PREGUNTAS DE EXAMEN

Academia Java

Descripción breve

Preguntas de examen de la semana 2 respondidas

Odraude Méndez Aguirre

Mendez.odraude.1dm@gmail.com

PREGUNTA 1: ¿CUÁLES SON LOS PASOS PARA IMPLEMENTAR SCRUM?

ELIGE UN RESPONSABLE DEL PRODUCTO

El responsable del producto debe ser capaz de comprender la visión del producto y tomar las mejores decisiones para que el equipo sepa como actuar, pues va a estar en constante comunicación con el cliente, y es el encargado de transmitir sus ideas con el equipo.

SELECCIONA UN EQUIPO

El equipo depende enormemente del alcance del proyecto, generalmente suelen ser 7 los integrantes, pues mientras mas recursos existan, mas lento se vuelve el equipo. Una buena selección de recursos ocasiona que el equipo sea Trascendente, Autónomo, e interfuncional.

ELIGE UN SCRUM MASTER

Un scrum master es la 2° cabeza importante de scrum, pues es el encargado de gestionar las tareas de los integrantes del equipo, apoyar resolviendo problemas y organizar de manera efectiva el flujo de trabajo. Debe ser una persona capas de empatizar con los integrantes del equipo, el scrum master no es un jefe, es un líder.

CREA Y PRIORIZA UNA BITÁCORA DEL PRODUCTO

La bitácora del producto no es mas que una lista ordenada donde se detallan los puntos que deben cumplirse para que el producto tenga éxito, esta lista es creada por el responsable del producto y debe estar seguro de que esos puntos representan lo que la gente necesita y es alcanzable para el equipo.

AFINA Y ESTIMA LA BITÁCORA DEL PRODUCTO

Es realmente importante que el equipo y todas las personas involucradas en los elementos de la bitácora revisen los puntos que se detallan en esta, y confirmar que efectivamente se pueden concretar en tiempo y forma, o si en cambio deben hacerse cambios para que los objetivos sean alcanzables.

PLANEACIÓN DEL SPRINT

La planeación de los Sprints es de las primeras reuniones de scrum, pues se examina el inicio de la bitácora y se calcula cuanto tiempo se puede llevar en completar el sprint, este no debe ser muy grande, se recomiendan acciones cortas para mejorar la

productividad. Una forma útil para calcular el tamaño de las tareas es usando la serie Fibonacci, así es más fácil para el cerebro saber cuando algo es rápido, y cuando una tarea podría llevar mas tiempo. Muchos equipos de desarrollo concretan al menos 2 Sprints por semana

VUELVE VISIBLE EL TRABAJO

Se trata de llevar una tabla donde se puedan ver los procesos pendientes, en proceso y terminados. También se puede llevar un diagrama de finalización (Una buena implementación debería formar una campana de gauss en el diagrama)

PARADA DIARIA O SCRUM DIARIO

Es una reunión de no mas de 15 minutos donde el equipo comparte sus aportes y dificultados contestando 3 preguntas sencillas: ¿Qué hiciste ayer para ayudar al equipo a terminar el sprint?, ¿Qué harás hoy para ayudar al equipo a terminar el sprint?, ¿Algún obstáculo te impide o impide al equipo cumplir la meta del sprint?

REVISIÓN DEL SPRINT O DEMOSTRACIÓN DEL SPRINT

En esta reunión pueden estar presentes las personas interesadas en el avance del proyecto, incluido el cliente, scrum master etc., ya que el objetivo es presentar los Sprints completados y como va avanzando el equipo en el proyecto, generalmente conlleva de 15 – 30 minutos

RETROSPECTIVA DEL SPRINT

En esta reunión es importante que los participantes presenten una actitud madura y dispuesta a recibir retroalimentación, pues se debe hacer análisis de las cosas que pudieran haber salido mal, retrasos y descuidos, así como también cuestionarse como mejorar la productividad y resolver conflictos, anotándolos y tomándolos en cuenta para el siguiente sprint y así determinar si en efecto ha ocurrido una mejora en el proceso.

