

**CODIFICACIÓN DE MÓDULOS DEL SOFTWARE STAND-ALONE, WEB Y MÓVIL
DE ACUERDO CON EL PROYECTO A DESARROLLAR**

GA7-220501096-AA3-EV01

PRESENTADO POR:

Sergio Andrés Palomino Silva.

PRESENTADO A:

Milton Iván Barbosa Gaona.

Tecnólogo en Análisis y Desarrollo de Software.

No. De ficha: 2977481

Centro de la Tecnología del Diseño y la Productividad Empresarial.

8 de octubre de 2025

INTRODUCCION

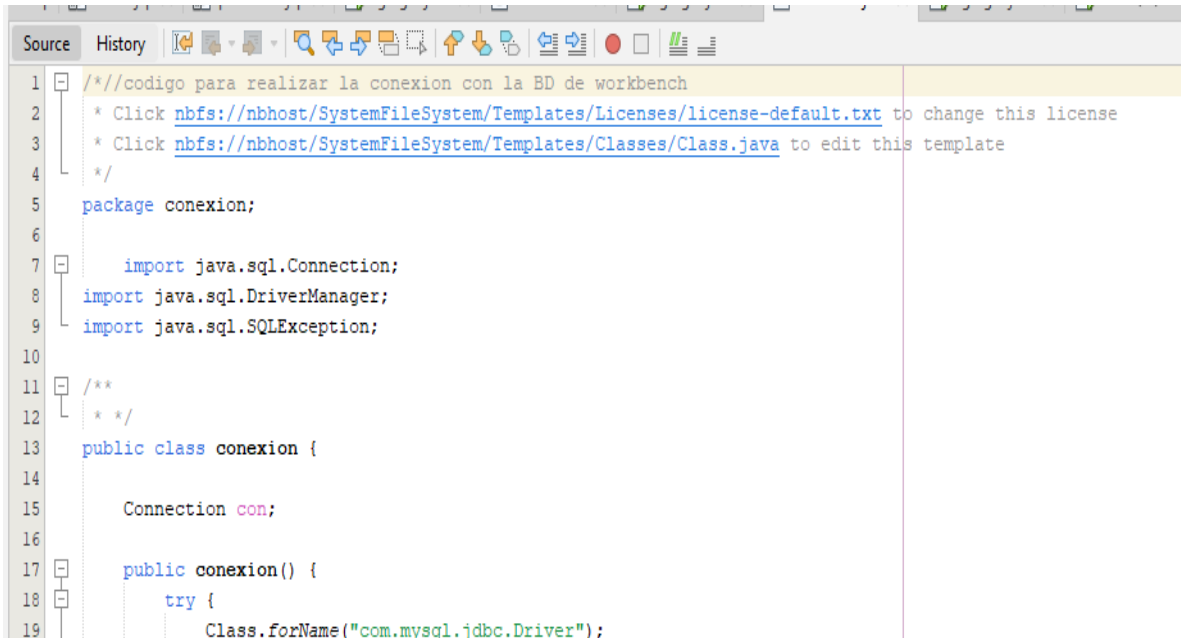
El desarrollo de software moderno exige la capacidad de adaptarse a diferentes plataformas, ya sean stand-alone, web o móviles. Cada tipo de aplicación tiene sus propios requisitos, desafíos y herramientas específicas que deben ser consideradas durante la fase de codificación. Este trabajo tiene como objetivo explorar la codificación de módulos para estos diferentes tipos de software, aprovechando frameworks y herramientas adecuadas para cada contexto. Se prestará especial atención a la importancia de seguir los ciclos de vida del software, asegurando que el código este bien estructurado, comentado y siga estándares reconocidos.

OBJETIVOS

- **Codificación modular:** Implementar módulos de software que sean fácilmente integrables, siguiendo buenas practicas de programación y utilizando frameworks específicos según el tipo de aplicación.
- **Optimización para la plataforma:** Asegurar que la codificación tenga en cuenta las particularidades de la plataforma objetivo, ya sea stand-alone, web o móvil.
- **Gestión de bases de datos:** Implementar una gestión efectiva con bases de datos, utilizando herramientas como MySQL Workbench para aplicaciones web y móviles, asegurando una correcta conexión e interacción.
- **Control de versiones:** Utilizar sistemas de control de versiones como (GIT) para mantener un seguimiento preciso de los cambios, facilitando la colaboración y asegurando la integridad del código a lo largo del proyecto.

