**Universidad Veracruzana**

Facultad de Estadística e Informática

**Proyecto Final:**

**S.I.G.A**

Principios de Construcción de Software

Fredy Castañeda Sánchez

Licenciatura en Ingeniería de Software

2 de Junio de 2017

Integrantes:

* Domínguez Delgado Ángel Eduardo
* Domínguez González José Andrés
* Hernández González Esmeralda Yamileth

**Tabla de contenido**

[Estándar de codificación en Java 3](#_Toc484204437)

[Revisión del código apegada al estándar y buenas prácticas de codificación. 6](#_Toc484204438)

[Modelo de clases 8](#_Toc484204439)

[Diagramas de Casos de Uso 9](#_Toc484204440)

[Diagrama de paquetes 9](#_Toc484204441)

[Asesor 9](#_Toc484204442)

[Alumno 10](#_Toc484204443)

[Coordinador 11](#_Toc484204444)

[Diagrama Relacional 12](#_Toc484204445)

[Caso de Uso 01: Reservar Actividad 13](#_Toc484204446)

[Descripción detallada 13](#_Toc484204447)

[Diagrama de secuencia 16](#_Toc484204448)

[Prototipo 17](#_Toc484204449)

[Caso de Uso 32: Registrar Bitácora 18](#_Toc484204450)

[Descripción detallada 18](#_Toc484204451)

[Diagrama de secuencia 21](#_Toc484204452)

[Prototipo 22](#_Toc484204453)

[Programación defensiva 23](#_Toc484204454)

[Pruebas 24](#_Toc484204455)

[Documentación 30](#_Toc484204456)

[Conclusión 32](#_Toc484204457)

[Referencias 33](#_Toc484204458)