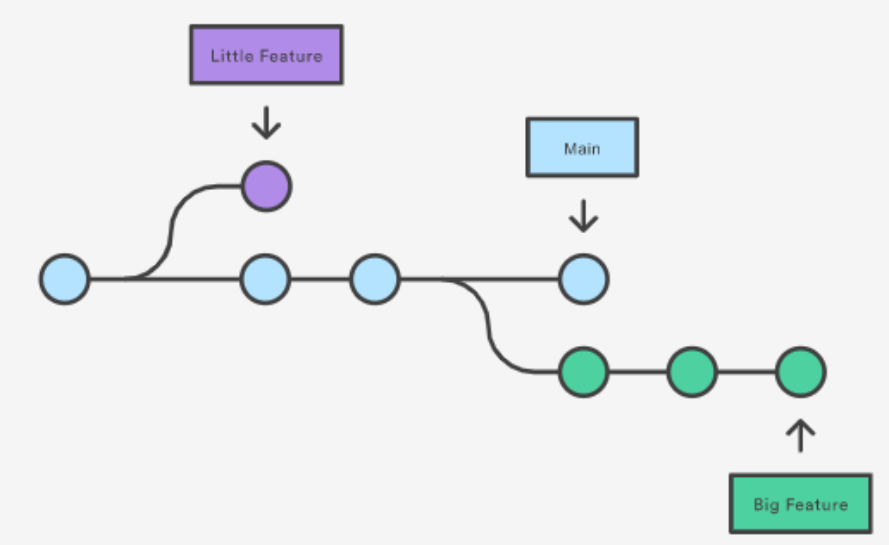
COMENZAR DE INMEDIATO EL CICLO DEL SIGUIENTE SPRINT

Una vez terminado el primer Sprint y habiendo seguido los pasos anteriores, es recomendable continuar con las siguientes actividades programadas para optimizar el tiempo y los resultados