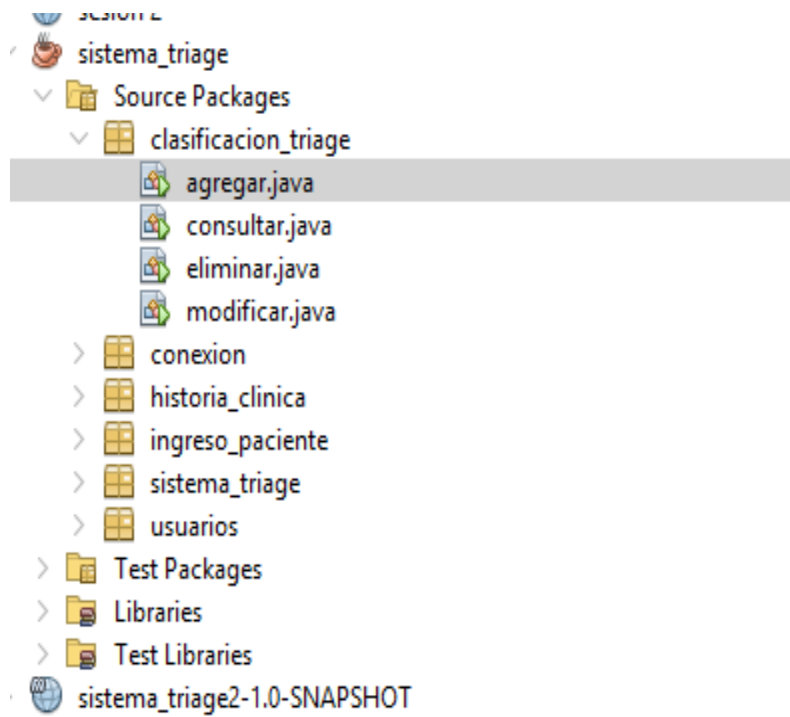
El proceso de codificación incluye:

- Realizamos la conexión con la base de datos, en la línea No. 1 podemos observar el comentario en el código para conservar las buenas practicas de programación.



```
1  /**codigo para realizar la conexion con la BD de workbench
2
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
4  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
5  */
6
7  package conexion;
8
9  import java.sql.Connection;
10 import java.sql.DriverManager;
11 import java.sql.SQLException;
12
13 /**
14  * */
15 public class conexion {
16
17     Connection con;
18
19     public conexion() {
20         try {
21             Class.forName("com.mysql.jdbc.Driver");
```

- Creamos los seis paquetes Java: clasificación_triage, conexión, historia_clinica, ingreso_paciente, sistema_triage, usuarios.



- **Paquete clasificacion_triage:** En la línea No. 1 podemos observar el comentario en el código para conservar las buenas prácticas de programación.

```

1  /**/nos permite clasificar la urgencia y asi poder dar un orden a la atencion de los pacientes
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package clasificacion_triage;
6  import conexion.conexion;
7  import java.sql.Connection;
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10 import java.sql.Statement;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13
14 /**
15  *
16  * @author milto
17  */
18 public class agregar {
19     public static void main (String[] args){

```

- **Paquete conexión:** //conecta NetBeans con la base de datos de Worckbench.

```

10
11 /**
12  */
13 public class conexion {
14
15     Connection con;
16
17     public conexion() {
18         try {
19             Class.forName("com.mysql.jdbc.Driver");
20             con=DriverManager.getConnection("jdbc:mysql://localhost:33060/sistema_triage", "root", "
21         }catch (ClassNotFoundException | SQLException e) {
22             System.out.println("No Conectado");
23         }
24     }
25     public Connection getConexion(){
26         return con;
27     }
28