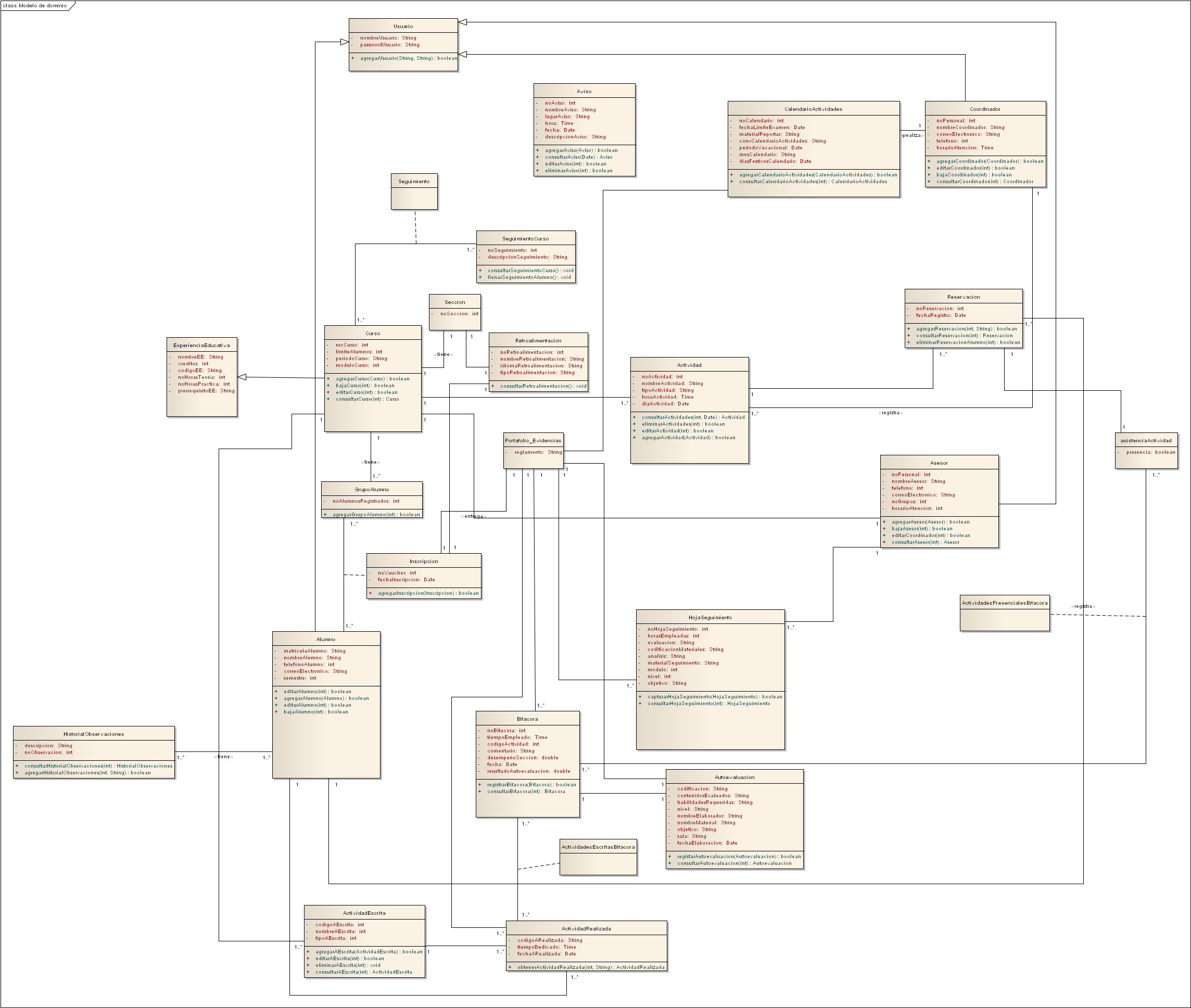
# **Estándar de codificación en Java**

|  |  |
| --- | --- |
| **Propósito** | Guiar el desarrollo de la codificación del proyecto SIGA utilizando el lenguaje de programación Java. |
| **Cabecera del programa** | Comienza todos los programas con un encabezado donde se describan datos generales. |
| **Formato de la cabecera** | /\*\*  \* Nombre: Nombre del programa a desarrollar  \* Versión: Versión actual del programa  \* Autor: Nombre del autor(es) del programa  \* Fecha: Fecha en que se inició el desarrollo  \* Descripción: Una corta descripción de lo que hace el programa  \*/ |
| **Lista de contenidos** | Proporciona un resumen de la lista de contenidos |
| **Ejemplo de contenidos** | /\*\*  \* Lista de contenidos:  \* Importaciones de paquetes  \* Declaraciones de clases:  \*  \* ClaseA  \*  \* ClaseB  \*  \*  \*/ |
| **Identificadores** | Usa nombres descriptivos para las variables, nombres de funciones, constantes, y otros identificadores. Evita las abreviaciones o letras independientes. |
| **Ejemplo de identificadores** | int numeroDeAnimales; /\*Esto está bien\*/  float a, b /\*Esto está mal\*/ |
| **Comentarios** | -Documenta el código para que el lector pueda entender cada operación que realizas.  -Los comentarios deben explicar el propósito y el comportamiento del código. |
| **Buenos comentarios** | If(flag == true) /\*Comprueba si la variable flag es verdadera, si lo es, realiza lo que está dentro de la condición. Flag proviene de la comprobación de archivos\*/ |
| **Malos comentarios** | If(flag == true) /\* Condicional if \*/ |
| **Bloques grandes** | Precede los bloques grandes de código con un comentario donde describas lo que realizas en esa sección. |
| **Ejemplo** | /\*\*  \* La siguiente sección del programa contiene la impresión en pantalla de \* las facturas de ventas realizadas hasta el momento  \*  \*/ |
| **Espacios en blanco** | -Escribe programas con suficiente espacio entre líneas (donde haga falta) para que no se vea amontonado.  -Separa cada constructor del programa con al menos un espacio en blanco. |
| **Indentación** | -Indenta las líneas del código que lo requieran y hagan falta. (Por ejemplo, en ciclos)  -Abre y cierra las llaves asegurando que queden alineadas verticalmente.  -Se le asignarán cuatro espacios a la indentación.  -Se le asignarán 100 caracteres al tamaño total de un renglón. |
| **Ejemplo de indentación** | if(flag == true) {  suma += 8;  if(suma == 10) {  return 10;  }  } |
| **Uso de mayúsculas** | -Las clases empiezan siempre con letra mayúscula  -Las variables y funciones empiezan con minúscula, si se llegarán a unir palabras, a partir de la segunda palabra comenzarían con mayúscula.  -Las constantes van con letras mayúsculas |
| **Ejemplo de uso de mayúsculas** | int variableUno;  final double PI = 3.1415  public void realizarSumaVariables(); |

# **Revisión del código apegada al estándar y buenas prácticas de codificación.**

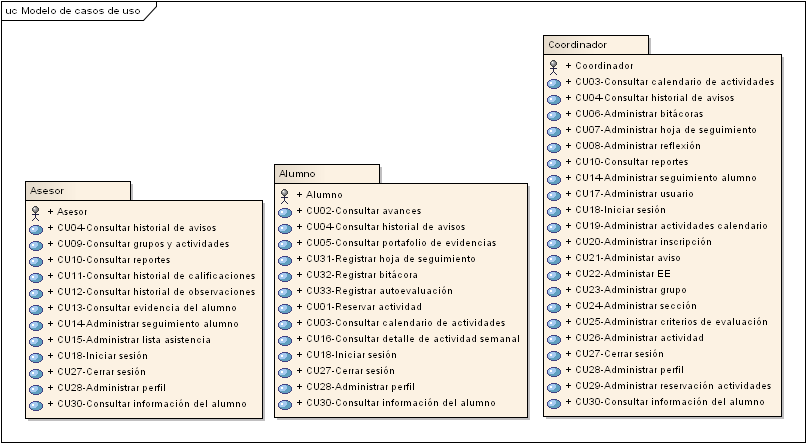
|  |  |  |
| --- | --- | --- |
| **Punto a evaluar** | **Sí** | **No** |
| **Estándar de Codificación** | | |
| El programa contiene una cabecera con los puntos propuestos | X |  |
| El contenido de las clases sigue el órden propuesto | X |  |
| Se utilizan nombres descriptivos para las variables y constantes | X |  |
| Se utilizan nombres descriptivos para los métodos | X |  |
| Los métodos tienen documentación completa | X |  |
| Los comentarios explican el propósito y comportamiento del código | X |  |
| En los bloques largos de código se tienen comentarios sobre su utilización | X |  |
| Existe sufieciente espacio entre líneas de código, evitando que se amontonen ente sí | X |  |
| Se separan los constructores de las clases con al menos un espacio en blanco | X |  |
| El código está indentado y ordenado de acuerdo al estándar | X |  |
| Se utilizan las mayúsculas adecuadamente | X |  |
| **Revisión de métodos** | | |
| El nombre de los métodos describe de forma general lo que estos hacen | X |  |
| El nombre de los métodos es claro, así como su función | X |  |
| Existen suficientes razones para haber creado los métodos | X |  |
| El número de parámetros de los métodos no excede de cinco | X |  |
| En los métodos, todos los parámetros son utilizados | X |  |
| Los métodos realizan una sola cosa | X |  |
| La mayoría del contenido de los métodos son únicos en el programa | X |  |
| Los métodos regresan valores válidos (si fuera el caso) | X |  |
| Extras | | |
| Las consultas de mySQL son parametrizadas para proteger la información de la base de datos | X |  |
| Se lleva a cabo un cifrado de contraseñas de usuario al almacenarlas en la base de datos | X |  |
| Se verifica el ingreso de información a los campos | X |  |
| Se separa el proyecto en modelo de capas | X |  |

