

IMPLEMENTACIÓN DE UN AMBIENTE INTEGRADO DE DESARROLLO ONLINE
COMO HERRAMIENTA DE APOYO AL PROCESO EDUCATIVO EN LA
MATERIA FUNDAMENTOS DE PROGRAMACIÓN, PERTENECIENTE AL PLAN
DE ESTUDIOS DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD
FRANCISCO DE PAULA SANTANDER.
(Proyecto de Grado Dirigido)

CARLOS ANDRÉS SEPÚLVEDA SÁNCHEZ

WILSON YESID RIVERA CASAS

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
CÚCUTA
2012

IMPLEMENTACIÓN DE UN AMBIENTE INTEGRADO DE DESARROLLO ONLINE
COMO HERRAMIENTA DE APOYO AL PROCESO EDUCATIVO EN LA
MATERIA FUNDAMENTOS DE PROGRAMACIÓN, PERTENECIENTE AL PLAN
DE ESTUDIOS DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD
FRANCISCO DE PAULA SANTANDER.
(Proyecto de Grado Dirigido)

CARLOS ANDRÉS SEPÚLVEDA SÁNCHEZ
Código 1150068

WILSON YESID RIVERA CASAS
Código 1150010

Trabajo de grado para ostentar el título de Ingenieros de Sistemas de la
Universidad Francisco de Paula Santander.

DIRECTOR
MARCO ANTONIO ADARME JAIMES
INGENIERO DE SISTEMAS
MAGÍSTER EN CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
CÚCUTA
2012

TABLA DE CONTENIDO

INTRODUCCIÓN

DATOS DE EJECUCIÓN

1. PRESENTACIÓN GENERAL DEL ANTEPROYECTO	2
1.2 PLANTEAMIENTO DEL PROBLEMA.....	2
1.3 JUSTIFICACIÓN.....	4
1.4 OBJETIVOS.....	6
1.4.2 Objetivos Específicos	6
1.5 ALCANCE Y DELIMITACIONES	6
1.5.1 ALCANCE	6
1.5.2 LIMITACIONES	7
2. MARCO TEÓRICO O REFERENCIAL	8
2.1 ANTECEDENTES EN LA SOLUCIÓN DEL PROBLEMA.....	8
2.1.1 “IDEWeb (Entorno de Desarrollo Integrado en Web)”	8
2.1.2 JAWAWIDE	8
2.1.3 Diseño de un entorno de desarrollo integrado para una unidad controladora de procesos	9
2.1.4 laboratorios <i>online</i> del DIT	9
2.1.5 IDE's COMERCIALES.....	10
2.2 MARCO TEÓRICO	10
2.2.1 IDE	10
2.2.2 JAVA	10
2.2.3 Software Web.....	12
2.2.4 Cloud Computing.....	14
2.2.5 <i>eXtreme Programming XP</i>	16
2.3 MARCO CONCEPTUAL	21
2.3.2 Internet	21

2.3.3	World Wide Web (WWW).....	21
2.3.4	Compilador.....	21
2.3.5	Lenguajes programación.....	21
2.3.6	Programación Orientada a Objetos	22
2.4.	FUNDAMENTOS LEGALES.....	22
2.4.1	Acuerdos De Uso	22
2.4.2	Contrato De Licencia De Código Binario. Sun Microsystems, Inc.	23
2.4.3	<i>GNU General Public License</i>	24
2.4.4	<i>GNU AFFERO General Public License</i>	24
2.4.5	Propiedad Intelectual.....	24
2.5	ANTECEDENTES	25
3.	DISEÑO METODOLÓGICO.....	28
3.1	TIPO DE INVESTIGACIÓN	29
3.2	FUENTES DE INFORMACIÓN	29
3.2.1	Fuentes de Información Primaria.....	29
3.2.2	Fuentes de Información Secundaria.....	29
3.3	RECOLECCIÓN DE LA INFORMACIÓN.....	30
3.4	ANÁLISIS DE INFORMACIÓN	31
4.	CARACTERÍSTICAS Y GENERALIDADES DE LOS IDE'S WEB.....	33
4.1.	ENTORNOS DE DESARROLLO INTEGRADOS PARA JAVA.....	33
4.2.	EVALUACIÓN DE LAS FUNCIONES DE LOS IDE'S.....	42
4.2.1.	Características Básicas	43
4.2.2.	Características avanzadas	46
4.2.3.	Matriz de Caracterización de los IDE'S	49
4.3.	TABULACIÓN Y ANÁLISIS	52
4.3.1.	Tabulación	52
4.3.2.	Análisis	54
5.	TRABAJO INVESTIGATIVO	58
5.1.	DETERMINACIÓN DE LA POBLACIÓN Y MUESTRA.....	58
5.1.1.	Población.....	58

5.1.2. Muestra	59
5.1.3. Estratificación de la Muestra	60
5.2. TABULACIÓN Y ANÁLISIS DE ENCUESTAS	61
5.2.1. Análisis de Resultados	61
5.2.2. Análisis General	74
6. GENERALIDADES DE JAVA.....	75
6.1. MIRANDO DENTRO DE LA PLATAFORMA DE JAVA	75
6.2. LA ARQUITECTURA DE LA MÁQUINA VIRTUAL DE JAVA	76
6.3. ENTORNOS DE COMPILACIÓN Y EJECUCIÓN	77
6.4. COMPILADOR DE JAVA.....	78
6.4.1. Funcionamiento del Compilador	79
6.5. ARCHIVOS DE FORMATO CLASS	81
6.2.1. Estructura General de Archivo .class	81
6.6. API DE JAVA.....	82
6.8 AMBIENTES DE EJECUCIÓN PARA PROGRAMAS JAVA	90
6.8.1. Ejecución Local	90
6.8.2. Ejecución Remota	90
6.8.2.1. Applets Firmados	94
6.8.2.2. Proceso de firmado de un applet	94
7. RESULTADOS OBTENIDOS.....	97
8. METODOLOGÍA DE DESARROLLO.....	103
8.1 EXPLORACIÓN.....	104
8.1.1. Identificación De Roles.....	104
8.1.2. Identificación De Actores Del Sistema.....	106
8.1.3. Familiarización Con Tecnologías, Herramientas y Prácticas.....	109
8.1.4. Historias de Usuario	121
8.5. PLANIFICACIÓN	125
8.5.1 Priorización y Estimación de las Historias de Usuario.....	125
8.5.2. Plan de Iteraciones.....	127
8.5.3. Metáfora del Sistema.....	128

8.6. ITERACIONES	128
8.6.1. Iteración 1	128
8.6.2. Iteración 2	131
8.6.3. Iteración 3	135
8.6.4. Iteración 4	137
8.6.5. Iteración 5	144
8.6.6. Iteración 6	148
8.6.7. Iteración 7	153
8.6.8. Iteración 8	157
8.6.9. Iteración 9	162
8.6.10. Iteración 10	165
8.6.11. Iteración 11	171
8.6.12. Iteración 12	175
9. IMPLEMENTACIÓN	181
9.1. ARQUITECTURA	181
9.2. PATRONES	182
9.3. PERSISTENCIA DE DATOS, ARCHIVOS DE INDEXACIÓN Y CONFIGURACIÓN	184
9.3.1. Usuarios En Espera	186
9.3.2. Usuarios Activos	188
9.3.3. Directorio de un Programador en el Servidor	190
9.3.4. Fichero XML de Configuración del Proyecto	192
9.3.5. Fichero XML de Librerías	193
9.3.6. Fichero XML para Otros Ficheros	194
9.3.7. Fichero XML para un Paquetes	194
9.3.8. Fichero XML para los Archivos Fuente	195
9.3.9. Fichero XML para Proyectos	196
9.3.10. Fichero XML para el Programador	197
9.3.11. Fichero XML Consola WEB	198
9.4. PROPERTIES	198

9.4.1. DBXMLProperties.properties	199
9.4.2. ExtensionFileUploadProperties.properties	199
9.4.3. KeyEncryptProperties.properties	200
9.4.4. MatchFolderProperties.properties	200
9.4.5. RegisterUserProperties.properties	201
9.4.6. SignerProperties.properties	201
9.4.7. UserLibrariesProperties.properties	202
9.4.8. UsersProperties.properties	202
9.4.9. WEBConsoleLibrarieProperties.properties	202
9.4.10. WebSiteProperties.properties	203
9.5. HERRAMIENTAS DE DESARROLLO	203
9.5.1. DHTMLX	204
9.5.2. Ace Editor	206
9.5.3. NodeJS	208
9.5.4. Socket.IO	208
9.5.5. ShareJS	208
9.6. ESPECIFICACIÓN DE FUNCIONALIDAD DE LA APLICACIÓN	210
9.6.1. Registro y Activación de Programador	211
9.6.1.1. Registro Temporal de Usuario	211
9.6.1.2. Activar Programador	213
9.6.2. Encriptar Contraseña	215
9.6.3. Login y Gestión de Sesión	215
9.6.3.1. Login	216
9.6.3.2. Validación de Usuarios Internos	219
9.6.3.3. Abrir Sesión	221
9.6.3.4. Validar Sesión	223
9.6.4. Restablecer y Cambiar Contraseña	225
9.6.4.1. Restablecer Contraseña	225
9.6.4.2. Cambiar Contraseña	227
9.6.5. Gestionar Usuarios	228

9.6.5.1. Mostrar Programadores y Usuarios Temporales.....	229
9.6.5.2. Filtrar Programadores por Última Fecha de Ingreso	231
9.6.5.3. Eliminar Usuarios	233
9.6.6. Notificaciones en Tiempo Real.....	233
9.6.6.1. Proceso	234
9.6.6.2. Estructura del Mensaje.....	236
9.6.6.3. Tipo de Operaciones	237
9.6.6.4. Tipos de Mensajes	237
9.6.6.5. Abstracción del Funcionamiento de Socket.IO.....	239
9.6.7. Gestión de Proyectos	241
9.6.7.1. Crear Proyecto	242
9.6.7.2. Abrir Proyecto	245
9.6.7.3. Compartir Proyecto	249
9.6.7.4. Dejar de Compartir Proyecto	253
9.6.7.5. Dejar de Participar en un Proyecto	255
9.6.7.6. Renombrar Proyecto	258
9.6.7.7. Eliminar Proyecto	261
9.6.7.8. Descargar Proyecto	263
9.6.8. Gestión de Ficheros	266
9.6.8.1. Crear Fichero	266
9.6.8.2. Renombrar Fichero	269
9.6.8.3. Eliminar Fichero	272
9.6.8.4. Mover Fichero	274
9.6.9. Concurrencia	274
9.6.9.1. Pioneros en Transformación Operacional	275
9.6.9.2. Integración	276
9.6.9.3. Proceso	277
9.6.10. Ejecutar Proyecto	280
9.6.10.1 Compilar Proyecto.....	283
9.6.10.2. Construir Jar Ejecutable	285

9.7. DIAGRAMA DE DESPLIEGUE.....	287
CONCLUSIONES	
RECOMENDACIONES	
REFERENCIAS BIBLIOGRÁFICAS	

TABLA DE FIGURAS

Figura 1	Diagrama conceptual de todos los componentes de Java SE Plataform	12
Figura 2	Interface gráfica de DrJava	35
Figura 3	Interface gráfica de BlueJ	35
Figura 4	Interface gráfica de JCreator	36
Figura 5	Interface gráfica de IntelliJ IDEA	37
Figura 6	Interface gráfica de Eclipse	37
Figura 7	Interface gráfica de NetBeans	38
Figura 8	Interface gráfica de JBuilder	39
Figura 9	Interface gráfica de Anjuta	39
Figura 10	Interface gráfica de Geany	40
Figura 11	Interface gráfica de SlickEdit	41
Figura 12	Interface gráfica de GreenFoot	42
Figura 13	Subcaracterísticas del Editor de Márgenes	44
Figura 14	Resultado de Análisis gráfico de las IDE	56
Figura 15	Factor Común en interface gráfica de los IDE.	57
Figura 16	Uniformidad de la Plataforma Base de Java sobre los Sistemas Operativos	76
Figura 17	Arquitectura Interna de la JVM	77
Figura 18	Secuencia de Compilación y Ejecución	78
Figura 19	Funcionamiento del Compilador de Java	80
Figura 20	Estructura General de un Archivo	81
Figura 21	Definición de funciones en CoffeeScript vs JavaScript	111
Figura 22	Composición de estructuras JSON	112
Figura 23	Resultado de la causalidad	114

Figura 24	Representación de la intención del usuario	115
Figura 25	Concurrencia parcial entre operaciones	115
Figura 26	Logo oficial NodoJS	116
Figura 27	Arquitectura de NodeJS	116
Figura 28	Arquitectura Base de ShareJS	118
Figura 29	Logo de JQuery	119
Figura 30	Pruebas de Unidad de Iteración	131
Figura 31	Resultado de Prueba de Unidad Iteración 2	134
Figura 32	Resultado de Prueba de Iteración 3	137
Figura 33	Resultado de Prueba de Unidad H2	140
Figura 34	Resultado de prueba de Unidad H3	143
Figura 35	Composición gráfica de la IDE	147
Figura 36	Resultado del Entorno Gráfico	147
Figura 37	Resultado Prueba de Unidad H4	150
Figura 38	Resultado de Prueba de Unidad H5	152
Figura 39	Resultado de Prueba de Unidad para H18	155
Figura 40	Resultado de Prueba de Unidad para H19	157
Figura 41	Resultado de prueba de Unidad para H6	158
Figura 42	Resultado de Prueba de Unidad para H8	160
Figura 43	Resultado de Prueba de Unidad para H9	162
Figura 44	Vista Lienzo y Paleta de Componentes	168
Figura 45	Vista de Código Fuente Generado del Lienzo	169
Figura 46	Vista Agregar Componentes al Lienzo	169
Figura 47	Vista de Propiedades de un Componente	170
Figura 48	Vista de Redimensionar Lienzo	170
Figura 49	Resultado de Prueba de Unidad para H11	172

Figura 50	XML Usuarios En Espera	187
Figura 51	Archivo XML de Usuarios Activos	188
Figura 52	Estructura de Carpeta de un Programador en el Servidor	190
Figura 53	Estructura Interna del Fichero config.xml	192
Figura 54	Estructura Interna del Fichero libs.xml	193
Figura 55	Estructura Interna del Fichero ohterFiles.xml	194
Figura 56	Estructura Interna del Fichero packages.xml	195
Figura 57	Estructura Interna del Fichero src.xml	195
Figura 58	Estructura Interna del Fichero projects.xml	196
Figura 59	Estructura Interna del Fichero user.xml	197
Figura 60	Fichero XML Indexador de Clases de Consola WEB	198
Figura 61	Property para la Persistencia de Usuarios	199
Figura 62	Property de Configuración de Extensión de Ficheros a Cargar	199
Figura 63	Property de Llaves para Encriptar y Desencriptar	200
Figura 64	Fichero Property para el Archivo de Emparejamiento	201
Figura 65	Fichero Property para la Activación de Usuarios Temporales	201
Figura 66	Fichero Property para el Firmador de Applets	202
Figura 67	Fichero Property de Librerías por Defecto	202
Figura 68	Fichero Property del Directorio de Usuarios en el Servidor	202
Figura 69	Fichero Property Consola WEB	203
Figura 70	Fichero Property de la Aplicación	203
Figura 71	Estructura XML del Mensaje de Notificaciones	236

TABLA DE GRÁFICOS

Gráfico 1	Resultado de la pregunta 1	63
Gráfico 2	Resultados de la pregunta 2	64
Gráfico 3	Resultados de la pregunta 3	65
Gráfico 4	Resultado de la pregunta 4	67
Gráfico 5	Resultados de la pregunta 5	69
Gráfico 6	Resultados de la pregunta 6	71
Gráfico 7	Resultados de la pregunta 7	72
Gráfico 8	Resultados de la pregunta 8	73

ÍNDICE DE TABLAS

Tabla 1	Matriz de Caracterización de Funciones Básicas de las IDE'S seleccionadas	50
Tabla 2	Matriz de Caracterización de Funciones Avanzadas de las IDE'S seleccionadas	51
Tabla 3	Porcentajes de Cumplimiento de Características Básicas	53
Tabla 4	Porcentajes de Cumplimiento de Características Avanzadas	53
Tabla 5	Ilustración de los niveles que componen la API de Java	83
Tabla 6	Descripción del Nivel de Kit de Herramientas de Interfaces de Usuario	84
Tabla 7	Descripción del nivel Librerías de Integración	85
Tabla 8	Descripción del Nivel Otras Librerías Base	86
Tabla 9	Descripción del Nivel Librerías Base de Utilidades	88
Tabla 10	Qué puede y qué no puede hacer un Applet no Firmado	93
Tabla 11	Resultado de las Características básicas y avanzadas de las IDE's	97
Tabla 12	Características Básicas determinadas para la IDE	98
Tabla 13	Especificación del Actor Visitante	107
Tabla 14	Especificación del Actor Visitante Interno	108
Tabla 15	Especificación del Actor Administrador del Sistema	108
Tabla 16	Especificación del Actor Programador	109
Tabla 17	Priorización y Estimación de Historias de Usuario	127
Tabla 18	Plan de Iteraciones	127
Tabla 19	Descripción de la Historia de Usuario H7	129
Tabla 20	Descripción de la historia de Usuario H10	131
Tabla 21	Descripción de la Historia de Usuario H1	135

Tabla 22	Descripción de la Historia de Usuario H2	138
Tabla 23	Descripción de la Historia de Usuario H3	141
Tabla 24	Descripción de la Historia de Usuario H12	145
Tabla 25	Descripción de la Historia de Usuario H4	148
Tabla 26	Descripción de la Historia de Usuario H5	151
Tabla 27	Descripción de la Historia de Usuario H18	153
Tabla 28	Descripción de la Historia de Usuario H19	156
Tabla 29	Descripción de la Historia de Usuario H6	157
Tabla 30	Descripción de la Historia de Usuario H8	159
Tabla 31	Descripción de la Historia de Usuario H9	161
Tabla 32	Descripción de la Historia de Usuario H14	163
Tabla 33	Lista de Chequeo de prueba de Unidad para H9	165
Tabla 34	Descripción de la Historia de Usuario H17	166
Tabla 35	Lista de Chequeo de Prueba de Unidad para H9	168
Tabla 36	Descripción de la Historia de Usuario H11	171
Tabla 37	Descripción de la Historia de Usuario H13	173
Tabla 38	Lista de Chequeo de la Prueba de Unidad para H13	175
Tabla 39	Descripción de la Historia de Usuario H15	176
Tabla 40	Lista de Chequeo de la Prueba de Unidad de H15	178
Tabla 41	Descripción de la Historia de Usuario H16	179
Tabla 42	Lista de Chequeo para H16	180
Tabla 43	Tipo de Operaciones en las Notificaciones	237
Tabla 44	Tipo de Mensajes en las Notificaciones	239

TABLA DE DIAGRAMAS

Diagrama 1	Actores del Sistema	107
Diagrama 2	Diagrama de Clases de la Iteración 1	130
Diagrama 3	Diagrama de Clases de la Iteración 2	133
Diagrama 4	Diagrama de Clases Iteración 3	136
Diagrama 5	Diagrama de Clases de la Iteración 4 - H2	139
Diagrama 6	Diagrama de Clases de la Iteración 4 - H3	142
Diagrama 7	Diagrama JavaScript Entorno Gráfico	146
Diagrama 8	Diagrama de Clases de Iteración 6 H4	149
Diagrama 9	Diagrama de Clases para H18	154
Diagrama 10	Flujo General de Edición Concurrente	164
Diagrama 11	Diagrama Javascript de Constructor de Interfaces Gráficas	167
Diagrama 12	Flujo Genérico de Notificación de Operaciones	177
Diagrama 13	Flujo Genérico del Chat para H16	180
Diagrama 14	Arquitectura de la IDE	181
Diagrama 15	Descripción del Proceso Registrar un Usuario Temporal	211
Diagrama 16	Descripción del Proceso Activar Programador	213
Diagrama 17	Descripción del Proceso de Inicio de Sesión	216
Diagrama 18	Descripción del Proceso de Validación de un Usuario Interno	220
Diagrama 19	Descripción del Proceso Abrir Sesión	221
Diagrama 20	Descripción del Proceso Validar Sesión	224
Diagrama 21	Descripción del Proceso Restablecer Contraseña	226
Diagrama 22	Descripción del Proceso Cambiar Contraseña	227

Diagrama	23	Descripción del Proceso Buscar Usuarios	229
Diagrama	24	Descripción del Proceso Consultar por Última Entrada	231
Diagrama	25	Descripción de Notificaciones En Tiempo Real	235
Diagrama	26	Abstracción de Funcionamiento de Socket.IO	240
Diagrama	27	Descripción del Proceso Crear Proyecto	242
Diagrama	28	Descripción del Proceso de Abrir un Proyecto	245
Diagrama	29	Descripción del Proceso Abrir Proyecto desde la IDE	248
Diagrama	30	Descripción del Proceso Compartir Proyecto	250
Diagrama	31	Descripción del Proceso Dejar de Compartir Proyecto	253
Diagrama	32	Descripción del Proceso Dejar de Participar en un Proyecto	256
Diagrama	33	Descripción del Proceso Renombrar Proyecto	259
Diagrama	34	Descripción del Proceso Eliminar un Proyecto	261
Diagrama	35	Descripción del Proceso Descargar un Proyecto	264
Diagrama	36	Descripción del Proceso Crear Fichero	267
Diagrama	37	Descripción del Proceso Renombrar Fichero	270
Diagrama	38	Descripción del Proceso Eliminar Fichero	272
Diagrama	39	Descripción del Proceso de Concurrencia	278
Diagrama	40	Descripción del Proceso Ejecutar Proyecto	281
Diagrama	41	Descripción del Proceso Compilar	284
Diagrama	42	Descripción del Proceso Generar Proyecto Ejecutable	286

INTRODUCCIÓN

Los Entornos de Desarrollo Integrados, también conocidos por sus siglas en inglés como IDE, son aquellas aplicaciones que les ofrecen a los desarrolladores independientemente de su proveedor, gran cantidad de herramientas, componentes y funciones que permiten agilizar el proceso de construcción de software. En el mercado existen tantos IDE como existen lenguajes de programación. De hecho, existen entornos que son multilenguaje, es decir, que soportan varios lenguajes para la creación de proyectos e incluso multiplataforma que permita desarrollar para ambientes desktop, web o dispositivos móviles.

En el contexto de desarrollo de software existen múltiples opciones de ambientes de desarrollo, algunos son licenciados, otros hacen parte de un paquete de herramientas que alguna empresa adquiere comprando alguna licencia de un proveedor de software y otros son libres y gratuitos; pero todos concuerdan en algunas funciones básicas, por ejemplo compilar y ejecutar.

Gran parte de los IDE's del mercado están pensados para usuarios avanzados, y para desarrollos profesionales. Esto implica que usuarios que incursionan en el campo de la programación, que cuentan con pocos conocimientos de programación, no tienen un buen rendimiento en estas herramientas, por lo cual para ellos es necesario ofrecerles una herramienta que se adapte a sus necesidades.

ECO - Entorno Colaborativo de Online - es una aplicación web pensada para aquellos programadores que no requieren de todas aquellas funciones avanzadas de los IDE's convencionales, pero que tampoco esperan un simple editor que compile y ejecute. Esta aplicación nace como herramienta de apoyo para el Plan de Estudios de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, que contribuya de alguna forma a mitigar problemas de mortalidad y deserción académica en materias orientadas a programación de computadores presentes. Es necesario mencionar que ECO no está pensada como aplicación educativa, es decir, en ningún momento de la ejecución de la aplicación, se presentarán juegos didácticos, ejercicios de desarrollo de lógica, y ninguna otra técnica de aprendizaje.

El lector podrá encontrar a través del contenido de este documento, las circunstancias que se presentaron para el proyecto naciera. Se fijan algunos objetivos con respecto a la puesta en marcha, se determina la población que proporcionará información que ayude a crear un producto que se adapte a sus necesidades y a lo que esperan. Se define el alcance y las limitaciones del

proyecto, se analizan algunas IDE's seleccionadas, se definen las características básicas y avanzadas más comunes, se aplica un instrumento de recopilación de información en una muestra de la población a la que está dirigida el proyecto y de allí, junto con el análisis a las IDE's se toman decisiones con respecto a lo que se debería o no tener en la aplicación.

Fue necesario realizar una pequeña investigación de las generalidades de Java, ya que este lenguaje sería usado en la implementación. Aquí se encontró la forma adecuada de llevar a cabo operaciones como compilar, ejecutar, construir archivos ejecutables, firmar archivos JAR, entre otras cosas.

En el documento se podrá encontrar dos capítulos muy importantes, en los cuales se define y se implementa la metodología de desarrollo, se describen los procesos desarrollados y por último se muestra la distribución de los nodos que interactúan en la ejecución de la aplicación.

ECO es una aplicación web donde un programador podrá registrarse, crear proyectos, compilarlos, ejecutarlo, eliminarlos, compartirlos con otros programadores para trabajar en modo colaborativo y simultáneamente. Cuando un grupo de programadores comparten un proyecto y trabajan sobre este en un momento dado, podrá ver las funciones que ECO les ofrece, sobre las notificaciones en tiempo real, que básicamente informa todo lo que sucede con el proyecto, y además podrán usar un función muy particular, que le permite a los programadores editar en tiempo real sobre la misma clase, y cada uno observar lo que en su momento están editando.

DATOS DE EJECUCIÓN

La aplicación se encuentra disponible en la siguiente URL:

<http://108.60.150.136:8087/IDEWeb/IDE/index.html>

El link de Descarga del Código Fuente está disponible en la siguiente URL:

<http://108.60.150.136:8087/descargas/>

ECO tiene por defecto instanciados los siguientes usuarios:

1. Administrador:

Usuario: "admin"
Contraseña:"xxxx"

2. Programadores:

Usuario: "cas@"
Contraseña:"xxxx"

Usuario:"wil@"
Contraseña:"xxxx"

PRESENTACIÓN DEL PROYECTO

1. PRESENTACIÓN GENERAL DEL ANTEPROYECTO

1.1 TÍTULO

IMPLEMENTACIÓN DE UN AMBIENTE INTEGRADO DE DESARROLLO ONLINE COMO HERRAMIENTA DE APOYO AL PROCESO EDUCATIVO EN LA MATERIA FUNDAMENTOS DE PROGRAMACIÓN, PERTENECIENTE AL PLAN DE ESTUDIOS DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER.

1.2 PLANTEAMIENTO DEL PROBLEMA

En las materias Fundamentos de Programación, Programación Orientada a Objetos y Estructura de datos, pertenecientes al Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, se ha percibido mediante observaciones indirectas, una alta tasa de mortalidad académica, además de algunos factores que probablemente están influyendo en esta. Dentro de dichos factores se pueden encontrar la falta de equipos de cómputo especializados para que los estudiantes realicen sus prácticas, la complejidad de la instalación y configuración de los IDE's (*Integrated Development Environment*, o por sus siglas en español *Entorno de Desarrollo Integrado*), además de las exigencias hardware y software presentadas y la incompatibilidad de algunas versiones de IDE's.

Estudios realizados por parte del grupo de desarrollo de software (GIDIS) de la Universidad Francisco de Paula Santander, formalizados posteriormente en un proyecto de grado¹ con el fin de determinar factores involucrados en la reprobación de materias relacionadas con fundamentos de programación en diferentes universidades de la ciudad de Cúcuta; se llegó a la conclusión, que los factores que más influían en dicha problemática eran: falta de dedicación por parte de los estudiantes, problemas de comprensión, dentro de los cuales se percibían la poca percepción y la abstracción, además de la falta de ambientes prácticos donde los estudiantes tuvieran la posibilidad de

¹ AMAYA TORRADO, Yegny Karina y HERRERA ANGARITA, Lady Torcoroma. Identificación de las causas que generan problemas en el aprendizaje de fundamentos de programación de computadores en las facultades de ingeniería de las universidades de la ciudad de Cúcuta. Trabajo de Grado Ingeniería de Sistemas. Cúcuta N.de S. Universidad Francisco de Paula Santander. Facultad de Ingeniería Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura TIS 371.334A489i.

llevar a cabo actividades para afianzar los conocimientos obtenidos en las aulas de clase.

A nivel del país, se pueden evidenciar estudios y proyectos realizados por diferentes universidades; como en los casos de la Universidad Tecnológica de Pereira, Universidad Nacional de Colombia, donde se presenta una situación similar en los índices de mortalidad académica en materias relacionadas con fundamentos de programación. Como respuesta a dicha problemática, evidenciada en los estudios, algunas universidades han tomado diversas medidas para tratar reducir las cifras develadas^{2,3}

En el programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, se observan los esfuerzos realizados para contribuir a disminuir la problemática, construyendo un plan piloto llamado “amigo académico” el cual pretende que estudiantes destacados en diversas áreas como programación, matemáticas discretas y cálculo, brinden asesoría académica a los estudiantes que inician los estudios en Ingeniería de Sistemas. Con dicho plan se pretende dar cierto tipo de seguimiento y acompañamiento a los estudiantes que inician su proceso de formación profesional.

Detectando los factores mencionados en el estudio realizado por el grupo GIDIS, se destaca que la falta de práctica por parte del estudiante, y la dificultad de realizar actividades de seguimiento, constituyen un problema que entorpece de alguna manera, el sólido proceso de formación diseñado por parte del Programa de Ingeniería de Sistemas de la UFPS. Con el fin de brindarle al estudiante mayores garantías y beneficios para llevar a cabo su proceso de formación académica, se ofrece una solución de accesibilidad de una herramienta vía Web que permite tener una IDE *online*.

De igual manera se ha observado que algunos estudiantes manifiestan tener cierto tipo de problemas relacionados con: falta de computador, dificultad en la instalación y configuración del software, incumplimiento de requerimientos de hardware para el correcto funcionamiento del software, entre otras. Así mismo la falta de soluciones a estos problemas de una manera en la que se

²VARGAS C., José Gilberto. BUSTOS RIOS Ligia Stella y MORENO LAVERDE, Ricardo. Propuesta para Aumentar el Nivel Académico, Minimizar la Deserción, Rezago y Repitencia Universitaria por Problemas de Bajo Rendimiento Académico en la Universidad Tecnológica de Pereira, en el Programa Ingeniería de sistemas y Computación. Scientia et Technica Año XI No 28 Octubre de 2005 UTP. ISSN 0122-1701. Disponible: <http://revistas.utp.edu.co/index.php/revistaciencia/article/view/6841/4029>

³Autoevaluación del programa de pregrado de Ingeniería de Sistemas de la Universidad Nacional de Colombia, Sede Bogotá. Facultad de Ingeniería. Departamento de Ingeniería de Sistemas e Industrial.2005. 63p. Disponible: <http://www.ing.unal.edu.co/viceacad/acreditacion/sistemas/SistemasCNAv1.pdf>

integren las nuevas tendencias y tecnologías de la información, tales como el *Cloud Computing*, en donde todas estas situaciones adversas presentadas por el estudiante, sean solucionadas en un modelo de servicios brindados por la Universidad como soporte al proceso de formación.

1.3 JUSTIFICACIÓN

El Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, ha identificado gracias a estudios realizados por estudiantes en sus proyectos de grado o en desarrollo de asignaturas de metodología y seminarios de investigación, ciertos aspectos causantes de deserción estudiantil y mortalidad académica, específicamente en cursos que involucran contenidos de programación de computadores¹. Algunos de estos resultados arrojaron que los estudiantes tienen deficiencias en ubicación espacial, dificultades para esquematizar y modelar, bajos niveles de raciocinio y carencia de comprensión verbal y lectora. Además de los factores mencionados, existen otros indicadores que afectan tanto al aprendizaje de los estudiantes como el desarrollo de las prácticas relacionadas con el contenido del curso, de los cuales se resalta la falta de equipos de cómputo con características y requerimientos de hardware adecuados para soportar herramientas IDE (Eclipse, NetBeans, JCreator, entre otros), las cuales son necesarias para garantizar el correcto funcionamiento de las mismas. Además, a causa de que los entornos de desarrollo más utilizados por los estudiantes son funcionalmente muy robustos, y están diseñados pensando en usuarios más experimentados en el desarrollo de software; ha generado cierto grado de complejidad en la usabilidad de las mismas, creando confusiones y bajos índices de productividad en los estudiantes, ya que no logran desenvolverse eficientemente con la herramienta.

Cabe mencionar que gracias a las ventajas que ofrece el desarrollo de aplicaciones web (WebApps), al alcance que tiene Internet en el mundo, a los grandes y significativos avances de estas tecnologías y a la migración de muchas aplicaciones de escritorio a la “nube”; es posible concebir propuestas de desarrollo de herramientas bajo esta arquitectura, que

¹ AMAYA TORRADO, Yegny Karina y HERRERA ANGARITA, Lady Torcoroma. Identificación de las causas que generan problemas en el aprendizaje de fundamentos de programación de computadores en las facultades de ingeniería de las universidades de la ciudad de Cúcuta. Trabajo de Grado Ingeniería de Sistemas. Cúcuta N.de S. Universidad Francisco de Paula Santander. Facultad de Ingeniería Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura TIS 371.334A489i.

aproveche cada uno de los beneficios que estas tienen, para generar soluciones óptimas a problemas presentes en nuestro entorno educativo.

Con lo anterior expuesto, surge la idea de construir una herramienta bajo arquitectura Web que permita a los estudiantes del Programa de Ingeniería de Sistemas, realizar prácticas de Programación Orientada a Objetos en lenguaje JAVA sobre una IDE desarrollada para ejecutarse en línea.

