Requisitos No Funcionales

...

Sebastián Bedoya Restrepo
Jose Andrés López Castaño
Luis Felipe Salazar Rios

Resiliencia

la resiliencia es la capacidad de proporcionar y mantener un nivel de servicio aceptable frente a fallas y desafíos para el funcionamiento normal . Las amenazas y desafíos para los servicios pueden variar desde una simple mala configuración por desastres naturales a gran escala hasta ataques dirigidos. Como tal, la resiliencia de la red toca una amplia gama de temas.

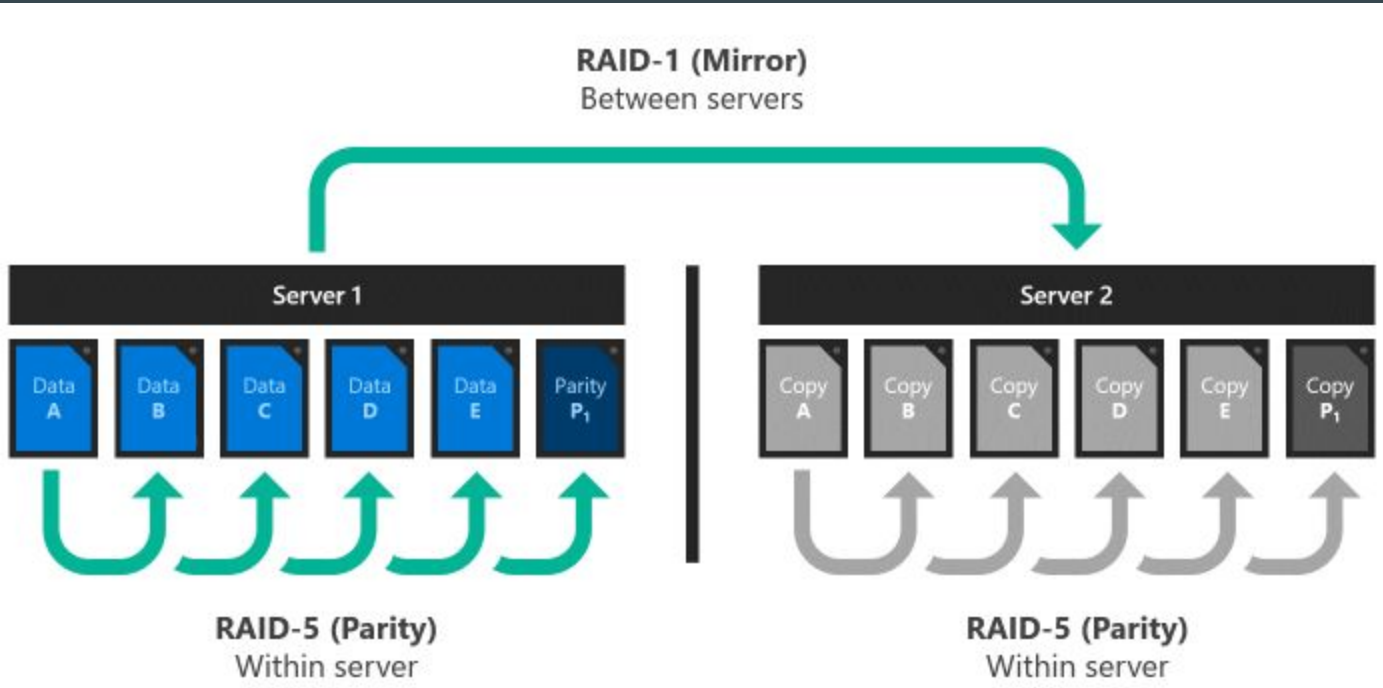
Portabilidad

La portabilidad es la facilidad con la que un sistema de software se puede transferir desde su entorno actual de hardware o software a otro entorno. Los requisitos de portabilidad abordan la preocupación del usuario sobre la facilidad con que se transporta el sistema. Al obtener los requisitos de portabilidad, tenga en cuenta los aspectos de la portabilidad con respecto a los datos, el programa, el usuario final y la documentación del desarrollador.

Eficiencia de almacenamiento

Consiste en obtener el mayor almacenamiento posible al menor costo posible. Una organización que es eficiente en almacenamiento puede almacenar más datos en sus unidades sin afectar el rendimiento o el costo en el sistema de red total.

Actualmente muchas empresas usan la tecnología de instantáneas, consiste en solo guardar los valores cambiados en un archivo, en lugar de guardar varias copias de un archivo alterado.