# **Modelo de clases**

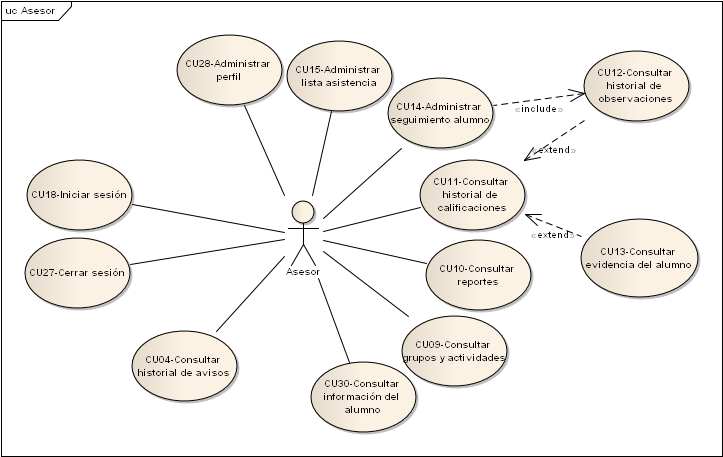


# **Diagramas de Casos de Uso**

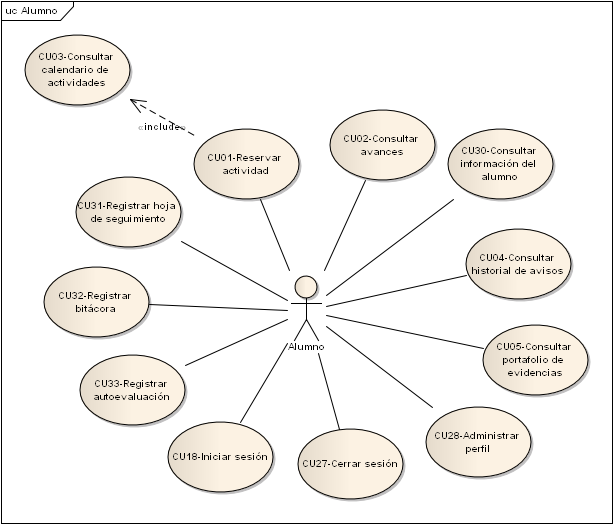
## Diagrama de paquetes



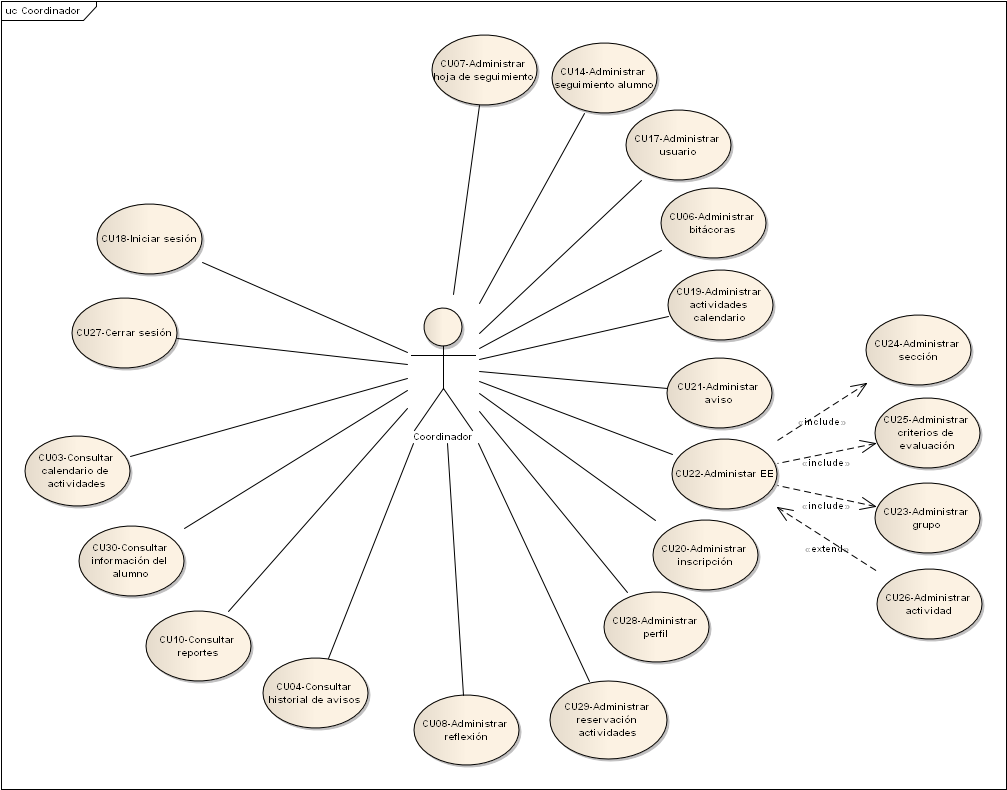
## Asesor



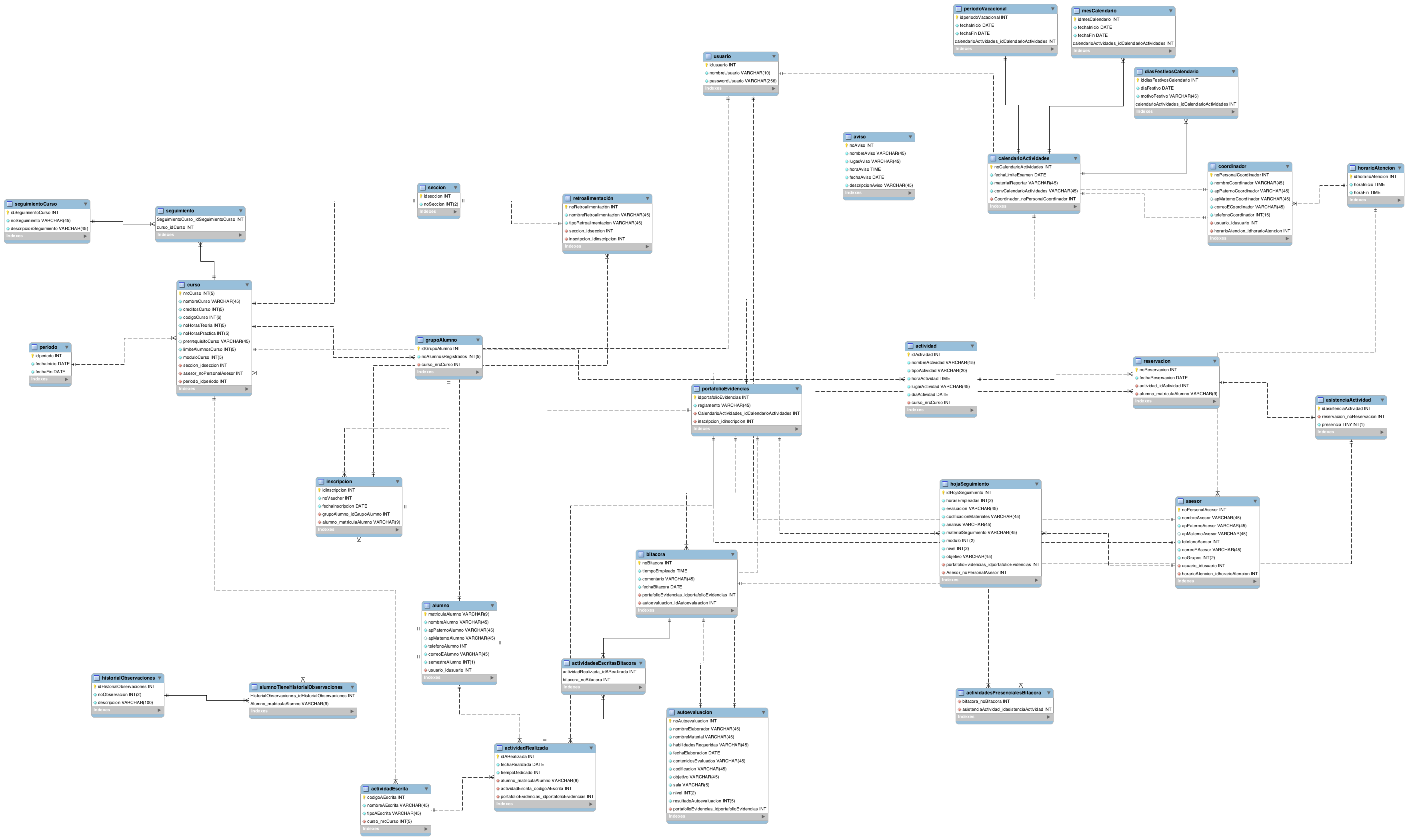
## Alumno



## Coordinador



# **Diagrama Relacional**

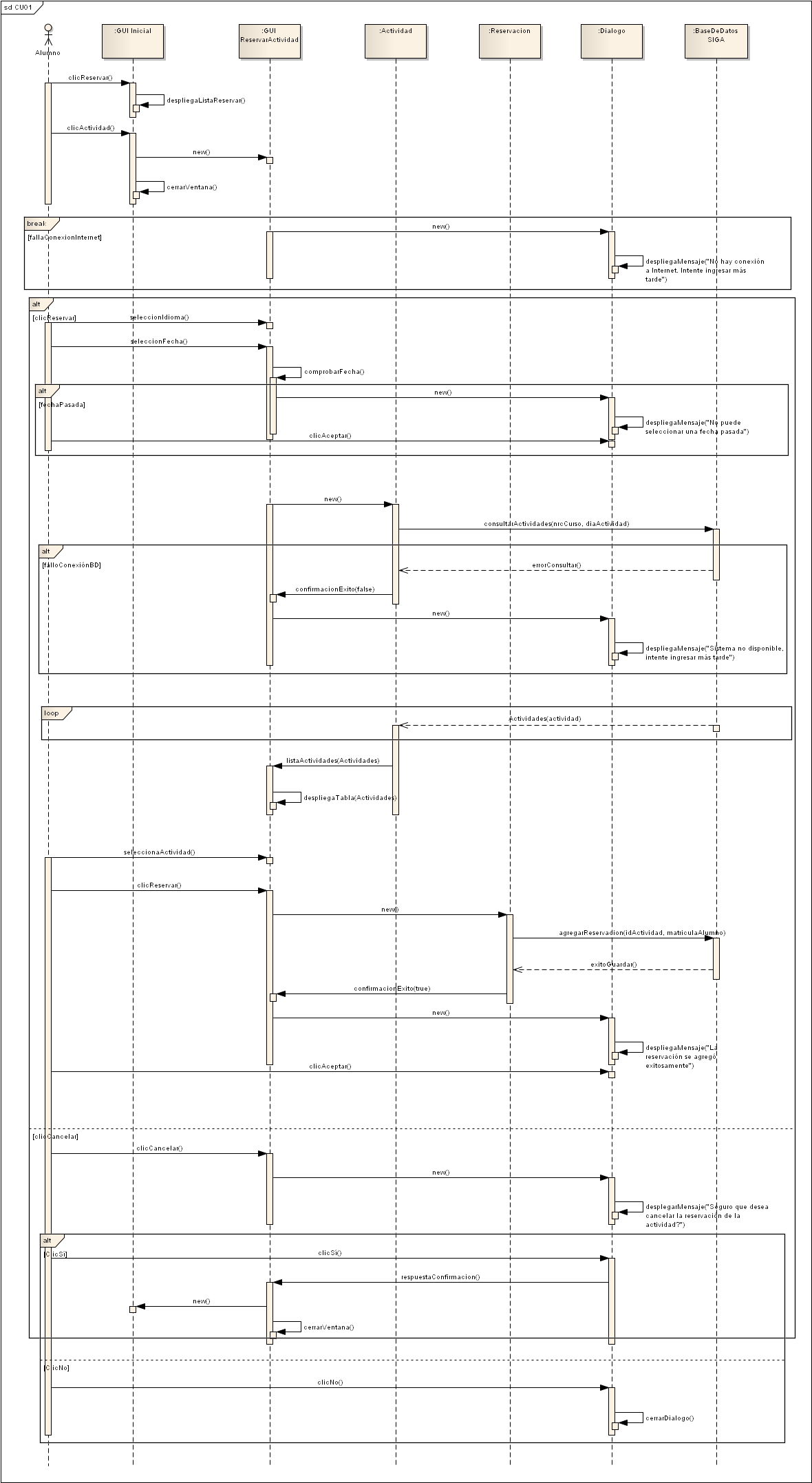


# **Caso de Uso 01: Reservar Actividad**

## Descripción detallada

|  |  |
| --- | --- |
| **ID:** | CU01 |
| **Nombre:** | Reservar Actividad |
| **Autor(es):** | Domínguez Delgado Ángel Eduardo  Domínguez González José Andrés  Hernández González Esmeralda Yamileth |
| **Fecha de creación:** | 15 de abril de 2017 |
| **Fecha de actualización:** | 30 de mayo de 2017 |
| **Actor(es):** | Alumno |
| **Descripción:** | Le permite al Alumno registrar la reservación de alguna actividad que desee llevar a cabo en el centro de autoacceso. |
| **Precondiciones:** | -Haber ingresado al sistema como usuario con privilegios de Alumno  -Tener registro de actividades disponibles en la base de datos. |
| **Flujo Normal:** | 1. El Alumno da clic en la lista desplegable “Reservar”. 2. El Sistema despliega la lista desplegable “Reservar”. 3. El Alumno hace clic en “Actividad”. 4. El Sistema cierra la ventana “Inicial” y crea la ventana “ReservarActividad”. 5. El Alumno selecciona el curso del que quiere reservar actividad en el ComboBox “Curso”. 6. El Alumno selecciona la fecha deseada en el calendario desplegable que se muestra. 7. El Sistema conecta con la base de datos y recupera la información de las ACTIVIDADES (nombreActividad, profesorActividad, tipoActividad y horaActividad) relacionadas con el curso que seleccionó el alumno, mostrándola en una tabla en la ventana. 