La herramienta le facilitará al estudiante el desarrollo de prácticas en lenguaje JAVA, de manera que pueda ser más productivo, mejorar o desarrollar destrezas, habilidades e ingenio para conseguir soluciones óptimas a los problemas propuestos, conforme avance el contenido programático de la materia. La herramienta Web, tendrá disponible un IDE online de fácil acceso y usabilidad, con requerimientos mínimos de hardware y software, incluyendo las funciones necesarias para garantizar una ejecución adecuada, permitiendo de esta forma, atacar aquellas circunstancias que se presentan respecto a la falta de equipos de cómputo adecuados y a las dificultades que encuentran los estudiantes cuando usan los IDE's más conocidos y utilizados.

Es muy importante, que los estudiantes del Programa de Ingeniería de Sistemas, cuenten con herramientas que les permitan reforzar, complementar los conocimientos adquiridos y desarrollar buenas prácticas, que junto al acompañamiento del docente a través de la supervisión, se logre conseguir un ambiente colaborativo y motivado de aprendizaje.

Con el desarrollo y despliegue de la herramienta Web, se beneficiarán los estudiantes que cursan asignaturas referentes a la Programación Orientada a Objetos en JAVA, los docentes que las imparten y el Programa de Ingeniería de Sistemas como tal; a causa de que aquellos equipos con características de hardware básicas, serán útiles para las actividades de práctica, permitiendo que mayor cantidad de alumnos puedan tener a su alcance un equipo de cómputo, que sumado al ambiente colaborativo, motivado, sencillo, y usable, podrían reforzar aquellos factores en los que pueden tener carencias. Un beneficio agregado es, que el producto que persigue el proyecto, ayudará a enriquecer el conjunto de aplicaciones del Programa de Ingeniería de Sistemas de la UFPS, lo cual contribuye a que los estudiantes tengan a su disposición una variedad de herramientas y productos software que apoyen su formación profesional.

1.4 OBJETIVOS

1.4.1 Objetivo General

Implementar un ambiente de desarrollo integrado online, como herramienta de apoyo al proceso educativo de los estudiantes en la materia Fundamentos de Programación, perteneciente al plan de estudios de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander.

1.4.2 Objetivos Específicos

- Identificar características básicas y avanzadas de los IDE's para programación en JAVA más utilizados por los estudiantes del Programa de Ingeniería de Sistemas de la UPFS.
- Definir las funciones básicas que la herramienta Web debe tener.
- Determinar las principales librerías de Java que el usuario puede incorporar a sus programas.
- Desarrollar los componentes pertenecientes a la aplicación Web.
- Elaborar las pruebas respectivas a la herramienta web para garantizar el correcto funcionamiento y el adecuado comportamiento en su ejecución.
- Desplegar la aplicación online para permitir el uso por parte de los estudiantes y docentes.

1.5 ALCANCE Y DELIMITACIONES

1.5.1 ALCANCE

El alcance del proyecto es obtener como producto una herramienta web que pueda ser utilizada por los estudiantes que cursan la asignatura de Fundamentos de Programación perteneciente al Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, brindándoles una IDE online en el cual pueden desarrollar sus actividades de programación.

Con el manejo de la herramienta Web, el estudiante podrá realizar los ejercicios propuestos, sin la necesidad de instalación de ningún tipo de

aplicación en sus equipos de cómputo, que pueda exigir considerables requerimientos del sistema, permitiéndole desarrollar programación en un ambiente online. La aplicación ofrecerá la funcionalidad básica de un IDE convencional, (digitación de código, compilación y ejecución del mismo) teniendo en cuenta las percepciones obtenidas de los estudiantes y docentes, respecto a la funciones que desean que la aplicación implemente.

El propósito que el proyecto busca con el desarrollo, implementación y despliegue de esta aplicación web, es establecer una herramienta que acompañe a través de la práctica, el fortalecimiento de los conceptos aprendidos, y sea convertida en una de las herramientas de trabajo de los estudiantes del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander (UFPS).

1.5.2 LIMITACIONES

Es importante mencionar que el propósito de herramienta Web no es pedagógico, ya que no está enfocada a enseñar fundamentos, conceptos y demás aspectos necesarios para instruir al estudiante en Programación Orientada a Objetos.

La herramienta Web está dirigida para los estudiantes de la asignatura de Fundamentos de Programación del Programa de Ingeniería de Sistemas de la UFPS, y es así, ya que es la etapa en que los estudiantes adquieren bases conceptuales y estas son complementadas solo con la práctica.

La herramienta Web para el desarrollo de prácticas de Programación Orientada a Objetos presentará una limitante respecto al diseño de Interfaces Gráficas de Usuario, es decir, la herramienta no contendrá un constructor de interfaces gráficas, que cuente con paletas de sus componentes, que puedan ser arrastrados, posicionados y alineados para diseñar frames y paneles.

2. MARCO TEÓRICO O REFERENCIAL

2.1 ANTECEDENTES EN LA SOLUCIÓN DEL PROBLEMA

2.1.1 “IDEWeb (Entorno de Desarrollo Integrado en Web)”

Proyecto presentado en la Universidad de Oviedo, en España, por Cesar Rodríguez Rodríguez y dirigido por Juan Ramón Pérez Pérez, donde se muestra el diseño de un IDE online con el fin de servir como soporte para los estudiantes de ingeniería técnica e informática.

“Se trata de un entorno de programación, que en vez de ser una aplicación de escritorio, como los tradicionales, es una aplicación Web. El usuario sólo necesita disponer de un navegador Web estándar para poder usar la aplicación, editar ficheros, compilarlos, obtener el fichero resultado, o en su caso una lista de los errores y advertencias obtenidos. El entorno emplea herramientas de análisis estático de código para realizar un examen del código fuente más exhaustivo que los entornos de programación tradicionales. Esto permite detectar problemas del código que, de otra forma, podrían quedar ocultos. El entorno también extiende el concepto de ayuda sobre los errores de los entornos tradicionales, ya que dispone de una base de conocimientos colaborativa para todos los errores y advertencias que se pueden producir, donde no sólo se recoge información general sobre el error, sino que todos los usuarios de la aplicación pueden añadir nuevos conocimientos a partir de los casos de error que detectan en su propio desarrollo y las soluciones que aportan para corregirlos, lo cual permite incrementar la experiencia de todo el grupo contribuyendo a la mejora de la calidad del código de los proyectos desarrollados”⁴.

2.1.2 JAVAWIDE

JavaWide es un proyecto que tiene como resultado un IDE llevado a la nube. Fue creado en Noviembre de 2007 como una extensión básica MediaWiki⁵. Desde entonces ha crecido como una aplicación robusta basado en la web,

⁴RODRÍGUEZ RODRÍGUEZ, César. IDEWeb (Entorno de Desarrollo Integrado en Web). Proyecto de grado. Universidad de Oviedo. Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo. Disponible:

http://www.di.uniovi.es/~juanrp/docencia/pfc/pfc/IDEWeb_Entorno_de_Desarrollo_Integrado_en_Web.pdf

⁵ MediaWiki: es un paquete de software libre de código abierto, escrito en PHP, escrito originalmente para Wikipedia. Es ahora utilizado por muchos más proyectos wikis existentes.

que lleva a cabo un eficiente uso en entornos cliente-servidor. Además está diseñado con el fin de brindar flexibilidad y escalabilidad⁶.

Para acceder a las funcionalidades de JavaWide, es necesario entrar en la página oficial (<http://www.javawide.org>) y descargar un archivo JNLP⁷, al ejecutar este fichero con código Java orientado a la web, se puede hacer uso de JavaWide.

Es necesario que la máquina en donde se pretende ejecutar JavaWide, cumpla con cierto requerimientos, como: una conexión a Internet, y como mínimo un JRE (J2SE Runtime Environment) 1.5 o superior.

2.1.3 Diseño de un entorno de desarrollo integrado para una unidad controladora de procesos

El proyecto TeleLAB, es una iniciativa de parte de la Escuela de Ingeniería Electrónica de Costa Rica, cuyo propósito es la creación de un laboratorio en línea enfocado a temas relacionados al área del Control Automático. Este laboratorio en línea, brindará a los estudiantes la posibilidad de realizar distintos experimentos de manera remota, ya sea, por medio de un sistema de red de área local (LAN) o Internet⁸.

TeleLAB pretende brindar al estudiante una aplicación web, que le permita hacer uso de una interfaz gráfica, que facilite el diseño de simulaciones de diversos tipos. Internamente cumple con la función de generar un código a partir del diseño creado por el usuario, enviarlo al servidor, para que allí sea compilado y ejecutado, evitándole al estudiante la necesidad de instalar software especializado en su máquina, y pasando solo a ser necesario un navegador web, además de una conexión a internet.

2.1.4 laboratorios *online* del DIT

Los laboratorios online de DIT (Departamento de ingeniería de sistemas de la Universidad Politécnica de Madrid), son una herramienta software brindada a los estudiantes de la Universidad Politécnica de Madrid. Dicha

⁶ JENKINS, Jam. BRANNOCK, Evelyn y DEKHANE, Sonal. Innovation in an Online IDE. Tutorial Presentation. En: Journal of Computing Sciences in Colleges. Vol. 25 Issues 5, Mayo 2010. ACM Digital Library.

⁷ JNLP: (Por sus siglas en inglés Java Network Launching Protocol) El archivo JNLP es un XML donde se describe la forma de descargar y cargar una aplicación en particular.

⁸ CARAVACA MORA, Oscar Mauricio. Diseño de un Entorno de Desarrollo Integrado para una Unidad Controladora de Procesos. Proyecto de Grado Ingeniería en Electrónica. Instituto Tecnológico de Costa Rica. Escuela de Ingeniería en Electrónica. 132p. 2010. Disponible:
http://www.ie.itcr.ac.cr/einteriano/control/Laboratorio/Proyectos/2010_IDE/IDE_Informe_Final.pdf

herramienta consiste en permitir que el estudiante tenga un acceso remoto a los equipo de cómputo presentes físicamente en la Universidad, siendo necesario así solamente un punto de acceso a internet para poder realizar sus prácticas desde cualquier lugar.

2.1.5 IDE's COMERCIALES

Por último, vale la pena resaltar algunos IDE's online encontrados, los cuales brindan una interface gráfica que facilita el uso por parte del usuario, y además brindan funciones tales como: soporte para varios lenguajes de programación, almacenamiento de archivos, y lo más importante que solo es necesario una conexión a internet para acceder a todos los servicios brindados evitando así la instalación de software y hardware especializado en el equipo.

Es importante resaltar que es poca la información encontrada acerca de su diseño y funcionamiento interno, además que algunos de estos son pagos.

Dentro de los IDE's online a los que se hace referencia se encuentran: CodeRun (<http://www.coderun.com>), Compilr (<http://www.compilr.com>), Cloud9 IDE (<https://c9.io/>), codeanywhere (<https://codeanywhere.net/>), entre otros.

2.2 MARCO TEÓRICO

2.2.1 IDE

Los entornos de desarrollo integrado (IDE), son aplicaciones que permiten desarrollar más fácilmente software, ya que brindan al usuario, un conjunto de herramientas que permiten realizar un trabajo más ágil y rápido. Dentro de las herramientas que poseen los IDE's, se puede encontrar: un editor de código, un compilador, un depurador, pero sin embargo, algunos poseen funcionalidades agregadas tales como constructor de interfaces gráficas, control de versiones, herramientas de generación de código, entre otras.

2.2.2 JAVA

Java es un lenguaje creado por Sun Microsystem en 1995, muy utilizado en la actualidad en todo el mundo, sobre todo gracias a su independencia de la plataforma en la que se ejecuta⁸.

⁸ VILLALOBOS, Jorge A y CASALLAS Rubby. Fundamentos de Programación. Aprendizaje Activo Basado en Casos. Un enfoque moderno usando Java, UML, Objetos y Eclipse. 1^a Edición Pearson Prentice Hall. 384p. 2007. ISBN 9702608465.

Esto quiere decir que el código producido por el compilador Java puede transportarse a cualquier plataforma que tenga instalada la máquina virtual Java y ejecutarse. Según lo expuesto, Java incluye dos elementos: un compilador y un intérprete; el compilador produce un código de bytes que se almacena en un fichero para ser ejecutado por el intérprete Java denominado máquina virtual de Java. Los códigos bytes de Java son un conjunto de instrucciones correspondientes a un lenguaje máquina que no es específico de ningún procesador, sino de la máquina virtual de Java (JVM)⁹.

– **JVM (Java Virtual Machine)**

La máquina virtual de Java (JVM) es el elemento principal de Java, ya que es la responsable de brindar independencia del hardware y del sistema operativo a los programas desarrollados en Java.

La JVM no sabe nada del lenguaje de programación Java, sólo de un formato binario particular, el formato de archivo de clase. Un archivo de clase contiene las instrucciones de la máquina virtual Java (o bytecodes) y una tabla de símbolos, así como otra información complementaria¹⁰.

Por seguridad, la JVM impone un formato, y fuertes restricciones estructurales en código, en un archivo de clase.

Sin embargo, cualquier lenguaje que se pueda expresar en términos de un archivo de clase válido, puede hacer uso de la JVM. Atraídos por el uso general de la plataforma independiente de máquina, los desarrolladores de lenguajes están recurriendo a la JVM como medio para correr sus lenguajes.

La plataforma Java 2 Standard Edition (J2SE) proporciona dos implementaciones de la JVM:

- Java HotSpot Client VM, es una aplicación que suele ser utilizada para plataformas cliente.
- Java HotSpot Server, es una JVM diseñada para la velocidad máxima en la ejecución de programas.

⁹ CEBALLOS SIERRA, Francisco Javier. Java 2. Curso de programación. 3^a edición. Librería y Editorial Microinformática. 880p. 2005. ISBN 8478976866.

¹⁰ ORACLE. Java SE Documentation. Java SE 7 Documentation.

<http://download.oracle.com/javase/7/docs/>

- **JRE y JDK**

Sun Microsoft system proporciona dos principales productos en la familia de Java™ 2 Platform Standard Edition (J2SE™¹¹). (Ver Figura 1)

- *J2SE Runtime Environment (JRE)*

El JRE ofrece librerías, la JVM y otros componentes necesarios para correr applets y aplicaciones escritas en el lenguaje de programación de Java. Este entorno de ejecución puede ser redistribuido para que las aplicaciones sean independientes.

- *J2SE Development Kit (JDK)*

El JDK incluye el JRE y herramientas de desarrollo tales como compiladores, depuradores que son necesarios y útiles para el desarrollo de aplicaciones.

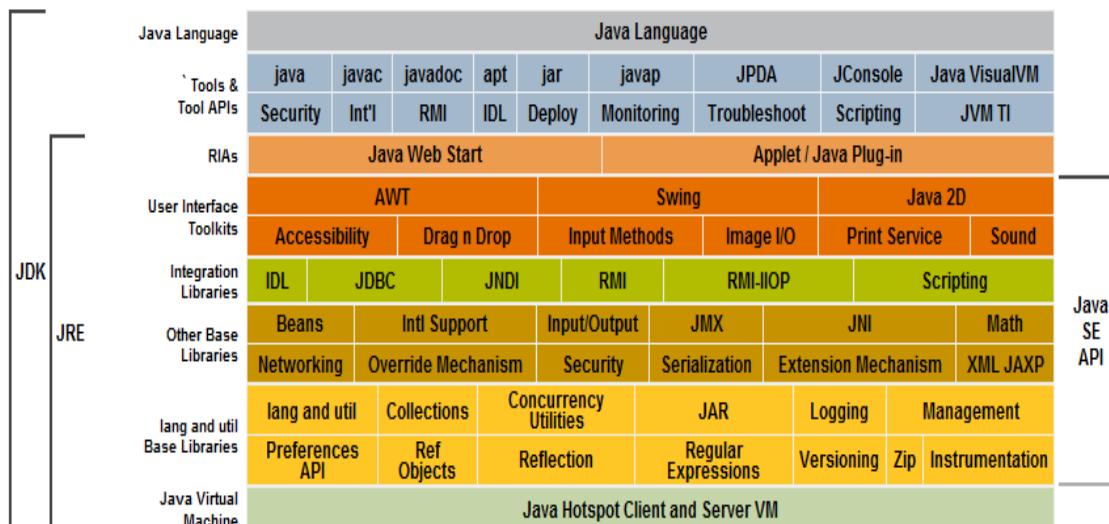


Figura 1. Diagrama conceptual de todos los componentes de Java SE Platform¹¹.

2.2.3 Software Web

- **NAVEGADOR WEB**

Un navegador web, es un programa altamente complejo, que cumple con la función de traducir y mostrar el código presente en una página web, que

previamente ha sido descargada desde un servidor. En otras palabras, la función más importante de un navegador web, es presentar las páginas web solicitadas por el usuario, y brindarle los controles necesarios para que pueda ser manejado¹¹.

Para transformar una página web, en un contenido gráfico, el navegador debe interpretar el código HTML que contiene la página, y desplegarlo en pantalla, con esta operación ejecuta fielmente el diseño y estilo original de la página web.

Un navegador, debe estar en la capacidad de manejar un dialogo HTTP, es decir crear correctamente las solicitudes HTTP, y saber interpretar las respuestas HTTP. Una simple conversación HTTP puede consistir en el acceso a un recurso referenciado en una página web u otro documento. En caso de que la URL solicitada sea incorrecta o el servidor presente errores, el navegador recibirá un mensaje de errores, y debe estar en la capacidad de informarlo al usuario.

Para manejar datos sobre sesiones del usuario, el navegador web se vale de ciertos recursos, como guardar archivos temporales en carpetas específicas, para que a la hora de volver a acceder a este, el tiempo de respuesta sea menor, también maneja un espacio de memoria dentro de la máquina del cliente, donde almacena algunos datos de vital importancia, como los correspondientes a los manejos de sesión.

Dentro de los navegadores más conocidos actualmente se pueden encontrar: Internet Explorer, Mozilla FireFox, Google Chromer, Opera, Safari, entre otros.

– SERVIDORES WEB

Un servidor web es una pieza fundamental del software en la web; el rol que desempeña este, es el de brindarle los recursos a un cliente web, tal como un navegador¹².

La primera responsabilidad de un servidor web, es interpretar las solicitudes HTTP y generar respuestas HTTP adecuadas. Una vez recibida una solicitud, la tarea es localizar el recurso solicitado.

Los servidores web deben poder manejar, cientos o miles de solicitudes concurrentes de diferentes clientes. Para cumplir con este requerimiento los

¹¹ GROVE, Ralp F. Web-Based Application Development. Editorial Jones & Bartlett Pub. 329P. ISBN 0763759406.

servidores no procesan las solicitudes secuencialmente, sino que lo hacen concurrentemente valiéndose de una técnica llamada *multitasking*. Los servidores web modernos, típicamente manejan el *multitasking*, manteniendo una cantidad de hilos activos o como su término en inglés, un *pool* de hilos.

– SERVIDORES DE BASE DE DATOS

La función de un servidor de base de datos, es la almacenar toda la información usada en la aplicación, además de brindar una copia de seguridad y servicios de recuperación de información, con el fin de ayudar a la seguridad de los datos. El software que lleva a cabo todo este proceso es llamado sistema manejador de base de datos (DBMS)¹².

Los DBMS deben estar en la capacidad de soportar SQL, el cual es un lenguaje para expresar operaciones en bases de datos, tales como leer o actualizar un registro. SQL fue originalmente desarrollado por IBM y ahora es un estándar ANSI / ISO (*American National Standards Institute / International Organization for Standardization*). La conexión entre la aplicación y el servidor de datos, es hecha mediante un ODBC (*Open Database Connectivity*), o un JDBC(*Java Database Connectivity*) para programas Java.

2.2.4 Cloud Computing

El término *Cloud Computing* hace referencia a la integración de red e infraestructura, no es otra cosa más que una orientación de funciones que puedan ser accedidas mediante internet¹².

Cloud Computing está utilizando servidores web para brindar servicios a terceros, tal como almacenamiento, desarrollo, y ejecución de aplicaciones. Algunos expertos dicen ver al *Cloud Computing* como un servicio, una plataforma y hasta un sistema operativo.

Actualmente los modelos de servicios que ofrece *Cloud Computing* se pueden clasificar en 3 tipos¹³¹⁴:

¹² Art. Cloud Computing: Security Risk.LaQuata Sumter. Florida A&M University. Department of Computer and Information Sciences. En: Proceedings of the 48th Annual Southeast Regional Conference. ACM Digital Library.

¹³ Art. Dave Durkee. Why Cloud Computing Will Never Be Free. En: Magazine Communications of the ACM. Vol. 53 Issues 5. Mayo 2010. ACM DIGITAL LIBRARY.

¹⁴ ALCOCER, Alberto. Blog Oficial SocieTIC Sociedad y Tecnología. Categoría Cloud Computing. <http://www.societic.com/category/cloud-computing>

- *Cloud Software As a Service (SaaS)*. SaaS es aquella aplicación ofrecida por un fabricante de software o proveedor de servicios informáticos a través de internet, para su uso o utilización por varios clientes. El fabricante es el que en última instancia se ocupa del manteniendo de la privacidad de los datos y la personalización de la aplicación.

En este modelo de servicio, el usuario paga por el uso y por la infraestructura necesaria (almacenamiento, seguridad, alojamiento, etc.) para el correcto funcionamiento de la aplicación y, a excepción de unos pocos parámetros de configuración, se limita a utilizar la herramienta y sus funcionalidades.

- *Cloud Platform As a Service (PaaS)*. Este modelo de nube amplía las prestaciones del caso anterior, de forma que el consumidor o usuario de esa nube, puede desplegar en ella aplicaciones desarrolladas o adquiridas por él mismo, en pos de ampliar las funcionalidades de dicha nube. Todo esto, por supuesto, se deberá desarrollar en aquellos lenguajes de programación que sean aceptados por el proveedor de la nube.

En este modelo de nube, el usuario no podrá gestionar la infraestructura de la nube, pero tendrá acceso tanto sobre las aplicaciones desplegadas en ella como sobre la configuración de las diversas herramientas que utilice.

- *Cloud Infrastructure As a Service (IaaS)*. En el IaaS, se parte de la idea de la externalización de servidores para espacio en disco, base de datos etc., en lugar de tener un control completo de los mismos con el DATA CENTER dentro de la empresa, u optar por un centro de datos y sólo administrarlo. Mediante este modelo de despliegue en Cloud, lo que se tiene es una solución basada en la virtualización, en la que se paga por el nivel de consumo de los recursos: espacio en disco utilizado, tiempo de CPU, espacio en base de datos, transferencia de datos.

La ventaja más inmediata de elegir este tipo de soluciones es la de desplazar una serie de problemas al proveedor relacionados con la gestión de las máquinas y llegar a un ahorro de costes importante, ya que pagaremos solo por lo consumido en función del nivel servicio que nos ofrezca dicho proveedor.

Otro aspecto fundamental a tener en cuenta, es que las Infraestructura como servicio pueden permitir una escalabilidad

automática o semiautomática, de forma que podamos contratar más recursos según los vayamos necesitando.

2.2.5 eXtreme Programming XP

“La eXtreme Programming, es una disciplina de desarrollo de software basado en los valores de la simplicidad, la comunicación, la retroalimentación, el coraje y el respeto. Actúa llevando a todo el equipo unido en presencia de prácticas simples, con suficiente información para que el equipo pueda ver dónde está y para ajustarlas a situaciones particulares¹⁵.

Uno de los creadores de esta metodología fue Kent Beck. La Programación Extrema, por su significado en español, es el proceso ágil de desarrollo de software más destacado, y se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Kent Beck dice:

“Tendremos éxito cuando tengamos un estilo que celebre un conjunto consistente de valores que sirvan a las necesidades tanto humanas como comerciales: comunicación, simplicidad, retroalimentación y coraje”¹⁶.

Estos valores que menciona Kent Beck conducen a un conjunto de doce prácticas, que esencialmente forman las reglas de la Programación Extrema. A continuación se describen y se agrupan de la siguiente manera:¹⁷

La Comunicación

- **Cliente en el sitio:**
 - Considerado un miembro más del equipo del proyecto.
 - Describe la funcionalidad que el sistema tendrá a través de historias de usuario.
 - Responde a las dudas de los detalles de las historias de usuario.
 - Negocia y planea los tiempos de las liberaciones.

¹⁵ Ronald E. Jeffries. *XProgramming.com An Agile Software Development Resource. “What is Extreme Programming?”*. <http://xprogramming.com/what-is-extreme-programming/>

¹⁶ Kent Beck, *Extreme Programming Explained. Embrace Change. Chapter 7 Four Values*. Addison-Wesley Professional ©2004.

¹⁷ Lisa Crispin, Tip House. *Testing Extreme Programming. Chapter 1 An Overview: Overview of XP*. Addison-Wesley Professional

- Toma las decisiones que afectan los objetivos del negocio.
- Ayuda con las pruebas funcionales.
- **Programación en Pareja.**
 - Equipo de dos programadores trabajan juntos escribiendo toda la producción del código.
 - Disminuye la probabilidad de cometer errores.
 - Aumenta la productividad
- **Estándares de Codificación.**
 - Código escrito bajo los mismos estándares.
 - Mantiene la consistencia.
 - Control en la estructura del código.
 - Mejora la compresión del código.
 - Facilita la refactorización del código por parte del equipo.

Simplicidad

- **Metáfora.**
 - Define cómo funciona el sistema completo.
 - Define el alcance y el propósito del sistema.
 - Expresa una visión evolutiva del proyecto.
- **Diseño Simple:**
 - Hacer las cosas simples, siempre y cuando funcionen.
 - Se enfoca en proporcionar un sistema que cubra las necesidades inmediatas del cliente ni más ni menos.
- **Refactorización.**
 - Permite mejorar el diseño del sistema a través de todo el proceso de desarrollo.
 - Continuamente se evalúa el diseño y recodifican lo que sea necesario.

Retroalimentación

- **Pruebas.**
 - Las pruebas de unidad son el corazón de XP.
 - Cada pieza del código tiene un conjunto automatizado de pruebas de unidad.
 - Los programadores escriben las pruebas de unidad antes que el código.

- Ninguna modificación o refactorización de código es completa hasta que el 100% de las pruebas de unidad corran satisfactoriamente.
- Las pruebas de aceptación validan grandes bloques de la funcionalidad del sistema.
- **Integración Continua.**
 - Adiciones y modificaciones del código son integradas en el sistema al menos diariamente.
 - Las pruebas de unidad deben correr un 100% satisfactoriamente, antes y después de cada integración.
- **Pequeños Lanzamientos.**
 - El conjunto más pequeño de funciones útiles es identificado para el primer lanzamiento
 - Los lanzamientos se llevan a cabo tan pronto como sea posible y a menudo con nuevas características cada vez.

Coraje

- **Planeando el juego.**
 - Los clientes y los desarrolladores cooperan en planear el orden y el tiempo para los lanzamientos.
 - El cliente provee las especificaciones del sistema.
 - Los desarrolladores estiman el tiempo y el esfuerzo.
 - Los clientes toman la decisión sobre qué historias implementar y en qué orden.
 - Los desarrolladores y los clientes conjuntamente negocian la secuencia de las iteraciones.
- **Propiedad colectiva del código.**
 - Una persona no es dueña de un módulo.
 - Todos los programadores deben ser capaces de trabajar en cualquier parte del sistema y en cualquier momento.
- **Ritmo sostenible.**
 - Trabajar a un ritmo que puede ser sostenido por un largo plazo sin sacrificar nuestra salud física y mental, la familia o la productividad.
 - Semanas consecutivas de horas extras se consideran un signo de que algo anda muy mal.

CICLO DE VIDA IDEAL DE UN PROYECTO XP

Kent Beck¹⁸ propone un ciclo de vida ideal para un proyecto desarrollado en XP. El autor plantea seis fases, y su objetivo es dar una idea de flujo general, en la que se puedan apoyar los equipos que adopten esta metodología. A continuación se describen:

- Exploración

Los clientes plasman la funcionalidad del sistema a través de historias de usuario, mientras que el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que podrían verse implicadas en el proyecto, las cuales se prueban y también exploran posibilidades de arquitectura. Esta fase termina cuando el cliente está seguro que ya hay suficiente material en las tarjetas de historias para realizar un buen primer lanzamiento, y los programadores se aseguran de que ya no se pueda hacer una mejor estimación. Esto puede llevarles pocas semanas o pocos meses.

- Planificación

El objetivo de esta fase es que los clientes y los programadores priorizan las historias de usuario de acuerdo al valor que proporcione al negocio, definan el alcance del lanzamiento, y lleguen a un acuerdo en la fecha en que los conjuntos de historias de usuario más pequeñas y las de más valor se puedan realizar. El plan para el primer lanzamiento podría tomar entre dos y seis meses, menos tiempo es probable que no solucionen los problemas significativos del negocio.

- Iteraciones para el primer Lanzamiento

Cada iteración está comprendida entre una y cuatro semanas, y produce una serie de casos de prueba funcionales, para cada historia de usuario programada en dicha iteración. La primera iteración debe contener las historias que le permitan crear la estructura del proyecto, es decir la arquitectura del sistema. El cliente decide qué historias y en qué orden se deben implementar, mientras el programador estima el esfuerzo para cada una, definiendo un cronograma. En esta fase se deben analizar cómo se está trabajando, con el fin de detectar cualquier signo de

¹⁸ Kent Beck. *eXtreme Programming eXplained: Embrace Change*. 2nd Edition. Addison-Wesley Professional. Chapter 21. Lifecycle of an Ideal XP Project.

desviación del plan, y si la hay es necesario ajustar la manera como se viene trabajando.

- Producción

En esta fase en lugar de iteraciones de tres semanas, se tendrán iteraciones de una semana. Se pueden tener reuniones diarias para que todo el equipo conozca lo que se está trabajando. Requiere de pruebas adicionales y revisiones de rendimiento antes de que esté listo para la entrega final. Si se detecta un problema en una de las pruebas, el cliente deberá recordar la forma como consiguió el problema, reportar la falla para que sea atacada y cuando se entregue la siguiente liberación rectificar el fallo. En este caso los desarrolladores atacan el problema y crean un nuevo conjunto de pruebas de unidad, partiendo de la prueba de aceptación en la que el cliente detectó la falla. Después que se consiga la primera liberación productiva para uso del cliente, es decir, que cumpla las pruebas de unidad y de aceptación, el proyecto debe mantener el funcionamiento mientras se desarrollan las siguientes iteraciones.

- Mantenimiento

Es considerada la fase en donde el proyecto XP se encuentra en un estado normal. Requiere de tareas de soporte para el cliente y puede requerir de nuevo personal y cambios de estructura del equipo. Puede intentar las refacturaciones que temía hacer antes, probar nueva tecnología o migrar a versiones recientes, experimentar con nuevas ideas de arquitectura, siempre en busca de aportar más valor al negocio. Es probable que la velocidad de desarrollo se afecte por tal razón se debe ser prudente en las estimaciones. Con respecto al movimiento del personal, debe hacerse poco a poco, y darles poca carga mientras se familiarizan con lo que se está trabajando, y una vez demuestren que pueden ofrecer más, es ahí el momento de aumentar la carga. Esto es necesario ya que si no fuese así, interrumpirían haciendo preguntas, tratando de entender lo que se viene desarrollando.

- Muerte

Se debe satisfacer las necesidades del cliente, tanto funcionales, de rendimiento y confiabilidad, a tal punto de que ya no hayan más por incluir en el sistema, aquí se considera que el proyecto ha muerto. Ahora bien se realiza la documentación final y no se efectúan más cambios a la arquitectura. Otro caso de muerte del proyecto, es cuando no se cumplen las expectativas del cliente, o cuando ya no hay más presupuesto para mantenerlo.

2.3 MARCO CONCEPTUAL

2.3.2 Internet

La Internet es una abreviatura para inter-network; lo que traduce una conexión de redes. Las redes de computadores fueron desarrolladas en 1960 como un camino de investigación para la transmisión de datos y la exploración de otras formas de transmisión de información. Debido a la cantidad de protocolos de conexión de redes utilizados, la Internet se desarrollada con el fin de proveer una solución para que las diversas redes pudiesen comunicarse entre sí. La primera red en la Internet fue ARPAnet¹².

2.3.3 World Wide Web (WWW)

Es una idea concebida y llevada a cabo por Tim-berrner lee, y puesta en marcha en 1990. Su función principal fue la de servir en la transmisión de documentos de investigación en modo hipertexto¹².

2.3.4 Compilador

Un compilador es un macro programa que traduce cadenas de un lenguaje a otro distinto. El compilador toma una cadena del programa original y lo traduce, incorporándola al programa traducido. Las cadenas son las frases del lenguaje. El programa original recibe el nombre de programa fuente y el generado por la compilación, es decir el traducido, se denomina programa objeto. El compilador traduce, sentencia a sentencia, el programa fuente al programa objeto.

2.3.5 Lenguajes programación

Los lenguajes de programación, como su nombre lo indica, son lenguajes creados con su propia gramática, y reglas, para dar órdenes al computador.

Los lenguajes de programación pueden clasificarse en 3 tipos:

- Lenguaje de máquina.
- Lenguaje ensamblador.
- Lenguajes de alto nivel.

En los primeros días de la computación, a menudo los programadores usaban lenguaje de maquina o lenguaje ensamblador. El lenguaje de

maquina usa código binarios, o cadenas de 0s y 1s, para ejecutar instrucciones de la CPU y referenciar a las direcciones de memoria. Este método es demasiado exigente y requiere de mucho tiempo dedicado. También el código de maquina no es portable para otras arquitecturas de computadores. La popularidad del lenguaje de maquina fue debido en gran parte a que los programadores no tenían otras opciones.