```

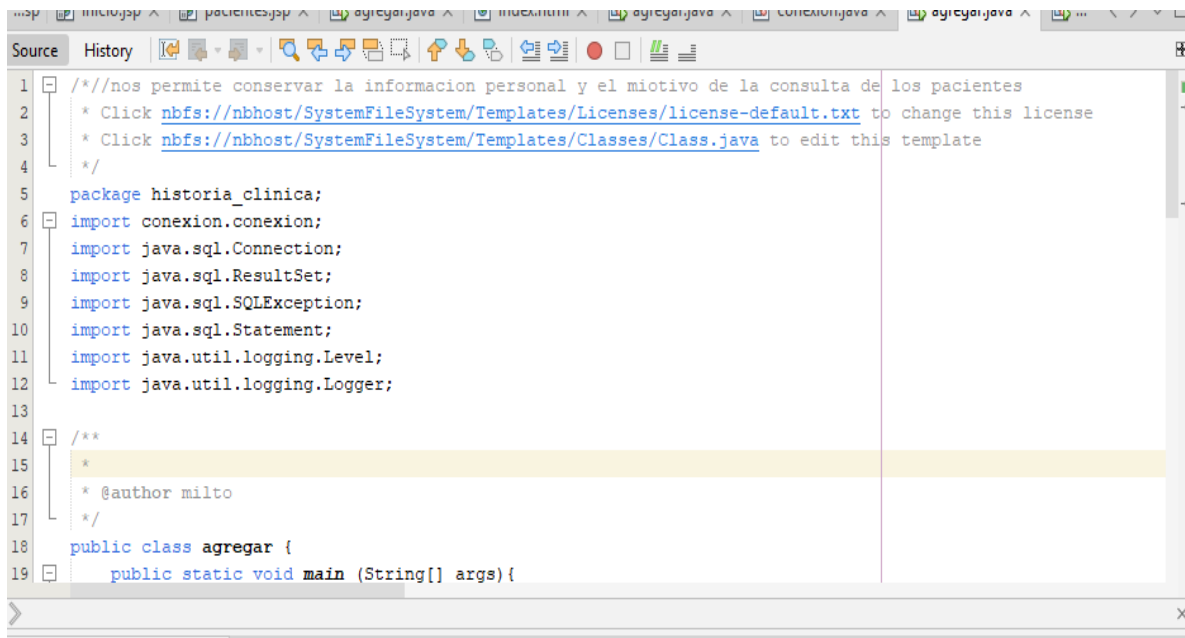
Output - sistema_triage (run) ×

```

run:
Loading class 'com.mysql.jdbc.Driver'. This is deprecated. The new driver class is 'com.mysql.cj.jdbc.Driver'. The
2:cefalea-fractura-nauseas-dolor pie derecho
3:cefalea-fractura-nauseas-dolor pie derecho
BUILD SUCCESSFUL (total time: 2 seconds)

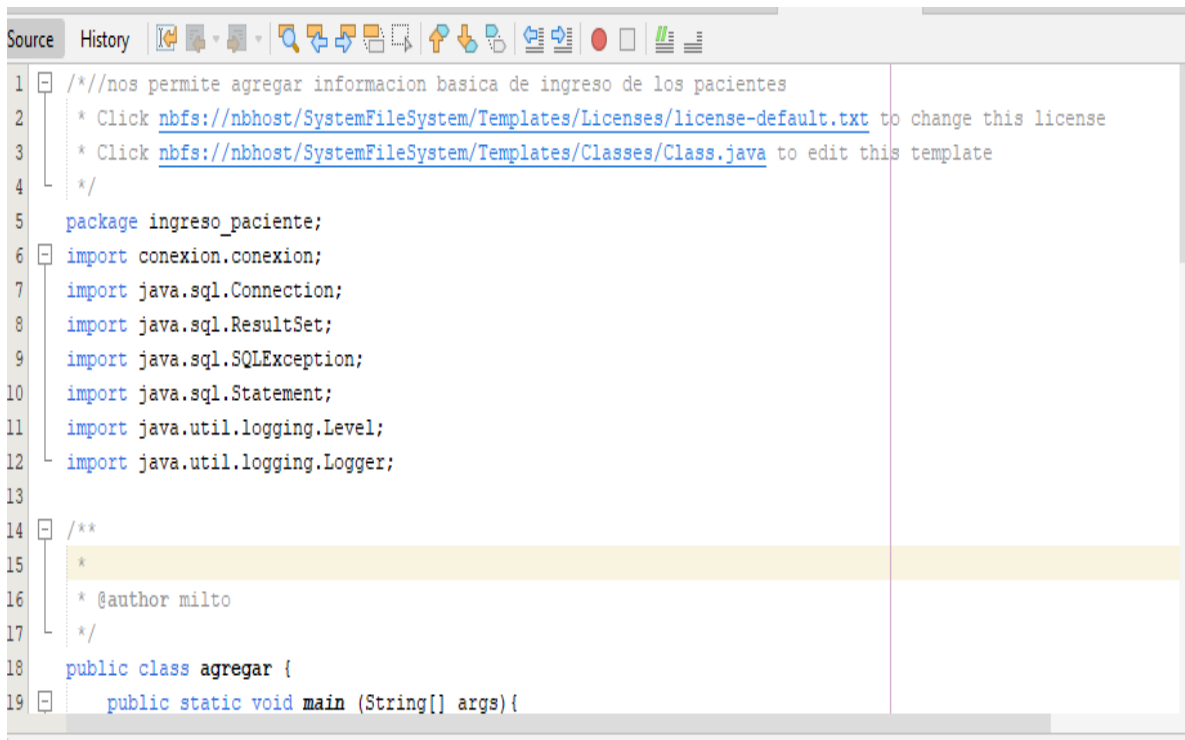
```