PREGUNTA 2: EXPLICACION DE BRANCH, MERGE

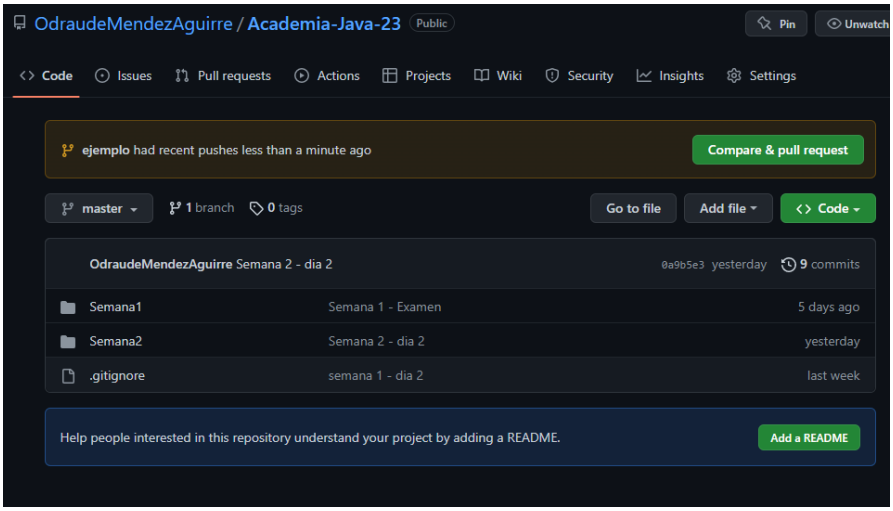
BRANCH

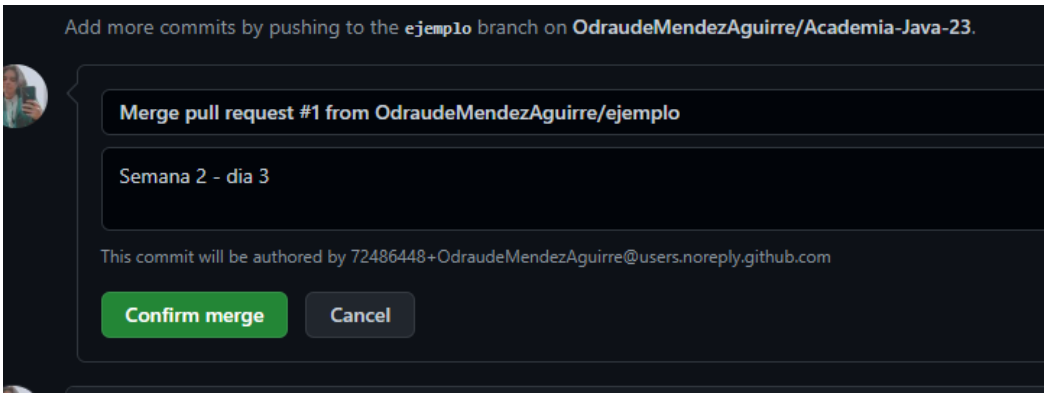
Un Branch es una forma de control de versiones en modelos de trabajo distribuidos, pues permite guardar cambios sin afectar al código principal, y tener un mejor control sobre errores de código inestable, otorgando la oportunidad de deshacer dichos cambios sin echar a perder el proyecto. El comando que utilizamos normalmente para crear una rama es "**git Branch <nombre>".**
para navegar entre ramas usamos "**git checkout <Branch>".**