8. El Alumno selecciona en la tabla la actividad que quiere reservar. 9. El Sistema habilita el botón “Reservar”. 10. El Alumno da clic en el botón “Reservar”. 11. Se crea una nueva RESERVACIÓN (fechaRegistro) y el Sistema la almacena en la base de datos. 12. El Sistema despliega un cuadro de diálogo que dice “La reservación se realizó exitosamente”. 13. El Alumno da clic en el botón “Aceptar”. 14. Fin del caso de uso. |
| **Flujos Alternos:** | 5.1. El Alumno da clic en el botón “Cancelar”.  5.1.1. El Sistema muestra un diálogo de confirmación que dice “¿Seguro que desea cancelar la reservación de la Actividad?”  5.1.1.1. El Alumno da clic en “Aceptar”.  5.1.1.1.1. El Sistema cierra la ventana “ReservarActividad” y crea la ventana “Inicial”.  5.1.1.2. El Alumno da clic en “Cancelar”.  5.1.1.2.1. El Sistema cierra el diálogo de confirmación.  6.1. El Alumno selecciona una fecha que ya pasó.  6.1.1. El Sistema muestra un diálogo de alerta que dice “No puede seleccionar una fecha pasada”.  6.1.2. El Alumno da clic en “Aceptar”. |
| **Excepciones:** | 4.1. El Sistema perdió la conexión con internet.  4.1.1. El Sistema muestra un mensaje en la ventana “ReservarActividad” que dice “No hay conexión a Internet, intente ingresar más tarde”.  7.2. El Sistema perdió la conexión con la base de datos.  7.2.1. El Sistema muestra un mensaje en la ventana “ReservarActividad” que dice “Sistema no disponible, intente ingresar más tarde”. |
| **Poscondiciones:** | El Alumno reserva una Actividad en el Sistema |
| **Prioridad:** | Alta |