El lenguaje ensamblador dio un paso por encima del lenguaje de máquina, usando nombres simbólicos para direcciones de memoria y nemotécnica para instrucciones de procesador, por ejemplo: BEQ (branch if equals), SW (Store), o LW (Load). Un programador de lenguaje de ensamblador convierte el código a lenguaje de maquina antes de ser ejecutado.

Los lenguajes de alto nivel, tal como Fortran, Pascal, Perl, C, C++, y Java trabajan con sentencias en inglés, y esto es convertido en lenguaje de máquina, haciendo mucho más fácil el uso para desarrolladores de software, y más portables entre diferentes CPU. Por esta razón los programadores han utilizado los lenguajes de alto nivel para más y más aplicaciones.

2.3.6 Programación Orientada a Objetos

La programación orientada a objetos es una técnica de estructuración. En ella los objetos son los principales elementos de construcción. Sin embargo, comprender simplemente lo que es un objeto o utilizar objetos en un programa no significa que se esté programando en una forma orientada a objetos. Lo que cuenta es la forma como los objetos se conectan entre sí.

El significado exacto de la programación orientada a objetos, dice que es aquella programación que se da por medio del envío de mensajes a objetos de tipo desconocido. Tales objetos se encontrarán en un arreglo o en una colección como es un escritorio. Todos los objetos de la colección comparten ciertas características¹⁹.

2.4. FUNDAMENTOS LEGALES

2.4.1 Acuerdos De Uso

El software, resultado de la ejecución del proyecto, se regirá bajo los marcos legales fijados por la licencia pública Creative Common (LPCC) que dicta las

¹⁹ VOSS Greg. Programación orientada a objetos. Una introducción. McGraw-Hill Companies Inc. 597p. 1194. ISBN 9701004930.

siguientes condiciones de uso, encontradas más detalladamente en el documento Atribución No Comercial Compartir Igual 2.5 (Colombia)²⁰:

- Se posee la libertad de compartir la obra, entendiendo compartir como la capacidad de copiar, distribuir, ejecutar y comunicar públicamente la obra.
- Además puede llevar a cabo obras derivadas de la original.
- Es obligatorio reconocer los créditos de la obra de manera especificada por el autor o el licenciatario, pero no de manera que sugiera que tiene el apoyo de este último, o que apoya el uso que hacen a su obra.
- No puede ser utilizada para uso comercial.
- Si se altera o transforma, o genera un derivado a partir de esta obra, solo puede ser distribuida bajo una licencia idéntica a la presente.

2.4.2 Contrato De Licencia De Código Binario. Sun Microsystems, Inc.

Licencia de uso de software concedidos por Sun Microsystem, para la utilización de Java SE, siempre y cuando sean aceptados los términos de licencia, expuestos en el documento SUN MICROSYSTEMS, INC. CONTRATO DE LICENCIA DE CÓDIGO BINARIO²¹.

“SUN MICROSYSTEMS, INC. (EN ADELANTE DENOMINADO “SUN”) LE CONCEDE LA LICENCIA DEL SOFTWARE DEFINIDO A CONTINUACIÓN ÚNICAMENTE CON LA CONDICIÓN DE QUE USTED ACEPTE TODOS LOS TÉRMINOS ESTIPULADOS EN EL PRESENTE CONTRATO DE LICENCIA DE CÓDIGO BINARIO Y TÉRMINOS DE LICENCIA ADICIONALES (EN CONJUNTO DENOMINADOS “CONTRATO”). POR FAVOR, LEA EL CONTRATO DETENIDAMENTE. EL USO DEL SOFTWARE SIGNIFICA QUE HA LEÍDO LAS CONDICIONES Y QUE LAS ACEPTE. SI ACEPTE ESTAS CONDICIONES EN REPRESENTACIÓN DE UNA COMPAÑÍA U OTRA ENTIDAD LEGAL, SIGNIFICA QUE ESTÁ EN POSESIÓN DE LA AUTORIDAD QUE LE PERMITE VINCULAR LA ENTIDAD LEGAL A ESTAS CONDICIONES. SI NO ESTÁ EN POSESIÓN DE Dicha AUTORIDAD O SI NO DESEA QUEDAR VINCULADO A ESTAS CONDICIONES, NO DEBE UTILIZAR EL SOFTWARE EN ESTA INSTALACIÓN NI EN NINGÚN OTRO SOPORTE EN EL QUE SE UBIQUE EL SOFTWARE”.

²⁰ Creative Commons. Código Legal. Atribución no comercial compartir igual 2.5 (Colombia).
<http://creativecommons.org/licenses/by-nc-sa/2.5/co/legalcode>

²¹ CONTRATO DE LICENCIA DE CÓDIGO BINARIO. Sun Microsystems, Inc. *Java 2 Platform Standard Edition Runtime Environment 5.0*. Disponible:
http://www.java.com/es/download/license_jre5.jsp

2.4.3 GNU General Public License

La Licencia Pública General de GNU (GNU GPL, por sus siglas en inglés) es una licencia libre y gratuita con derecho de copia para software y otros tipos de obras.

Las licencias para la mayoría del software y otras obras de índole práctica están diseñadas para privarle de la libertad para distribuir y modificar las obras. Por el contrario, la Licencia Pública General de GNU garantiza la libre distribución y modificación de todas las versiones de un programa, a fin de asegurarle dicha libertad a todos los usuarios. En la Fundación para el Software Libre utilizamos la Licencia Pública General de GNU para la mayoría de nuestro software; también se aplica a cualquier otra obra publicada de esta manera por sus autores. Usted también puede aplicarla a sus programas.

Cuando hablamos de software libre, nos referimos a la libertad, no al precio. Nuestras Licencias Públicas Generales están diseñadas para garantizarle a usted la libertad de distribuir copias de software libre (y cobrar por ellas, si así lo desea), obtener el código fuente, o tener la posibilidad de obtenerlo, modificar el software o utilizar partes del mismo en nuevos programas libres, y saber que puede hacer estas cosas²².

2.4.4 GNU AFFERO General Public License

La licencia pública general de Affero está basada en GPL de GNU, pero añade una cláusula adicional para permitir a los usuarios que interactúan con el programa licenciado a través de una red, recibir el código fuente de ese programa. Recomendamos que se considere utilizar la licencia AGPL de GNU para cualquier programa que vaya a utilizarse a través de una red. La última versión es la 3.²³.

2.4.5 Propiedad Intelectual

De acuerdo con lo estimulado en el artículo 156 del acuerdo 065 del 26 de 1996, correspondiente al estatuto estudiante de la Universidad Francisco de Paula Santander, el cual dicta de la siguiente manera:

²² GNU GENERAL PUBLIC LICENSE. <http://www.gnu.org/licenses/>

²³ GNU AFFERO GENERAL PUBLIC LICENSE . <http://www.gnu.org/licenses/>

“Los trabajos de grado son propiedad intelectual de la Universidad y su uso estará sujeto a las normas que para tal fin estén vigentes”²⁴.

2.5 ANTECEDENTES

En la Universidad Francisco de Paula de Santander existen algunos estudios realizados como proyectos de grados que otorgaron título de Ingeniero(a) de Sistemas a los estudiantes que los desarrollaron. Estos aportaron ciertos indicadores, factores, causas, entre otras cosas, que permitieron crear una perspectiva de cierta problemática que permite crear estrategias que logren aportar un grano de arena a buscarle solución. A continuación se referencian y se describen de manera general:

ESTUDIO SOBRE DESERCIÓN – RETENCIÓN DE LOS ESTUDIANTES DEL PLAN DE ESTUDIOS DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER Y PERFIL OCUPACIONAL DEL EGRESADO. Esta investigación desarrollada en el año 2001 por los estudiantes Euline Esperanza Rosales Sales, Germán Darío Suárez García y Diana Carolina Velasco Sánchez, señala factores que precipitan la deserción estudiantil y propicia la permanencia del estudiante retenido en el Plan de Estudios de Ingeniería de Sistemas de la UPFS y además identifica el perfil ocupacional del egresado como requisito al proceso de acreditación de la carrera.²⁵

ESTUDIO SOBRE DESERCIÓN – RETENCIÓN DE LOS ESTUDIANTES DEL PLAN DE ESTUDIOS DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER Y PERFIL OCUPACIONAL DEL EGRESADO. Esta investigación desarrollada como actualización en el año 2007 por los estudiantes Marcia Karina Nieto Ropero y Sixto Tulio Barbosa Hernández, señala los factores o causas que hacen que se presenten los fenómenos de deserción y retención estudiantil en el Plan de Estudios de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander. Además identifica el perfil ocupacional del egresado, las impresiones de los empleadores que tienen a su cargo egresados de la

²⁴ Universidad Francisco de Paula Santander. Estatuto Estudiantil. Acuerdo 065 26 de agosto de 1996.

²⁵ ROSALES SALES, Euline Esperanza. VELASCO SÁNCHEZ, Diana Carolina y SUÁREZ GARCÍA, German Darío. Estudio sobre deserción - retención de los estudiantes del Plan de Estudios de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander y perfil ocupacional del egresado. Proyecto de Grado Ingeniería de Sistemas. Universidad Francisco de Paula Santander. Facultad de Ingeniería. Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura TIS 371.2913 R788E

U.F.P.S y se mencionan una serie de estrategias que pueden contribuir a disminuir estos fenómenos.²⁶

IDENTIFICACIÓN DE LAS CAUSAS QUE GENERAN PROBLEMAS EN EL APRENDIZAJE DE FUNDAMENTOS DE PROGRAMACIÓN DE COMPUTADORES EN LAS FACULTADES DE INGENIERÍA DE LAS UNIVERSIDADES DE LA CIUDAD DE CÚCUTA. Esta investigación fue desarrollada en el año 2003 por los estudiantes Yegny Karina Amaya Torrado y Lady Torcoroma Herrera Angarita. Este proyecto analiza las causas que generan problemas en el aprendizaje de Fundamentos de Programación de Computadores en las universidades de la ciudad de Cúcuta, tales como: Universidad Francisco de Paula Santander, Universidad Simón Bolívar, Fundación Universitaria San Martín y Corporación Universitaria Santander. Debido a la relevancia que dicho curso posee en la formación de profesionales cuyo desempeño se encuentra ligado a la solución de problemas, se hace necesario utilizar estrategias que permitan dar solución a los posibles factores que inciden en el rendimiento de los cursos (Grado de maduración intelectual, enseñanza tradicional empleada por el docente, etc.).¹

CREACIÓN DE UN SOFTWARE EDUCATIVO PARA EL DESARROLLO DE LAS HABILIDADES DEL RAZONAMIENTO ABSTRACTO. Este proyecto desarrollado en el año 2005 por los estudiantes Gina Paola Vergel Arévalo y Fernando Alberto Tarazona Claro, menciona que en la actualidad las aulas de clase de la Universidad Francisco de Paula Santander están requiriendo la introducción de nuevos métodos o herramientas de enseñanza que estimulen al estudiante a desarrollar habilidades básicas del pensamiento cognoscitivo, ya que a través de estudios realizados al interior del grupo de investigación y desarrollo de software en su línea de software educativo, se identificó un problema, el cual es la mortalidad académica de los estudiantes en las materias relacionadas con fundamentos de programación de computadores.²⁷

²⁶ AMAYA TORRADO, Yegny Karina y HERRERA ANGARITA, Lady Torcoroma. Identificación de las causas que generan problemas en el aprendizaje de fundamentos de programación de computadores en las facultades de ingeniería de las universidades de la ciudad de Cúcuta. Trabajo de Grado Ingeniería de Sistemas. Cúcuta N. de S. Universidad Francisco de Paula Santander. Facultad de Ingeniería Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura TIS 371.334A489i.

¹ AMAYA TORRADO, Yegny Karina y HERRERA ANGARITA, Lady Torcoroma. Identificación de las causas que generan problemas en el aprendizaje de fundamentos de programación de computadores en las facultades de ingeniería de las universidades de la ciudad de Cúcuta. Trabajo de Grado Ingeniería de Sistemas. Cúcuta N.de S. Universidad Francisco de Paula Santander. Facultad de Ingeniería Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura TIS 371.334A489i.

²⁷ VERGEL ARÉVALO, Gina Paola y TARAZONA CLARO, Fernando Alberto. Creación de un software educativo para el desarrollo de las habilidades del razonamiento abstracto. Proyecto de Grado Ingeniería de Sistemas. Universidad Francisco de Paula Santander. Facultad de Ingeniería. Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura: TIS 005.1 V495c

CREACIÓN DE UN PORTAL ACADÉMICO COMO HERRAMIENTA DIDÁCTICA DE APOYO A LOS PROCESOS EDUCATIVOS EN LOS PLANES DE ESTUDIO DE PREGRADO MODALIDAD PRESENCIAL DE LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER QUE INTEGRE NUEVAS TECNOLOGÍAS DE COMUNICACIÓN E INFORMACIÓN. Desarrollado en el año 2003 por Mabel Hernández Molina y Milton Jesús Vera Contreras, corresponde a un Desarrollo de Software sobre plataforma WEB con una Arquitectura en 5 Capas (Almacenamiento, Acceso a Datos, Servicios, Control y Visualización), fundamentada en el paradigma Orientado a Objetos, siguiendo el Proceso Unificado (UP) y el lenguaje UML e integrando Nuevas Tecnologías de Comunicación e Información (NTCI): Java, Servlets, JSP, XML, XSLT. El producto pretende servir como herramienta didáctica de apoyo a los procesos educativos en los Planes de Estudio de Pregrado Modalidad Presencial de la UPFS.²⁸

²⁸ VERA CONTRERAS, Milton de Jesús y Hernández Molina Mabel. Creación de un portal académico como herramienta didáctica de apoyo a los procesos educativos en los planes de estudio de pregrado modalidad presencial en la Universidad Francisco de Paula Santander que integre nuevas tecnologías de comunicación e información. Proyecto de Grado Ingeniería de Sistemas. Universidad Francisco de Paula Santander. Facultad de Ingeniería. Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura: TIS 005.1 H558c

3. DISEÑO METODOLÓGICO

Es claro que existen diferentes contextos (científicos, sociales, culturales, educativos, religiosos, entre otros) en el entorno que nos rodea, los cuales ofrecen misterios, incógnitas, cuestionamientos acerca de comportamientos anómalos, causas que conducen a que ciertos hechos inesperados, fenómenos o circunstancias no estudiadas. El campo de la investigación puede abarcar cualquiera de estos contextos, con el único propósito de encontrar soluciones, recomendaciones, explicaciones, orígenes o causas. Además, existen ciertos tipos de investigación que pueden aplicarse según sea el caso del contexto al que se esté enfrentando; por ejemplo, la investigación aplicada, que en términos generales está enfocada en la adquisición de conocimientos referentes algún tema o problema, para aplicarlos con el fin de generar soluciones a problemáticas presentadas en el contexto que se investiga.

Luego de esta breve mención sobre investigación, se determina que la población que interviene o al cual está dirigido este proyecto, son los estudiantes del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, específicamente aquellos que cursan primer semestre del proceso de formación profesional. Esto es así, ya que es la fase inicial que le permite a los estudiantes, conocer y entender el Paradigma de la Orientación a Objetos, enfrentando circunstancias y dificultades que pueden ser resueltas a través del interés, la atención, la curiosidad, la aclaración de dudas y muy importante, el refuerzo de conceptos a través de la práctica.

Es importante mencionar que, el desarrollo del proyecto está incluido dentro de:

- la Ingeniería de Software y Programación (áreas o líneas de investigación establecidas por el Programa de Ingeniería de Sistemas de la UPFS).

Se considera así, ya que el producto que persigue el proyecto es una aplicación web que formará parte de las herramientas de trabajo de los estudiantes, y por lo tanto, requiere la aplicación de la Ingeniería de Software.

3.1 TIPO DE INVESTIGACIÓN

El desarrollo del proyecto el cual se basa en la creación de una IDE Web para llevar a cabo prácticas actividades y ejercicios de Programación Orientada a Objetos, es catalogado como una investigación aplicada, ya que a raíz de una problemática que aqueja al Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, este busca aportar una alternativa de solución a través de la adquisición de conocimientos, producto de la investigación que puedan ser aplicados en el desarrollo de esta aplicación.

3.2 FUENTES DE INFORMACIÓN

3.2.1 Fuentes de Información Primaria

Claramente la información obtenida a través de este tipo de fuentes brinda bases fundamentadas, que permitirá que el desarrollo del proyecto sea fiable y argumentado. Las fuentes primarias que intervienen en el desarrollo del proyecto, y teniendo que en cuenta que no es un proceso netamente investigativo son:

- Los estudiantes de primer semestre del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, los cuales son los protagonistas de la problemática que se busca atacar con el desarrollo del proyecto.
- Los docentes adscritos al Departamento de Sistemas e Informática que imparten cursos en el Programa de Ingeniería de Sistemas, referentes a fundamentos de Programación Orientada a Objetos.

Con la selección de las fuentes primarias mencionadas anteriormente, se pretende obtener información que brinde un perspectiva de la problemática, desde el punto de vista de los actores directamente implicados, y de este modo enfocar el desarrollo del proyecto a la consecución de logros que en lo posible cubran las necesidades manifestadas por estos.

3.2.2 Fuentes de Información Secundaria

Las fuentes secundarias que intervienen en el desarrollo del proyecto son:

- Libros concernientes a: Lenguajes de Programación, Ingeniería de Software, Ingeniería de Software en arquitecturas Web y Lenguaje Unificado de Modelado.
- Artículos de revistas indexadas y especializadas en el área de desarrollo de software y desarrollo de aplicaciones web.
- Páginas oficiales de los organizaciones y/o empresas creadoras de los distintos IDE'S.
- Aplicaciones Web comerciales que tienen disponibles IDE'S para programar en diferentes lenguajes de programación.

3.3 RECOLECCIÓN DE LA INFORMACIÓN

Un aspecto muy importante dentro del desarrollo de un proyecto, sean cuales sean los objetivos que persiguen, es obtener la información confiable y veraz que fundamenta el desarrollo del mismo. Por lo tanto, es necesario aclarar que el proyecto no está interesado en obtener información que brinde respuestas a ciertas hipótesis acerca de algún fenómeno social y/o académico.

El desarrollo de una aplicación web requiere de conocimientos claros, metodologías, análisis profundo, técnicas, trabajo en equipo, buenas prácticas, entre otras que permitan conseguir un producto software de calidad.

Con el fin de alcanzar los objetivos del proyecto y obtener la información adecuada, es necesario realizar una revisión literaria, búsquedas avanzadas en Internet, observaciones a proyectos existentes que tengan ciertas semejanzas y opiniones profesionales. El procedimiento para la recolección de la información básicamente consistirá en analizar el producto de la revisión literaria (libros, documentos, artículos, documentación, entre otras) para identificar cuáles son los aspectos relevantes para el proyecto.

Es importante mencionar que debido a que la aplicación Web está pensada para ser una herramienta disponible para los estudiantes de materias relacionadas con fundamentos de programación y dirigido por sus respectivos docentes, es necesario interactuar con estos, para extraer información vital acerca del funcionamiento que debería tener dicha herramienta.

3.4 ANÁLISIS DE INFORMACIÓN

El proceso de análisis de información consiste en escudriñar la documentación, con el propósito de identificar características, propiedades, conceptos, metodologías, alternativas de solución, que permita abstraer lo necesario y adecuado para implementar la aplicación.

Además, es necesario llevar a cabo un pequeño proceso de tabulación y análisis de cierta información que será suministrada por parte de los estudiantes directamente involucrados. El resultado de dichos análisis será de vital importancia para el desarrollo de los pasos posteriores dentro del proyecto.

ESTADO DEL ARTE DEL PROYECTO

4. CARACTERÍSTICAS Y GENERALIDADES DE LOS IDE'S WEB

4.1. ENTORNOS DE DESARROLLO INTEGRADOS PARA JAVA

Una *IDE* está conformada por un conjunto de herramientas de programación que son diseñadas con el fin de simplificar la tarea del programador²⁹.

Para entender mejor la evolución de los ambientes integrados de desarrollo, es necesario realizar un análisis a la propia historia de los lenguajes de programación, su evolución y la incorporación de herramientas en el desarrollo de software.

Los primeros programas fueron escritos usando lenguaje de máquina, instrucciones construidas dentro de un circuito eléctrico de un computador particular. Se escribía en código binario y los programadores debían recordar las combinaciones de las instrucciones. Debido a que escribir en lenguaje de máquina era muy tedioso, algunos programadores dedicaron parte de su tiempo a desarrollar herramientas que contribuyeran a hacer su trabajo más fácil; así fue creado el primer lenguaje artificial, y fueron llamados lenguajes ensambladores³⁰.

Con la creación de los lenguajes ensambladores, se dio paso a los lenguajes de alto nivel como COBOL y FORTAN. La introducción de los lenguajes de alto nivel brindó un vehículo para que los mismos programas corrieran en más de un computador, pero para que esto pudiera ser posible, era necesaria la presencia de un programa que pudiese traducirlo y ejecutarlo, denominado compilador²⁹. En el instante que el hardware presentó un gran avance, se da una repercusión en el desarrollo de software, ya que estaban construyendo equipos más potentes, por lo cual se tenía el camino libre para la creación de software más robustos, exigiendo tener una mayor efectividad a la hora de crear programas, dando como resultado un proceso constante, encaminado en la integración de herramientas que le den mayor efectividad al programador, teniendo hoy en día las IDE que integran y brindan a los programadores un completo *kit* de desarrollo, dentro del cual se encuentra una extensa variedad de herramientas presentes en algunos y carentes en otros, que entre otras están: el editor de texto, resaltador de sintaxis, compilador, depurador, analizador de rendimiento, diseñador de interfaces

²⁹ *The Computer Science And Engineering-Handbook*. Editor-In-Chief, Allen B. Tucker, Jr. A CRC Handbook Publishes in Cooperation with ACM, The Association for Computing.

³⁰ Nell B Dale, John Lewis. Computer Science Illuminated. 2th Edition. Editorial Jones & Bartlett Learning, 2004 ISBN 0763707996, 9780763707996, 699 págs.

gráficas, generador de código, control de versiones, consola de diálogos de errores y sugerencias, capacidad para ejecución del programa³².

La característica principal de los *IDEs* es que además de brindar las herramientas mencionadas anteriormente, permiten una gran integración y cohesión entre ellas.

Así por ejemplo en Java se cuenta con una gran variedad de *IDEs* que permiten llevar a cabo el desarrollo de software mediante el uso de este lenguaje de programación. Durante la búsqueda y exploración se encontraron diversos *IDEs*, algunos con más funcionalidades que otros, además tanto privativos como gratuitos. Al final fueron escogidos once (11) *IDEs*, tomando en cuenta diversos factores como lo fueron la popularidad, facilidad de descarga e instalación, orientación hacia la enseñanza, entre otros.

A continuación se presentan los *IDEs* objeto de estudio:

– **DrJava**

Desarrollado por: Grupo JavaPLT de *Rice University*³¹.

Es un ambiente de desarrollo ligero, para escribir programas en Java. Está diseñado principalmente para estudiantes, proporcionando una interfaz intuitiva y la capacidad de evaluar código interactivamente. También ofrece un conjunto de características para usuarios más avanzados. Está disponible gratuitamente, bajo la licencia de “[BSD License](#)” y actualmente se encuentra en desarrollo activo por parte del grupo JavaPLT de *Rice University*³². Ver figura 2.

– **BlueJ**

Desarrollado por: University of Kent in Canterbury, UK³³.

Es un entorno de desarrollo amigable, creado como parte de un proyecto de investigación acerca de la enseñanza de la programación orientada a objetos.

³¹ Rice University JavaPLT. Sitio web oficial del grupo JavaPLT de Rice University, en donde se encuentra disponible información acerca de proyectos e investigaciones llevadas a cabo por dicho grupo. <http://www.cs.rice.edu/~javaplt/>

³² Sitio web oficial de DrJava. <http://www.drjava.org/>

³³ School of Computing. Teaching, learning, research and innovation. Sitio web de la escuela de computación de University of Kent, y donde se encuentra almacenada la información de proyectos e investigaciones informáticas llevadas a cabo en la universidad. <http://www.cs.kent.ac.uk/>

Actualmente está siendo desarrollado y mantenido por un grupo conjunto de *La Trobe University, Melbourne, Australia* y la *University of Kent de Canterbury*, en Reino Unido, además es importante resaltar que el proyecto es apoyado por Sun Microsystem³⁴. Ver figura 3.

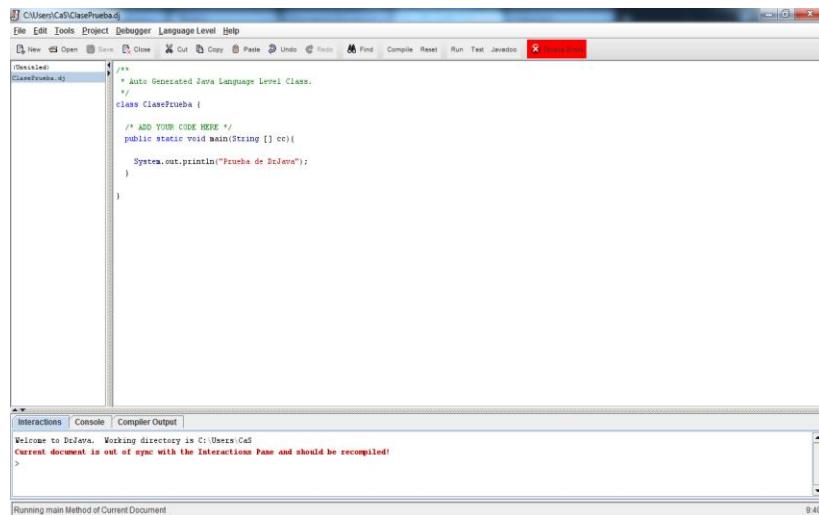


Figura 2. Interface gráfica de DrJava

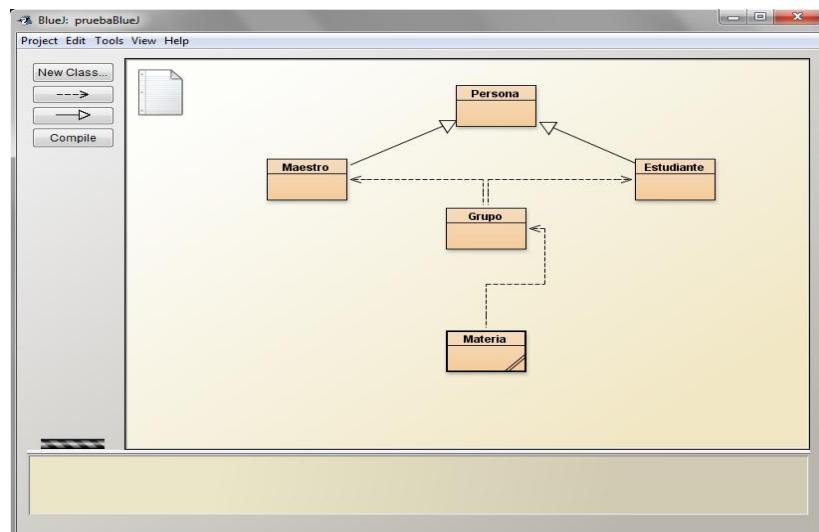


Figura 3. Interface gráfica de BlueJ

³⁴ BlueJ – The interactive Java environment. Entorno integrado de Java diseñado específicamente para la enseñanza introductoria. Sitio web oficial <http://www.bluej.org/>

– JCreator

Desarrollado por: Xinox Software.

Entorno de desarrollo que integra amplias funcionalidades, está desarrollado completamente en C++ y ofrece dos versiones, una *free software* y otra con costo de adquisición³⁵. Ver figura 4.

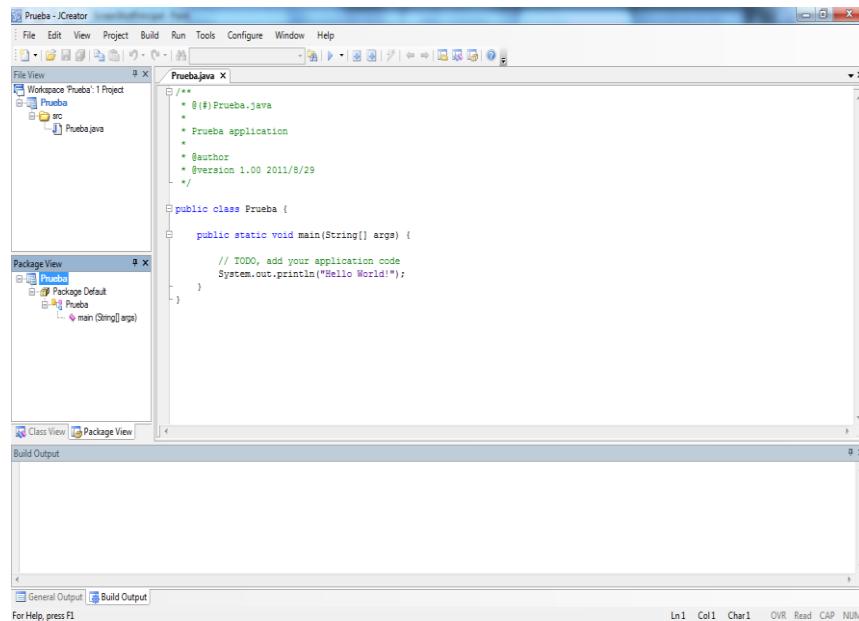


Figura 4. Interface gráfica de JCreator.

– IntelliJ IDEA

Desarrollado por: JetBrains³⁶.

Es un entorno de programación integrado, que brinda al usuario una gran variedad de herramientas, pensadas para favorecer la productividad de programador. Posee dos versiones, una paga y otra gratuita bajo el nombre de IntelliJ IDEA Community Edition³⁷. Ver figura 5.

³⁵ Sitio web oficial de JCreator. <http://www.jcreator.com/>

³⁶ Sitio web oficial de JetBrains. En este apartado se encuentra información acerca de la compañía. <http://www.jetbrains.com/company/index.html>

³⁷ Sitio web oficial de IntelliJ IDEA. <http://www.jetbrains.com>

– Eclipse

Desarrollado por: IBM³⁸.

IDE de código abierto. Fue desarrollado por IBM y escrito en Java. Proporciona una amplia gama de capacidades y características para hacer más fácil el trabajo del desarrollador³⁹. Ver figura 6.

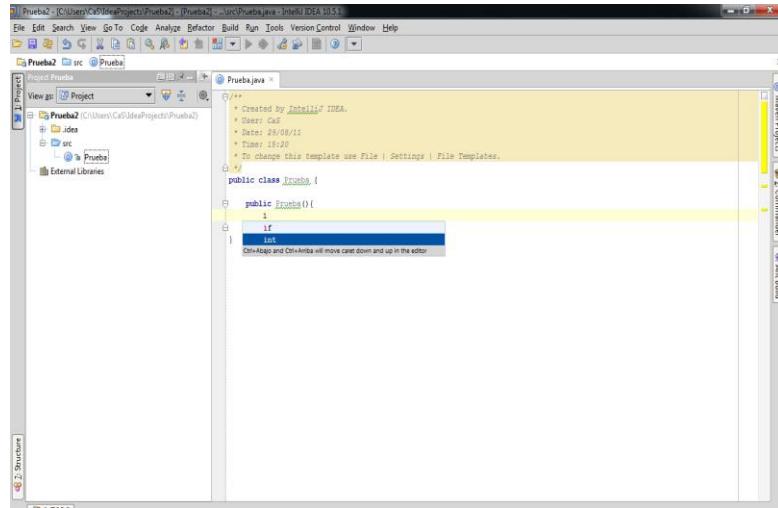


Figura 5. Interface gráfica de IntelliJ IDEA.

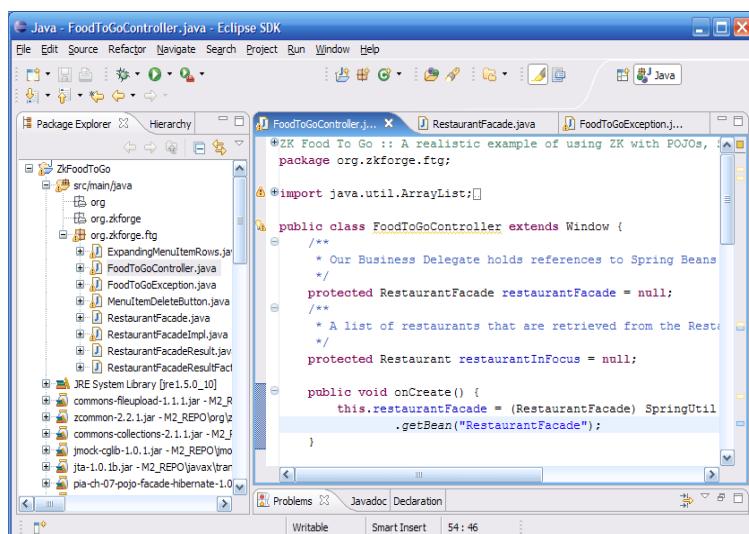


Figura 6. Interface gráfica de Eclipse

³⁸ Sitio web oficial de eclipse. En este apartado se encuentra información acerca de la historia del software entre otros. <http://www.eclipse.org/org/#about>

³⁹ Sitio web oficial de Eclipse. Eclipse.org es un portal donde se pueden encontrar diversos plugins y recursos importantes para facilitar el desarrollo de software. <http://www.eclipse.org/>

– NetBeans

Desarrollado por: Sun Microsystem⁴⁰.

El IDE NetBeans es un reconocido entorno de desarrollo integrado disponible para Windows, Mac, Linux, Solaris. El proyecto Netbeans está formado por un IDE de código abierto y una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones web, empresariales, de escritorio y móviles utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails, Groovy y Grails y C/C++⁴¹. Ver figura 7.