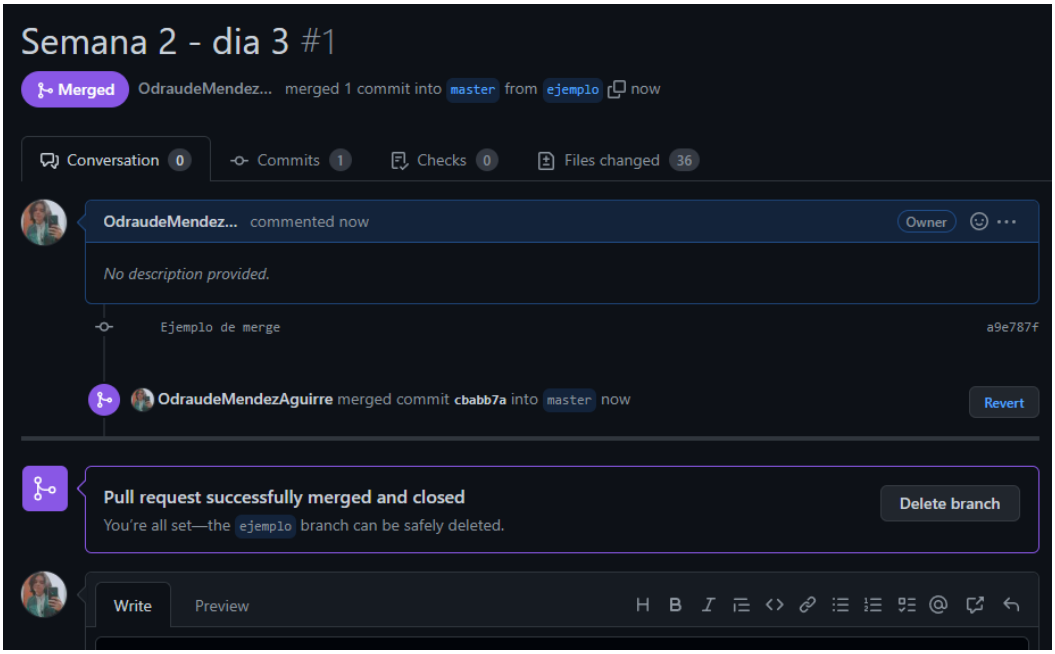


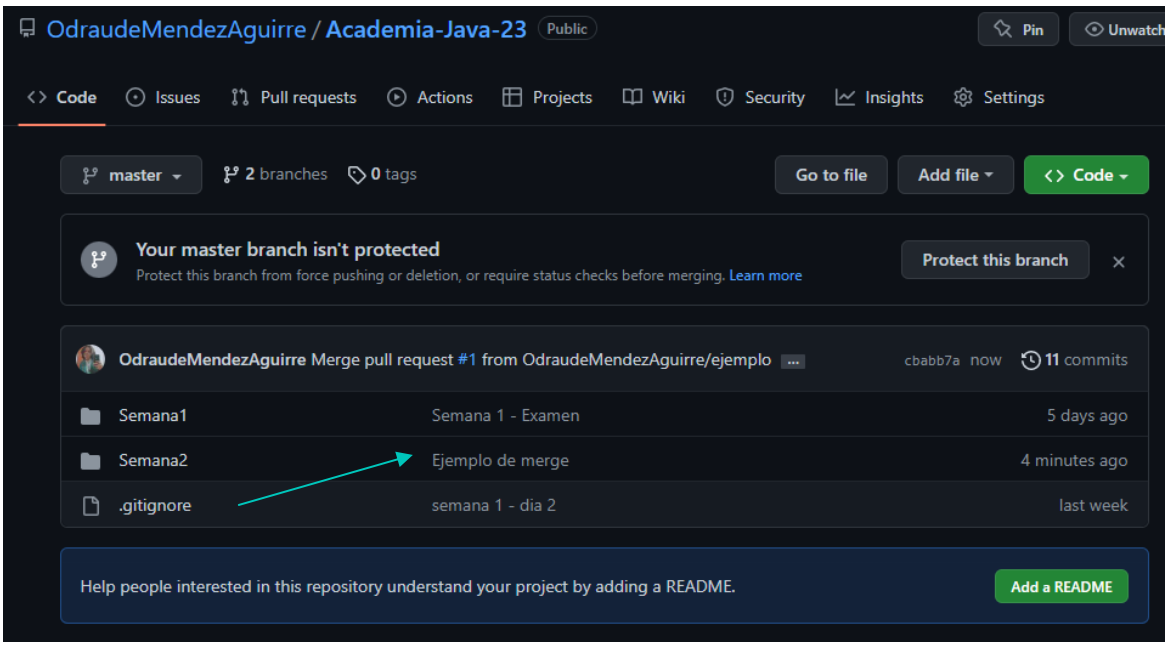
MERGE

Merge significa "Unir", y cuando hablamos de git, nos referimos a unir el contenido de una rama con otra. Generalmente se utiliza cuando un cambio ha sido aprobado por el responsable del repositorio y se requiere implementar las nuevas funciones al proyecto.




Add more commits by pushing to the ejemplo branch on OdraudeMendezAguirre/Academia-Java-23.
Merge pull request #1 from OdraudeMendezAguirre/ejemplo
Semana 2 - dia 3
This commit will be authored by 72486448+OdraudeMendezAguirre@users.noreply.github.com
Confirm merge
Cancel


Semana 2 - dia 3 #1
Merged
OdraudeMendez... merged 1 commit into master from ejemplo now
Conversation 0
Commits 1
Checks 0
Files changed 36
OdraudeMendez... commented now
Owner
No description provided.
Ejemplo de merge
a9e787f
OdraudeMendezAguirre merged commit cbabb7a into master now
Revert
Pull request successfully merged and closed
You're all set—the ejemplo branch can be safely deleted.
Delete branch
Write
Preview


OdraudeMendezAguirre / Academia-Java-23
Public
Pin
Unwatch
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
2 branches
0 tags
Go to file
Add file
Code
Your master branch isn't protected
Protect this branch from force pushing or deletion, or require status checks before merging. Learn more
Protect this branch
OdraudeMendezAguirre Merge pull request #1 from OdraudeMendezAguirre/ejemplo
cbabb7a now
11 commits
Semana1
Semana 1 - Examen
5 days ago
Semana2
Ejemplo de merge
4 minutes ago
.gitignore
semana 1 - dia 2
last week
Help people interested in this repository understand your project by adding a README.
Add a README