## Diagrama de secuencia



## Prototipo

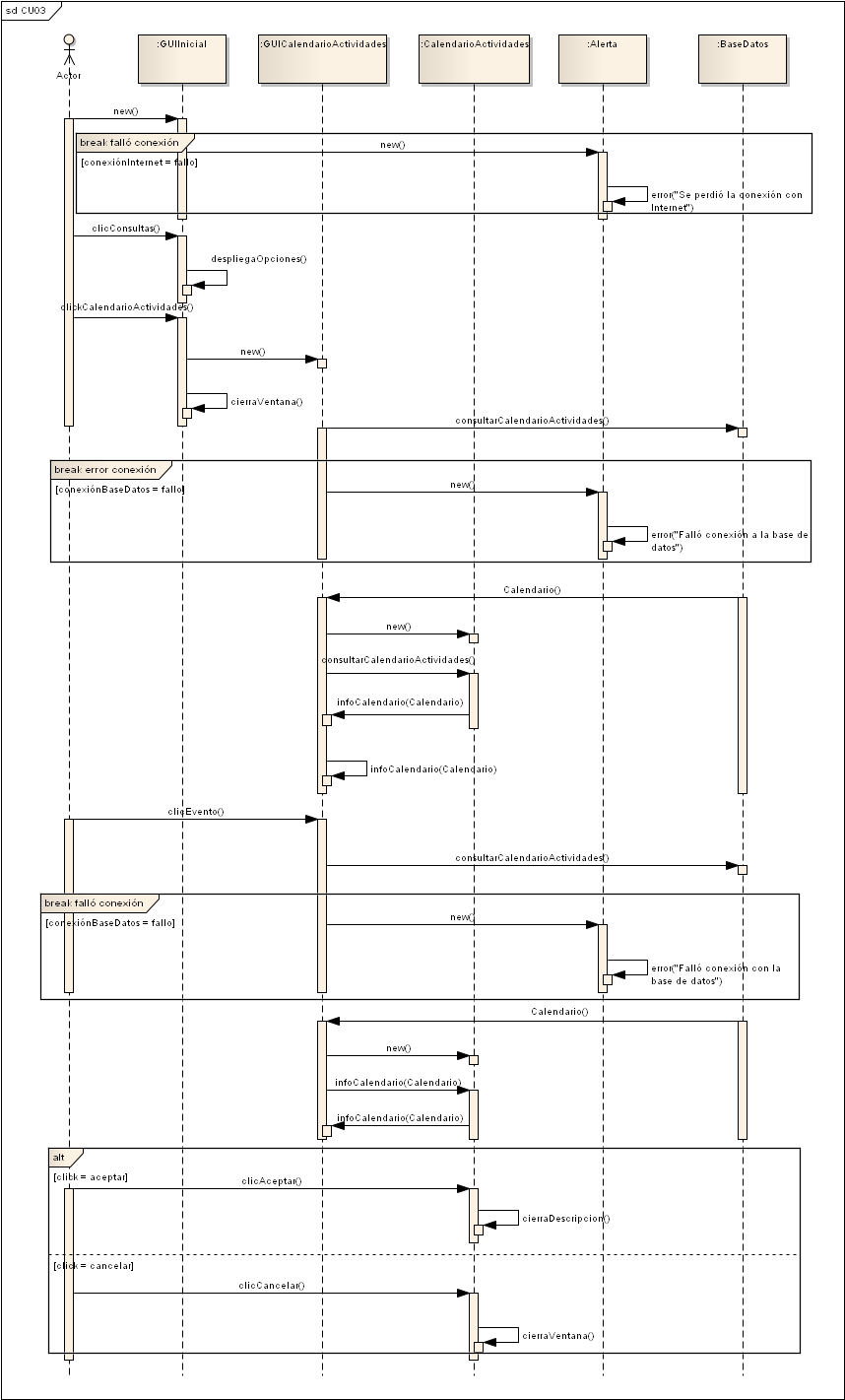


# **Caso de Uso 32: Registrar Bitácora**

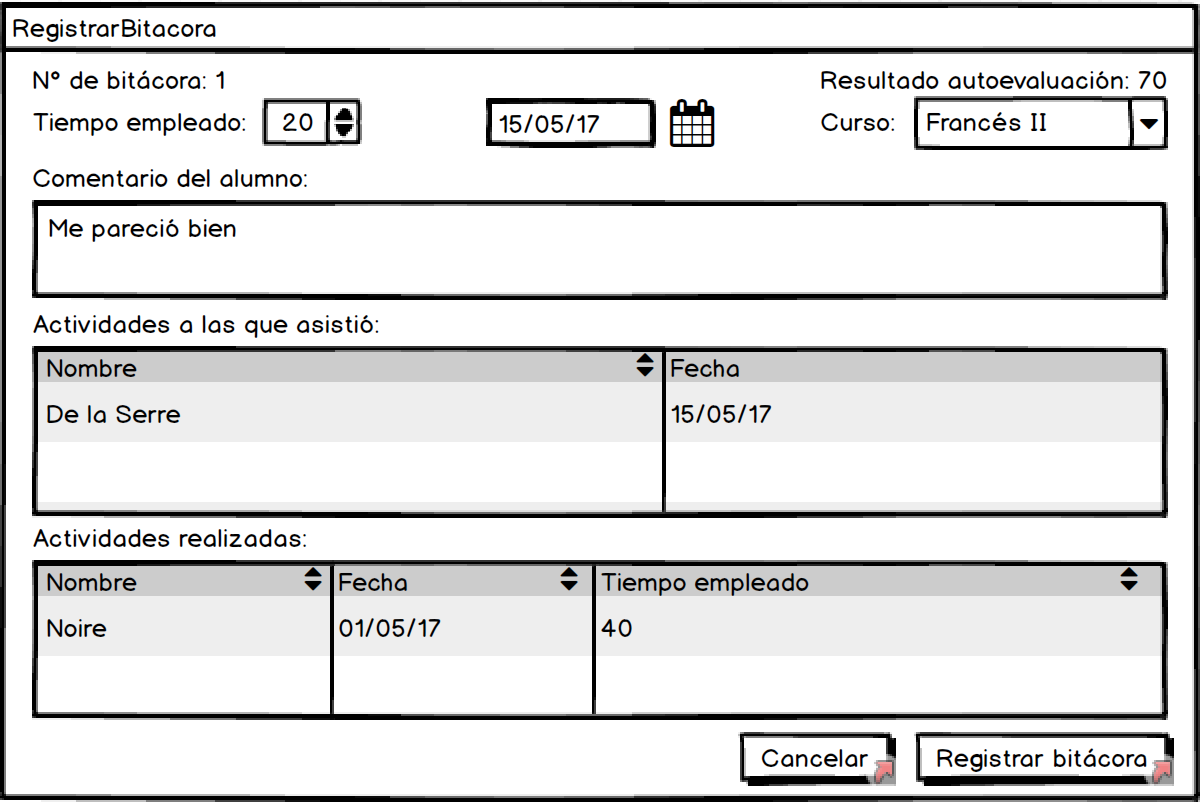
## Descripción detallada

|  |  |
| --- | --- |
| ID: | CU35 |
| Nombre: | Registrar Bitácora |
| Autor(es): | Domínguez Delgado Ángel Eduardo  Domínguez González José Andrés  Hernández González Esmeralda Yamileth |
| Fecha de creación: | 22 de abril de 2017 |
| Fecha de actualización: | 1 de junio de 2017 |
| Actor(es): | Alumno |
| Descripción: | Le permite al Alumno registrar la bitácora (fecha de visita, minutos empleados, comentarios) de la sección actual en el Sistema. |
| Precondiciones: | -El Alumno debe estar inscrito por lo menos en una sección |
| Flujo Normal: | 1. El Alumno da clic en la lista desplegable “Registrar”. 2. El sistema despliega la lista desplegable “Registrar”. 3. El Alumno da clic en “Bitácora”. 4. El Sistema crea la ventana “RegistrarBitacora” y cierra la ventana “Inicial”. 5. El Sistema conecta con la base de datos y recupera las ACTIVIDADESESCRITAS (codigoAEscrita, nombreAEscrita, tipoAEscrita) que haya realizado el Alumno y las muestra en la tabla “Actividades a las que asistió”. 6. El Sistema conecta con la base de datos y recupera las ACTIVIDADESREALIZADAS (codigoARealizada, fechaARealizada) escritas que haya realizado el alumno y las muestra en la tabla “Actividades realizadas”. 7. El Alumno completa los campos (“Tiempo empleado”, “Fecha de la actividad”, “Curso” y “Comentarios del alumno”). 8. El Sistema verifica que la fecha seleccionada sea válida (que sea anterior al día actual y que no se pase de la fecha límite entrega). 9. El Alumno da clic en el botón “Registrar bitácora”. 10. El Sistema conecta con la base de datos y almacena los datos de la BITACORA (noBitacora, tiempoEmpleado, codigoActividad, comentario, desempenoSeccion, fecha, resultadoAutoevaluacion). 11. El Sistema muestra una alerta que dice: “La bitácora se guardó de manera exitosa”. 12. El Alumno da clic en el botón “Aceptar” de la alerta. 13. El Sistema vacía todos los campos y actualiza el número de bitácora. 14. Fin del caso de uso. |