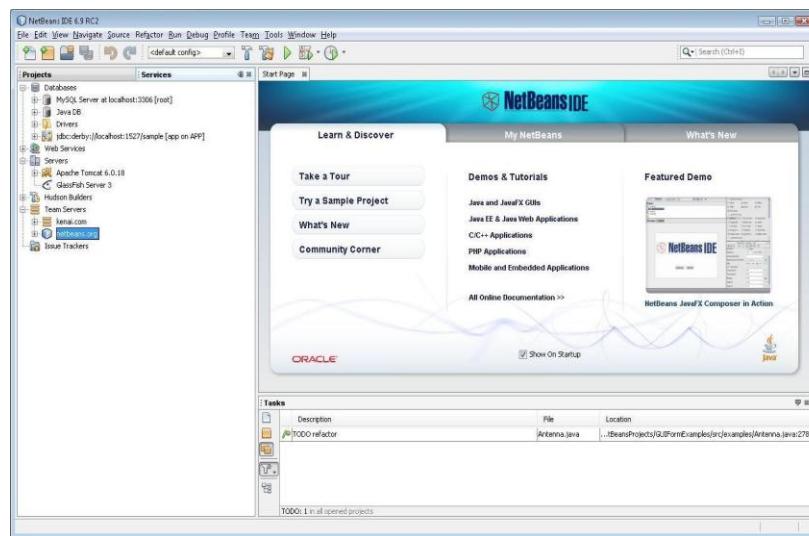


Figura 7. Interface gráfica de NetBeans.

– JBuilder

Desarrollado por: Originalmente por la empresa Borland, pero el propietario actual es la empresa embarcadero.

JBuilder es un completo IDE para el desarrollo de software, además brinda un gran conjunto de herramientas para hacer más efectivas las actividades a la hora de programar. Dentro de las herramientas, se encuentra generadores de código, gestores gráficos para el modelado de clases e interfaces entre otros⁴². Ver figura 8.

⁴⁰ Sitio web oficial de Netbeans. En este apartado se encuentra información acerca de la creación y evolución que ha tenido el software de NetBeans. <http://netbeans.org/about/history.html>

⁴¹ Sitio web oficial de NetBeans. netbeans.org es el portal de la comunidad de código abierto dedicado a construir un IDE de primera clase. <http://www.netbeans.org/>

⁴² Sitio oficial de JBuilder. <http://www.embarcadero.com/products/jbuilder>

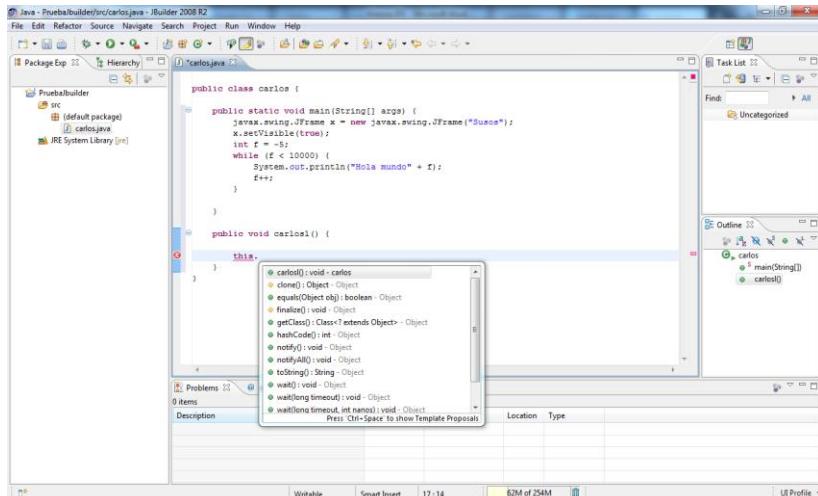


Figura 8. Interface gráfica de JBuilder.

– Anjuta

Desarrollado por: Naba Kumar (fundador)⁴³.

Es un software de desarrollo con un gran número de características de avanzada, tales como manejo de proyectos, interactividad con el depurador, editor de código, control de versiones y editor de GUI entre otras. Está enfocado en proveer una interface simple y útil, para potenciar la eficiencia en el desarrollo⁴⁴. Ver figura 9.

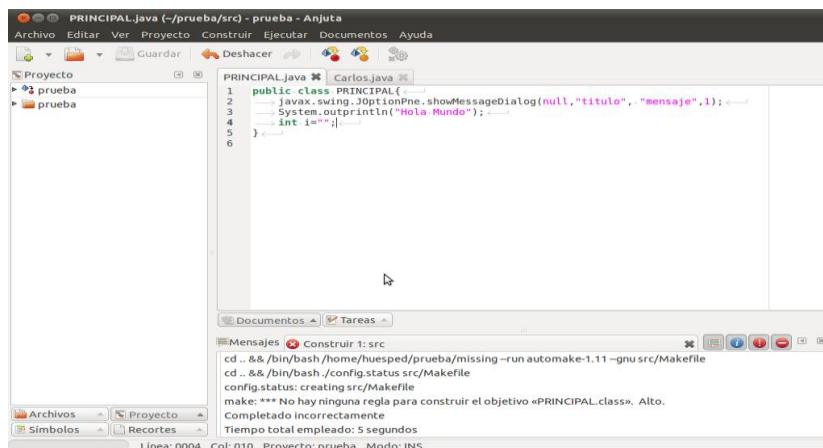


Figura 9. Interface gráfica de Anjuta.

⁴³ Sitio oficial de Anjuta. Sección dedicada a compartir información acerca del equipo de desarrollo que se encuentran vinculados al proyecto. <http://www.anjuta.org/team.html>

⁴⁴ Sitio oficial de Anjuta DevStudio. <http://www.anjuta.org/index.html>

– Geany

Desarrollado: Actualmente por Nick Treleaven, Enrico Tröger, Colomban Wendling, Frank Lanitz.⁴⁵

Geany es un ambiente de desarrollo integrado sencillo, está pensado para brindar facilidad y velocidad, teniendo pocas dependencias con otros paquetes para garantizar su efectividad. Otro objetivo es ser tan independientes como sea posible, requiriendo solo las librerías de ejecución GTK2⁴⁶. Ver figura 10.

Además corre en múltiples plataformas, tales como: Linux, FreeBSD, MacOS X, Solaris Express y Windows. Generalmente debería correr sobre todas las plataformas soportadas por librerías GTK (*GIMP Toolkit*), solo en Windows se pierden algunas características⁴⁵.

El código fuente está bajo términos de licencia *General Public Licence (GNU)*.

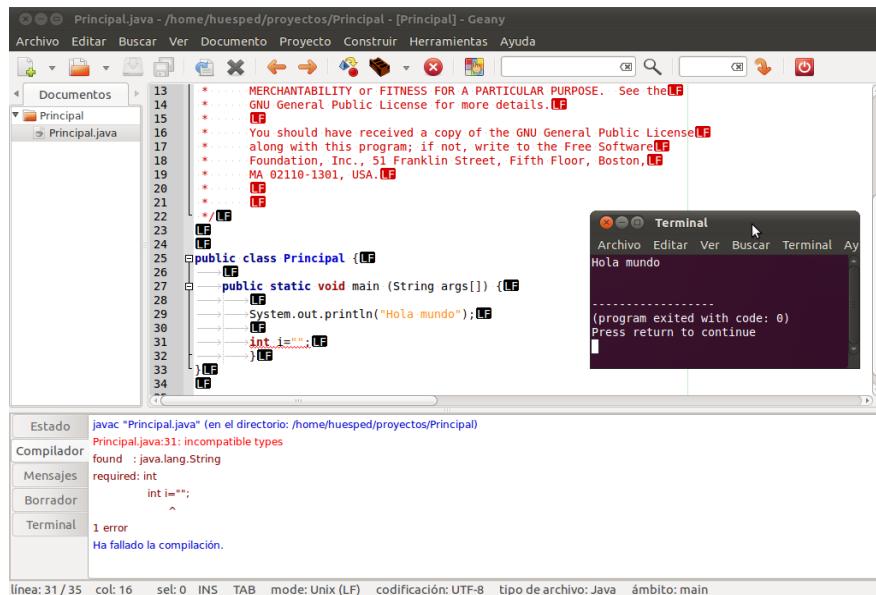


Figura 10. Interface gráfica de Geany

⁴⁵ Sitio oficial de Geany. Apartado dedicado a informar acerca de los autores del proyecto. <http://www.geany.org/Main/Authors>

⁴⁶ Sitio oficial de Geany. Apartado dedicado acerca del proyecto. <http://www.geany.org/Main/About>

– SlickEdit

Desarrollado por: SlickEdit Inc⁴⁷.

IDE de uso privativo, multiplataforma, multilenguaje, con potentes herramientas para el programador, permitiéndole crear, navegar, modificar y depurar código fácil y rápido. Puede ser utilizado como un editor de texto complementario o como una completa herramienta de desarrollo. Ver figura 11.

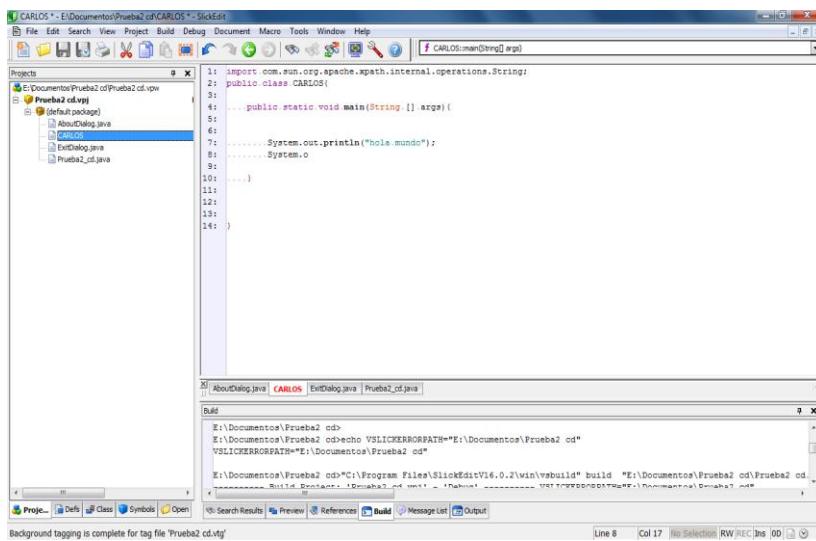


Figura 11. Interface gráfica de SlickEdit.

– GreenFoot

Desarrollado por: University of Kent at Canterbury (UK) y Deakin University, Melbourne⁴⁸.

Es proyecto llevado a cabo por University of Kent at Canterbury (UK) y Deakin University, Melbourne (Australia), además es apoyado por Sun Microsystems.

No es propiamente un IDE convencional, sino que como sus autores lo definen, es una especie de framework orientado a la enseñanza de

⁴⁷ Sitio web oficial de SlickEdit Inc. En este apartado se encuentra información acerca de la compañía y sus productos software. <http://www.slickedit.com/about>

⁴⁸ Sitio oficial de GreenFoot. En esta sección se especifican las entidades y personas involucradas con el proyecto. <http://www.greenfoot.org/about/contributors.html>

programación. En él se pueden diseñar programas, orientados a los juegos, donde el usuario puede observar desde un punto de vista diferente la programación orientada a objetos en java. Además de esto el software brinda al usuario un conjunto de herramientas presentes en los IDEs convencionales tales como editor de texto, resaltador de sintaxis, depurador, constructor de .jar, entre otros⁴⁹. Ver figura 12.

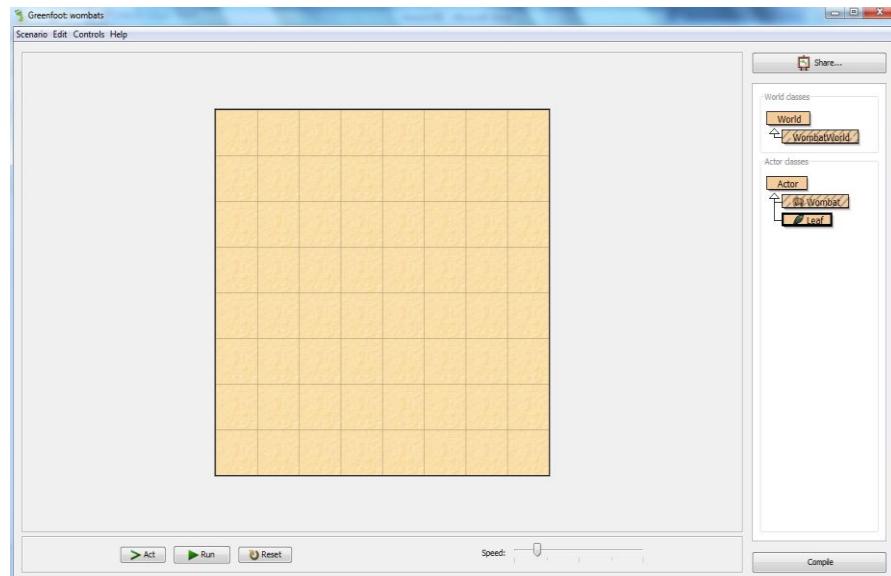


Figura 12. Interface gráfica de GreenFoot.

4.2. EVALUACIÓN DE LAS FUNCIONES DE LOS IDE'S

Como se observó anteriormente, las IDEs están conformadas por un conjunto de funciones que facilitan el trabajo del programador. Por lo tanto, para la realización de la evaluación de las respectivas funcionalidades se tendrán en cuenta las IDEs descritas anteriormente a excepción de GreenFoot, debido a que tiene características especiales orientadas al aprendizaje de Java mediante la creación de juegos y no es el objeto de estudio de este trabajo. A continuación se presentará el conjunto de

⁴⁹ Sitio oficial de GreenFoot. Sección dedicada a especificar los aspectos y características del proyecto, más específicamente lo correspondiente a ¿Qué es GreenFoot?.
<http://www.greenfoot.org/about/whatis.html>

funciones que fueron escogidas para ser evaluadas en cada IDE. Recopilando esta información se obtiene lo siguiente:

4.2.1. Características Básicas

- **COMPILADOR:** Mediante esta función, el IDE permite al usuario llevar a cabo el proceso de compilación de su código fuente de una manera transparente, además dicha función está ligada con una consola de salida donde se puede observar el resultado de aplicar la operación al fichero con el código fuente.
- **EXECUTER-LINKER:** Ofrece la posibilidad al usuario de realizar la ejecución del código.
- **CONSTRUCTOR DE JAR:** Al hacer uso de esta función, la IDE permite al usuario realizar dos operaciones diferentes, la primera es: realizar un comprimido de extensión .jar donde se almacenen las clases que se encuentra desarrollando, y la segunda opción, requiere que el usuario le indique la clase principal que se ejecutara al iniciar la aplicación y automáticamente se genera un archivo ejecutable de extensión .jar.
- **EDITOR**

Editor de Texto: Permite escribir el código fuente del archivo que se ha creado, en texto plano y sin la aplicación de un formato. Editor de texto trabaja en paralelo con otras características que la complementan, con el fin de brindar una presentación ordenada, legible y estructurada del código fuente.

Editor de Márgenes: El editor de márgenes son guías agregadas para facilitar la navegación y ubicación al desarrollador en el código fuente. Está compuesto por las siguientes subcaracterísticas: Véase Figura 13.

- **Número de Línea:** Esta función enumera cada línea de código fuente escrito y saltos línea, con el propósito de establecer puntos de ubicación directos a las filas de código fuente.

- **Marcadores:** Consiste en la aplicación de una serie de símbolos definidos por el IDE, en la línea de código que corresponda, para representar ya sea la ubicación actual del cursor, errores, advertencias, excepciones, sugerencias, entre otras.
- **Bloques Desplegables:** Esta guía le permite al desarrollador percibir el bloque que abarca ya sea la clase, un método, una bifurcación, un bucle, entre otros. Además estos bloques se despliegan para comprimir el espacio en líneas de todo el archivo fuente.
- **Final de Línea:** Guía que determina con la representación de un símbolo, el final de cada línea en el código.
- **Marcador de Tabulador:** Esta guía marca los espacios del tabulador o sangría en cada línea de código.

```

8 | public class Principal { [F]
9 |   .....public static void main(String args[]){ [F]
10|     .....int i=0; [F]
11|     .....[F]
12|     .....System.out.println("hola mundo"); [F]
13|     .....int s=Integer.valueOf("jashdkja"); [F]
14|     .....[F]
15|   } [F]
16| } [F]

```

→ Marcador de Tabulador
 → Final de Línea
 → Bloques Desplegables
 → Marcadores
 → Número de Línea

Figura 13. Subcaracterísticas del Editor de Márgenes.

Resaltador de sintaxis: Es una función que tiene como objetivo mejorar la legibilidad y el contexto del código fuente, a través de colores y efectos aplicados a la fuente, de acuerdo a las categorías de los términos del lenguaje, es decir, definir color y tipo de fuente para las palabras reservadas, operadores, atributos y demás elementos que conforman la sintaxis de un lenguaje de programación.

Auto-formato de código fuente: Consiste en establecer espacios de tabulación y sangrado, saltos de líneas en finalización de código de

acuerdo a las configuraciones que el IDE establezca a las etiquetas, instrucciones, sentencias y demás aspectos del lenguaje que soporta. Esto le da un aspecto limpio al código fuente, aumenta la legibilidad, reduce la probabilidad de errores de sintaxis, y sobre todo ahorra tiempo.

Sugerencias de Finalización de Código: Su único fin es el de facilitar y agilizar la escritura de código por parte del desarrollador. Consiste en un seguimiento a la escritura del desarrollador para sugerirle el siguiente término a través de un cuadro de dialogo, reduciendo en cada paso las posibilidades, hasta llegar al final de la sentencia. Todo esto es llevado a cabo teniendo en cuenta la API (*Application Programming Interface*) del lenguaje de programación que se esté utilizando.

- **MANEJADOR DE PROYECTOS:** Esta funcionalidad permite al usuario realizar la respectiva gestión de proyectos, dentro de lo que se encuentran funciones básicas tales como: crear un proyecto, abrir un proyecto existente, eliminar un proyecto, renombrar un proyecto, visualizar los archivos contenidos dentro de las carpetas pertenecientes al proyecto.
- **AYUDAS DE MANEJO:** Función de la cual el usuario puede valerse al momento de presentar inconvenientes dudas o inquietudes acerca del manejo o configuración de los diferentes componentes y funcionalidades del IDE.
- **CONSOLA DE SALIDA:** Permite al usuario llevar a cabo la visualización de los diferentes mensajes generados durante el proceso de desarrollo. La consola de salida de encuentra compuesta por 4 elementos que algunas veces se visualizan de manera independiente, dichos elementos son:

Consola de compilación: Muestra los mensajes correspondientes al proceso de compilación.

Consola de ejecución: Despliega los mensajes generados durante el proceso de ejecución.

Consola de excepciones: Permite visualizar los mensajes correspondientes a excepción generadas durante los diversos procesos que se ejecuten.

Consola de advertencias: Muestra las advertencias generadas durante la ejecución de los diferentes procesos.

4.2.2. Características avanzadas

- **ANÁLISIS DE RENDIMIENTO:** Función que se enfoca en el cálculo y determinación de serie de variables de consumo de recursos (Cant. de memoria utilizada, porcentaje de CPU, número de invocaciones de un método, entre otras), que le sirven al desarrollador como índices de rendimiento y comportamiento en la ejecución de la aplicación en desarrollo.
- **CONTROL DE VERSIONES:** Es la gestión que el IDE permite realizar a los cambios sobre los elementos que conforman un proyecto. Esto facilita la administración de las distintas versiones realizadas, estableciendo puntos de cambios realizados y los elementos afectados por dichos cambios.
- **CONSTRUCTOR DE GUI:** Es una de las funciones avanzadas más consideradas en los IDE, ya que le permite al desarrollador despreocuparse por las interminables líneas de código que se deben escribir para implementar componentes gráficos. Este generalmente ofrece una paleta de elementos visuales donde el desarrollador puede arrastrarlos y organizarlos, mientras el IDE de forma automática y transparente genera el código fuente respectivo.
- **HERRAMIENTAS DE MODELADO:** Algunas IDE incluyen en su funcionalidad componentes que permiten crear diagramas de modelado ya sean particulares o basados en UML (*Unified Modeling Language*).
- **DESARROLLO WEB:** Esta característica es implementada en algunos IDE, para brindar la opción de crear un proyecto web, o en otros casos archivos fuente con extensión utilizada para esta

plataforma. Esta característica se ve en aquellos IDE que generalmente soporta varios lenguajes de programación, ofreciendo una variedad de alternativas de desarrollo al programador para evitarle la adquisición de un IDE por cada lenguaje que va a utilizar.

- **ADMINISTRADOR DE COMPLEMENTOS:** Algunos IDE ofrecen una funcionalidad predeterminada, pero le permiten al desarrollador obtener ciertos complementos que se crean necesarios para personalizar el IDE a las necesidades del programador. Estos complementos son suministrados por un gestor que les genera un listado de ellos, o la posibilidad de búsqueda de alguno en particular. Para tener una idea mejor sobre esta característica se puede mencionar como ejemplo un complemento de UML para el IDE, donde este le permita crear diagramas basados en dicho estándar.
- **IMPORTACIÓN DE PROYECTOS DE OTROS IDE:** Cada IDE establece la estructura de la construcción de sus proyectos, lo cual provoca incompatibilidades entre ellos, excepto en aquellos que proponen una estructura similar a la establecida, generalmente por los IDE más posicionados en el mercado. Esto es posible, ya que en algunos de estos, sus fabricantes los han establecido como proyectos de código abierto, permitiendo que el grupo desarrollador de otro IDE pueda adaptarlo al suyo, con el fin de que sean compatibles entre ellos.
- **TRAZABILIDAD:** Función agregada como ayuda para permitirle al desarrollador la visualización del código fuente de las distintas clases que componen la API del lenguaje, pero están estrictamente restringidas a la edición.
- **GENERAR CÓDIGO:** Función establecida como ayuda al programador para evitar la escritura de código complementario y así pueda concentrarse en escribir el código que corresponda a la lógica del problema a resolver. Algunas permiten entre otras opciones generar una estructura mínima de una clase cuando esta es creada, en otros casos generan a partir de la declaración de atributos globales, los respectivos métodos *getter* y *setter* e incluso

constructores. Las opciones de generación de código son establecidas por los IDE que soportan esta función.

- **NAVEGADOR:** Ofrecerle al desarrollador una vista global de cómo está compuesto el archivo fuente que está editando, esto es una ayuda de ubicación importante. Algunos IDE implementan esta función brindando una visualización de cada elemento (atributos, métodos, etc...) que compone el archivo fuente, algunos además agregan opciones de traslado como: ir a la línea, ir al método entre otras.
- **GENERADOR DE JAVADOC:** Algunos IDE permiten generar la documentación de la API en formato HTML a partir de comentarios presentes en el proyecto o el archivo fuente que estén trabajando. Esta documentación permite obtener una vista generalizada del contenido de proyecto, logrando visualizar parámetros, datos que devuelve cada método entre otras cosas.
- **GESTOR DE PRUEBAS (JUnit):** Funcionalidad agregada por algunos IDE que le permite realizar al desarrollador la ejecución de su proyecto o archivos fuente de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos o funciones se comportan como se espera. Los IDE que tienen esta función integran las bibliotecas de JUnit, y básicamente operan evaluando en función de algún valor de entrada el valor retornado esperado; si cumple con la especificación, entonces la prueba de ese método fue exitosa; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, se devolverá un fallo en el método correspondiente.
- **GESTIÓN DE LIBRERÍAS EXTERNAS:** En ocasiones es necesario integrar cierta funcionalidad a nuestra aplicación y conocemos de cierta librería que suple esa necesidad, pero para integrarla al proyecto es necesario que el IDE permita agregarla. Algunos IDE admite la adición de librerías al proyecto a través de bibliotecas o archivos con extensión .jar ó .war.
- **DEPURADOR:** Función que permite examinar el código fuente, para analizar su comportamiento, el estado de los elementos de los métodos en ejecución y variables, estableciendo puntos de

interrupción en el cual el desarrollador desea que el depurador se detenga.

- **DISPONIBILIDAD PARA MÚLTIPLES PLATAFORMAS:** Esta es una de las características que los desarrolladores tienen en cuenta para adquirir un IDE. Consiste en la disponibilidad que un IDE tiene para los distintos y más populares sistemas operativos del mercado; algunos solo están disponibles para una sola plataforma lo cual le da cierto grado de limitación, frente a los IDE multiplataforma.
- **SOPORTE PARA MÚLTIPLES LENGUAJES DE PROGRAMACIÓN:** Permite al desarrollador crear proyectos de programación en distintos lenguajes, usando el mismo IDE, facilita su trabajo ya que evita tener muchos IDE para cada uno de los lenguajes en los que trabaja.

4.2.3. Matriz de Caracterización de los IDE'S

Los resultados de la evaluación de los diferentes IDE, fueron recopilados mediante una matriz, cuyas filas corresponden a las diversas características mencionadas anteriormente, y sus columnas hacen referencias a los IDE evaluados. Las celdas correspondientes a la intersección entre una característica y un IDE, pueden contener una (X) en caso de que cumpla, y celda vacía en ausencia.

Las tablas 1 y 2 corresponden a las matrices del análisis de las características básicas y avanzadas de las IDEs seleccionadas:

Características Básicas		Netbeans	BlueJ	DrJava	Eclipse	IntelliJava	JBuilder	JCreator	Anjunta	Geany	SlickEdit
Compilador		X	X	X	X	X	X	X	X	X	X
Executer-linker		X	X	X	X	X	X	X	X	X	X
Constructor (JAR)		X	X	X	X	X	X	X	X	X	X
Editor	Editor de Texto	X	X	X	X	X	X	X	X	X	X
	Número de línea	X	X		X	X	X	X	X	X	X
	Marcadores	X	X		X	X	X	X	X	X	X
	Bloques desplegables	X	X		X	X	X	X	X	X	
	Marcador de tabulador				X	X	X	X	X	X	X
	Final de línea	X			X		X		X	X	X
	Resaltador de sintaxis	X	X	X	X	X	X	X	X	X	X
	Auto-formatos de código fuente	X	X	X	X	X	X	X	X	X	X
	Sugerencias de finalización de código	X			X	X	X				X
	Manejador Proyectos(organización de paquetes y clases)	X	X	X	X	X	X	X	X	X	X
Ayudas Contenido, Manejo		X	X	X	X	X	X	X	X	X	X
Consola de salida	Compilación	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	X	X	X	X	X	X
	Excepciones	X	X	X	X	X	X	X	X	X	X
	Advertencias	X	X		X	X	X	X	X	X	X

Tabla 1. Matriz de Caracterización de Funciones Básicas de las IDE'S seleccionadas

Características Avanzadas	NetBeans	BlueJ	DrJava	Eclipse	IntelliJava	JBuilder	JCreator	Anjunta	Geany	SlickEdit
Análisis de rendimiento	X			X	X	X				
Control de versiones	X			X	X	X				X
Constructor de GUI	X			X	X	X		X		X
Herramientas de modelado (CASE)	X	X		X		X				
Desarrollo web	X			X	X	X			X	X
Administrador de complementos (Plugins)	X			X		X		X	X	
Importación de proyectos de otros IDEs	X	X			X	X				
Visualización de clases de la API	X									
Generador de Código	X	X		X	X	X	X		X	X
Navegador	X	X	X	X	X	X	X	X	X	X
Generador JavaDoc	X	X	X	X	X	X				X
Gestor de pruebas (JUnit)	X		X		X	X				X
Gestión de librerías externas	X			X	X	X		X		X
Depurador	X	X	X	X	X	X			X	X
Disponibilidad para múltiples plataformas	X	X	X	X	X	X			X	X
Soporte para varios lenguajes de programación	X			X	X	X		X	X	x

Tabla 2. Matriz de Caracterización de Funciones Avanzadas de las IDE'S seleccionadas

4.3. TABULACIÓN Y ANÁLISIS

Con el fin de obtener información que brinde herramientas necesarias, para enfocar el trabajo a la búsqueda de una solución orientada a satisfacer las necesidades expresadas por el estudiante, y con base en las funcionalidades con las que cuentan los IDEs presentes en el mercado; se utilizan dos estrategias para la recolección de información. La primera de ellas consiste en el análisis de las características que presentan los IDEs comerciales, mediante tablas de comparación, y la segunda estrategia, es mediante encuestas. Cabe resaltar que la información obtenida mediante la depuración de los datos de ambas estrategias, brinda una perspectiva acerca de lo que ofrece el mercado, y la opinión de los estudiantes acerca de ello.

4.3.1. Tabulación

A partir de la información obtenida mediante la evaluación de las características de los IDE seleccionados, presentada anteriormente en matrices (ver Tablas 1 y 2), se realizó una posterior tabulación que será presentada en las tabla 3 y 4, donde cada fila corresponde a las características evaluadas, y la columna hace referencia al porcentaje de IDE seleccionados que cumplen con dicha característica.

Las tablas 3 y 4 presentan la siguiente estructura:

- La columna izquierda hace referencia a las características definidas de las IDEs, mencionadas en el apartado 4.2. EVALUACIÓN DE LAS FUNCIONES DE LOS IDE'S.
- A su vez, la columna derecha representa el porcentaje de las IDEs evaluadas que poseen tal característica. Por ejemplo: El 100% de las IDEs tienen la característica básica “compilador”.

De la evaluación se obtiene que:

Características Básicas		Porcentaje de los IDEs evaluados que cumplen con la característica (%)
Compilador		100
Executer-linker		100
Constructor (JAR)		100
Editor	Editor de Texto	100
	Numero de línea	90
	Marcadores	90
	Bloques desplegables	90
	Marcador de tabulador	60
	Final de línea	60
	Resaltador de sintaxis	100
	Auto-formatos de código fuente	100
	Sugerencias de finalización de código	50
	Manejador Proyectos(organización de paquetes y clases)	100
Ayudas Contenido, Manejo		100
Consola de salida	Compilación	100
	Ejecución	100
	Excepciones	100
	Advertencias	90

Tabla 3. Porcentajes de Cumplimiento de Características Básicas

Características Avanzadas		Porcentaje de los IDEs evaluados que cumplen con la característica (%)
Análisis de rendimiento		40
Control de versiones		50
Constructor de GUI		60
Herramientas de modelado (CASE)		40
Desarrollo web		60
Administrador de complementos (Plugins)		50
Importación de proyectos de otros IDEs		40
Visualización de clases de la API		10
Generador de Código		80
Navegador		100
Generador JavaDoc		70

Gestor de pruebas (JUnit)	50
Gestión de librerías externas	60
Depurador	80
Disponibilidad para múltiples plataformas	80
Soporte para varios lenguajes de programación	70

Tabla 4. Porcentajes de Cumplimiento de Características Avanzadas

4.3.2. Análisis

De acuerdo a los resultados observados, fueron definidos 4 grupos de características teniendo en cuenta los porcentajes presentados en las anteriores tablas. Estos grupos están conformados de la siguiente manera:

- Grupo 1 (100%): Características analizadas en las cuales el 100% de los IDE evaluados coinciden.
- Grupo 2 (entre 80%-90%): Características analizadas en las cuales entre un 80% y 90% de los IDE evaluados coinciden.
- Grupo 3 (entre 60%-70%): Características analizadas en las cuales entre un 60% y 70% de los IDE evaluados coinciden.
- Grupo 4 (menor o igual a 50%): Características analizadas en las cuales un porcentaje igual o menos al 50% de los IDE evaluados coinciden.

A continuación se mostraran las características que conforman cada grupo.

GRUPO 1 (100%): Está conformado por las siguientes características.

- Compilador
- Executer-Linker
- Constructor (.jar)
- Editor de texto
- Resaltador de sintaxis
- Auto-formatos de código fuente.
- Manejador de proyectos (organización de paquetes y clases)
- Ayudas contenido, manejo
- Consola de compilación
- Consola de ejecución
- Consola de excepciones
- Navegador

GRUPO 2 (entre 80% - 90%): Está conformado por las siguientes características.

- Número de línea
- Marcadores
- Bloques desplegables
- Advertencias
- Generador de código
- Depurador
- Disponibilidad para múltiples plataformas.

GRUPO 3 (entre 60% - 70%): Está conformado por las siguientes características.

- Marcador de tabulador
- Final de línea
- Constructor de GUI
- Desarrollo web
- Generador de JavaDoc
- Gestor de librerías externas
- Soporte para varios lenguajes de programación

GRUPO 4 (menor o igual a 50%): Está conformado por las siguientes características.

- Sugerencias de finalización de código
- Análisis de rendimiento
- Control de versiones
- Herramientas de modelado (CASE)
- Administrador de complementos
- Importación de proyectos de otros IDE
- Visualización de clases de la API
- Gestor de pruebas (JUnit)

Durante el análisis de los IDE seleccionados, resultaba evidente cierto patrón de diseño en las distintas interfaces gráficas observadas. El resultado obtenido a través de la observación arrojó que el 90% de los IDE coinciden en la presentación de tres paneles correspondientes a: Ventana del Editor, Ventana de Proyectos y Ventana de Salida. (Ver Figura 14).



Figura 14. Resultado de Análisis gráfico de las IDE

Además de la semejanza en el diseño de la interface gráfica de las IDEs presentados en la figura 14, surgen otros dos componentes gráficos compartidos, que corresponden a una barra de menús, en donde depositan toda la funcionalidad de la IDE, y una barra de funciones en donde se ofrecen las funciones básicas más importantes (compilar, ejecutar, depurar, construir). A partir de los aspectos gráficos encontrados durante el análisis y observación de las IDEs, se establece como resultado un factor común gráfico presentado en la Figura 15.