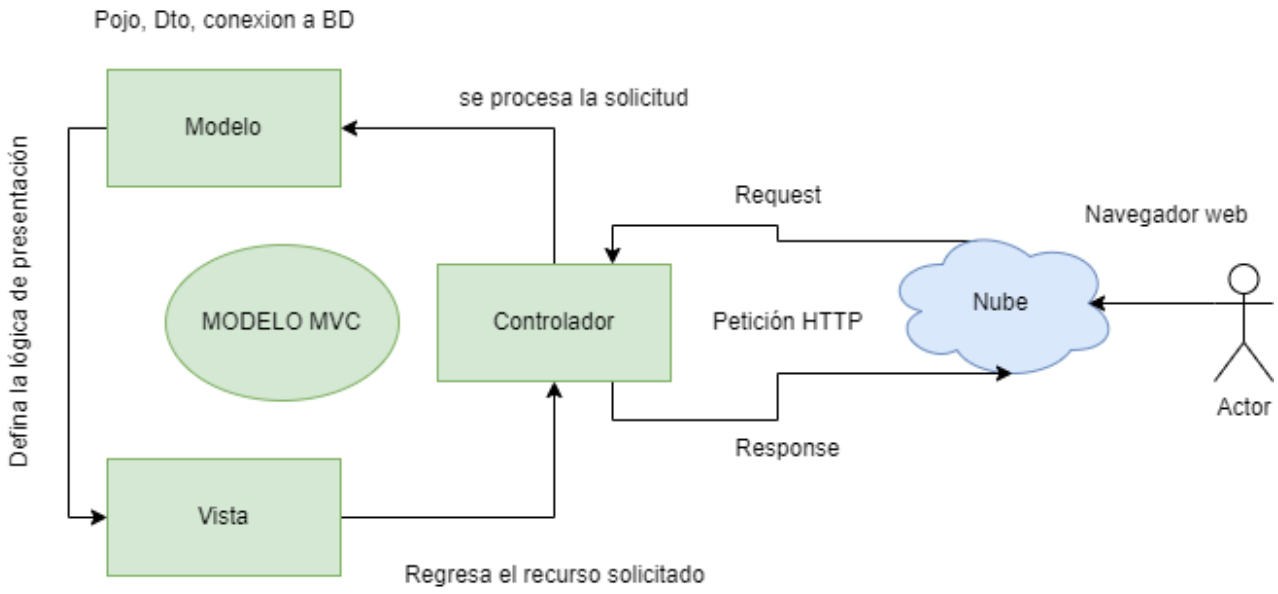
RESOLUCION DE CONFLICTOS

Suelen ser provocadas por realizar cambios en la misma línea del mismo archivo en ramas diferentes del repositorio de GitHub, para resolverlo podemos hacerlo yendo directamente al repositorio donde se nos mostraran las líneas que han sido modificadas, y deberemos decidir que versión mantener para poder hacer el merge, si queremos identificar el error por línea de comandos, debemos posicionaros en el repositorio de git local en donde se haya formado el conflicto, y ahí, ejecutar el comando "git status", la terminal nos marcara los archivos conflictuados y solo deberemos abrir un editor de código y modificar las líneas marcadas.

PREGUNTA 3: DIAGRAMA Y EXPLICA EL MODELO MVC

MODELO VISTA CONTROLADOR

El patrón de arquitectura mvc nos ayuda a separar el código por capas, las cuales se encargan de procesos concretos. La idea principal es trabajar con 3 capas: el Modelo (generalmente son los llamados pojos, definiremos las entidades necesarias que nuestro proyecto requiere para funcionar), la Vista (normalmente relacionada con la presentación de los datos al cliente cuando se habla de interfaces de usuario, pero también se relaciona con modelar la información del lado del back para que sea mostrada en front, ya sea HTML, json, txt, etc.), y el Controlador (Encargado de gestionar las peticiones http que lleguen de lado del front (Request), y devolver el recurso solicitado (Response), las peticiones http más comunes son GET, POST, PUT, DELETE).