RAID-1 (Mirror)
Between servers
Server 1
Data A
Data B
Data C
Data D
Data E
Parity P1
Server 2
Copy A
Copy B
Copy C
Copy D
Copy E
Copy P1
RAID-5 (Parity)
Within server
RAID-5 (Parity)
Within server

Navegabilidad

Es la sencillez con la que un usuario puede moverse a través del conjunto de elementos que conforman el software. Son todas aquellas características que posibilitan que los usuarios se desplacen con facilidad y de manera intuitiva. Se relaciona directamente con la experiencia del usuario final.

Legibilidad

La legibilidad se refiere a la facilidad con la que un lector humano puede comprender el propósito, el flujo de control y el funcionamiento del código fuente. Afecta los aspectos de calidad , incluida la portabilidad, la usabilidad y, lo más importante, la capacidad de mantenimiento.

Ejemplo

```java
package com.banco.gaia.web.service.impl;
import com.banco.gaia.web.dao.AbsenteeismDao;

public class NEvClaSSWIthUSer {


Long VARIABLEForHel;

    @Autowired
    AbsenteeismDao ABD;

        public String METHOD(Long var1, Long var2) {
        String VARIABLERETORNAR = "";
        Long var3 = ABD.deleteAbsenteeism(var1);
if(var3 == 1) { //Compara si la primera operación se realizo
VARIABLERETORNAR = "Se elimino el registro: " + var1;
}else { Long var = ABD.deleteAbsenteeismEndOlds(); VARIABLERETORNAR = "Se eliminaron todos "+ var; }
        return VARIABLERETORNAR;
    }
        String VAr;
        Long Var2;
    public void method2() {
    Absenteeism A = new Absenteeism();
        ABD.getListAbsenteeismEnd(); ABD.truncateTable(); ABD.getListToCompare(A);} //Este metodo reporta una lista de los objetos, agrega un obj
}
```

```java
package com.banco.gaia.web.service.impl;
import com.banco.gaia.web.dao.AbsenteeismDao;

public class NEvClaSSWIthUSer {


Long VARIABLEForHel;

@Autowired
AbsenteeismDao ABD;

public String METHOD(Long var1, Long var2) {
String VARIABLERETORNAR = "";
        Long var3 = ABD.deleteAbsenteeism(var1);
if(var3 == 1) { //Compara si la primera operación se realizo
VARIABLERETORNAR = "Se elimino el registro: " + var1;
}else { Long var = ABD.deleteAbsenteeismEndOlds(); VARIABLERETORNAR = "Se eliminaron todos "+ var; }
return VARIABLERETORNAR;
}
String VAr;
Long Var2;
public void method2() {
Absenteeism A = new Absenteeism();
    ABD.getListAbsenteeismEnd(); ABD.truncateTable(); ABD.getListToCompare(A);} //Este metodo reporta una lista de los objetos, agrega un objeto
}
```

```java
package com.banco.gaia.web.service.impl;

import java.text.SimpleDateFormat;
/**
 * Servicio de la tabla AbsenteeismExceptions con la lógica de negocio para el manejo de la tabla absenteismo
 *
 * @author Wilson Sarrazola wsarrazo@bancolombia.com.co
 *
 */

@Service("absenteeismExceptionsService")
@Transactional
public class AbsenteeismExceptionsServiceImpl implements AbsenteeismExceptionsService {

    private static final String YYYY_MM_DD = "yyyy-MM-dd";
    final SimpleDateFormat sdf = new SimpleDateFormat(YYYY_MM_DD);

    private static final Logger logger = LogManager.getLogger(AbsenteeismExceptionsServiceImpl.class);

    @Autowired
    AbsenteeismExceptionsDao absDao;

    @Override
    public boolean addAbsenteeismExceptions(Absenteeism abs) {
        //Valida que la fecha de finalización del asentismo sea mayor a la fecha actual.
        if(validatedEndDate(abs.getEndDateNovelty())) {
            return absDao.addAbsenteeismExceptions(abs);
        }else {
            return false;
        }
    }

    @Override
    public long deleteAbsenteeismExceptions(Long id) {
        return ((AbsenteeismExceptionsDao) absDao).deleteAbsenteeismExceptions(id);
    }

    @Override
    public List<Absenteeism> getListToAbsenteeismExceptions() {
        return absDao.getListToAbsenteeismExceptions();
    }

    /**
     * Valida que la fecha de finalización del asentismo sea mayor a la fecha actual.
     * @param endDate fecha
     * @return Si la fecha es validad o no
     */
    public Boolean validatedEndDate(String endDate) {
        Boolean answer = false;
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern(YYYY_MM_DD);

        try {
            LocalDate today= LocalDate.now();
            LocalDate endNovelty = LocalDate.parse(endDate, formatter);
            if(today.isEqual(endNovelty)) {
                return true;
            }else if(today.isBefore(endNovelty)) {
                return true;
            }
        } catch (DateTimeParseException ex) {
            Logger.error("Error "+ ex.getMessage());
        }
        return answer;
    }
}
```