| Flujos Alternos: | 7.1. La fecha ingresada por el Alumno es una fecha que aún no ocurre.  7.1.1. El Sistema muestra un mensaje informándole que: “Esa fecha aún no llega”.  7.2. El Sistema detecta que la fecha límite de entrega ya pasó CALENDARIOACTIVIDADES(fechaLimiteExamen) y le muestra un aviso al Alumno que dice: “La fecha límite ya ha pasado, no podrás subir las bitácoras de esta sección”.  7.2.1. El Alumno hace clic en el botón “Aceptar”.  7.2.2. El Sistema cierra la ventana “RegistrarBitacora”.  9.1. El Alumno da clic en el botón “Cancelar”.  9.1.1. El Sistema crea la ventana “Inicial” y cierra la ventana “RegistrarBitacora”. |
| Excepciones: | * 1. El Sistema perdió la conexión con Internet.      1. El Sistema muestra un mensaje en la ventana “Inicial” que dice “Se perdió la conexión con Internet, inténtelo nuevamente”.   5.1. El Sistema perdió la conexión con la base de datos     5.1.1. El Sistema muestra un mensaje en la ventana “RegistrarBitacora” que dice “Sistema no disponible, intente ingresar más tarde”.  6.1. El Sistema perdió la conexión con la base de datos     6.1.1. El Sistema muestra un mensaje en la ventana “RegistrarBitacora” que dice “Sistema no disponible, intente ingresar más tarde”.  10.1. El Sistema perdió la conexión con la base de datos     10.1.1. El Sistema muestra un mensaje en la ventana “RegistrarBitacora” que dice “Sistema no disponible, intente ingresar más tarde”. |
| Poscondiciones: | -La bitácora quedó registrada exitosamente en la base de datos |
| Prioridad: | Alta |

## Diagrama de secuencia



## Prototipo



# **Programación defensiva**

En este proyecto se utilizaron principalmente dos técnicas para hacer el sistema más seguro, el cifrado de las contraseñas del usuario y la implementación de consultas parametrizadas.

Es bien sabido dese hace tiempo que los maneadores de bases de datos y, por lo tanto, los sistemas que dependen de ellos son susceptibles a inyecciones de código, es decir, introducir código mediante los propios medios del sistema para obtener información sensible, modificar permisos, etc. Una manera de evitar la inyección de código es mediante la implementación de consultas parametrizadas. Estas consultas son consultas comunes de MySQL, pero en lugar de concatenar los datos que se necesitan en nuestro programa, se definen parámetros con un tipo específico, por lo que cualquier valor extraño es rechazado automáticamente antes de que la consulta se lleve a cabo si quiera.