PREGUNTA 4: EXPLICAR Y DIAGRAMAR LOS TIPOS DE EXCEPTIONS

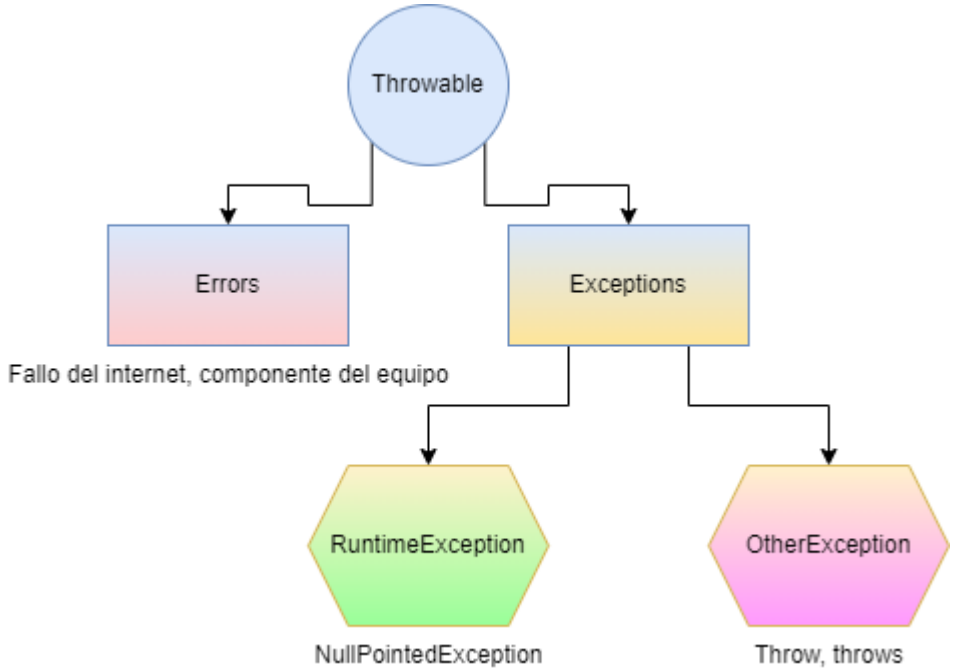
Existen 3 tipos de excepciones:

Los Errors: estos errores no los podemos controlar ni evitar, ya que son errores fuera del alcance del programa (Fallo del internet repentino, el equipo se apague en medio de un proceso, fallo de algún componente en el equipo, etc.).

Runtime Exceptions: Son operaciones "Unchecked" es decir, errores que el compilador no nos obliga a darles un tratamiento por que suelen ocurrir durante la ejecución del código (NullPointerException, ArithmeticException, IllegalArgumentException, etc.).

Other Exceptions: Son operaciones las cuales el compilador te exige darles tratamiento porque el método al que se hace referencia arroja explícitamente una Excepción y es necesaria cacharla.

Las clases Error y Exceptions heredan de la clase Throwable, y las Runtime y other Exceptions heredan de la clase Exceptions.



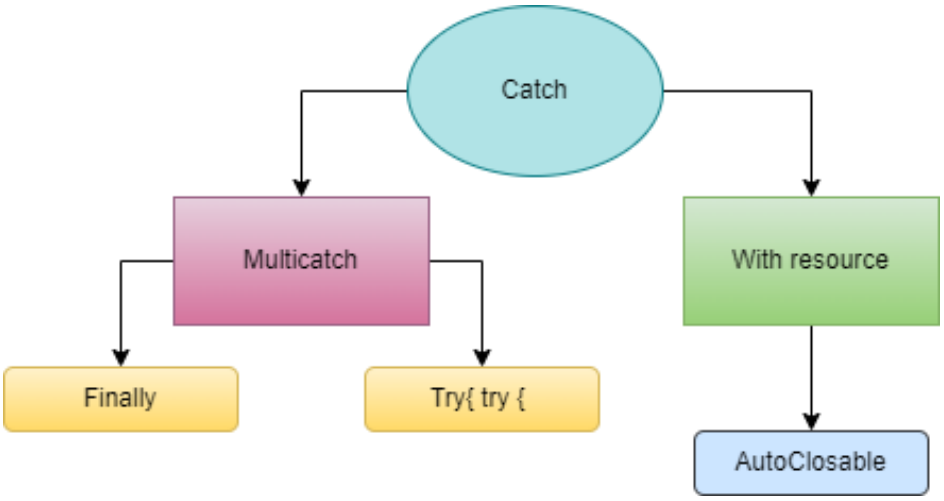
PREGUNTA 5: EXPLICAR MULTICATCH Y TRY WITH RESOURCES

El try-catch nos sirve para "cachar" las excepciones que el programa nos arroja y prevenir posibles errores del usuario, sin embargo, existen 2 formas de hacerlo:
El multicatch hace referencia a un catch que encadena otro catch, generalmente se utiliza cuando usamos un final, o necesitamos cerrar un objeto que hayamos abierto.

```
try {
    res = calcularDiv(x,y);
}catch(CeroException ax) {
    System.out.println(ax);
}finally {
    System.out.println("Siempre paso por el finally");
}
System.out.println(res);
```