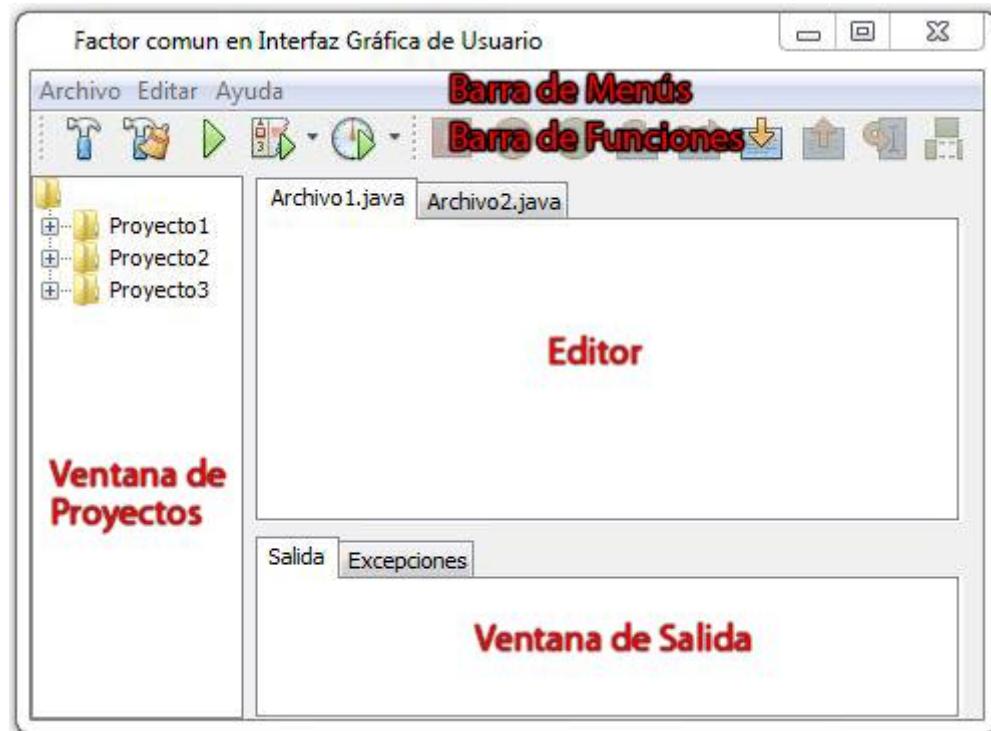


Figura 15. Factor Común en interface gráfica de los IDE.

5. TRABAJO INVESTIGATIVO

5.1. DETERMINACIÓN DE LA POBLACIÓN Y MUESTRA

Debido a la necesidad de conocer la experiencia del estudiante con los IDE, se llevará a cabo la ejecución de un instrumento de recolección de información que permita conocer los inconvenientes presentados a la hora de utilizar un IDE, y las características mínimas con las que este debe cumplir. Es por esto que se toma la siguiente población como objeto de estudio.

5.1.1. Población

Estudiantes de la Universidad Francisco de Paula Santander de programación orientada a objetos, estructuras de datos, y fundamentos de programación que aún no han logrado culminarla con éxito.

La población a tener en cuenta para ejercer la recolección de información fue escogida por las siguientes razones:

- **Estudiantes de Programación Orientada a Objetos:** Son aquellos estudiantes que ya tuvieron la oportunidad de cursar la asignatura Fundamentos de Programación y conocen los problemas que se pueden presentar haciendo uso de IDE específico.
- **Estudiantes que han cursado Fundamentos de Programación y no la han culminado con éxito:** Son estudiantes que ya tuvieron la oportunidad de ver la asignatura Fundamentos de Programación Orientada a Objetos, pero que a diferencias del anterior grupo, no lograron culminarla con éxito y por tal razón deben cursarla de nuevo.
- **Estudiantes de Estructuras de Datos:** Son aquellos estudiantes que han culminado con éxito Programación Orientada a Objetos, y se encuentran cursando la asignatura Estructura de Datos. Las

actividades desarrolladas en este curso por lo general exigen que el estudiante utilice algún IDE, debido a la necesidad de agilizar el trabajo haciendo uso de ciertas funciones avanzadas tales como diseñador de interfaces, generador de JavaDoc, entre otros. Por lo tanto conocen a la perfección los problemas y necesidades a la hora de hacer uso de un IDE.

A continuación se relaciona los estudiantes de la población, que están matriculados en el segundo semestre del año 2011:

Estudiantes de programación orientada a objetos = 71.

Estudiantes repitentes de fundamentos de programación = 59.

Estudiantes de estructura de datos = 49.

Población total = 179 estudiantes.

5.1.2. Muestra

Debido a la dificultad para llevar a cabo la recolección de datos a toda la población seleccionada, es necesario tomar una muestra para facilitar dicha actividad. Como es bien sabido, la muestra es en esencia un subgrupo de la población, y por lo tanto pertenece a ese conjunto definido que contiene las características necesarias que toman importancia en el transcurso de la recolección de datos⁵⁰.

En este caso se va a proceder a tomar una muestra probabilística, ya que todos los elementos pertenecientes a la población tienen la misma posibilidad de ser elegidos y debido a que son esenciales en las investigaciones de tipo cuantitativo donde se pretenden hacer uso de encuestas.

Para determinar el tamaño de una muestra probabilística es necesario conocer y entender los siguientes términos:

Población: Conjunto de elementos (N).

⁵⁰ METODOLOGÍA DE LA INVESTIGACIÓN. Roberto Hernández Sampieri, Carlos Fernández Collado, Pilar Baptista Lucio. Mc Graw Hill.

Muestra: Es un subconjunto de la población (n).

Interesa conocer valores promedio en la población, lo cual se expresa como:

\bar{y} = al valor de una variable determinada (y) que nos interesa conocer.

También interesa conocer:

V = la varianza de la población con respecto a determinadas variables.

se = desviación estándar de la distribución muestral.

$(se)^2$ = es la fórmula que sirve para calcular la varianza (V) de la población(N).

S^2 : varianza de la muestra, la cual podrá determinarse en términos de probabilidad donde $S^2=p(1-p)$.

Tamaño provisional de la muestra $n` = \frac{s^2}{V^2}$

Tamaño de la muestra $n = \frac{n`}{1+n`/N}$

Por lo tanto:

N=179 estudiantes.

$se = 35\% (0.035)$

$V=(0.035)^2=0.001225$

$s^2=p(1-p)=0.9(1-0.9)=0.09$

$$n` = \frac{0.09}{0.001225} = 73.46$$

$$n = \frac{n`}{1+n`/N} = \frac{73.46}{1+73.46/179} = 52.09$$

n = 52. $\cong 52$

5.1.3. Estratificación de la Muestra

Luego se determinar el número de estudiantes pertenecientes a la muestra, se procede a realizar la estratificación de dicha muestra con el fin de que los elementos de análisis posean un determinado atributo evaluable. Por lo tanto, el procedimiento a seguir, es dividir la muestra en submuestras o estratos⁵¹.

Debido al interés de realizar una recolección de información donde se brinde igual prioridad a cada uno de los tres grupos que conforman la población, se ha optado por escoger una división estratificada de la muestra. Su representación estará dada por la proporción correspondiente al número de integrantes de dicho grupo muestral, es decir:

Población = 179 estudiantes.

- 59, es el número de estudiantes repitentes de programación 1, lo que corresponde a un 32.96% de la población.
- 71, es el número de estudiantes de programación orientada a objeto, lo que corresponde a un 39.66% de la población.
- 49, es el número de estudiantes de estructura de datos, lo que corresponde a un 27.37% de la población.

Muestra = 52 estudiantes.

De estos datos se obtiene que:

- La cantidad de Estudiantes de repitentes de fundamentos de programación que harán parte de la muestra es igual a: $52 \times 32.96/100 = 17.13 = 17$.
- La cantidad de Estudiantes de programación orientada a objetos que harán parte de la muestra es igual a: $52 \times 39.66/100 = 20.62 = 21$.
- La cantidad de Estudiantes de estructura de datos que harán parte de la muestra es igual a: $52 \times 27.37/100 = 14.23 = 14$.

5.2. TABULACIÓN Y ANÁLISIS DE ENCUESTAS

5.2.1. Análisis de Resultados

A partir de la información recopilada por medio de la encuesta aplicada a los estudiantes seleccionados, se llevó a cabo el proceso de tabulación; que para efectos de legibilidad, será presentado de la siguiente manera:

El análisis de cada pregunta contiene: enunciado, objetivo, resultado y conclusiones. Por último se presentará un análisis general en donde serán expuestas las conclusiones.

Con el fin de presentar los resultados gráficamente, fue necesaria la utilización de dos tipos de gráficos estadísticos:

- Gráficos de Barras (comparación), con el fin de expresar datos en los cuales cualquiera de las opciones tiene la posibilidad de obtener un porcentaje de participación del 100%.
- Gráfico Circular, que permite visualizar el porcentaje de participación de cada uno de los elementos, y la suma de cada sector equivale al 100%

Pregunta 1:

“Seleccione con cuál de los siguientes entornos de desarrollo integrado (IDEs) para JAVA está familiarizado o ha usado alguna vez” (Respuesta de selección múltiple).

Para la cual se presentaron las siguientes opciones: (Para efectos de tabulación, cada respuesta abierta dada por un estudiante, es asociada con una letra consecutiva).

- | | |
|---|---|
| <input type="checkbox"/> BlueJ. (a) | <input type="checkbox"/> Netbeans. (f). |
| <input type="checkbox"/> DRjava ó Doctor Java (b). | <input type="checkbox"/> JBuilder. (g). |
| <input type="checkbox"/> Eclipse. (c). | <input type="checkbox"/> Anjuta. (h). |
| <input type="checkbox"/> IntelliJava (d). | <input type="checkbox"/> Geany. (i). |
| <input type="checkbox"/> JCreator (e). | <input type="checkbox"/> SlickEdit. (j). |
| <input type="checkbox"/> Otros: _____ (k) | |

Objetivo: Conocer las IDEs con los cuales el estudiante se encuentra familiarizado, de esta manera tomar puntos de referencia con respecto a las funciones y posible estructura gráfica, basados en los IDE seleccionados.

El gráfico 1 muestra los resultados obtenidos:

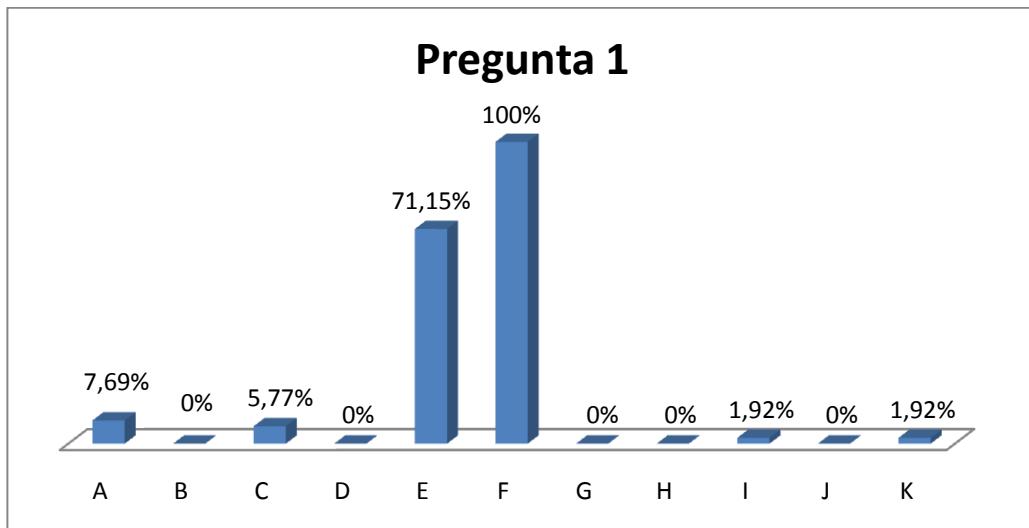


Gráfico 1. Resultado de la pregunta 1.

Del gráfico 1 se observa que:

- El 100% de los estudiantes se encuentra familiarizado con Netbeans.
- El 71,15% de los estudiantes conoce de alguna forma Eclipse.
- Solo el 4% de los estudiantes conoce al menos una IDE diferente a Netbeans y Eclipse.
- Se concluye que los estudiantes se encuentran familiarizados principalmente con dos IDEs, las cuales son Netbeans y Eclipse, teniendo un porcentaje del 100% y 71% respectivamente.

Pregunta 2:

“Que tan de acuerdo está la siguiente afirmación: Los requisitos mínimos (cantidad de memoria RAM, espacio en disco, tecnología del procesador, etc.) que debe cumplir un computador para que se pueda instalar un IDE y que se ejecute bien, son muy exigentes:” (Única respuesta).

Para la cual se presentaron las siguientes opciones:

- a. Totalmente de acuerdo.
- b. De acuerdo.

- c. Ni de acuerdo, ni en desacuerdo.
- d. En desacuerdo.
- e. Totalmente en desacuerdo.
- f. No tengo en cuenta los requisitos de instalación.

Objetivo: Conocer si los estudiantes consideran que los requisitos de *hardware* y *software* necesarios para la correcta instalación y funcionamiento de los IDE, son exigentes. Ya que como es bien sabido, algunos ambientes de desarrollo poseen diversas funcionalidades, exigiendo características funcionales en las maquinas, que muchas veces no se cumplen a cabalidad.

El gráfico 2 muestra los resultados obtenidos:

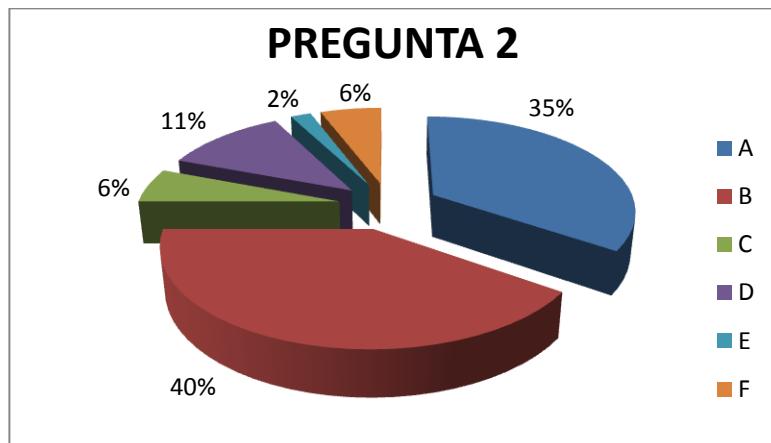


Gráfico 2. Resultados de la pregunta 2.

Se puede concluir que:

- Los estudiantes consideran que los requerimientos de *hardware* mínimos, exigidos para garantizar una correcta instalación y óptimo funcionamiento de las IDEs son altos. Esto se ve reflejado en el 75% obtenido por las opciones A (35%) y B (40%).
- Solo el 13% de los estudiantes consideran que estos requerimientos no son exigentes.
- El 12% de los estudiantes no tienen una opinión clara con respecto a esta afirmación.

Pregunta 3:

“Teniendo en cuenta los requisitos mínimos para instalar un IDE, mencionados en la anterior pregunta. ¿Considera que el equipo de cómputo que regularmente utiliza los cumple?” (Única respuesta).

Para la cual se presentaron las siguientes opciones:

- a. Sí.
- b. No.
- c. No lo sé.

Objetivo: conocer si los equipos de cómputo que regularmente son utilizados por los estudiantes cumplen con las características exigidas por las IDE para correcto funcionamiento.

El gráfico 3 muestra los resultados obtenidos:

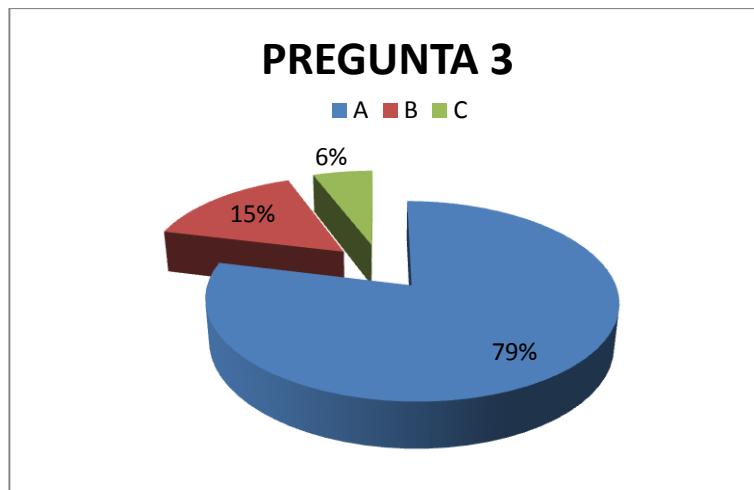


Gráfico 3. Resultados de la pregunta 3.

Se concluye que:

- El 79% de los estudiantes manifiesta tener equipos de cómputo que cumplen con las características exigidas por las IDEs para su correcto funcionamiento, lo cual entra en contradicción con respuestas dadas en posteriores preguntas, que reflejan inconformidad con el rendimiento de sus equipos al momento de hacer uso de ciertas funcionalidades brindadas por la IDE, tales como compilar y ejecutar, además del tiempo de carga requerido para iniciar la aplicación; lo

cual evidencia un posible desconocimiento de las características de computo requeridas.

- El 15% de los estudiantes posee equipos de cómputo que no cumplen con las características mínimas, exigidas para su correcto funcionamiento.
- El 6% de los estudiantes no tienen una opinión clara con respecto a esta pregunta lo cual concuerda con el 6% de estudiantes que manifestaron no tener en cuenta los requerimientos de hardware exigidos.

Pregunta 4:

“Basado en el uso que ha tenido con algún o algunos IDE, con cuál o cuáles de los siguientes inconvenientes se ha enfrentado:” (Múltiple respuesta)

Para la cual se presentaron las siguientes opciones (Para efectos de tabulación, cada respuesta abierta dada por un estudiante, es asociada con una letra consecutiva a la última utilizada):

- Se bloquea el equipo. (a).
- Cuando ejecuto el IDE se demora en cargar. (b).
- Se demora en compilar. (c).
- Se demora en ejecutar. (d).
- No encuentro fácilmente funciones que necesito. (e).
- Si abro un proyecto de una versión diferente del mismo IDE no funciona bien. (f).
- Me Confundo al utilizarlo. (g).
- No carga archivos .jar que necesito para la aplicación. (h).
- La versión del IDE que instalé tiene fallas. (i).
- Daña los archivos fuente de JAVA. (j).

- No encuentra la máquina la virtual de JAVA. (k).
- El IDE es incompatible con la versión de la Máquina Virtual. (l).
- Las respuestas de error o de advertencia del compilador no las entiendo. (m).
- No encuentra librerías que la aplicación necesita y muestra un mensaje de advertencia. (n).
- El IDE se cierra mientras lo estoy usando. (o).
- Otros: _____ (p) .

Objetivo: Conocer los problemas que presentan frecuentemente los estudiantes a la hora de hacer uso de una IDE.

El gráfico 4 muestra los resultados obtenidos:

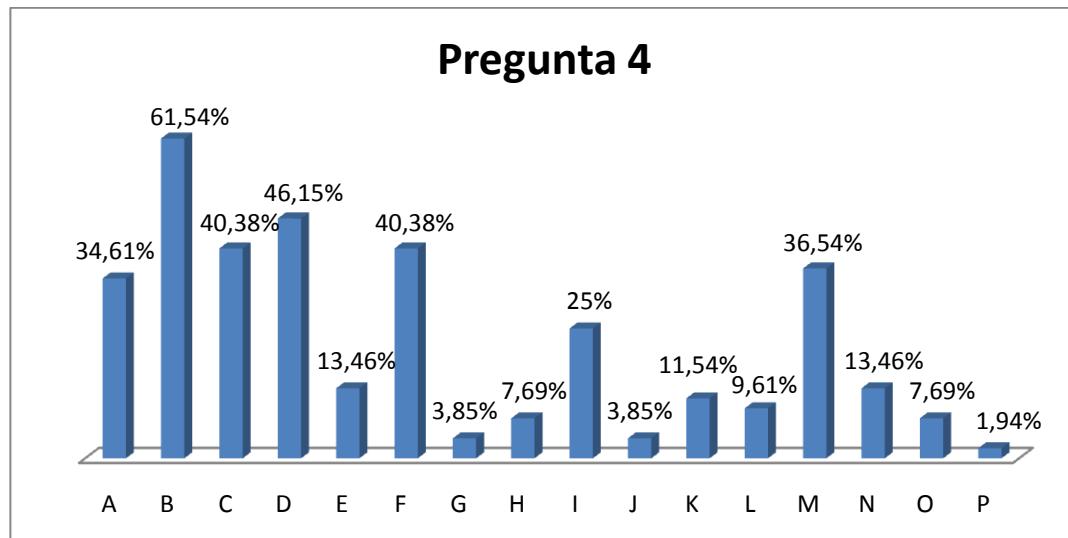


Grafico 4. Resultado de la pregunta 4.

Por último, se puede concluir que:

- El problema que más presentan los estudiantes a la hora de hacer uso de una IDE, con un 61,53%, es que “Cuando lo ejecutan se demora en cargar”.
- Se encontraron otros grupos de respuestas, que si bien poseen un porcentaje inferior al mencionado anteriormente, si tienen una cantidad realmente significativa. Los cuales son: el grupo número uno está conformado por porcentajes superiores al 40%, y el grupo numero dos está compuesto por porcentajes superiores al 25% e inferiores al 40%. A continuación se especificaran los grupos uno y dos.
 1. El grupo uno está conformado por las respuestas: “Se demora en compilar” (40,38%), “Se demora en ejecutar” (46,15%), “Si abro un proyecto de una versión diferente del mismo IDE no funciona bien” (40,38%).
 2. El grupo dos está conformado por las respuestas: “Se bloquea el equipo” (34,61%), “La versión del IDE que instale tiene fallas” (25%), “Las respuestas de error o de advertencia del compilador no las entiendo” (36,53%).
- Se resalta que todas las opciones de respuesta presentadas al estudiante, fueron seleccionadas, si bien con un porcentaje mínimo, pero lo que se traduce en que todos los problemas planteados los han presentado los estudiantes al menos una vez.

Pregunta 5:

“Si necesitara un IDE para programar. ¿Qué características debe tener para que usted lo elija?.” (Múltiple respuesta)

Para la cual se presentaron las siguientes opciones (Para efectos de tabulación, cada respuesta abierta dada por un estudiante, es asociada con una letra consecutiva):

- Que la Interfaz gráfica sea agradable. (a).
- Que las funciones básicas estén a la vista. (b).
- Que los tiempos de respuesta sea bajo. (c).
- Que sea fácil de usar. (d).

- Que su rendimiento en equipos de gama baja sea eficiente. (e).
- otros_____ (f).

Objetivo: conocer los criterios de selección de los estudiantes a la hora de elegir un IDE.

El gráfico 5 muestra los resultados obtenidos:

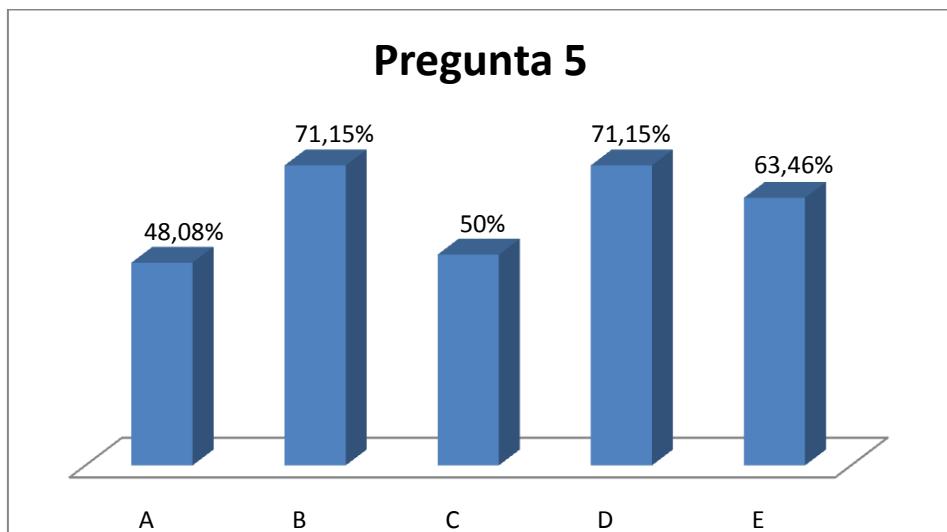


Gráfico 5. Resultados de la pregunta 5.

Se concluye que:

- La respuesta de la mayoría de los estudiantes estuvo orientada a la facilidad de uso y sencillez en la interface, representadas por las opciones “Que las funciones básicas estén a la vista” y “Que sea fácil de usar” respectivamente, ambas con un porcentaje de participación del 71,15%.
- El 63,46% marcó la opción “Que su rendimiento en equipos de gama baja sea eficiente” convirtiéndola en un porcentaje realmente significativo. Además se presentaron otros dos porcentajes a tener en cuenta los cuales fueron: 48,07% y 50%, correspondientes a las respuestas “Que la interfaz gráfica sea agradable” y “Que los tiempo de respuesta sean bajos”, respectivamente.

- Es necesario resaltar que todas las opciones propuestas como respuesta, obtuvieron un porcentaje de participación superior al 45%, lo cual significa que tienen gran relevancia al momento de elegir un IDE.

Pregunta 6:

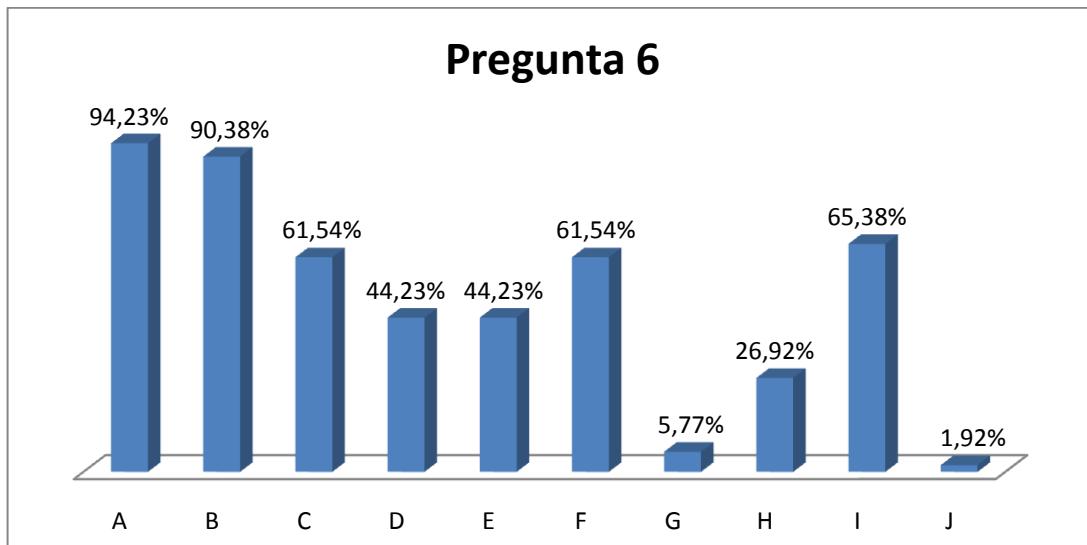
“De las siguientes opciones marque las que considere debe tener un IDE para JAVA:” (Múltiple respuesta)

Para la cual se presentaron las siguientes opciones: (Para efectos de tabulación, cada respuesta abierta dada por un estudiante, es asociada con una letra consecutiva).

- Compilar. (a).
- Ejecutar. (b).
- Depurar. (c).
- Generar o construir un .jar. (d).
- Generar javadoc (Documentación o API). (e).
- Generar código fuente. (f).
- Auto-formatear código fuente. (g).
- Visualización por consola. (h).
- Constructor de interfaces gráficas. (i).
- Otros: _____ (j) .

Objetivo: Conocer los componentes software que espera encontrar el estudiante en una IDE, a diferencia de la pregunta anterior que está más enfocada a características relacionadas con la usabilidad.

El gráfico 6 muestra los resultados obtenidos:



Gráfica 6. Resultados de la pregunta 6.

Por último, se puede concluir que:

- Las opciones “compilar” y “ejecutar”, fueron seleccionadas por un porcentaje de estudiantes superior al 90%, teniendo “compilar” 94,23% y “ejecutar” 90,38%, con lo que se concluye que estos dos componentes software son considerados como fundamentales en una IDE.
- Otro grupo de respuestas son aquellas que presentaron un porcentaje de participación superior al 60%, las cuales fueron: “Depurar” 61,54%, “Generar código fuente” 61,54%, “Constructor de interfaces graficas” 65,38%. Las anteriores opciones se convierten en funciones consideradas como importantes por los estudiantes.
- Las opciones D y E, correspondientes a “Generar o construir un .jar” y “Generar javadoc”, ambas obtuvieron un porcentaje del 44,23%, que si bien es inferior a las opciones mencionadas en ítems anteriores, si hace referencia a un porcentaje considerable, lo cual indica que dichas funciones deben ser tenidas en cuenta.
- Para finalizar se observa que la opción “Visualización por consola” obtuvo un porcentaje del 26,92%, lo que no la convierte en una función considerada como vital para el 73,08% de los estudiantes. Las demás opciones tuvieron un porcentaje de participación inferior al 6%.

Pregunta 7:

“¿Considera que muchas de las funciones avanzadas de una IDE, como análisis de rendimiento, control de versiones, gestor de pruebas, gestor de complementos, entre otras, son necesarias para usted? (Única respuesta)

Para la cual se presentaron las siguientes opciones:

- a. Sí.
- b. No.
- c. No lo sé.

Objetivo: determinar si los estudiantes consideran necesarias las funciones avanzadas en una IDE. A continuación los resultados obtenidos, expresados mediante un gráfico circular:

El gráfico 7 muestra los resultados obtenidos:

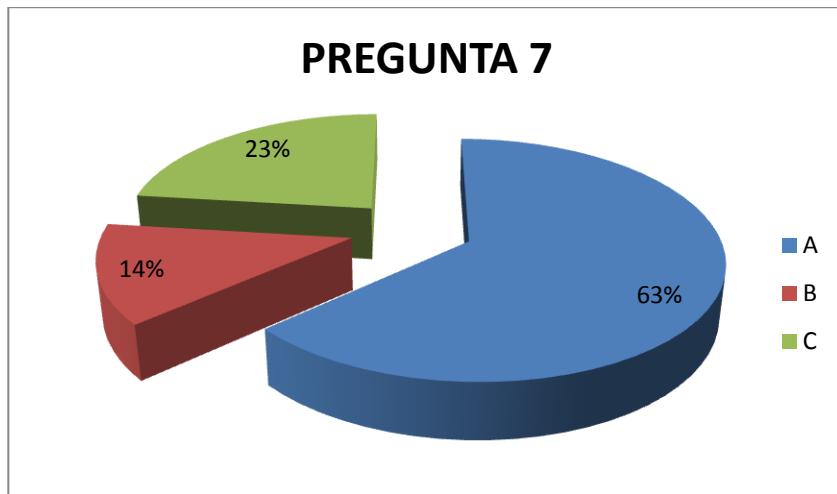


Gráfico 7. Resultados de la pregunta 7

Por último, se puede concluir que:

- La mayoría de los estudiantes (63%) considera que las funciones avanzadas son necesarias al momento de manejar la IDE.
- Un 37% de los estudiantes consideran que: las funciones avanzadas no son necesarias a la hora de hacer uso de una IDE (14%), o no saben si realmente las necesitan (23%).

Pregunta 8:

“¿Le gustaría contar con una IDE en un entorno web, para sus actividades de programación?” (Única respuesta)

Para la cual se presentaron las siguientes opciones:

- a. Sí.
- b. No.
- c. No lo considero importante

Objetivo: contar con una IDE en línea para realizar sus actividades de programación.

El gráfica 8 muestra los resultados obtenidos.



Gráfico 8. Resultados de la pregunta 8.

Por último, se puede concluir que:

- El 94% de los estudiantes le gustaría contar con una IDE en un entorno web para llevar a cabo sus actividades de programación. Lo que se convierte en un indicador vital para el desarrollo del proyecto ya que demuestra que la idea de la construcción de un IDE en un ambiente web es bien recibida por el público a quien va dirigida.
- Por otro lado se observa que tan solo el 6% manifestó que: no le gustaría contar con una IDE en un entorno web (2%), o simplemente no lo consideran importante (6%).

5.2.2. Análisis General

Luego de la respectiva tabulación y análisis realizado a cada una de las preguntas hechas a los estudiantes seleccionados, se puede observar que los resultados brindan una gran guía acerca de:

- La estructura gráfica de los IDE con los que el estudiante se encuentra familiarizado: Netbeans y Eclipse, que con la evaluación a las IDE, arroja como resultado un bosquejo de la interface gráfica que debería ser tomada en cuenta para sacar un mayor provecho de la aplicación a construir (Véase sección 4.3.2). Ya que el estudiante se encuentra familiarizado con una estructura gráfica, lo cual debe ser aprovechado y tenido en cuenta al momento de llevar a cabo el análisis y diseño de la interface. Además los estudiantes manifestaron la necesidad de una interface fácil de usar.
- Los problemas con los cuales el estudiante se enfrenta a la hora de hacer uso de las IDE tales como: interfaces complejas, desconocimiento de características de cómputo para correcto funcionamiento, pérdida de tiempo esperando la culminación de ciertos procesos como la carga de la aplicación, compilación y ejecución de proyectos, entre otros. Brindan una guía que ayudará a fijar la atención en la manera de solucionar los anteriores problemas en el desarrollo del proyecto.
- Las características que el estudiante considera que una IDE debe tener, encierra tanto consideraciones de usabilidad (“Que las funciones básicas estén a la vista” y “Que sea fácil de usar”), como de funcionalidades requeridas (“compilar” y “ejecutar”), que junto con el estudio realizado anteriormente a las IDE, arroja información vital acerca de requerimientos con los que debe cumplir la aplicación, pero de igual forma adaptándose también a los alcances y limitaciones presentadas anteriormente.