- **Paquete historia clínica:** En la línea No. 1 podemos observar el comentario en el código para conservar las buenas prácticas de programación.



```
1  /**nos permite conservar la informacion personal y el motivo de la consulta de los pacientes
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package historia_clinica;
6  import conexion.conexion;
7  import java.sql.Connection;
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10 import java.sql.Statement;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13
14 /**
15 *
16 * @author milto
17 */
18 public class agregar {
19     public static void main (String[] args){
```

- **Paquete ingreso_paciente:** : En la línea No. 1 podemos observar el comentario en el código para conservar las buenas prácticas de programación.



```
1  /**nos permite agregar informacion basica de ingreso de los pacientes
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package ingreso_paciente;
6  import conexion.conexion;
7  import java.sql.Connection;
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10 import java.sql.Statement;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13
14 /**
15 *
16 * @author milto
17 */
18 public class agregar {
19     public static void main (String[] args){
```

- **Paquete usuarios:** : En la línea No. 1 podemos observar el comentario en el código para conservar las buenas prácticas de programación.

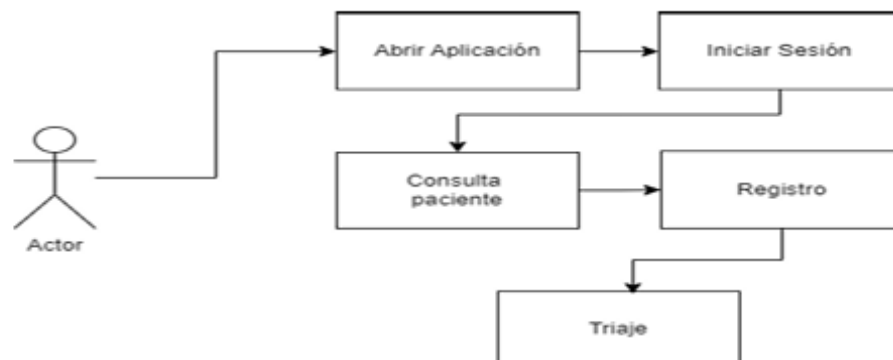
```

1  /**permite agregar usuario y contraseña a los usuarios para que puedan ingresar
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package usuarios;
6   import conexion.conexion;
7   import java.sql.Connection;
8   import java.sql.ResultSet;
9   import java.sql.SQLException;
10  import java.sql.Statement;
11  import java.util.logging.Level;
12  import java.util.logging.Logger;
13
14  /**
15   *
16   * @author milto
17   */
18  public class agregar {
19      public static void main (String[] args){