El Try with resources nos ayuda a cerrar automáticamente el objeto que le pasamos el cual arroja explícitamente un exception, y así reducimos código.

```
//Auto closable
try (Pato pato = new Pato()){
    System.out.println(pato);
    getExcption();
}catch(Exception e) {
    e.printStackTrace();
}
```



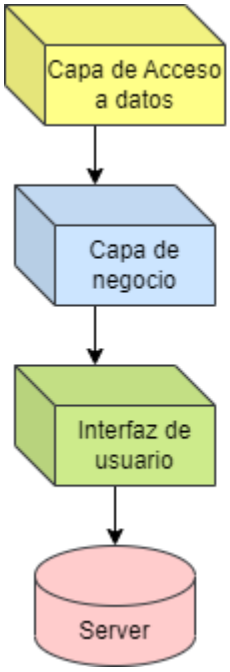
PREGUNTA 6: DIFERENCIAS ENTRE MONOLITICO Y MICROSERVICIOS

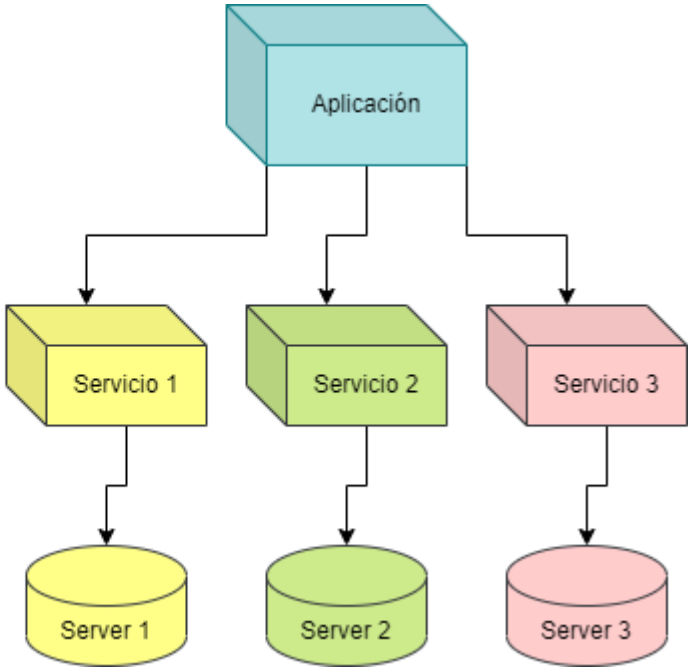
APLICACIONES MONOLITICAS

Una aplicación monolítica es una aplicación donde las capas de Interfaz de usuario, negocio y transferencia de datos están en el mismo programa y sobre el mismo servidor. Es la forma más fácil de hacer aplicaciones, pero tiene problemas en escalabilidad cuando se tratan de miles de peticiones a un mismo recurso.

APLICACIONES CON MICROSERVICIOS

Una aplicación con microservicios separa sus capas entorno a la funcionalidad y las alojan en contenedores diferentes, lo que permite un bajo acoplamiento entre los servicios y la aplicación principal, además, cada uno de estos servicios pueden estar en diferentes servidores y contener a su vez otros servicios. Algunas de las herramientas mas usadas para crear aplicaciones con microservicios son Docker y Cabernets.





PREGUNTA 7: MENCIONA LOS TIPOS DE COLLECTIONS

CLASE LIST

La clase List hereda de Collections, y este tipo de lista va formando un **orden** conforme vamos agregando registros, cada uno de esos registros los podemos identificar gracias a su índice en la lista.

CLASE SET

También Hereda de Collections En las listas de tipo Set no existe un orden, solamente se agregan los datos en memoria, pero no hay una forma de saber exactamente en qué posición están guardados, por esta razón no podemos tener duplicados dentro de una lista Set.

CLASE QUEUE

En este tipo de listas proveen de una inserción y extracción de datos adicional, pues cada uno de esos métodos existen de dos formas: uno arroja una excepción si la operación falla, y el otro retorna un valor en especial. La ventaja de esta lista es que podemos hacer inserciones por arriba y por abajo. También hereda de Queue.

CLASE MAP

Esta clase no hereda de Collections, sin embargo, la manera de guardar datos es muy semejante a una tabla de base de datos, pues manejamos un identificador de registro, una clave y su valor.