Además se hace necesario señalar que gracias a la ejecución, tabulación y análisis de las encuestas, se tiene información certera acerca de lo que realmente desea el estudiante y que problemas presenta; evitando así partir de supuestos que de ser erróneos entorpecerían el resultado final del proyecto.

6. GENERALIDADES DE JAVA

6.1. MIRANDO DENTRO DE LA PLATAFORMA DE JAVA⁵¹

La plataforma de Java cuenta con la *Java Virtual Machine*, y la API de Java, que hacen parte fundamental de su arquitectura. A continuación se describirán estos conceptos: (Ver figura 16)

- **Máquina Virtual de Java** – Es la piedra angular del lenguaje de programación Java. Es el componente de tecnología responsable de la multiplataforma, del pequeño tamaño del código compilado, y la capacidad de proteger a los usuarios de los programas maliciosos⁵². Es una máquina abstracta diseñada para ser implementada sobre procesadores existentes.
- **API de Java**⁵⁴ – Forma una interface estándar para las applets y aplicaciones, independientemente del sistema operativo en el que subyace. La API es el marco fundamental para el desarrollo de aplicaciones, especificando un conjunto de interfaces esenciales, en un creciente número de áreas clave que los desarrolladores utilizan para construir sus aplicaciones. La API está compuesta por:
 - **API Base de Java**–Proporciona un lenguaje muy básico, utilidades, Entrada/Salida, red, interface gráfica de usuario, y servicios applet.
 - **API Java de Extensión Estándar** - Extiende las capacidades de Java más allá de la API Base, pero eventualmente se integrarán a esta. Otras extensiones no estándar pueden ser proporcionadas por un applet, una aplicación, o el sistema operativo. A medida que cada extensión de especificación API se publica, se pondrá a disposición para la revisión de la industria y los comentarios antes de ser liberada.

⁵¹ KRAMER Douglas. *The Java Platform A White Paper. A Look Inside the Java Platform.* 1996.
<http://java.sun.com/docs/white/platform/javaplatform.doc2.html>

⁵² *The Java Virtual Machine Specification.* LINDHOLM Tim, YELLIN Frank. Second Edition. Chapter 1-Introduction. View HTML in:
<http://docs.oracle.com/javase/specs/jvms/se5.0/html/Introduction.doc.html>

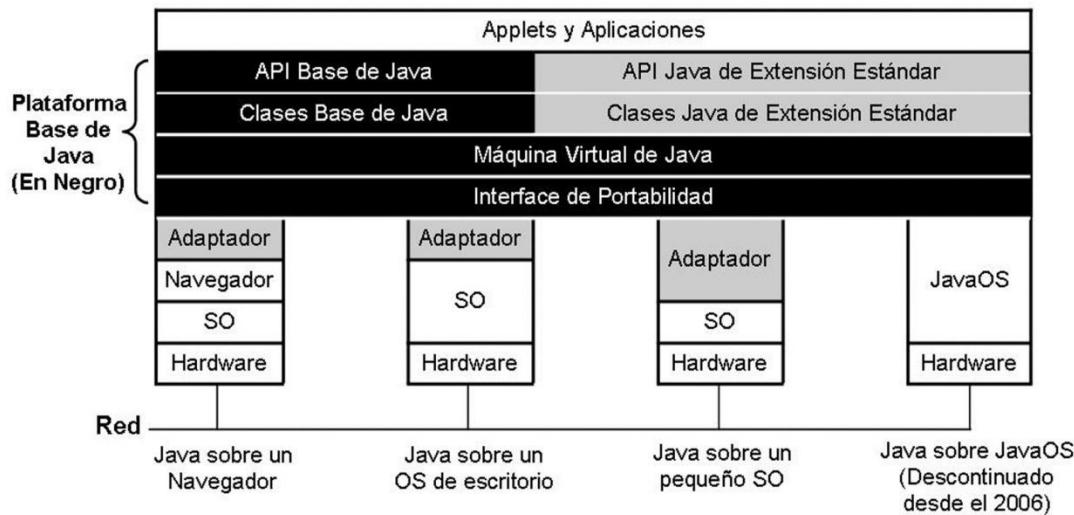


Figura 16. Uniformidad de la Plataforma Base de Java sobre los Sistemas Operativos⁵⁴

6.2. LA ARQUITECTURA DE LA MÁQUINA VIRTUAL DE JAVA

Java fue diseñado con el objetivo de facilitar la portabilidad de aplicaciones, es decir, que cualquier programa escrito se pudiese ejecutar en cualquier máquina y/o dispositivo, sin tener la necesidad de volver a compilarlo o en el peor caso reescribirlo. La dependencia al sistema o plataforma donde se ejecutan aplicaciones, está completamente ligada con la JVM (*Java Virtual Machine*), es decir, para que una aplicación escrita en este lenguaje pueda ejecutarse, debe existir una JVM implementada para el ambiente en que se desea hacer la ejecución, esto permite que el desarrollador se enfoque en cómo solucionar el problema propuesto y no en dónde se va a ejecutar esta solución.

El comportamiento de una instancia de la JVM es descrita en términos de subsistemas, áreas de memoria, tipos de datos e instrucciones. Estos componentes describen una arquitectura abstracta interna para la máquina virtual. El propósito de estos componentes no es más que dictar una arquitectura interna para las implementaciones.⁵³

La Figura 17. Muestra un diagrama de bloques de la JVM, que incluye los principales subsistemas y áreas de memoria. Cada JVM tiene un subsistema

⁵³ VENNERS, Bill. Inside the Java Virtual Machine. Capítulo 5.The Java Virtual Machine. The Architecture of the Java Virtual Machine.McGraw-Hill. 1997

cargador de clases: un mecanismo de tipo de carga dado los nombre completamente calificados, y además también cuenta con un motor de ejecución: un mecanismo responsable de ejecutar las instrucciones contenidas en los métodos de la clases cargadas.

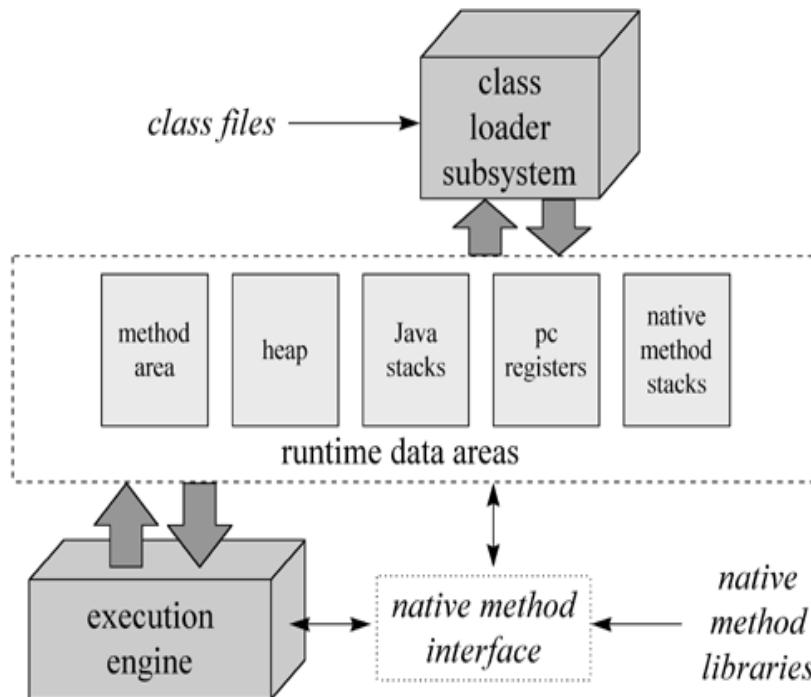


Figura 17. Arquitectura Interna de la JVM.⁵⁴

Cuando la máquina virtual ejecuta un programa, necesita memoria para almacenar muchas cosas, incluyendo los bytecodes y otra información extraída desde el cargador de archivos class, instancias de objetos del programa, parámetros de los métodos, valores de retorno, variables locales, y resultados intermedios de computo. La máquina virtual organiza la memoria necesaria para ejecutar un programa en varias áreas de datos de ejecución.

6.3. ENTORNOS DE COMPILACIÓN Y EJECUCIÓN

Un entorno de desarrollo de lenguaje Java incluye tanto un ambiente de compilación y el ambiente de ejecución⁵⁴, como lo ilustra la Figura 18. El

⁵⁴ KRAMER Douglas, The Java™ Platform A White Paper. Java Compile and Runtime Environments. 1996. Link: <http://java.sun.com/docs/white/platform/javaplatform.doc3.html>

programador escribe el código fuente, generando archivos .java y estos posteriormente se compilan en bytecodes creando un archivo .class, los cuales son las instrucciones para la máquina virtual, o también puede ser visto análogamente como, el lenguaje máquina para la JVM.

El transporte de los bytecodes se puede presentar tanto de manera local, o a través de una red. Una vez transportados los bytecodes, estos se cargan en memoria y luego son verificados por seguridad, antes de entrar en la JVM.

Ya en la JVM, los bytecodes son interpretados por el *Java Interpreter*, u opcionalmente se convierte en código máquina a través del generador de código *Just-in-Time (JIT)*, más conocido como el compilador JIT. El intérprete y el compilador JIT operan en el contexto del sistema de ejecución (hilos, memoria, y otros recursos del sistema), que actúa sobre el sistema operativo, que a su vez está en un nivel superior al hardware. Algunas clases de las bibliotecas de clases de Java (API) se cargan dinámicamente según sea necesario por la aplicación.

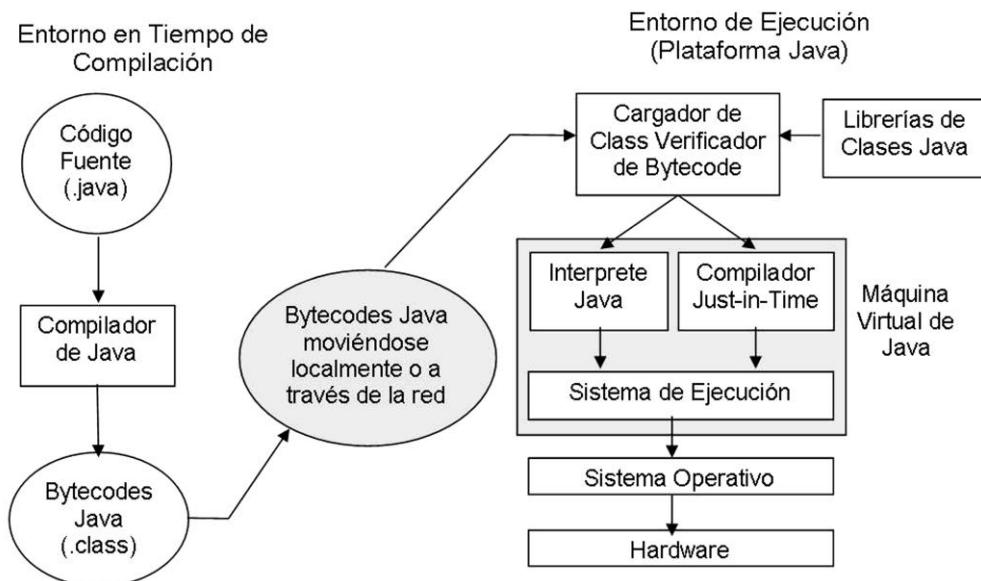


Figura 18. Secuencia de Compilación y Ejecución⁵⁵

6.4. COMPILADOR DE JAVA

Se trata de una herramienta de desarrollo llamada javac incluida en el JDK (*Java Development Kit*), siendo este un conjunto de programas que permiten

desarrollar, compilar y ejecutar. El JDK es distribuido actualmente de manera gratuita a través de *Oracle Corporation*⁵⁵.

6.4.1. Funcionamiento del Compilador

El comportamiento del compilador de Java, básicamente consiste en escanear recursivamente los directorios fuente y de destino, en busca de archivos fuente de Java (.java) para compilar. Sólo aquellos archivos .java que no tienen correspondencia con un archivo .class o cuando el archivo .class es más antiguo que el archivo .java, serán compilados. Ahora, si los archivos fuente hacen parte de un paquete, la estructura de directorios del árbol de fuentes, debería seguir la jerarquía de paquetes.⁵⁶

El compilador de Java es una herramienta que convierte código fuente en archivos .class (bytecode), para su ejecución por el intérprete de Java como se muestra en la figura 19. El compilador lee cada línea del código fuente y se asegura de que se hayan cumplido todas las reglas (sintácticas y/o semánticas)⁵⁷.

Todos los programas escritos en Java utilizan clases que están definidas fuera del archivo de código fuente. En muchas ocasiones estas clases externas se encuentran en la biblioteca de Java conocida como la API. Sin embargo, estas clases también pueden ser aquellas que se han desarrollado previamente, y se han reutilizado en el programa. En estas circunstancias, el compilador debe ser capaz de localizar estas clases externas, con el fin de determinar cómo compilar el código que hace referencias a ellas.

Todas las declaraciones de Java, tal como clases, interfaces, y excepciones, son organizadas en unidades lógicas llamadas paquetes. Una clase o interface deben ser identificadas como públicas para ser usadas fuera de su

⁵⁵ Aprenda Java como si estuviera en primero. Tecnun. Campus Tecnológico de la Universidad de Navarra. Escuela Superior de Ingenieros. SAN Sebastian 2000. Pág 3.

<http://www.tecnun.es/asignaturas/Informat1/Ayudalnf/aprendainf/Java/Java2.pdf>

⁵⁶ The Apache Ant Project. Apache Ant™ 1.8.2 Manual. Description Javac.

<http://ant.apache.org/manual/Tasks/javac.html>

⁵⁷ GREANIER, Todd. *Java Foundations*. Editorial John Wiley & Sons. ISBN 9780782143737. Pág 24.

paquete. Solo aquellas clases o interfaces públicas pueden ser referenciadas en un archivo fuente determinado.

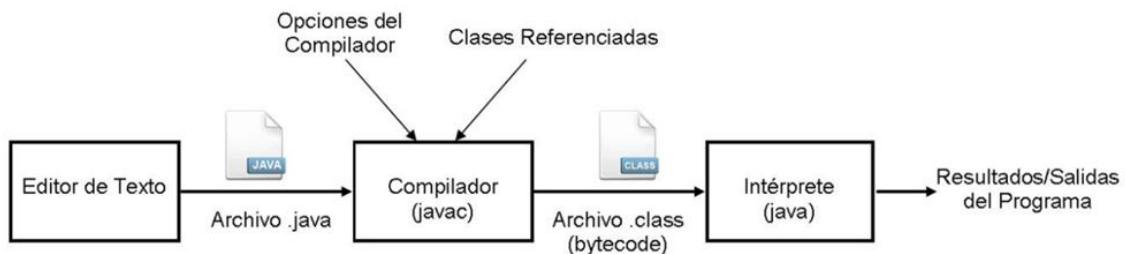


Figura 19. Funcionamiento del Compilador de Java.

Una vez el compilador identifica la ruta en la cual se encuentran los archivos fuente, este analiza sus contenidos en busca de posibles errores. Si este análisis arroja errores, no crea el archivo .class, y notifica el resultado del análisis, describiendo el tipo del error, la línea donde se cometió, la posición del carácter donde está el error en la línea y en algunos casos da unas sugerencias de solución de acuerdo al tipo. Ahora, si el análisis fue satisfactorio, el compilador creará los archivos .class que contiene los bytecode, correspondientes a los archivos fuentes analizados.

Por conveniencia, el compilador nombra los archivos con extensión *.class de acuerdo al nombre de los archivos fuente, por ejemplo, si el archivo fuente fue nombrado Test.java, el compilador analizará su contenido en busca de errores, y si no los encuentra, nombrará de la misma manera al archivo que contiene los bytecode así: Test.class.⁵⁸

Existen ciertas consideraciones acerca del proceso de compilación, como por ejemplo: Javac permite una clase pública por archivo e insiste que el archivo tenga el mismo nombre que el de la clase declarada como pública; esto es así siempre y cuando en un mismo archivo fuente, exista la declaración de más de una clase, pero cuidado, no es recomendable que existan demasiadas clases en un mismo archivo.⁵⁹

En el caso de que en un archivo existan más de una definición de una clase, el javac analiza verificando que el archivo sólo contenga una clase pública, y seguirá analizando en busca de otros errores. Cuando javac termina el

⁵⁸ NIEMEYER, Patrick; KNUDSEN Jonathan. Learning Java. Edición 3. Editorial O'Reilly Media Inc. 2005. ISBN 9780596008734. Capítulo 3. Pág. 72

análisis y ha sido satisfactorio, este crea tantos archivos .class con el nombre correspondiente a las clases definidas en el archivo analizado.

6.5. ARCHIVOS DE FORMATO CLASS

Un archivo class es precisamente un formato de archivo binario definido para programas hechos en Java, representando cada uno de estos, una descripción completa de una clase o interface.⁵⁹ Las clases internas y las clases que no son públicas también serán compiladas en su propio archivo class.

6.2.1. Estructura General de Archivo .class

Un archivo class es una secuencia binaria de bytes de 8 bits, por lo tanto todas las cantidades de, 32 y 64 bit son construidas por la lectura de dos, cuatro, y ocho bytes de 8 bits consecutivos respectivamente. Los elementos de datos son almacenados secuencialmente en el archivo class sin espacio entre elementos adyacentes, y aquellos que ocupan más de un byte, se dividen en varios bytes consecutivos en orden *big-endian* (El más alto en bytes primero).^{60,60} Los archivos class son principalmente divididos en cuatro partes: bytes iniciales, pool de constantes, banderas & “punteros”, y las propiedades de la clase, como lo muestra la figura 20.

<i>0xcafebabe</i>		<i>version</i>	
<i>constant pool</i>			
<i>accessflags</i>	<i>this</i>	<i>super</i>	<i>interfaces</i>
<i>fields</i>			
<i>methods</i>			
<i>attributes</i>			

Figura 20. Estructura General de un Archivo .class⁶¹

⁵⁹ VENNERS Bill. *Inside the Java Virtual Machine*. McGraw-Hill. Chapter 6 – The Java Class File. ISBN 0079132480

⁶⁰ *The Java Virtual Machine Specification*. LINDHOLM Tim, YELLIN Frank. Second Edition. Chapter 4- The Class File Format.

⁶¹ Barte Van Rompaey. *Java and .NET: A Look Into Today's Virtual Machine Technology*. Universiteit Antwerpen. Departement Wiskunde en Informatica. Chapter 5 – File Formats. 2003-2004.

- **0xcafebabe:** Los primeros 4 bytes de un archivo class siempre son 0xcafebabe. Estos bytes representan o identifican un archivo class y es conocido como el número mágico. Si un archivo no se inicia con 0xcafebabe, definitivamente no es un archivo class.⁶⁰
- **Version:** Los segundos 4 bytes contienen de versión mayor y menor, y es así porque la tecnología Java está en constante evolución, de manera que las nuevas características se pueden agregar al archivo class.⁶⁰
- **El *constant pool*:** contiene una lista de constantes: nombre de clases, métodos, variables, y también los nombres de los atributos presentes en partes del archivo class. Cada entrada consiste en una etiqueta, que define el tipo de entrada, el tamaño y finalmente el contenido mismo. Ejemplo de estas etiquetas son: Class, Fieldref, Integer, Utf8,....⁶²
- La siguiente parte contiene los modificadores de acceso de las clases (abstract, static, public, etc..) y los campos *this*, *superclass* e *interfaces* que apuntan en el *constant pool*, las constantes que contienen el nombre de la clase de herencia (super), el nombre de la propia clase (*this*), y los nombres de las interfaces que la clase implementa.⁶²
- Lo siguiente es la información de los campos y métodos de la clase, y además ambas tienen una lista de atributos. Un atributo es una estructura adicional de longitud variable perteneciente al método o miembro. Un método tienen un atributo de código, en el que lista las instrucciones para que el método se almacene. Otros ejemplos de atributos son el atributo de excepción, indicando cuales excepciones un método podría lanzar.⁶²

6.6. API DE JAVA

API – *Application Programming Interface* es una gran colección de componentes de software listos para usar que ofrecen muchas capacidades útiles. Se agrupan en librerías de clases e interfaces, que se conocen como paquetes⁶². Cuando el programador escribe su aplicación, este asume que todas los archivos class de la API, estarán disponibles en JVM, de manera

⁶² API - Java Application Programming Interface- definition-
<http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html>

que siempre tenga los privilegios para ejecutar su programa; este supuesto es válido puesto que la JVM y la API son los componentes necesarios de cualquier implementación de la plataforma de Java.

La API de Java cuenta con una serie de paquetes que agrupan ciertas funcionales similares, ofreciendo al programador una visión más estructurada de lo que puede reutilizar al momento de implementar alguna solución a un problema determinado.

La Tabla 5, muestra la forma como está establecida la API a través de niveles. Estas librerías en sus diferentes versiones presentarán algunos refinamientos y/o adiciones de nuevos elementos.

API de Java (<i>Java Platform Standard Edition 7</i>)	
Nivel	Paquetes
Kit de Herramienta para Interfaces de Usuario	<i>AWT</i> <i>Swing</i> <i>Java 2D</i> <i>Accessibility</i> <i>Drag and Drop</i> <i>Input Methods</i> <i>Image I/O</i> <i>Print Service</i> <i>Sound</i>
Librerías de Integración	<i>IDL</i> <i>JDBC</i> <i>JNDI</i> <i>RMI</i> <i>RMI-IIOP</i> <i>Scripting</i>
Otras Librerías Base	<i>Beans</i> <i>Internationalization Support</i> <i>Input/Output</i> <i>JMX</i> <i>JNI</i> <i>Math</i> <i>Networking</i> <i>Override Mechanism</i> <i>Security</i> <i>Serialization</i> <i>Extension Mechanism</i>

XML JAXP	
Librerías de Utilidades (<i>lang</i> y <i>util</i>)	<i>lang and util</i> <i>Collections</i> <i>Concurrency Utilities</i> <i>JAR</i> <i>Logging</i> <i>Management</i> <i>Preferences API</i> <i>Reference Objects</i> <i>Reflection</i> <i>Regular Expressions</i> <i>Versioning</i> <i>Zip</i> <i>Instrumentation</i>

Tabla 5. Ilustración de los niveles que componen la API de Java⁶³

A continuación se describirán cada uno de los paquetes por cada nivel, con el fin de proporcionar una idea global de las funcionales que cada uno cumple.

Kit de Herramienta para Interfaces de Usuario	
Paquete	Descripción
AWT	El <i>AbstractWindowToolkit</i> soporta la programación de interfaces de Usuario (Eventos, gráficos, imágenes, formas, color, fuentes, distribución de espacios, entre otras) a través de un conjunto de componentes nativos.
Swing	Al igual que la AWT, implementa un conjunto de componentes para la construcción de interfaces gráficas de usuario, adicionando funcionalidad gráfica sofisticada e interactiva a las aplicaciones. Está totalmente implementado en Java.
Java 2D	Es un conjunto de clases para el manejo de imágenes y gráficos avanzados en 2D.
Accessibility	Es una librería destinada para facilitar el desarrollo de aplicaciones orientadas a personas con discapacidad. Son compatibles con lectores de pantalla, sistemas de reconocimiento de voz y visualizadores braille.

⁶³ Oracle Corporation. *Java Platform Standard Edition 7 Documentation*.
Link: <http://download.oracle.com/javase/7/docs/>

<i>Drag and Drop</i>	Permite la transferencia de información entre aplicaciones Java y aplicaciones nativas, también entre aplicaciones Java o en la misma aplicación.
<i>Input Methods</i>	Proporcionan una colaboración entre los componentes de edición de texto y los métodos de entrada para el ingreso de texto. Esta librería es útil cuando estas entradas son a través de caracteres y/o dispositivos de ingreso de texto no convencionales, por ejemplo, caracteres en chino, japonés o coreano.
<i>Image I/O</i>	Ofrece una arquitectura para trabajar con imágenes almacenadas en archivos y accedidas a través de la red.
<i>Print Service</i>	Está diseñada para soportar la impresión en todas las plataformas Java, incluyendo las plataformas que requieren espacios reducidos, como en un perfil de J2ME.
<i>Sound</i>	Potente API para capturar, procesar, y reproducir datos de audio y MIDI (<i>Musical Instrumental Digital Interface</i>). Es soportada por un eficiente motor de sonido que garantiza una alta calidad de mezcla de audio y capacidad de síntesis de MIDI para la plataforma.

Tabla 6. Descripción del Nivel de Kit de Herramientas de Interfaces de Usuario⁶³.

Librerías de Integración	
Paquete	Descripción
IDL	Es una tecnología para objetos distribuidos, es decir, objetos interactuando sobre diferentes plataformas a través de una red. Permite interactuar a los objetos sin importar si están escritos en Java o en cualquier otro lenguaje orientado a Objetos.
JDBC	<i>Java Database Connectivity</i> , brinda acceso virtualmente a cualquier fuente de datos, desde base de datos relacionales, hojas de cálculo y archivos planos.
JNDI	<i>Java Naming and Directory Interface</i> proporciona la funcionalidad de asignación de nombres y directorios a las aplicaciones. Está diseñado para ser independiente de cualquier especificación de nombres o implementación de servicios de directorios.
RMI	<i>Java Remote Method Invocation</i> permite el desarrollo de

	aplicaciones mediante el establecimiento de la comunicación remota entre programas escritos en Java. La diferencia en comparación con IDL, es que en RMI solo puede interactuar con objetos distribuidos escritos enteramente en Java.
RMI-IIOP	Permite la Programación de Servidores y aplicaciones CORBA (<i>Common Object Request Broker Architecture</i> la cual es una arquitectura estándar para sistemas de objetos distribuidos).
Scripting	Esta librería permite que implementaciones de scripting y tipos de lenguajes que generan bytecodes, puedan ser ejecutado en la plataforma de Java.

Tabla 7. Descripción del nivel Librerías de Integración⁶³.

Otras Librerías Base	
Paquete	Descripción
<i>Beans</i>	Contiene clases relacionadas con el desarrollo de beans (componente software que se puede reutilizar y manipular por una herramienta de programación de lenguaje Java), que son componentes basados en arquitectura JavaBeans.
<i>Internationalization Support</i>	Permite el desarrollo de aplicaciones internacionalizadas, de manera que pueda ser adaptado a distintos idiomas y regiones sin que requiera cambios de ingeniería.
<i>Input/Output</i>	Son las librerías que incluyen las siguientes características: Entrada y salida a través de flujo de datos, serialización y el sistema de archivos. Juego de caracteres, decodificadores y codificadores, para traducir bytes y caracteres Unicode. Acceso a archivos, a sus atributos y a los archivos del sistema. APIs para la creación de servidores escalables.
JMX	<i>Java Management Extensions</i> es una API estándar para gestionar y monitorear los recursos, de las aplicaciones, dispositivos, servicios y la máquina virtual de Java. Los usos típicos para la tecnología JMX

	incluye, la configuración de consultas y cambios de la aplicación, acumulación de estadísticas sobre el comportamiento de las aplicaciones y ponerlas a disposición, la notificación de cambios de estado y las condiciones erróneas.
JNI	<i>Java Native Interface</i> , es una interface estándar de programación para escribir métodos nativos y máquina virtuales embebidas en aplicaciones nativas. El objetivo principal, es la compatibilidad binaria de las librerías de los métodos nativos en todas las implementaciones de máquina virtual en una plataforma determinada.
Math	Corresponde a la librería matemática, y permiten el manejo de funciones y constantes matemáticas, desde la trigonometría básica hasta el soporte aritmético de precisión arbitrario.
Networking	Esta librería proporciona clases para manejar la funcionalidad de red, incluyendo direccionamiento para usar URLs y URIs, clases socket para la conexión a los servidores, la funcionalidad de la seguridad de las redes y mucho más.
Override Mechanism	Ofrecen un medio para que las versiones posteriores de clases e interfaces, que implementan estándares aprobados puedan ser incorporados en la plataforma Java.
Security	Incluye un amplio conjunto de APIs, herramientas, e implementaciones de algoritmos de seguridad de uso común, mecanismos y protocolos. La API de seguridad, abarca una amplia gama de áreas, incluyendo criptografía, infraestructura de clave pública, comunicaciones seguras, autenticación y control de acceso.
Serialization	Admite la codificación de los objetos y los objetos accesibles desde ellos, en un flujo de bytes. <i>Serialization</i> es usada para la persistencia ligera y para la comunicación a través de sockets o <i>Java Remote Methods Invocation</i> (RMI).
Extension Mechanism	Usada para soporte de paquetes opcionales. Permite a la máquina virtual (JVM), utilizar clases de extensión

	opcional de la misma manera como la JVM, utiliza las clases de la plataforma Java. También proporciona una forma para la necesidad de paquetes opcionales que se recuperan de la URL cuando estos no están instalados en el JDK o JRE.
XML JAXP	Proporciona un rico conjunto de APIs para el procesamiento de documentos XML y datos.

Tabla 8. Descripción del Nivel Otras Librerías Base⁶³.

Librerías Base de Utilidades (lang y util)	
Paquete	Descripción
<i>lang and util</i>	Estos paquetes proporcionan la funcionalidad básica usada por casi todas las aplicaciones. El paquete <i>lang</i> provee de clases que son fundamentales para la programación, tal como <i>String</i> , <i>Math</i> , y soporte básico de ejecución para hilos y procesos. El paquete <i>util</i> ofrece una API para colecciones, formateo de impresión y escaneo, utilidades de manipulación de array, modelo de evento, facilidades de tiempo y fecha, internacionalización, y diversas clases de utilidades.
<i>Collections</i>	Es una arquitectura unificada para la representación y la manipulación de las estructura de datos, lo que les permite manipular independientemente de los detalles de su representación. Reduce el esfuerzo de programación, mientras aumenta el rendimiento fomentando la reutilización.
<i>Concurrency Utilities</i>	Proveen una poderosa y extensible base de utilidades de <i>threading</i> de alto rendimiento. Este paquete libera al programador de la necesidad de crear estos servicios, ya que estas librerías están hechas y disponibles para la reutilización.
JAR	Provee de clases para la lectura y escritura de archivos en formato JAR, ya sea de solo compresión de paquetes y clases o también como ejecutable.
<i>Logging</i>	Facilitan el servicio y mantenimiento de software en sitios de clientes, mediante la producción de informes de registro adecuado para el análisis por parte de los usuarios finales, administradores, ingenieros de servicio y

	los equipos de desarrollo de software. Esta API captura información como fallos de seguridad, errores de configuración, cuellos de botella y/o errores en la aplicación o en la plataforma.
<i>Management</i>	Esta librería ofrece una interface para el monitoreo y gestión de la Máquina Virtual. Por ejemplo, número de clases corriendo e hilos en ejecución, tiempo de funcionamiento, propiedades del sistema, estado de subprocessos, estadísticas, etc.
<i>Preferences API</i>	Este paquete proporciona una forma para que las aplicaciones almacenen y recuperen las preferencias de los usuarios y del sistema, además de los datos de configuración.
<i>Reference Objects</i>	Permite mantener la referencia de algún otro objeto, de tal manera que este último pueda ser reclamado por la <i>Garbage Collection</i> .
<i>Reflection</i>	Permite descubrir información del código fuente, acerca de métodos, campo y constructores, de las clases cargadas y usan esa información reflejada, para operar en base a sus homólogos, dentro de las restricciones de seguridad.
<i>Regular Expressions</i>	Son clases para hacer coincidir la secuencia de caracteres con los patrones especificados por una expresión regular.
<i>Versioning</i>	Permite a nivel de paquete, el control de versiones para que las aplicaciones y los applets puedan identificar la versión en tiempo de ejecución de JRE (<i>Java Runtime Environment</i>), Máquinas Virtuales y paquete de clases.
<i>ZIP</i>	Proporciona clases para leer y escribir el estándar para archivos en formato ZIP y GZIP. También incluye clases para la compresión y descompresión de datos utilizando el algoritmo de compresión DEFLATE, que es utilizado por estos formatos.
<i>Instrumentation</i>	Provee herramientas para aplicaciones instrumentales de Java - por ejemplo, para controlar o recopilar información de rendimiento. Herramienta que usan este paquete para modificar el archivo class- generalmente, mediante la inserción en el código de bytes de los métodos más el código de bytes se llevará a cabo la instrumentación.

Tabla 9. Descripción del Nivel Librerías Base de Utilidades⁶³.

6.8 AMBIENTES DE EJECUCIÓN PARA PROGRAMAS JAVA

En el apartado 6.3 de las generalidades de Java se describe la manera en la cual se lleva a cabo el proceso de ejecución de un programa escrito en Java, al nivel de su plataforma. Ahora, es necesario describir algunas alternativas de cómo se puede lanzar una ejecución de un programa, ya sea local o remotamente.

La JDK de Java ofrece dos utilidades básicas para permitir ejecutar un programa, estas son: *javac* y *java*. Este proceso se puede llevar a cabo, a través de una consola usando comandos establecidos, donde primero es ejecutado *javac*, que compila y crea un archivo .class, que luego es usado como entrada para la ejecución usando *java*.

