flowchart? (thanhmo.com)](https://thanhmo.com/flowchart-la-gi/)

# **Front-end:**

## Có biết về html css js không?. đã dùng js để validate chưa?

* Có biết về HTML CSS JS không: có (HTML là ngôn ngữ đánh dấu siêu văn bản, CSS là ngôn ngữ tạo phong cách cho trang web, JS là một ngôn ngữ lập trình giúp cho trang web trở nên sống động hơn và đẹp hơn)
* Đã dùng JS để validate chưa: rồi (sử dụng đoạn mã regex hoặc dựng các function kiểm tra tính hợp lệ của dữ liệu đầu vào)

## Có biết về bootstrap không?. grid, flex,...

* Bootstrap là một framework kết hợp của HTML, CSS và JS cho phép người dùng tạo thiết kế website một cách dễ dàng và theo một chuẩn nhất định.
* Grid: là một hệ thống lưới gồm có các hàng và các cột, các hàng sẽ phụ thuộc vào thiết kế, còn các cột sẽ luôn mặc định là 12 cột.
* Flex: là một kiểu dàn trang, tự động cân đối kích thước các phần tử bên trong nó để có thể hiển thị trên mọi kích thước màn hình

## Phân biệt Spring boot và Spring MVC

* Spring MVC: là một khung chương trình web model view controller dựa trên spring, cung cấp các tính năng sẵn sàng để xây dựng ứng dụng web, sử dụng nhiều file cấu hình, chỉ định dependency riêng biệt và mất nhiều thời gian hơn khi phát triển.
* Spring boot: cung cấp cấu hình mặc định để dựng một ứng dụng web, không yêu cầu có các file cấu hình, kết hợp các dependencies lại với nhau trong một đơn vị duy nhất, giảm thời gian phát triển và tăng hiệu suất.

## Phân biệt Spring & Struts

* Spring: là một lightweight framework, không hỗ trợ tag library, dễ dàng mở rộng và phân tách module, sử dụng theo kiến trúc MVC (model, view, controller) nhờ đó mà hiệu suất xử lý cũng sẽ nhanh hơn.
* Struts: là một heavyweight framework, hỗ trợ tag library directive, khó mở rộng hoặc phân tách các module, không có kiến trúc phân lớp.

## https, http, phương thức nào của http lưu được cookie

* Https: là một giao thức truyền tải siêu văn bản kết hợp chứng chỉ bảo mật SSL nhằm mã hoá thông điệp giao tiếp để tăng tính bảo mật.
* Http: là giao thức truyền tải siêu văn bản và hoạt động theo mô hình Client-Server
* Tất cả các phương thức của HTTP đều có thể lưu được cookie

## Một vài câu hỏi liên quan như annotation, transaction...

* Annotation: là một dạng chú thích hoặc một dạng siêu dữ liệu và không ảnh hưởng trực tiếp tới mã mà sử dụng nó.
* Transaction: là một tiến trình xử lý dữ liệu có điểm đầu và điểm cuối và được chia thành nhiều operation nhỏ, khi có bất kỳ một operation nào bị lỗi thì tất cả dữ liệu của tiến trình sẽ được khôi phục lại như ban đầu.

## Kĩ thuật : multi thread, timer, socket...

* Multi thread: là một chương trình có ít nhất 2 tiến trình đang chạy song song cùng lúc, mỗi tiến trình sẽ được gọi là một luồng.
* Timer: lên lịch thực thi một tiến trình theo một khoảng thời gian nhất định trong tương lai.
* Socket: là một cách thức giao tiếp giữa 2 chương trình kết nối tới cùng một cổng server.

## Sử dụng biểu thức Lambda?

* tên tham số -> code thực thi

## Callback function là gì? Tại sao lại sử dụng callback?

* Callback function: là một hàm A sẽ gọi tới hàm B thông qua tham số truyền vào sẽ là một function.
* Nên sử dụng Callback: thay vì chờ phản hồi khi tiếp tục thực hiện thì callback sẽ có thể thực hiện ngay khi đang lắng nghe các sự kiện khác

## Design pattern là gì? Em đã làm việc với những design pattern nào?

* Design pattern: là một giải pháp tổng thể cho các vấn đề chung và có thể tái sử dụng lại trong thiết kế phần mềm.
* Em đã làm việc với Design pattern: singleton

## Restful API?

* Là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau.

## Mô hình Microservice?

* Phân chia phần mềm thành các service nhỏ hoạt động độc lập, khi có service bị lỗi thì các service khác vẫn hoạt động bình thường.

## Sử dụng JPA thế nào? Làm thế nào để lấy dữ liệu từ db bằng JPA?

* Sử dụng JPA thông qua các ORM framework như Hibernate, Spring Data JPA, Mybatis,...
* Để lấy dữ liệu từ DB bằng JPA cần sử dụng các entity để hứng dữ liệu trả về hoặc sử dụng kiểu dữ liệu tương ứng nếu dữ liệu trả về từ db là dữ liệu đơn.

## Làm thế nào để lấy dữ liệu từ nhiều bảng khác nhau bằng jpa khi không có entity phù hợp?

## Sử dụng Thymeleaf như thế nào? Lấy ví dụ về cách truyền dữ liệu

* Sử dụng thymeleaf: thông qua file html, các biến dữ liệu sẽ được xử lý qua các tagname bên trong thẻ html và có chứa tiền tố th:
* Cách truyền dữ liệu: Truyền dữ liệu từ các file java qua thymeleaf thông qua tham số Model

## Sử dụng api như thế nào? lấy ví dụ về cách truyền dữ liệu

* Sử dụng API: Sử dụng annotation RestController hoặc annotation Controller kết hợp annotation ResponseBody, dữ liệu trả về của một API sẽ ở dưới dạng JSon.
* Cách truyền dữ liệu: Truyền vào thông qua các tham số có chứa annotation @RequestBody và gửi đi thông qua kiểu dữ liệu trả về và phương thức có chứa annotation @ResponseBody

## Làm thế nào để tự động quét các user hết hạn và thông báo cho họ?

* Khi có 1 phiên đăng nhập sẽ thêm một giá trị id tương ứng với người dùng đó vào session và setTimeOut cho phiên làm việc đó.
* Thêm HttpSessionListener khi phiên làm việc bị đóng sẽ gửi message bằng socket dựa theo id đã thêm khi phiên làm việc bắt đầu đến client tương ứng để thông báo.

## Giải thích về @Controller, @RestController

* @Controller:
* @RestController: