ÍNDICE

Implementación de Scrum.

1. Para implementar scrum como primer paso hay que definir quién será el **responsable de producto,** esta persona es quien tiene la idea y visión del producto final es decir que sabe lo que necesita el cliente final, teniendo en cuenta los beneficios y riesgos que puede haber en su desarrollo.
2. El segundo paso es escoger el **equipo de trabajo** el cual se encargará de desarrollar el producto, tomando en cuenta las habilidades necesarias para el mismo.
3. El tercer paso es seleccionar al **Scrum Master** el cual se centra en capacitar al equipó de trabajo en el tema de scrum y de igual forma eliminar todo aquello que atrase al equipó.
4. El cuarto paso es crear una **bitácora del producto,** es decir se enlistan todos los puntos necesarios para el desarrollo del producto, de igual forma esta va cambiando conforme el estado de desarrollo en el que se encuentre el producto .
5. Para el quinto paso **(Afina y estima la bitácora del producto)** en este paso se estiman los tiempos, esfuerzo y si se cuenta con lo necesario para desarrollar las actividades.
6. Para el sexto paso se lleva a cabo **Planeación del sprint**, la cual consiste en una reunión del responsable del producto , el scrum master y el equipo de trabajo en la cual consiste en la estimación de los tiempos que llevará realizar.
7. El séptimo paso es hacer **visible el trabajo**, la forma más común de hacer visible el trabajo es usando la tabla de scrum la cual consiste en tres columnas con títulos como Pendiente, proceso y Terminado además de usar posticks en los cuales se pondrán las tareas a realizar.
8. Para el octavo paso se consideran las **reuniones diarias o scrum diario**, en estas reuniones se reúne el equipo de trabajo y el scrum master las cuales buscan ver que se hizo el equipo ayer, que harán hoy y si existe algún inconveniente que evite cumplir con la meta.
9. El noveno paso llamado **demostración del sprint**, de igual forma que el paso anterior consiste en una reunión con la diferencia de que en esta puede estar toda persona de interés en el proyecto como dirección, cliente etc. En esta reunión el equipo y el scrum master deberán mostrar lo terminado.
10. El décimo paso Retrospectiva del sprint consiste en enviar al cliente lo terminado para recibir la retroalimentación y seguidamente hacer una reunión en la cual se busquen factores a mejorar sin buscar culpables de las deficiencias causadas por estos.
11. Por último se comienza seguidamente el siguiente spring teniendo en cuenta todo la experiencia adquirida del anterior.

Git comandos

Git Branch
El comando git branch nos permite crear diferente ramas en la cuales podremos trabajar simultáneamente sin perjudicar a los demás desarrolladores y mantener una rama con la última versión estable del proyecto.

git merge
El comando git merge nos permite combinar las ramas creadas por git branch teniendo como esto una rama de desarrollo ya estable la cual deberá combinarse con la rama main para volver la nueva versión estable.

Resolver conflictos.
Los conflictos en git suelen darse al hacer merge o push y varias dos personas han trabajado sobre un mismo archivo.

con cat podremos ver

- <<<<<<< HEAD
- =======
- >>>>>>> rama principal

la línea ===== divide el conflicto teniendo como head el contenido de la rama a la que queremos hacer el merge o el pull , mientras que lo de abajo de la línea es la rama o el archivo en el que nos encontramos.

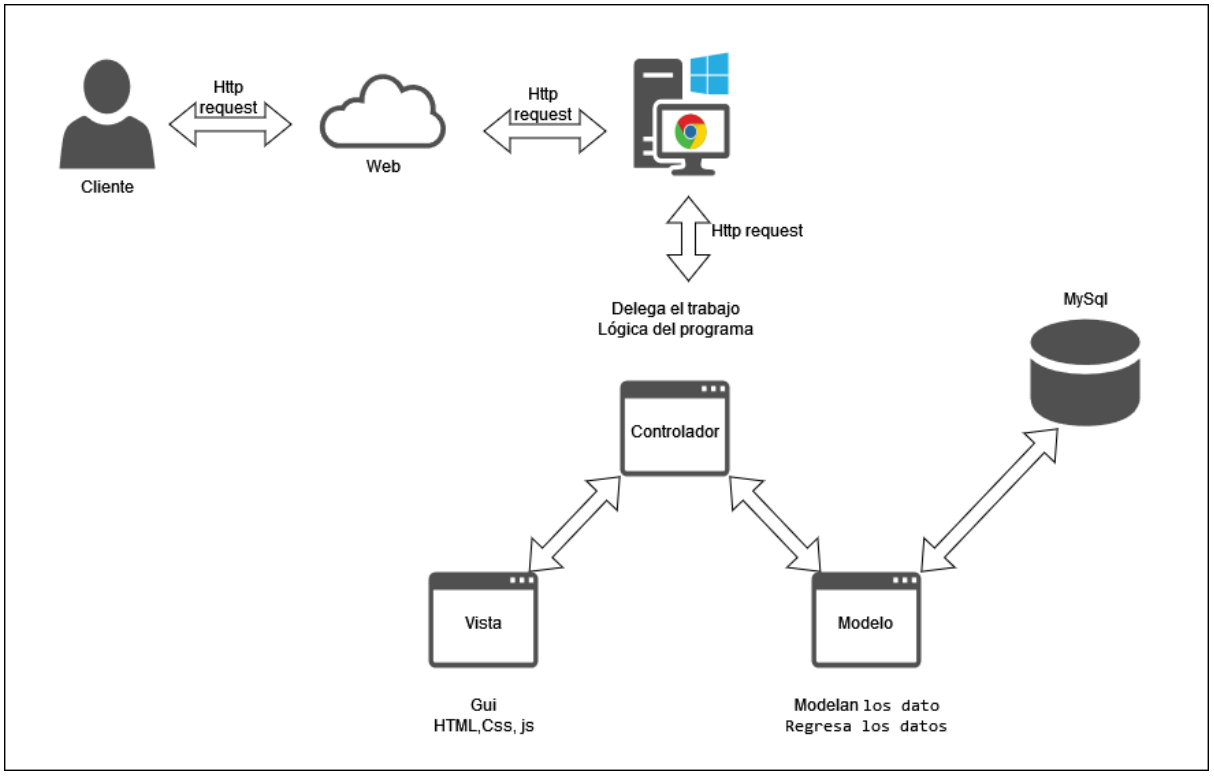
Una manera de resolver estos conflictos es modificar el archivo en el que nos encontramos de modo que al hacer la el pull o merge no afecte al archivo subido.

Otros comandos que nos ayudan a ver la diferencia entre los archivos sería el git diff, de igual forma otro comando sería el git reset que restablece el archivo modificado a como estaba antes de los cambios.

MVC

El MVC(Modelo vista controlador) es una arquitectura que divide una aplicación en tres secciones específicas dividiendo la interfaz de usuario (**Vista**), Lógica de control (**Controlador**) y los datos (**Modelo**).

- Controlador: Este es el que contiene la mayoría de la lógica del programa, se centra en gestionar la comunicación entre el modelo y la vista.
- Modelo: El modelo como tal es quien como lo dice su nombre modela los datos antes de enviarlos al controlador.
- Vista: La vista muestra toda la información recibida del modelo mediante el controlador para luego ser enviada al cliente.