6.8.1. Ejecución Local

Un programa Java puede ejecutarse localmente a través de una consola, donde el programador, previamente ha configurado algunas variables de entorno del sistema, para que se pueda compilar e interpretar el programa. Para esto se deben usar las dos aplicaciones mencionadas anteriormente (*javac*, *java*), o a través de un programa que efectúe dicha operación, por ejemplo, una IDE, en donde allí el programador escriba su programa, lo compile y lo ejecute sin salir del contexto de la aplicación.

6.8.2. Ejecución Remota

En este contexto, el código fuente está alojado en un computador en una ubicación diferente con respecto aquel equipo que desea ejecutarlo, de manera que debe encontrar una forma de alcanzar dicho programa para llevar a cabo esta operación. A continuación se presentarán 4 maneras de lograr ejecutar un programa Java remotamente:

- **Escritorio Remoto**

En la web existe una variedad de software que realizan conexiones remotas con otros computadores, los cuales le permiten al usuario interactuar con el equipo como si lo estuviera haciendo físicamente en este. Cuando el programador logra tener acceso remoto, podrá realizar la operación como lo haría localmente.

- **Protocolo Secure Shell (SSH)**

SSH es un protocolo de acceso remoto seguro y de servicios de red seguros a través de una red insegura. Está constituido por tres componentes principales:⁶⁴

- SSH-TRANS: Protocolo de capa de transporte que proporciona autenticación del servidor, confidencialidad e integridad.
- SSH-USERAUTH: Protocolo de autenticación de usuario en el servidor, que corre sobre la capa de transporte.
- SSH-CONNECT: Protocolo de conexión que multiplexa un túnel encriptado en varios canales lógicos. Este corre sobre la capa de autenticación.

El cliente envía una solicitud de servicio una vez la capa de transporte ha establecido una conexión segura. Una segunda petición es enviada después de que la autenticación del usuario ha sido completada. El protocolo de conexión proporciona los canales que pueden ser usados para un amplio rango de propósitos.

A través del protocolo SSH se pueden ejecutar programas escritos en Java que estén alojados en un computador que actuará de servidor de manera remota. El programador puede conectarse remotamente a través de una consola SSH y ejecutar el programa de forma manual, es decir, que ejecute las utilidades del JDK para compilar y ejecutar el programa o creando un proceso que ejecute el programa automáticamente a través del envío de parámetros.

⁶⁴ T. Ylonen, C. Lonwick. *The Seccure Shell (SSH) Protocol Architecture. Introduction.* <http://www.ietf.org/rfc/rfc4251.txt>

- **JNLP Java Network Launching Protocol**

JNLP permite que una aplicación Java sea lanzada en el escritorio de un cliente mediante el uso de recursos que están alojados en un servidor web remoto⁶⁵. El ambiente de la aplicación lanzada a través de JNLP está definido en un conjunto común de servicios y configuraciones del sistema para que sea confiable, es decir, que garantice que su ejecución no contiene secuencias maliciosas⁶⁶.

JNLP es usado por un software llamado *Java Web Start*, el cual permite descargar y ejecutar aplicaciones Java desde la Web a través de una simple activación de un clic, garantizando siempre la ejecución en su última versión y eliminando complejos procedimientos de instalación y actualización.

- **Applets Java**

Un Applet es un programa escrito en Java que puede ser incluido en una página HTML. Cuando se usa un navegador que soporte la tecnología Java, podrá verse el applet, de manera que el código se transfiere al sistema del cliente y se ejecuta en la máquina virtual (JVM) del navegador⁶⁷.

Un inconveniente que está estrechamente ligado a la naturaleza de la web, es que no se conoce el software que usa el usuario, de manera que no garantiza que la versión del intérprete de Java que tenga el usuario sea la misma con la cual el applet se ejecutó satisfactoriamente, e incluso no se puede determinar con seguridad si es posible ejecutar subprogramas de Java.⁶⁸

En el contexto de los applets, es muy recomendado usar archivos JAR, ya que reducen notablemente la carga, al ser archivos

⁶⁵ Oracle Corporation. *The Java Tutorials. Java Network Launch Protocol*.

<http://docs.oracle.com/javase/tutorial/deployment/deploymentInDepth/jnlp.html>

⁶⁶ Sun Java Software. *Java Network Launching Protocol & API Specification (JSR-56). Application Environment*. <http://jcp.org/aboutJava/communityprocess/first/jsr056/jnlp-1.0-proposed-final-draft.pdf>

⁶⁷ Oracle Corporation. *Code Samples and Apps. Applets*. <http://java.sun.com/applets/>

⁶⁸ Jorge Sánchez. Java 2 incluye Swing, Threads, programación en red, JavaBeans, JDBC y JSP/Servlets. Applets. <http://www.jorgesanchez.net/programacion/manuales/Java.pdf>

comprimidos. Los archivos JAR están basados en el formato de archivo ZIP, pero se diferencia por contener un directorio opcional llamado META-INF. Los archivos JAR no solo son usados para almacenar clases y otros recursos, también son usados como bloques de construcción de aplicaciones y extensiones, usando el directorio META-INF, para almacenar datos de configuración sobre los paquetes, las extensiones, seguridad, versiones y servicios.⁶⁹

El modelo de seguridad entre los applets ha sido diseñado con el objetivo de proteger al usuario de applet maliciosos. Los applets que no son firmados con un certificado de seguridad se consideran de no confianza, y cuando son firmados, los applets operan dentro de un “recinto” de seguridad que sólo les permite un conjunto de operaciones seguras.⁷⁰

La tabla 10 muestra las operaciones permitidas y no permitidas para un applet que no está firmado con un certificado de seguridad:

Permitido	No Permitido
<ul style="list-style-type: none"> • Hacer conexiones de red para el host de donde provienen. • Ver documentos HTML. • Invocar métodos públicos de otros applets. • Aquellos applets que son cargados desde el sistema de archivos local no tienen ninguna de las restricciones que se presentan en los applets que son cargados en una red. • Pueden leer las propiedades de seguridad del sistema. • Cuando se lanzan usando el Protocolo JNLP (<i>Java Network Launching Protocol</i>) pueden realizar lo siguiente: <ul style="list-style-type: none"> ○ Abrir, leer y guardar archivos en el cliente. ○ Acceso compartido a todo el sistema de portapapeles. ○ Acceso a funciones de impresión. 	<ul style="list-style-type: none"> • No tienen acceso a recursos del cliente, como sistema de archivos local, archivos ejecutables, portapapeles e impresoras. • No pueden conectarse o recuperar los recursos de cualquier servidor de terceros. • No pueden cargar librerías nativas. • No pueden cambiar el administrador de seguridad. • No pueden crear un <i>ClassLoader</i>. • No puede leer ciertas propiedades del sistema.

Tabla 10. Qué puede y qué no puede hacer un Applet no Firmado⁷⁰.

⁶⁹ Oracle Corporation. *JAR File Specification. Introduction*.

<http://docs.oracle.com/javase/1.3/docs/guide/jar/jar.html#Intro>

⁷⁰ Oracle Corporation. *What Applets Can and Cannot DO*.

<http://docs.oracle.com/javase/tutorial/deployment/applet/security.html>

6.8.2.1. Applets Firmados

Los navegadores a través de un programa incluido en ellos, llamado *Security Manager*, controlan las operaciones de los applets. Si un Applet no quiere tener restricciones como las mencionadas en la tabla anterior, debe ser firmado. Esto consiste que a través de una firma digital, donde se autentique el autor, pueda brindarle al usuario confianza de que el applet que se cargará no es peligroso, de manera que si el cliente que confíe en ese autor aceptará el certificado y dará los correspondientes privilegios; de otra manera el applet seguirá comportándose como no firmado.

6.8.2.2. Proceso de firmado de un applet

Para firmar un applet se debe tener en cuenta dos utilidades dispuestas en el JDK de Java llamadas **keytool** y **jarsigner**, que permiten manejar toda una base de datos de identidades, exportando certificados y firmar archivos JAR respectivamente. A continuación se presenta a través de un ejemplo la manera cómo se debe firmar un applet.

Un equipo desarrollador de la Universidad Francisco de Paula Santander, implementa un Applet de Prueba, el cual incluye un archivo JAR ejecutable que utiliza componentes gráficos de la librería Swing de la API de Java. Con el propósito de que este programa se ejecute en el cliente, necesariamente debe hacer uso de la Máquina Virtual del cliente, por lo tanto el applet debe ser firmado para que la aplicación tenga privilegios de acceso a ella. Para llevar a cabo esta operación, ellos deben firmar el applet certificándole al usuario que puede confiar en la fuente. Para esto se debe seguir los siguientes pasos:⁷¹

- 1. Compilar el Applet**, usando el comando `javac` de la siguiente manera:

⁷¹ Oracle Corporation. Chapter 10: Signed Applets.

<http://java.sun.com/developer/onlineTraining/Programming/JDCBook/signed.html>

```
javac AppletDemoUFPS.java
```

2. **Construir el archivo JAR:** El anterior paso produce un archivo con extensión class, con el mismo nombre así: *AppletDemoUFPS.class*. Ahora con este archivo se construirá el JAR con los siguiente comando y opciones:

```
jar cvf AppletDemoUFPS.jar AppletDemoUFPS.class
```

3. **Generar Claves:** El archivo generado (*AppletDemoUFPS.jar*) es firmado con una clave privada por parte del creador, y es verificada por el usuario con el par, que es una clave pública. Estas claves deben existir en la base de datos del almacén de claves (*keystore*), antes de firmar o verificar el JAR. Entonces el equipo crea la base de datos *keystore*, que tiene una entrada por cada par de llave privada y pública generada, con la llave pública en un certificado usando el comando *keytool*, así:

```
keytool -genkey -alias UFPS -validity 120 -v
```

keytool= comando

-genkey= comando de invocación para generar los pares de claves.

-alias= es el alias para las claves

-validity= días de validez del certificado, en este caso 120 días.

A partir de aquí *keytool* hará una serie de preguntas que posteriormente irán en el certificado que el usuario podrá ver, y generará un archivo con extensión *.keystore*:

- Clave para el almacén.
- Nombre y apellidos.
- Nombre del departamento de la empresa.
- Nombre de la empresa.
- Localidad.
- Provincia.
- Código del país de dos letras.

- Pregunta si los datos ingresados son correctos.

4. Firmar el archivo JAR

Una vez creado el certificado se debe firmar el JAR y para ello se utiliza la utilidad jarsigner del JDK de Java, de la siguiente manera:

```
jarsigner AppletDemoUFPS.jar UFPS –verbose
```

Luego nos pide la clave del almacén. Ya terminado todos los anteriores pasos, cuando el navegador intente cargar el applet en la página, detectará que está firmado y mostrará un aviso el cual es el certificado, en donde el usuario verá el autor y demás información suministrada cuando se creó el almacén.

7. RESULTADOS OBTENIDOS

El estudio que soporta el proyecto, permitió llevar a cabo el cumplimiento de los primeros tres objetivos, que hacen parte de los seis definidos en la sección 1.4.2 del presente documento. Estos tres objetivos, hacen referencia a la parte investigativa y de trabajo de campo del proyecto; los cuatro objetivos restantes, se encuentran estrechamente ligados con los procesos de construcción, pruebas y despliegue de software.

A partir de lo obtenido en las secciones 4 y 5, correspondiente al estado del arte y trabajo de campo respectivamente, se identificaron las características básicas y avanzadas de las IDE's con las que los estudiantes están más familiarizados, partiendo del resultado obtenido en la tabulación de la primera pregunta de la encuesta aplicada, el cual determinó que el 100% y 71,15% de ellos están familiarizados con Netbeans y Eclipse respectivamente. (La población de estudiantes fue designada en aquellos matriculados a los cursos de Fundamentos de Programación, Programación Orientada a Objetos y Estructura de Datos, pertenecientes al Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander en el segundo semestre del año 2011). De acuerdo a este resultado, se muestran y se categorizan las funciones básicas y avanzadas de Netbeans y Eclipse, obtenidas de la matriz de características de las tablas 1 y 2 de la sección 4.2.3.

Características Determinadas por la Investigación	
Básicas	Avanzadas
<ul style="list-style-type: none">• Compilador.• Executer-linker.• Constructor del JAR.• Editor<ul style="list-style-type: none">◦ Editor de Texto◦ Editor de Márgenes<ul style="list-style-type: none">▪ Número de Línea.▪ Marcadores▪ Bloques Desplegables.▪ Marcador de Tabulador.▪ Final de Línea.◦ Resaltador de Sintaxis◦ Auto-formato de código fuente.◦ Sugerencias de finalización de código.• Manejador de Proyectos.• Ayudas de Contenido.• Consolas de Salida (compilación,	<ul style="list-style-type: none">• Análisis de Rendimiento.• Control de Versiones.• Constructor de GUI.• Herramientas de Modelado.• Desarrollo Web.• Administrar Complementos.• Importación de proyectos de otras IDE's.• Visualización de clases de la API.• Generador de Código.• Navegador.• Generador de JavaDoc.• Gestor de Pruebas (JUnit)• Gestión de Librerías Externas.• Depurador.• Disponibilidad para múltiples

ejecución, excepciones y advertencias)	plataformas. <ul style="list-style-type: none"> • Soporte para varios lenguajes de programación.
--	--

Tabla 11. Resultado de las Características básicas y avanzadas de las IDE's

Ahora bien, teniendo en cuenta este resultado, es necesario, analizar y filtrar algunas de ellas, con el fin de obtener una funcionalidad práctica, para la implementación de la IDE online. Esta funcionalidad básica, permitirá crear una visión global acerca del comportamiento que la IDE podrá brindar a los usuarios.

Para alcanzar este objetivo además de tener en cuenta el anterior resultado, es necesario enfatizar en el análisis del resultado del estudio de las IDE's seleccionadas, (sección 4.3), además de ciertos factores importantes encontrados en la aplicación de la encuesta a los estudiantes, de lo cual se obtiene el siguiente resultado:

Características Básicas determinadas para la IDE
<ul style="list-style-type: none"> • Compilador. • Executer-linker. • Constructor (.jar) • Editor de texto. • Resaltador de Sintaxis. • Número de Línea. • Multiplataforma. • Auto-formato de código fuente. • Manejador de Proyectos. • Consola de Compilación. • Consola de Ejecución. • Consola de Excepciones.

Tabla 12. Características Básicas determinadas para la IDE

Estas características fueron seleccionadas respondiendo a la siguiente pregunta: ¿Cuáles deben ser las funciones más necesarias para escribir, compilar y ejecutar aplicaciones Java?, además, se concluye que las funcionales que presentan mayor grado de coincidencia entre las diferentes IDE's, tienden a convertirse en unidades mínimas de funcionalidad. Por lo tanto, un porcentaje mayor o igual a 80% de coincidencia, es determinante para identificar qué funciones deben estar presentes en una IDE de programación. Para la selección se han exceptuado algunas características que pertenecen a este rango, por desconocimiento de la complejidad de su implementación, además de la relevancia e impacto de su presencia.

Un aspecto importante para tener en cuenta es el entorno gráfico en el cual están familiarizados los estudiantes, de manera que un breve análisis comparativo de las distintas interfaces gráficas de las IDE's involucradas, determinó un factor común entre ellas (Figura 15), lo cual es muy conveniente para la usabilidad que pueda tener la aplicación.

La aplicación del instrumento de recopilación de la información (encuesta) reveló algunos índices significativos que deben considerarse en el desarrollo del proyecto. Consideraciones tales como:

- Requisitos de hardware de las IDE's son altos.
- La mayoría de estudiantes cuentan con equipos que se adecuan a esos requisitos de hardware.
- Tiempo de espera largos entre cada operación, por ejemplo compilar, ejecutar, limpiar y construir un archivo JAR, abrir un proyecto, etc...
- Incompatibilidad entre versiones.
- La interfaz gráfica de usuario debe ser limpia, agradable y usable.
- Que el rendimiento de la IDE en equipos considerados de gama baja sea eficiente.
- Compilar y ejecutar son funciones fundamentales para una IDE.
- El Constructor de interfaces gráficas es 65,38% necesario para una IDE.
- Las funciones avanzadas son consideradas necesarias.
- Alto porcentaje de aceptación frente a una IDE online para Java.

Es claro que la IDE debe permitir escribir, compilar y ejecutar aplicaciones Java, por lo tanto, es importante complementar el estudio realizado a las IDE's, estudiando las generalidades de este lenguaje de programación, con el propósito de determinar las librerías, que podrían verse implicadas en los programas desarrollados por el usuario final. En este punto surgían preguntas como:

- ¿Cuál es el procedimiento para compilar una clase y un paquete?
- ¿Cómo ejecuto una clase?
- ¿Cómo funciona el compilador?
- ¿Cómo se construye un archivo JAR?
- ¿Qué contiene la API de Java, y qué podría ser útil de ella?
- ¿Cuál es la arquitectura Java?

La sección 6, contiene una compilación de información obtenidas de varias fuentes, que dan respuestas a las mencionadas inquietudes, ayudando a orientar sobre la forma en que se deberían definir aquellas funcionalidades deseadas para la IDE. Se logró conocer cómo está compuesta la plataforma de Java, la importancia y la arquitectura de su Máquina Virtual, los entornos

de compilación y ejecución, el funcionamiento del compilador, cómo está constituido un archivo .class, y las maneras en que se pueden ejecutar aplicaciones Java.

Gracias al estudio hecho a las generalidades de Java, se logra definir que para las funcionalidades básicas de la IDE (compilar, construir un archivo jar, ejecutar), se podrían utilizar clases disponibles en la API, evitando la ejecución de las utilidades que la JDK ofrece para hacer estas operaciones.

Una conclusión importante respecto a las librerías que puede utilizar el usuario en sus programas hechos en la IDE fue:

- En lugar de restringir el uso de ciertas librerías de la API para evitar sobrecargar el servidor al ejecutar un programa, se podría optar por transferirle esa carga al cliente. Si esto se lograra, no habría inconvenientes con el uso de librerías que el usuario quiera utilizar en sus programas, ya que estas, se encuentran disponibles en la API de la JRE(*Java Runtime Environment*) instalada en el cliente.

Con respecto a la ejecución de aplicaciones Java, era necesario encontrar la manera en la cual se pudiese iniciar la ejecución en la web, de un programa escrito y compilado, por lo cual, se encontraron alternativas de ejecución (sección 6.8.2), tales como:

- Protocolo SSH⁶⁵.
- Protocolo JNLP^{66,67}.
- Applets⁶⁸.

Después de revisar documentación, algunos ejemplos y demos donde aplicaban estos métodos, se determinó la manera en la cual se llevaba a cabo la ejecución en cada una de ellas, lo cual permitió encontrar características y comportamiento que permitirán decidir cuál de ellas implementar para la aplicación. A continuación se describe lo encontrado:

- El uso del protocolo SSH a través de una consola remota en un navegador, presenta dos dificultades muy significativas, a raíz de que la ejecución se hace en el lado del servidor. La primera es que necesariamente se debe restringir librerías de la API instalada en el servidor, para evitar la carga del mismo, para lo cual no se cuenta con los conocimientos necesarios para llevar a cabo esta operación. El segundo inconveniente, parte de que se debe analizar y determinar cuáles librerías debe restringirse, ya que podrían crearse programas que tengan acceso a los recursos del servidor, tomar control remoto del mismo, y crear secuencias maliciosas.

- El protocolo JNLP permite lanzar aplicaciones java por la web, pero no la ejecuta en el navegador. En este protocolo la aplicación se centraliza y se distribuye a través de la web, posteriormente se descarga e instala en el cliente, y a partir de allí se ejecutará como una aplicación normal. Una aplicación que es ejecutada a través de JNLP, tiene restricciones de seguridad en el contexto del cliente. Estas restricciones son establecidas para evitar aplicaciones maliciosas ejecutándose en el cliente. Para que se puedan ejecutar aplicaciones con este protocolo, el cliente debe contar con un software llamado *Java Web Start*, el cual se ejecuta automáticamente y guarda la aplicación localmente, en la memoria caché del equipo.
- Por último los applets pueden lanzar aplicaciones Java para su ejecución en un navegador, de manera que el usuario no tenga que salir del contexto del navegador, para ubicar un archivo descargado para ejecutarlo, tal como lo hace JNLP. Los applets desde luego tienen restricciones de seguridad, definidos por un programa incluido en los navegadores llamado *Security Manager*⁷² que implementa políticas de seguridad, por lo tanto deben firmarse, dándole certificación al usuario de que applet pertenece a una fuente confiable.

Establecidas las desventajas de cada alternativa, se define los applets como la manera adecuada para ejecutar una aplicación a través de la web, debido a que la carga que requiere la ejecución es asumida por el cliente. Además, los applets tienen ventaja con respecto a JNLP, esto es que permite que el usuario se mantenga en el navegador mientras la instancia del programa es dada, sin necesidad de descargar ningún archivo en su equipo, siempre y cuando el navegador que utilice, sea compatible con tecnología Java y cuente con el respectivo plugin que le permita visualizar el applet. Ahora, con el fin de encontrar una solución a las restricciones de seguridad, el estudio hecho a Java permitió conocer la manera de proporcionarle privilegios a un applet sobre el sistema del cliente ([sección 6.8.2.2](#)). Esto consiste básicamente en pedir la autorización al usuario para que el applet tenga acceso a sus recursos, a través de un certificado de autenticidad que garantice que el applet no tiene código malicioso, que pueda poner en peligro su sistema; entonces, la ejecución sólo dependería de los privilegios que el usuario le otorgue al applet que contiene el código ejecutable de su aplicación.

⁷² *SecurityManager*. <http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/SecurityManager.html>

DESARROLLO DEL PROYECTO

8. METODOLOGÍA DE DESARROLLO

El proyecto se desarrolló usando *eXtreme Programming*, XP (Programación Extrema) como metodología, teniendo en cuenta su capacidad de planear de manera flexible la funcionalidad de la aplicación, el manejo y control de cambios que en el ciclo de vida del proyecto se puedan presentar; pero siempre en continua comunicación con el cliente. Es importante mencionar que esta metodología se adaptó a las necesidades del proyecto, ya que este implica la construcción de módulos base que sustentarán la funcionalidad de la IDE, que posteriormente se integrarán a implementaciones de control, y manejo de vistas sofisticadas para emular un entorno *Desktop* en un ambiente *Cloud*.

Teniendo en cuenta que el conocimiento para conseguir ciertas funcionalidades de la IDE resulta difícil prever, de igual forma establecer un diseño que las resuelva, por tal razón, XP se adecua a esta circunstancia, ya que durante las iteraciones se pueden adquirir los conocimientos necesarios, generando un diseño flexible a cambios que brinde una solución a los requerimientos planteados.

Durante el desarrollo del proyecto se obtendrán por cada iteración prototipos funcionales, después de respectivas pruebas de unidad y aceptación, en donde algunos de ellos serán base para el desarrollo de otros, y en paralelo se desarrollarán e integrarán nuevos prototipos que enriquecerán la funcionalidad de la aplicación.

El desarrollo de nuevos prototipos implicará buscar tecnologías que suplan ciertas necesidades que el proyecto requiere, de manera que con ciertos ajustes puedan integrarse aportando características básicas, avanzadas o colaborativas.

Otra consideración por la cual se utilizará XP como metodología es que en comparación a otras, esta es considerada ágil, ya que es más flexible en la determinación de los artefactos a utilizar en la documentación, de tal manera que los diseños sean sencillos durante cada iteración, siempre en busca de la simplicidad, y así tener ideas claras para una implementación eficiente y a tiempo, que es realmente su objetivo. Los equipos de trabajo en XP son pequeños y rotables, en cuanto a roles se refiere, y está pensada para proyectos cortos, en contra parte con otras metodologías, por ejemplo RUP

(*Rational Unified Process*)⁷³, la cual está orientada a proyectos más robustos, basándose mucho en la documentación, teniendo consideraciones tanto organizativas, operativas y del mismo desarrollo, exigiendo roles definidos para el equipo de trabajo que por lo general son grandes.

8.1 EXPLORACIÓN

8.1.1. Identificación De Roles

El desarrollo de este proyecto cuenta con un equipo de trabajo pequeño, otra razón por la cual se escogió a XP como metodología, por la tanto, basándose en ella se describen los roles que intervienen en este proyecto:

Programadores:

El equipo de programadores está compuesto por dos estudiantes de Ingeniería de Sistemas de la Universidad Francisco Paula Santander.

- Carlos Andrés Sepúlveda Sánchez
- Wilson Yesid Rivera Casas

Cliente

El rol Cliente juega un papel importante en el equipo de desarrollo, ya que él participa de manera activa en el proceso.

Para este proyecto el rol cliente no está definido como tal, es decir, el cliente no va a ser una persona física que está disponible en el equipo de desarrollo, aportando las historias de usuario, efectuando pruebas, resolviendo dudas, negociando entregas, etc... Entonces, para contrarrestar la necesidad de obtener esas historias que son vitales para el desarrollo, y todo lo demás, se llevó a cabo una un procesos investigativo, el cual se puede ver en detalle en la fase de estado del arte del proyecto. Esto permitió obtener la información y datos necesarios para organizar y crear las historias de usuario, con la participación de un grupo de personas descritas a continuación, a las cuales se les aplicaron una encuesta:

⁷³ Philippe Kruchten, *The Rational Unified Process An Introduction*, Addison Wesley, 2001

- Estudiantes de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, matriculados en los cursos de Programación Orientada a Objetos, Estructura de Datos y estudiantes repitentes de la asignatura de Fundamentos de Programación del segundo semestre de 2011.

Ahora surge una inquietud: ¿quién se hará cargo de las pruebas de aceptación? El ingeniero Marco Antonio Adarme Jaimes llevará a cabo esta actividad. Es él la persona a la cual se le presentarán las liberaciones de cada iteración, para las respectivas pruebas y también para aclaración de dudas que puedan surgir. En la etapa de planeación, el equipo de trabajo definirá qué historias tienen mayor prioridad y estimarán e tiempo y el esfuerzo que sean necesarios.

Director del Proyecto

Este rol está a cargo del Msc. Marco Antonio Adarme Jaimes, el cual será responsable de orientar y detectar si el proyecto se está desviando de su objetivo general, debe estar al tanto de todo el proceso y proponer alternativas de solución a dificultades que se presenten, entre otras funciones.

Consultor

En este rol están aquellas personas especialistas en ciertas áreas que se ven implicadas en el desarrollo del proyecto, que brindarán alternativas de solución a inconvenientes que se puedan presentar. Es necesario tener en cuenta este rol en el proyecto, ya que se utilizarán una serie de tecnologías y librerías de terceros que brindarán ciertas funcionalidades. Las personas que actuarán en este rol, son los desarrolladores o expertos en dichas tecnologías:

- **Joseph Gentle.** Ingeniero de Software de la *University of New South Wales, Computer Science, 2003-2008*. Ex ingeniero de Google Inc, trabajó en el proyecto Google Wave en donde se implementó el concepto de Transformación Operacional (*Operational Transform, OT*), que más adelante se describirá. Desarrolló una librería llamada ShareJS en donde implementa OT sobre texto plano y objetos JSON.

- **Harutyun Amirjanyan.** Desarrollador de complementos para el navegador web Mozilla, como firebug, acebug, entre otros. Contribuye en el grupo dispuesto por los creadores de la librería de Ace (Ajax.org Cloud9 Editor. Editor de código fuente independiente escrito en JavaScript) en su sección recursos para el usuario.

8.1.2. Identificación De Actores Del Sistema

Un actor es una idealización de un persona externa, un proceso o cualquier cosa que interactue con un sistema, subsistema o clase⁷⁴, por lo tanto su clara identificación permitirá establecer y enmarcar las funcionalidades y comportamientos que el sistema deberá tener, ya que este se debe desarrollar conforme a las necesidades que se definan para cada uno de los actores identificados. A continuación se presenta una descripción de los actores que estarán involucrados en la interacción con la aplicación web:

- **Visitante.** Es una persona que ingresa al sitio web con la intención de obtener información general del sitio. Este puede ser:
 - **Visitante Interno:** Visitante que ingresa a la aplicación e inicia sesión en esta. Dependiendo del rol, el Visitante Interno puede ser : Programador o Administrador del Sistema.
- **Programador:** Visitante interno que ingresa a la aplicación con la intención de hacer uso de su funcionalidad, tal como la IDE, la administración de proyectos, ayudas de contenido entre otras.
- **Administrador del Sistema:** Visitante interno con privilegios para realizar tareas especiales, ligadas con la administración y control de la aplicación.

El diagrama 1 se presentan los actores del sistema.

⁷⁴ *The Unified Modeling Language Reference Manual.* RUMBAUGH James, JACOBSON Ivar, BOOCHE Grady. La guía completa del proceso unificado escrita por sus creadores. Chapter 5, Use Case View. Pág 63.

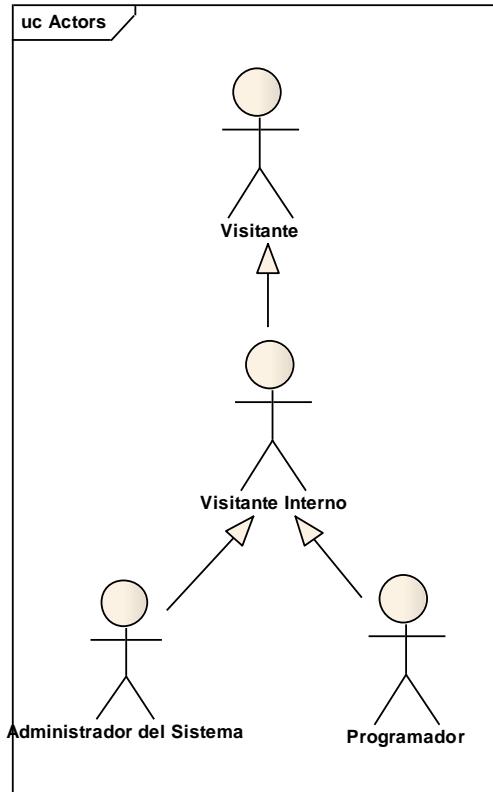


Diagrama 1. Actores del Sistema.

Especificación de Usuarios

<p>uc Actors</p> <pre> classDiagram actor Visitante </pre>	<p>SuperClase: -----</p> <p>Descripción: Cualquier persona abra la página index de la aplicación, a través su navegador web.</p>
	<p>SubClase: Visitante Interno.</p> <p>Necesidades y Expectativas:</p> <ul style="list-style-type: none"> • Informarse acerca de la aplicación, mediante la navegación por el sitio web. • Registrarse en la aplicación, creando una cuenta de usuario. • Activar la cuenta creada.
<p>Tabla 13. Especificación del Actor Visitante</p>	

 <p>uc Actors</p> <p>Visitante Interno</p>	<p>SuperClase: Visitante</p> <p>Descripción: Es un Visitante que se encuentra registrado en la aplicación. Posee un nombre de usuario y contraseña para hacer inicio de sesión en la aplicación.</p>
	<p>SubClase: Administrador del Sistema, Programador.</p>
<p>Necesidades y Expectativas:</p> <p>Las heredadas y además:</p> <ul style="list-style-type: none"> • Iniciar sesión en la aplicación. • Cerrar sesión en la aplicación. • Consultar ayudas acerca del uso de la aplicación. • Restablecer Contraseña y/o usuario. 	

Tabla 14. Especificación del Actor Visitante Interno

 <p>uc Actors</p> <p>Administrador del Sistema</p>	<p>SuperClase: Visitante Interno</p> <p>Descripción: Persona encargada de administrar la aplicación. Sus funciones están únicamente ligadas con la administración del sistema.</p>
	<p>SubClase: -----</p>

Necesidades y Expectativas:

Las heredadas y además:

- Gestionar Programadores.
- Gestionar de Proyectos de los Programadores.
- Administrar de Acceso de los Programadores.

Tabla 15. Especificación del Actor Administrador del Sistema.

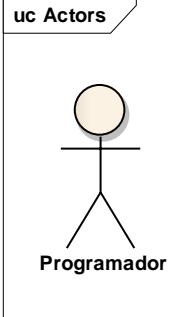
	SuperClase: Visitante Interno. Descripción: Es un visitante interno que hace uso de las funcionalidades de la aplicación dirigidas a la administración de proyectos y el uso de la IDE.
SubClase: - - - - - Necesidades y Expectativas: Las heredadas y además: <ul style="list-style-type: none">• Administrar proyectos de programación.• Uso compartido de Proyectos con otros programadores.• Hacer uso de la IDE.• Administrar su perfil.• Editar colaborativamente en proyectos compartidos.	

Tabla 16. Especificación del Actor Programador.

8.1.3. Familiarización Con Tecnologías, Herramientas y Prácticas

Es importante que el equipo desarrollo analice, pruebe y defina la tecnología, las herramientas y prácticas que serán utilizadas por los programadores para implementar la aplicación. En el proyecto intervendrá una serie de tecnologías y herramientas que a continuación serán descritas.

- **Java**

Es un lenguaje de programación de propósito general, concurrente, basado en clases y orientado a objetos. Está diseñado para ser lo suficientemente simple con el fin que los programadores logre mayor fluidez en poco tiempo. Para mayor información sobre Java diríjase a la sección 2.2.2 y la sección 6 de este documento.

- **JSP**

Java Server Pages es una tecnología que permite a los desarrolladores y diseñadores web, implementar páginas web dinámicas rápidamente, de fácil mantenimiento, ricas en información. Como parte de la familia Java, permite el desarrollo de aplicaciones basadas en la web que son independientes a la plataforma.⁷⁵

- **XML**

Es un lenguaje de etiquetado extensible muy simple pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.⁷⁶

- **JavaScript**

JavaScript es un lenguaje scripting basado en prototipos, que actualmente tiene muchas posibilidades. Inicialmente fue pensado para crear páginas más dinámicas e interactivas, y a medida que surgían nuevas versiones fueron potenciando el lenguaje con nuevos componentes.