Con la implementación de las consultas parametrizadas es muy complicado que algún usuario malicioso sea capaz de recuperar información sensible del sistema, pero como medidas de seguridad adicional, las contraseñas de los usuarios se encuentran cifradas mediante un algoritmo criptográfico de 256, en concreto sha-2. Las contraseñas se envían y se reciben cifradas a través de parámetros, por lo que incluso, si alguien pudiera tener acceso a ellas, no serían legibles.

# **Pruebas**

**UsuarioTest.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre método | Tipo de retorno esperado | Valores de entrada | Resultado esperado | Resultado obtenido |
| consultaUsuario() | Boolean | String nombre = “zS15011624”;  String pass = “angeldominguez”; | True | True |
| obtenerPassword() | String | String pass = “angeldominguez”; | "c45bb4815941a3722ca2182bf4e707afc74bbdc745077722cbeff4db3fa4ce7c" | "c45bb4815941a3722ca2182bf4e707afc74bbdc745077722cbeff4db3fa4ce7c" |
| existeUsuario() | Boolean | String nombre = “zS15011624”; | True | True |

**RegistrarBitacoraTest.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre método | Tipo de retorno esperado | Valores de entrada | Resultado esperado | Resultado obtenido |
| registrarBitacora() | Boolean | Integer tiempoEmpleado = 100;  String comentario = “ejercicios de audio”;  Java.sql.Date fecha = java.sql.Date.valueOf(“2017-06-01”);  int idPortafolio = 1;  int idAutoevaluacion = 1; | True | True |

**BitacoraTest.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre método | Tipo de retorno esperado | Valores de entrada | Resultado esperado | Resultado obtenido |
| recuperarNoActualBitacora() | int | int noBitacora = 2; | 3 | 3 |
| recuperarUltimoNoBitacora() | int | int noBitacora = 2; | 2 | 2 |
| guardarActividadesAsistidas() | boolean | int nrcCurso = 37182: | True | True |
| guardarActividadesEscritas() | boolean | int nrcCurso = 1; | True | True |
| fechaActual() | LocalDate |  | LocalDate.now() | LocalDate.now() |
| comprobarBitacoraExistente() | boolean | int nrcCurso = 39182; | False | False |

**ActividadRealizadaTest.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre método | Tipo de retorno esperado | Valores de entrada | Resultado esperado | Resultado obtenido |
| consultarActividades() | ObservableList<ActividadRealizada> | int nrcCurso = 37182;  String matricula = “S15011624”; | listaActividades | listaActividades |

**ActividadTest.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre método | Tipo de retorno esperado | Valores de entrada | Resultado esperado | Resultado obtenido |
| consultarActividades() | ObservableList<Actividad> | int nrcCurso = 37182;  java.sql.Date fecha = java.sql.Date.valueOf("2017-06-01"); | listaActividades | listaActividades |
| consultarActividadesAsistidas() | ObservableList<Actividad> | int nrcCurso = 37182;  String matricula = "S15011624"; | listaActividades | listaActividades |
| obtenerIDActividad() | int | String nombreActividad = “OUI”;  Java.sql.Time.valueOf(“15:00:00”); | 3 | 3 |

**CalendarioActividadesTest.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre método | Tipo de retorno esperado | Valores de entrada | Resultado esperado | Resultado obtenido |
| recuperarFechaLimiteExamen() | LocalDate | int nrcCurso = 37182;  int idPortafolio = 1;  int idCalendario = 1;  LocalDate fecha = LocalDate.parse(“2017-03-10”); | “2017-03-10” | “2017-03-10” |

**AutoevaluacionTest.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre método | Tipo de retorno esperado | Valores de entrada | Resultado esperado | Resultado obtenido |
| obtenerNoAutoevaluacion() | int | int nrcCurso = 39182;  String matricula = "S15011624"; | 1 | 1 |
| obtenerResultadoAutoevaluacion() | int | String matricula = "S15011624"; | 95 | 95 |

**CursoTest.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre método | Tipo de retorno esperado | Valores de entrada | Resultado esperado | Resultado obtenido |
| obtenerCursos() | ObservableList<Curso> | String matricula = "S15011624"; | [Francés I, Inglés II] | [Francés I, Inglés II] |

**ReservacionesTest.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre método | Tipo de retorno esperado | Valores de entrada | Resultado esperado | Resultado obtenido |
| agregarReservacion() | boolean | int idActividad = 2;  String matricula = "S15011624"; | True | True |
| comprobarReservaciones() | boolean | String matricula = “S15011624”;  int idActividad = 2; | False | False |

**PortafolioEvidenciasTest.java**

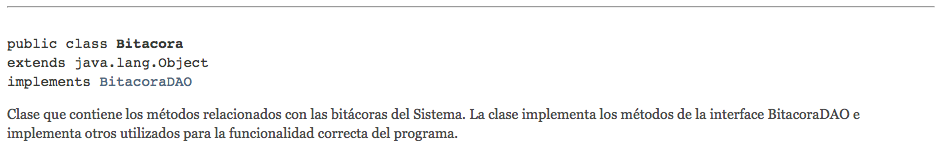
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre método | Tipo de retorno esperado | Valores de entrada | Resultado esperado | Resultado obtenido |
| recuperarIDPortafolio() | int | int nrcCurso = 37182; | 1 | 1 |

# **Documentación**

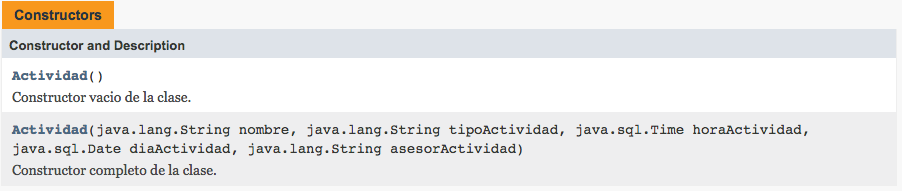
"If your program isn't worth documenting, it probably isn't worth running"

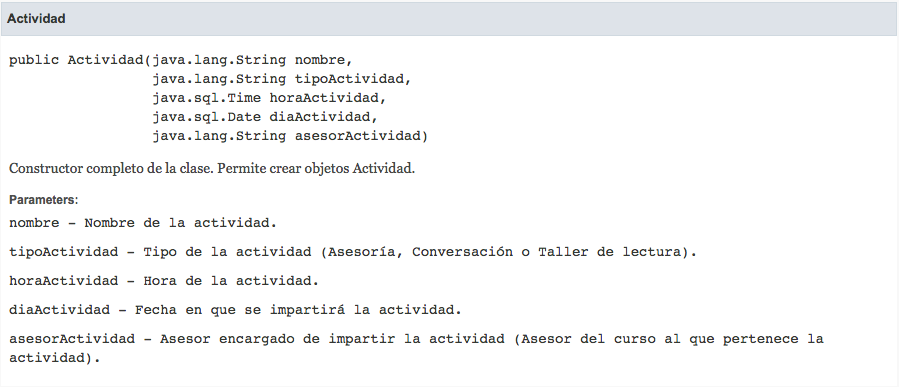
J. Nagler. 1995

La documentación de este proyecto fue un tema importante desde el comienzo, al tratarse de un proyecto en equipo y no poder estar en el mismo espacio físico, es necesario que todos los integrantes comprendan cómo funcionan los métodos, que parámetros recibirán y que valores deberían devolvernos, por lo tanto, todas las clases y métodos fueron debidamente documentados. Además, en las partes confusas del código se aclaró su funcionamiento mediante comentarios.