```

Diagrama de casos de uso del proyecto de software

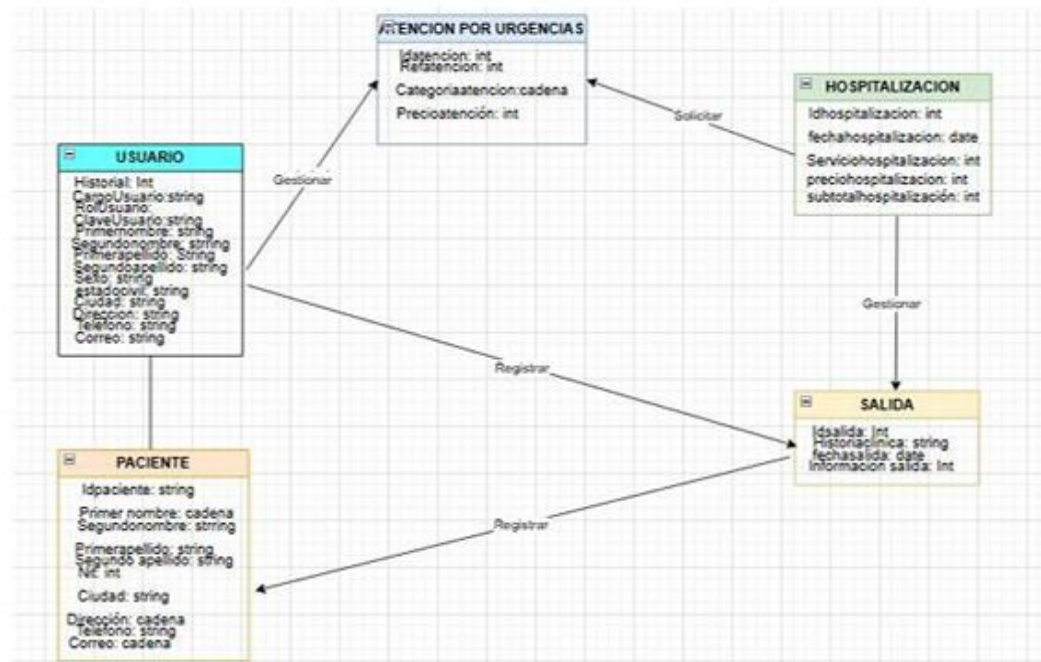
Diagrama 1 Casos de uso Triage



Nota: Diagrama creado por Sergio Andrés Palomino Silva.

Diagrama de clases del proyecto de software

Diagrama 2 Diagrama de clases sistema Triage



Nota: Diagrama creado por Sergio Andrés Palomino Silva.

1. INFORME TECNICO DE PLAN DE TRABAJO PARA CONSTRUCCION DE SOFTWARE

1.1 Control de Versiones Centralizado

1.1.1 Subversión (SVN)

Descripción: Subversión es un sistema de control de versiones centralizado

ampliamente utilizado. Permite rastrear cambios en archivos y directorios a través de un servidor central, lo que facilita la colaboración.

Ventajas:

- Soporte para versionamiento de directorios completos.
- Seguimiento de cambios en binarios.

- Acceso a revisiones anteriores.

Desventajas:

- Requiere conexión constante al servidor central.
- Menos eficiente en la gestión de ramas.

1.2 Control de Versiones Distribuido

1.2.1 Git

Descripción: Git es un sistema de control de versiones distribuido ampliamente utilizado.

Cada desarrollador tiene una copia completa del repositorio, lo que permite un trabajo sin conexión y una gestión eficiente de ramas.

Ventajas:

- Alta velocidad y eficiencia.
- Facilita la colaboración y la gestión de ramas.
- Ampliamente adoptado en la industria.

Desventajas:

- Curva de aprendizaje pronunciada para principiantes.
- No es ideal para gestionar grandes archivos binarios.

2. Plataformas de Hospedaje de Repositorios

2.1 GitHub

Descripción: GitHub es una plataforma de desarrollo colaborativo que utiliza Git como sistema de control de versiones. Ofrece características como seguimiento de problemas, solicitudes de extracción y una amplia comunidad de desarrolladores.

Ventajas:

- Facilita la colaboración entre equipos.
- Integración con numerosas herramientas y servicios.
- Amplia comunidad y recursos de aprendizaje.

Desventajas:

- Algunas funciones avanzadas requieren una suscripción de pago.

2.2 GitLab

Descripción: GitLab es una plataforma similar a GitHub que proporciona una gestión completa de ciclo de vida del desarrollo de software. Puede ejecutarse en servidores propios o en la nube.

Ventajas:

- Ofrece opciones tanto en la nube como en instalaciones locales.
- Gestión de CI/CD integrada.
- Herramientas de gestión de proyectos y seguridad.

Desventajas:

- Menos usuarios y proyectos en comparación con GitHub.

3. Control de Versiones para Datos

3.1 DVC (Data Version Control)

Descripción: DVC es una herramienta de control de versiones diseñada

específicamente para datos y modelos de aprendizaje automático. Permite rastrear cambios

en archivos de datos y modelos de manera eficiente.

Ventajas:

- Gestión de datos y modelos de manera eficiente.
- Integración con sistemas de almacenamiento en la nube.
- Control de versiones sin duplicidad de datos.

Desventajas:

- Enfoque específico para proyectos de datos y ML.

CONCLUSION

La codificación de módulos para aplicaciones stand-alone, web y móviles requiere una adaptación específica según la plataforma objetivo. Utilizando frameworks y herramientas apropiadas, es posible crear soluciones eficientes, seguras y escalables. El uso de buenas prácticas de codificación y la adherencia a estándares reconocidos son cruciales para asegurar la calidad del software, independientemente de la plataforma. Finalmente, la integración de sistemas de control de versiones como Git garantiza un proceso de desarrollo colaborativo y ordenado, que es esencial en desarrollo de proyectos de software modernos.