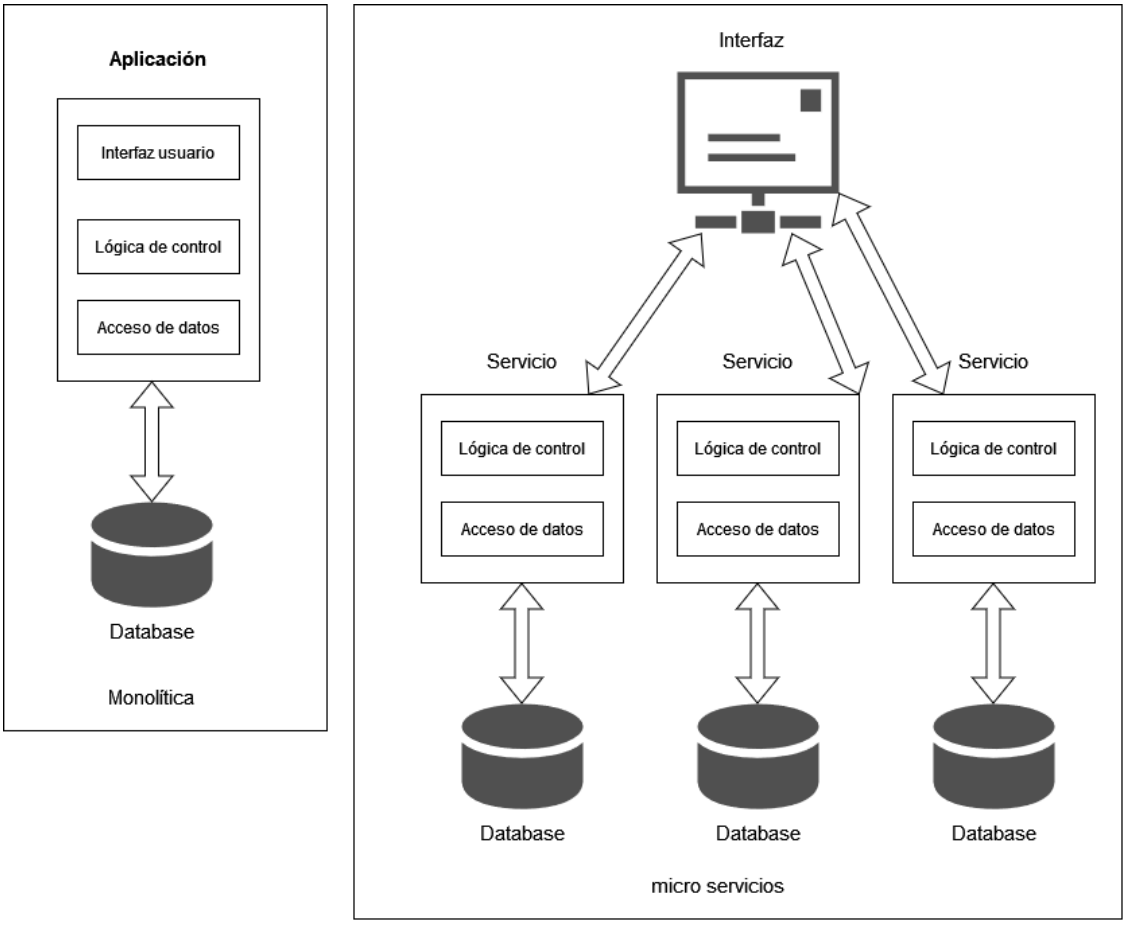


Diferencia entre monolítico y microservicios

la arquitectura monolítica es aquella en la que todos los servicios tales como interfaz de usuarios, lógica de control y acceso de datos se encuentran en una sola aplicación teniendo todos los datos en una sola base de datos, mientras que una aplicación basada en microservicios es aquella que divide todos sus servicios en aplicaciones independientes es decir el control de lógica vendría siendo una aplicación y el acceso a los datos otras, de igual forma cada aplicación o microservicio tiene su propia base de datos y a pesar de que estén divididas en aplicaciones estos se comunican entre sí logrando actuar al final como una sola aplicación

diferencias.

- Todos los servicios de la aplicación monolítica se encuentran en la misma aplicación, mientras que la de microservicios crea pequeñas aplicaciones para cada servicio.
- La escalabilidad de una aplicación de microservicios suele ser más fácil que una aplicación monolítica además de que se puede escalar sin perjudicar los otros servicios.
- Una aplicación monolítica suele tener una sola base de datos a diferencia de las aplicaciones de microservicios que llegan a tener una base por cada servicio.



Tipos de excepciones

Runtime Exception son todos aquellos errores que no piden ser tratados por el compilador de java que se esté usando como por ejemplo intentar dividir entre cero, exceder la longitud de un array o ejecutar funciones de un objeto con una variable de referencia que no apunta a ningún objet.
Other Exception o Excepciones Checked son todas aquellas excepciones que nos obliga el compilador de java a tratarlas con trycatch, estas son aquellas que definimos en los métodos mediante las palabras claves throw y throws.

Multicatch y trywithResource

MultiCatch consiste en tener un solo catch para capturar las excepciones en las cuales compartan el mismo comportamiento al suceder estas, si en llegado caso no compartieran el mismo comportamiento al ser tratadas se opta por tener varios catch.

```
} catch (Exception e1) {
        System.out.println(e1);
} catch (Exception e2) {
        System.out.println(e2);
}
```

trywithResource
Algunos objetos se crean y se destruyen luego de usarse pero otros no lo hacen lo cual genera que se usen recursos aunque estos ya no sean usados tales como conexiones a db o buffers de archivos, para solucionar esto se cierran dichos objetos pero el código de estos llega a crecer al grade de ser como el siguiente ejemplo:

```
catch (SQLException e) {
throw newException;
} finally {
if (resultSet != null) {
try {
resultSet.close();
} catch (Exception e) {
}
}
if (statement != null) {
try {
statement.close();
} catch (Exception e) {
}
}
if (connection != null) {
try {
connection.close();
} catch (Exception e) {
```

```
}
}
}
```

De ahí nace **trywithResource** el cual nos permite cerrar dichos objetos cuando suceda algún error reduciendo la sintaxis a tan solo una linea y dejando más limpio el código.

```
try (Connection connection = getConnection();
Prepared pStatement = connection.prepareStatement(sql);
ResultSet resultSet = pStatement.executeQuery())
```

Tipos de colecciones.

List
La colección de este tipo permite agregar datos repetidos y cada uno de los elementos u objetos cuentan con un índice, este tipo de colección permite remover y agregar valores de manera aleatoria y de igual forma acceder a ellos.

Qeue
Esta colección tiene el comportamiento de First In - First Out el cual nos limita a sólo poder acceder al primero y el último de esta lista.

Set
La colección set no permite agregar valores repetidos, y a diferencia del list no permite agregar valores de manera aleatoria y eliminar, es decir solo permite agregar valores secuencialmente y borrar los valores dependiendo el objeto o valor no el index.

Map
En la colección map todos los objetos o valores tienen una key la cual funcionará como índice, lo que permite que se obtenga el valor u objeto más rápido, no permite valores con llaves duplicada a modo que si existen varias key de uno toma el valor de un único valor.