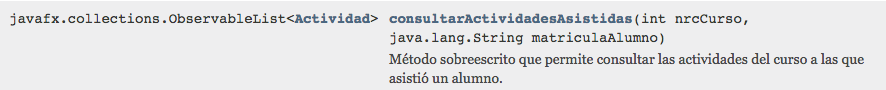


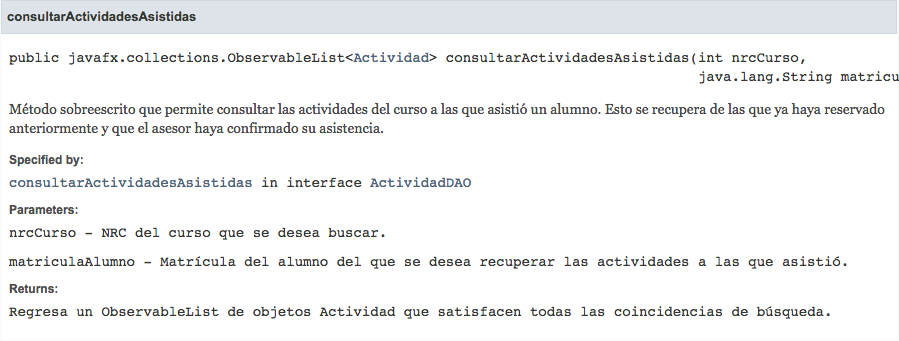
En la imagen anterior se muestra la documentación de la clase “Bitacora”, se describe la función de la clase, la implementación que hace de la interfaz “BitacoraDAO” y se aclara que además de los métodos que se sobrescriben de la interfaz, la clase cuenta con los propios.





Los constructores se muestran de manera resumida con todos los parámetros que llevan y una descripción y se puede ver una descripción detallada de los mismos donde se especifica cual es la función de cada uno de los parámetros.





La documentación de los métodos es similar a la de las clases y constructores, se define cual es la función del método y se define el tipo, el nombre y el significado de los parámetros de entrada, así como los valores que retornarán y cuál es el significado de los mismos.

# **Conclusión**

**Domínguez Delgado Angel Eduardo**

“If it can be used incorrectly, it will”.

-Es la frase de la Ley de Murphy aplicada al código de cualquier software existente.

El presente proyecto es el inicio de nuestras buenas prácticas de programación. A lo largo del curso vimos desde lo básico (como cuidar la simplicidad y eficencia de los métodos) hasta temas medios como debuguear o integrar proyectos. Puedo decir que hubo temas muy aburridos, pero también hubo temas muy interesantes, personalmente, lo que más me intereso fue la programación defensiva y cómo el código puede ser “tronado” de diferentes maneras, incluso mientras el usuario no busque hacerlo.

Todo lo que se vio en el curso fue aplicado a este proyecto final, incluso temas que no pudieron tocarse a profundidad, como inyección SQL. Además, sinceramente, pusimos mucho énfasis en él ya que es nuestra base para proyectos futuros y la forma en cómo vamos mejorando nuestra manera de programar y volvernos cada vez mejores programadores. Se nota nuestra mejoría porque no codificamos de la misma manera que lo hacíamos al iniciar el curso que ahora que lo estamos finalizando.

Es importante proteger y defender nuestro código, nunca sabemos quién pueda llegar y hacer cualquier cosa que se le antoje y tenga la posibilidad de “romperlo”, eso habla mal de un programador y cae su reputación, siempre es importante protegerlo lo más que podamos, aunque eso implique más recursos de los necesarios. Leí en un artículo que vale mucho más un código que haga pocas cosas pero que siempre funciona y esté muy protegido a uno que hace demasiadas cosas, pero falla ocasionalmente.

Está claro que aún debemos acostumbrarnos a diseñar completamente y de una manera correcta, y después empezar la codificación. No es nada fácil adaptarse a eso, ya que al inicio puedes saltarte varias cosas de las cuales te percatas hasta que lo estás codificando y requieres de ellas. Espero que con más práctica el diseño se vaya volviendo más claro, y haya menos “huecos” o errores al momento de codificar, permitiendo cambiar lo mínimo en los diagramas cuando se esté escribiendo todo el programa.

En conclusión, aprendimos muchísimas cosas con este proyecto final, que de ahora en adelante tendremos siempre presentes a la hora de escribir código, porque al no añadirlas estaríamos ignorando buenos principios de programación y le restaríamos orden, seguridad, e incluso simplicidad al código.

# **Referencias**

Autor desconocido. Debugging. Recuperado del archivo PDF proporcionado en el curso.

Autor desconocido. Design in Construction. Recuperado del archivo PDF proporcionado en el curso.

Autor desconocido. Integration. Recuperado del archivo PDF proporcionado en el curso.

Brooks, F. (1986). No Silver Bullet – Essence and Accident in Software Engineering. University of North Carolina at Chapel Hill.

Google Java Style Guide. Recuperado de https://google.github.io/styleguide/javaguide.html

Java™ Platform, Standard Edition 7 API Specification. Recuperado de https://docs.oracle.com/javase/7/docs/api/

Khalid, M. & Lian, N. (2012). Transforming Object Oriented Data Model to Relational Data Model. University of Sultan Zainal Abidin.

Long, F., Mohindra, D., Seacord, R., Sutherland, D. & Svoboda, D. (2014). Java Coding Guidelines. United States of America: Pearson Education.

Martin, R. (2009). Clean Code. A Handbook of Agile Software Craftmanship. United States of America: Pearson Education.

McConnell, S. (2004). Code Complete. Washington: Microsoft Press.

Stafford, J. (2013). Data Access Objects (DAOs). An Introduction to DAOs, DB Schema, and SQL.