JavaScript ha dejado de ser las mismas líneas de código que se copian y pegan de internet, para hacer efectos visuales, validar datos, etc. y se ha convertido en una gran alternativa para el desarrollo web. JavaScript

⁷⁵ Oracle Corporation. *JavaServer Pages Overview. Contents.*

<http://www.oracle.com/technetwork/java/overview-138580.html>

⁷⁶ Guía Breve de Tecnologías XML. <http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML>

ha tomado gran importancia gracias a sus bondades y se han implementado nuevas tecnologías a partir de él, dejando atrás aquel código que sólo estaba orientado en el lado del cliente, y toma importancia en el lado del servidor, permitiendo el desarrollo de servidores que son mucho más rápidos y eficientes con respecto a los servidores más conocidos. Hoy en día se puede observar el surgimiento de tecnologías muy poderosas que son implementadas en JavaScript, algunas de ellas permitiendo ambientes colaborativos en tiempo real.

- **CoffeeScript**

Es un pequeño lenguaje que compila en JavaScript, de manera que es un intento de exponer las partes buenas de JavaScript de una manera sencilla. El código se compila uno a uno al equivalente en JavaScript, y no hay interpretación en tiempo de ejecución. Se pueden usar cualquier librería JavaScript sin problemas desde CoffeeScript y viceversa. El código JavaScript resultado de la compilación de CoffeeScript es legible y ordenado, sin advertencias y tiende a ejecutarse tan rápido o más que el código JavaScript escrito a mano⁷⁷.

La siguiente figura muestra un ejemplo de la manera como CoffeeScript simplifica la escritura del código. Allí se observa la definición de funciones tanto en CoffeeScript como su equivalente en JavaScript.

<pre>square = (x) -> x * x cube = (x) -> square(x) * x</pre>	<pre>var cube, square; square = function(x) { return x * x; }; cube = function(x) { return square(x) * x; };</pre>
--	--



JavaScript

Figura 21. Definición de funciones en CoffeeScript vs JavaScript

⁷⁷ Sitio oficial de CoffeeScript. <http://coffeescript.org/>

- **JSON**

JavaScript Object Notation (Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. JSON es un formato de texto que es completamente independiente del lenguaje, utilizando convenciones de lenguajes como C, C++, C#, Java, JavaScript, Perl, Python, y otros más. Esto permite que JSON sea ideal para el intercambio de datos.⁷⁸

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor.
- Una lista ordenada de valores.

Los componentes de JSON se representan de la siguiente manera:

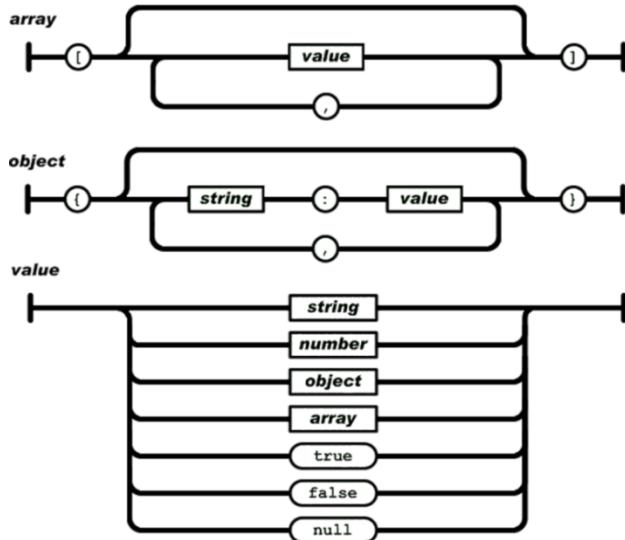


Figura 22. Composición de estructuras JSON⁷⁹

- **Transformación Operacional (*Operational Transformation OT*)**

Es una tecnología que soporta funciones informáticas y aplicaciones colaborativas⁷⁹. Considera n-sitios, en donde cada uno de ellos tiene una

⁷⁸ Introducing JSON. <http://json.org/>

⁷⁹ OT FAQ. *Operational Transformation Frequently Asked Questions and Answers. What is OT?*. <http://cooffice.ntu.edu.sg/otfaq/>

copia de los objetos compartidos. Cuando un objeto es modificado en un sitio, la operación es ejecutada inmediatamente y envía a los demás sitios que se encuentran geográficamente dispersos y conectados en una red de comunicación, para que sea ejecutada nuevamente. Cada una de estas operaciones es procesada en cuatro pasos:⁸⁰

- a) Generación de la operación en un sitio.
- b) Difusión o *broadcast* para lo demás sitios conectados.
- c) Recepción por parte de los otros sitios.
- d) Ejecución en los demás sitios.

Mantenimiento de la Coherencia

Con el fin de mantener la coherencia de los objetos copiados, durante el procesamiento de las operaciones, se tienen en cuenta las siguientes consideraciones:⁸¹

Nota: Para las ilustraciones se pondrá como ejemplo un editor de texto colaborativo, donde las operaciones son *insert(p,c)* y *delete(p)*, donde p es la posición en el string y c es el carácter que intervendrá.

- Preservación de la Causalidad

Mantiene el control de precedencias entre la ejecución de las operaciones. Entonces, si una operación op_1 precede a otra op_2 , en cada sitio en donde se hayan difundido, la ejecución de op_1 debe preceder a la ejecución de op_2 . Se define un vector de estados, a partir del cual se deduce el orden de las ejecuciones de las operaciones. En la figura 23 se observa el comportamiento del procesamiento de operaciones en ausencia de preservación de la causalidad, y con el uso de la misma.

⁸⁰ Pascal Molli, Gérald Oster, Hala Skaf-Molli, Abdessmad Imine. “Using the Transformation Approach to Build a Safe and Generic Data Synchronizer”. *Transformation Approach*. ACM New York, NY, USA ©2003.

⁸¹ Nicolas Vidot, Michelle Cart, Jean Ferrié, Maher Suleiman. “Copies Convergence in a distributed real-time collaborative environment”. *General Problems*. ACM New York, NY, USA ©2000

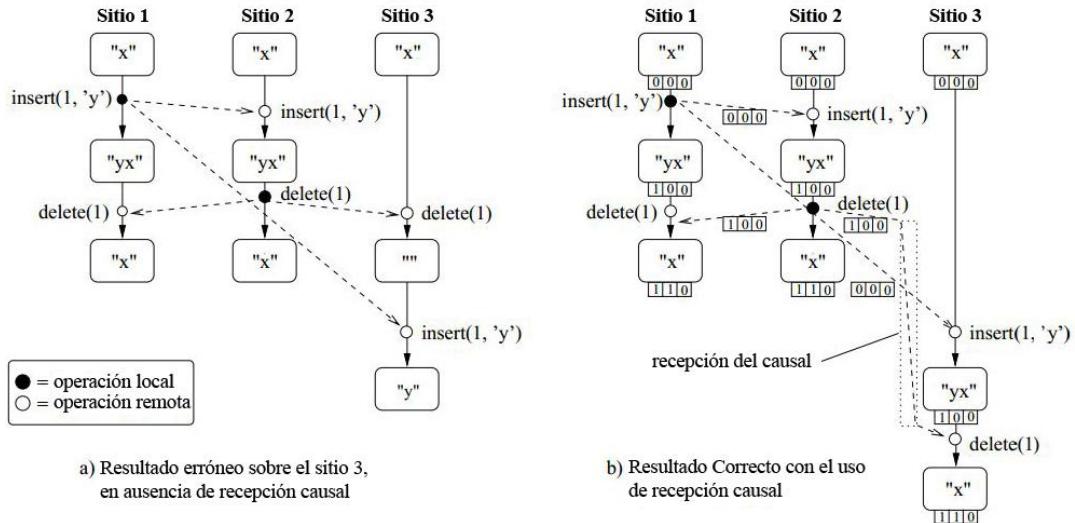


Figura 23. Resultado de la causalidad⁸²

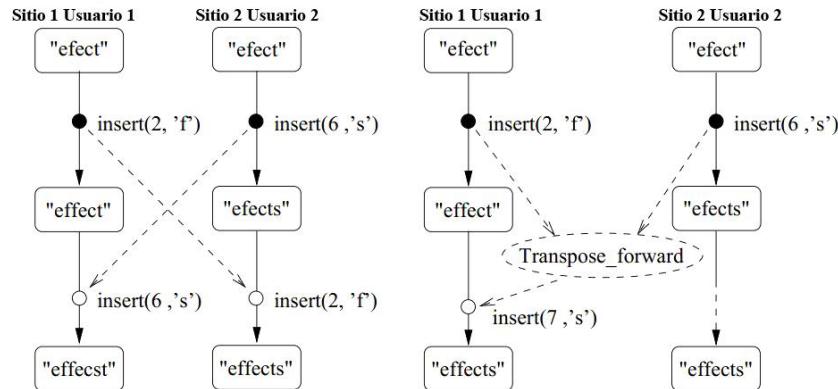
– Preservación de la Intención del Usuario

En este caso se consideran las operaciones que son concurrentes, en donde ninguno depende del efecto del otro. Dos operaciones concurrentes se definen de la siguiente manera:

$$\text{concurrentes}(\text{op1}, \text{op2}) \leftrightarrow \neg(\text{precede}(\text{op1}, \text{op2})) \wedge \neg(\text{precede}(\text{op2}, \text{op1}))$$

Lo anterior indica que op1 y op2 son concurrentes, si y solo si, op1 no precede a op2 , y viceversa. Si un sitio ejecuta op1 antes que op2 , se debe tener en cuenta los cambios hechos por op1 cuando se ejecute op2 , con el fin de respetar la intención del usuario que genera op2 . La figura 24 muestra un ejemplo de dos usuarios trabajando simultáneamente sobre el mismo objeto cuyo estado es “effect”. La intención del usuario 1 es adicionar una ‘f’ con la siguiente operación $\text{insert}(2,f)$. La intención del usuario 2 es adicionar ‘s’ con la operación $\text{insert}(6,s)$. Cuando la operación es liberada y ejecutada en el sitio 1, el nuevo estado es “effecst”, lo cual no es lo esperado por el usuario 2. Entonces, para respetar la intención del usuario 2, la operación $\text{insert}(6,s)$ necesita ser transformada por $\text{insert}(7,s)$. Entonces, la solución está en transformar la operación para ser ejecutada teniendo en cuenta las modificaciones hechas por todas las operaciones concurrentes serializadas antes. De manera similar

funciona con las operaciones de borrado, las figura 25, muestra la manera incorrecta y la correcta de llevar a cabo estas operaciones siempre respetando la intención del usuario.



- a) No respeta la intención del usuario 2, b) Uso de *de forward transposition* para asegurar el respeto de la intención del usuario, sobre el sitio 1

Figura 24. Representación de la intención del usuario⁸².

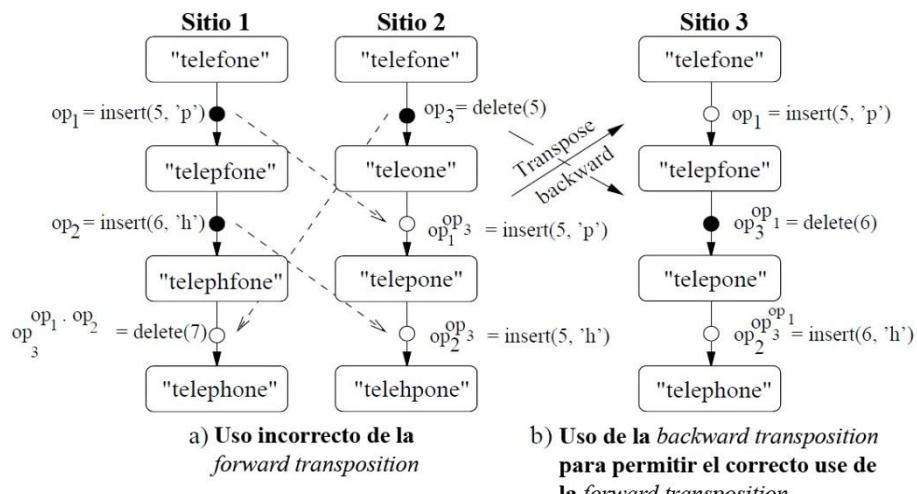


Figura 25. Concurrencia parcial entre operaciones⁸²

– Convergencia de las Copias

Teniendo en cuenta la causalidad y la intención del usuario no es suficiente para lograr ejecuciones que garanticen en todos los casos la convergencia de las copias en todos los sitios. De manera que sea cual sea las operaciones, $op_1, op_2, op_3, \dots, op_n$ que sean concurrentes, su

transformación no debe depender del orden usado para la serialización de las mismas, de manera que, sea cual sea el orden el resultado del efecto debe ser equivalente en todas las copias.

- **NodeJS**

NodeJS es una plataforma que permite la construcción fácil y rápida de aplicaciones para redes que sean escalables. Es ideal para el uso intensivo datos en aplicaciones en tiempo real que se ejecutan a través de dispositivos distribuidos.*



Figura 26. Logo oficial NodoJS*

NodeJS es parte del entorno de Javascript en el lado del cliente, y extiende la API de Javascript para ofrecer las funcionales habituales de un servidor. Está basado en el motor de javascript V8 de Google escrito en C++ y es orientado a eventos⁸².

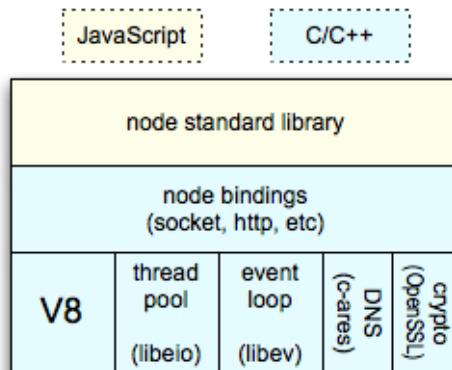


Figura 27 Arquitectura de NodeJS.

El objetivo de NodeJS es ofrecer una fácil y segura manera de construir aplicaciones de red de alto rendimiento y escalables en JavaScript, lo cual es posible gracias a su arquitectura que ofrecen las siguientes características:⁸³

* Definición tomada del sitio oficial de Node.js. <http://nodejs.org/>

⁸² Reuven M. Lerner. At the Forge. Nodejs. Linux Journal, volumen May 2011 Issue 205.

⁸³ Nagi Letaifa Dans. A Full Javascript Architecture, Part One-NodeJS. Zenika Architecture Informatique.

- **Un solo Hilo:** NodeJS usa un solo hilo para ejecutar, a diferencia de otros servidores que crean un hilo por cada solicitud, así evita la sobrecarga de la CPU y masivas pilas de ejecución en memoria.
- **Bucle de Eventos:** usa la librería libev de Marc Lehmann, para el mecanismo escalable de notificación de eventos, lo cual lo hace asincrónico.
- **No Bloqueo de E/S:** Evita la pérdida de tiempo de CPU producto de esperas de respuestas de procesos de E/S gracias a su asincronía.

- **Socket.IO**

Socket.IO tiene como objetivo hacer aplicaciones en tiempo real que sean posibles en cada navegador y dispositivo móviles, cerrando la brecha entre los diferentes mecanismos de transporte. Está implementado 100% en JavaScript⁸⁴.

Socket.IO es un módulo de NodeJS que brinda la capacidad de gestionar la conectividad en tiempo real en todos los navegadores, seleccionando automáticamente el mejor tipo de conexión entre el cliente y el servidor. Es usado tanto en el lado servidor como en el lado cliente.

- **ShareJS**

ShareJS es una librería de Transformación Operacional para NodeJS y navegadores. Le permite fácilmente hacer en tiempo real edición concurrente en la aplicación que la integre⁸⁵. El servidor corre sobre NodeJS y el cliente trabaja en NodeJS o un navegador web.

Tomando como ejemplo un editor de texto, a medida que este se edite, ShareJS genera operaciones que son como pequeños cambios o actualizaciones en el documento, por ejemplo, *insert:'hi', position:50*. Similar a un servidor de subversión, ShareJS tiene un número de versión, para manejar la exclusión entre los usuarios conectados. Si

⁸⁴ Introducing SocketIO v.8. <http://socket.io/>

⁸⁵ Joseph Gentle. ShareJS-Live concurrent editing in your app. <http://sharejs.org/>

varios usuarios están presentes en la misma versión, una de las operaciones se aplicará directamente, mientras que las demás el servidor las transforman automáticamente y luego se aplican.

Para el usuario todo esto es transparente, él verá las modificaciones en su navegador de inmediato. El algoritmo es muy cuidadoso para asegurarse que al terminar las ediciones, todos los usuarios vean el mismo documento, sin importar en qué orden se efectuaron dichas operaciones.

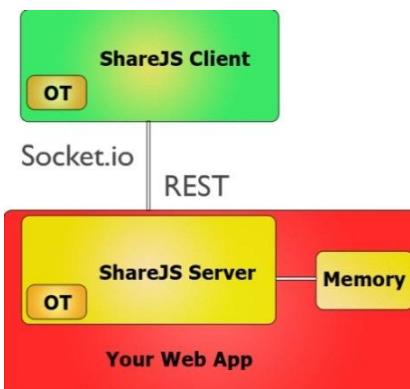


Figura 28. Arquitectura Base de ShareJS⁸⁶

- **NoSQL**

El término base de datos NoSQL fue elegido como una clase específica de almacenes de datos no relacionales. Estas bases de datos en su mayoría no utilizan SQL como lenguaje de consulta, de ahí su nombre. En un sentido más amplio, esta categoría de base de datos también incluye base de datos XML, base de datos gráficas o base de datos de documentos y base de datos de objetos.⁸⁷

Los sistemas NoSQL generalmente tienen seis características claves:⁸⁸

⁸⁶ Joseph Gentle. ShareJS: library for Google Wave –style collaboration. ShareJS launch talk slider. The Universal Runtime

⁸⁷ Jaroslav Pokorný, “NoSQL Databases: a step to database scalability in Web environment” Charles University, Malostranské, Czech Republic. ACM New York, NY, USA ©2011

⁸⁸ Rick Cattell. “Scalable SQL and NoSQL Data Stores”. ACM SIGMOD Record. Volume 39 Issue 4, December 2010.

1. La capacidad de escalar horizontalmente, en operaciones simples de lectura y escritura distribuida sobre varios servidores.
 2. La capacidad de replicar y distribuir (particiones) datos sobre varios servidores.
 3. Un simple llamado a nivel de interface o protocolo (en contraste al uso de SQL, como los motores convencionales).
 4. Un modelo de concurrencia más débil que las transacciones ACID (*Atomicity, Consistency, Isolation, Durability*) de la mayoría de sistemas de base de datos relacionales (SQL).
 5. Uso eficiente de indexes distribuidos y memoria RAM para almacenar los datos.
 6. La capacidad de adicionar dinámicamente atributos a los registros de datos.
- **JQuery**

JQuery es una librería de JavaScript rápida y concisa que simplifica la manera de manipular un documento HTML, manejo de eventos, animaciones e interacciones AJAX que permiten el desarrollo web rápido.⁸⁹



Figura 29. Logo de JQuery

- **Ace Editor⁹⁰**

Es un editor de texto escrito en JavaScript, coincide y extiende las características, la usabilidad y el rendimiento de los editores nativos que

⁸⁹ JQuery write less, do more. <http://jquery.com/>

⁹⁰ Ace Ajax.org Cloud9 Editor. <http://ace.ajax.org/>

existen. Es fácil de integrar en cualquier página web y/o aplicación JavaScript. Entre sus más destacadas características se encuentran:

- Resaltador de sintaxis.
- Asignar y anular sangrías automáticamente.
- Línea de comandos opcional.
- Trabaja sin problemas con documentos grandes (100000 líneas o más).
- Combinación de teclas personalizable de acuerdo a la plataforma.
- Manejo de temas visuales.
- Busca y reemplaza con expresiones regulares.
- Resalta paréntesis, llaves y corchetes coincidentes.
- Resalta línea activa y palabra seleccionada.
- Plegado de código entre llaves y caracteres de comentarios.
- Arrastra y suelta texto usando el mouse.

Ace Editor soporta la sintaxis de lenguajes como: JavaScript, HTML, CSS, XML, Python, PHP, Java, Ruby, C++, CoffeeScript, ya demás cuenta con un comprobador de sintaxis por el momento solo para JavaScript, CoffeeScript y CSS.

- **DHTMLX Suite**

DHTMLX Suite es una librería JavaScript que proporciona colecciones completas de componentes para construcción de interfaces gráficas de usuario, lo que permite obtener interfaces web robustas, gestionar las comunicaciones cliente-servidor con Ajax, e implementar la lógica del lado del servidor.⁹¹

- **Metaprogramación**

Metaprogramación es escribir programas que representan y manipulan otros programas (ejemplo, compiladores, programas generadores, e intérpretes) o ellos mismos (reflexión). Estos programas escritos que generan código fuente son llamados metaprogramas.⁹²

⁹¹ *JavaScript Ajax Library for Building Great Web Apps.*

<http://dhtmlx.com/docs/products/dhtmlxSuite/index.shtml>

⁹² Johannes Koskinen. *Metaprogramming in C++: Terminology.* March 9, 2004

8.1.4. Historias de Usuario

Las historias de usuario determinan las distintas funciones que la aplicación deberá ofrecerle al usuario final. Estas se obtienen con la participación del cliente, el cual se encarga de definirlas, describiendo claramente las tareas que la aplicación deberá cumplir para generarle valor al negocio.

Para el desarrollo del actual proyecto, el rol cliente no es una persona que sea parte del grupo de trabajo, de manera que la definición de las historias de usuario estará a cargo del grupo de desarrollo, partiendo de una investigación previa, analizando características de proyectos similares, consultando las opiniones de los posibles usuarios, atendiendo sugerencias del director del proyecto, y utilizando ideas de integración de tecnologías vigentes que aportarían valor e interacción a la aplicación.

A continuación se describirán las historias de usuario, que brindarán una visión general del comportamiento que la aplicación tendrá.

H1. Compilar Proyecto

El programador podrá compilar un proyecto (creado o compartido) abierto desde la IDE para examinar si tiene algún error, excepción o advertencia. La aplicación tiene que notificarle respuesta de compilación al programador, ya sea satisfactoria o denegada, que en este caso debe presentarle los errores, excepciones o advertencias ocurridas.

H2. Construir archivo JAR

La aplicación debe permitirle al programador construir un archivo JAR del proyecto que desee, siempre y cuando esté abierto en la IDE. En caso de ocurrir alguna eventualidad que lo impida notificarle al programador lo qué sucedió.

H3. Ejecutar Proyecto

El programador podrá ejecutar un proyecto que tenga abierto en la IDE. Si esta operación no se lleva a cabo exitosamente la aplicación debe notificarle lo que sucedió.

H4. Crear Cuenta de Programador.

El visitante del portal podrá crear una cuenta de programador, ingresando su nombre, correo electrónico, y contraseña. El visitante no debe ingresar a la aplicación hasta que no active su cuenta. Ahora, si el visitante interno olvida o han descubierto su contraseña, la aplicación debe permitir restablecerla, permitiéndole registrar una nueva. La aplicación debe implementar una estrategia de seguridad que gestione las cuentas de programadores, para posibles ataques de vulnerabilidad de acceder a sus datos privados.

H5. Gestionar Sesiones

El visitante interno podrá iniciar sesión en la aplicación, ingresando su correo electrónico y su contraseña en un formulario. La aplicación debe controlar que dos visitantes no usen la aplicación al mismo tiempo con la misma cuenta. Además la aplicación debe permitirle al visitante interno cerrar su sesión, ya sea a través de un botón, o al cerrar el navegador.

H6. Editar Perfil de Programador

El Programador podrá cambiar su imagen de avatar para la aplicación, además de tener un vínculo para cambiar su contraseña si lo desea.

H7. Gestionar Proyectos

El programador podrá a través de la IDE: crear proyectos, abrir un proyecto creado o compartido, cerrarlo, renombrarlo y eliminarlo. Estas dos últimas opciones serán posibles siempre que el programador sea el autor del proyecto. Estas operaciones deben comportarse como lo hacen las IDE *Desktop* convencionales, por ejemplo Netbeans y Eclipse.

H8. Compartir Proyecto

Un programador podrá compartir un proyecto con otros programadores, siempre y cuando sea el autor. El programador que comparte asigna los privilegios que los demás programadores tendrán sobre el proyecto (solo lectura o lectura/escritura). El proceso de dejar de compartir puede ser realizado tanto por el autor, como el programador a quien se le compartió, es decir, el autor decide a quién o quienes dejar de compartir, o un programador al cual se le ha compartido un proyecto podrá decidir no participar más en este.

H9. Descargar Proyecto

El programador cuenta con dos opciones de descarga de proyectos que haya creado o que tenga compartido. Una es descargar el ejecutable del proyecto, y la otra descargar el directorio que contiene el proyecto, después de esto se iniciará la descarga.

H10. Gestión de Ficheros

Un fichero puede ser una clase, un paquete, una librería, archivos planos o archivos con una extensión permitida para la aplicación. Para realizar operaciones con ficheros debe existir al menos un proyecto abierto. Teniendo en cuenta lo anterior, un programador autor o con privilegios de lectura/escritura sobre un proyecto, podrá: crear un tipo de fichero, agregar al proyecto una clase, una librería o cualquier fichero con extensión permitida, abrir una clase o archivo de texto plano en un editor, renombrar un fichero y eliminarlo.

H11. Mover Fichero

El programador autor o con privilegios de lectura/escritura sobre un proyecto, podrá mover ficheros entre paquetes, e incluso entre proyectos abiertos. Esta operación debe hacerse solo en movimientos permitidos y visualizar los cambios. Los programadores con privilegios de solo lectura podrán copiar un fichero hacia otro proyecto.

H12. Diseñar GUI

Es necesario que el entorno gráfico de la aplicación sea similar a un ambiente de aplicación Desktop. Por lo tanto la vista para IDE debe ser similar y adaptarse al factor común gráfico que muestra la figura 15 de la sección 4.3.2. Es necesario que el programador cuente con elementos gráficos como un árbol de proyectos, editor con resaltador de sintaxis y formateo de código, una consola de salida para mostrar los resultados de compilación, entre otros elementos.

H13. Generar Código Fuente

La aplicación debe ofrecerle al usuario una operación automática que le genere código fuente, con el propósito de ahorrarse tiempo de trabajo. Esto puede implementarse cuando se crea una clase, escribiéndole una plantilla básica de su estructura. También en el constructor de interfaces gráficas de usuario, cada vez que agregue un componente al lienzo, debe generarle el código fuente que involucre ese elemento agregado.

H14. Edición Concurrente

La aplicación debe permitir la edición colaborativa, a través de la participación de dos o más programadores editando en un proyecto en tiempo real. Esto debe permitirse, si estos programadores comparten el proyecto y tienen privilegios de lectura/escritura sobre este.

H15. Notificación de Operaciones

La aplicación debe notificar las operaciones que los programadores están efectuando sobre un proyecto compartido y que se encuentran trabajando concurrentemente. Esto es necesario para mantener al tanto a todos los programadores de lo que sucede con el proyecto en tiempo real. Por ejemplo, si un programador con privilegios de lectura/escritura elimina una clase, todos deben ser notificados para evitar confusiones.

H16. Chat

La aplicación debe ofrecerles un chat por proyecto, a los programadores que trabajan colaborativamente, con el propósito de que cuenten con un medio de comunicación en tiempo real, entre los que estén trabajando en el proyecto compartido.

H17. Constructor de Interfaces Gráficas de Usuario

La aplicación debe contar con una herramienta que le permita al programador diseñar interfaces gráficas para sus programas, a través de un lienzo y una paleta de componentes gráficos que puedan ser arrastrados y organizados. Esta herramienta debe ser colaborativa.

H18. Consultar Accesos a la Aplicación

La aplicación debe mantener un registro de ingreso de los programadores, que permita consultar los accesos para que el administrador pueda disponer de ellos. Esto le servirá al administrador para tomar decisiones sobre un usuario, partiendo del tiempo en el cual el programador no ha usado la aplicación, posiblemente considerándolo como abandono de la cuenta.

H19. Gestionar Programadores

La aplicación debe brindarle al administrador del sistema, un módulo de gestión de programadores que le permita buscar programadores, listarlos, ver sus perfiles, eliminarlos y ver sus proyectos.

8.5. PLANIFICACIÓN

8.5.1 Priorización y Estimación de las Historias de Usuario

En esta etapa es necesario establecer prioridades y estimaciones de esfuerzo para cada historia de usuario, con el fin determinar el tiempo o el alcance que tendrá el proyecto y el esfuerzo que debe invertirse en este.

La prioridad es establecida por el cliente de acuerdo a su importancia para el cumplimiento de los objetivos del proyecto descritas de la siguiente manera:

- **Alta:** La historia de usuario debe implementarse para cumplir con los objetivos del proyecto.
- **Media:** La historia de usuario tiene un impacto medible sobre los objetivos del proyecto.
- **Baja:** La historia de usuario aumentará la satisfacción del cliente, pero no se ha justificado explícitamente.

La estimación del esfuerzo por cada historia de usuario está a cargo del equipo desarrollador, y se determina por semanas.

La priorización y estimación de cada historia de usuario está definida en la Tabla 17.

Historia	Nombre	Prioridad	Estimación
H1	Compilar Proyecto	Alta	3
H2	Construir archivo JAR	Alta	1
H3	Ejecutar Proyecto	Alta	2
H4	Crear Cuenta de Programador	Media	1
H5	Gestionar Sesiones	Media	1
H6	Editar Perfil de Programador	Baja	1
H7	Gestionar Proyectos	Alta	3
H8	Compartir Proyecto	Baja	1
H9	Descargar Proyecto	Baja	1
H10	Gestión de Ficheros	Alta	3
H11	Mover Fichero	Baja	1
H12	Entorno Gráfico	Media	3
H13	Generar Código Fuente	Baja	2
H14	Edición Concurrente	Baja	3
H15	Notificación de Operaciones	Baja	1

H16	Chat	Baja	1
H17	Constructor de Interfaces Gráficas de Usuario	Baja	2
H18	Consultar Accesos a la Aplicación	Media	1
H19	Gestionar Programadores	Media	1

Tabla 17. Priorización y Estimación de Historias de Usuario

8.5.2. Plan de Iteraciones

Basado en las prioridades y estimaciones descritas anteriormente, se determina que cada iteración tendrá una duración máxima de tres semanas, dando como resultado un prototipo funcional, al cual se le aplicarán pruebas de unidad y de aceptación. A continuación se definen las iteraciones que el proyecto presentará durante todo su desarrollo.

Iteración	Historias de Usuario
Iteración 1	H7
Iteración 2	H10
Iteración 3	H1
Iteración 4	H2, H3
Iteración 5	H12
Iteración 6	H4, H5
Iteración 7	H18, H19
Iteración 8	H6, H8, H9
Iteración 9	H14
Iteración 10	H17
Iteración 11	H11, H13
Iteración 12	H15, H16

Tabla 18. Plan de Iteraciones

8.5.3. Metáfora del Sistema

Aplicación bajo ambiente *Cloud* que lleva la funcionalidad y apariencia de una IDE *desktop* a un ambiente web, para el desarrollo de programas en Java. Permite a los visitantes crear cuentas de programador, crear proyectos, gestionarlos y compartirlos con otros para trabajar colaborativamente. La IDE está provista de un constructor de interfaces gráficas básicos, para la construcción de JFrames sencillos. Está orientada para la edición, compilación y ejecución en un ambiente colaborativo.

8.6. ITERACIONES

8.6.1. Iteración 1

- Historia de Usuario Abarcadas**

Historia de Usuario	
Número: H7	Usuario: Programador
Nombre de la Historia: Gestionar Proyectos	
Prioridad en Negocio (Alta/Media/Baja): Alta	Estimación: 3
Descripción: Permitirle al programador crear, abrir, cerrar, renombrar y eliminar un proyecto. Estas dos últimas opciones serán posibles siempre que el programador sea el autor del proyecto. Las operaciones deben comportarse como lo hacen las IDE <i>Desktop</i> convencionales (Netbeans y Eclipse).	
Observaciones: Se debe determinar la estructura de carpetas y archivos de configuración para un proyecto, considerando que la administración de los proyectos debe tener en cuenta consideraciones de concurrencia entre los programadores que estén trabajando en un proyecto.	

Actividades:

- Analizar la estructura carpetas y archivos de configuración para un proyecto.
- Investigar sobre archivos XML, para gestionar archivos.
- Crear un archivo XML y manipularlo a través de las librerías disponibles en la API de Java, para gestionar los ficheros del proyecto.
- Implementar una rutina que permita crear los respectivos directorios cuando un proyecto es creado.
- Descargar de memoria un proyecto, cuando el programador lo cierre desde la IDE.
- Buscar un componente gráfico que permita visualizar en forma de árbol de contenidos, los paquetes y ficheros del proyecto.
- Implementar la opción de renombrar un proyecto (esta operación solo puede ser llevada a cabo por el programador que ha creado el proyecto).
- Eliminar un proyecto, lo cual implica eliminar su directorio, sus respectivos subdirectorios y archivos, actualizando los XML que gestionan los proyectos del programador. Es importante que eliminar un proyecto solo es permitido para el programador que lo ha creado.
- Actualizar todos los XML relacionados con la gestión de proyectos.
- Guardar cambios automáticamente y en tiempo real.
- Verificar qué programadores están trabajando en un proyecto en un momento dado, con el fin de notificar los respectivos cambios que se van efectuando sobre el proyecto. Esto también es útil para la funcionalidad del chat entre los programadores de un proyecto.

Tabla 19. Descripción de la Historia de Usuario H7

- **Diagrama de Clases Iteración 1**

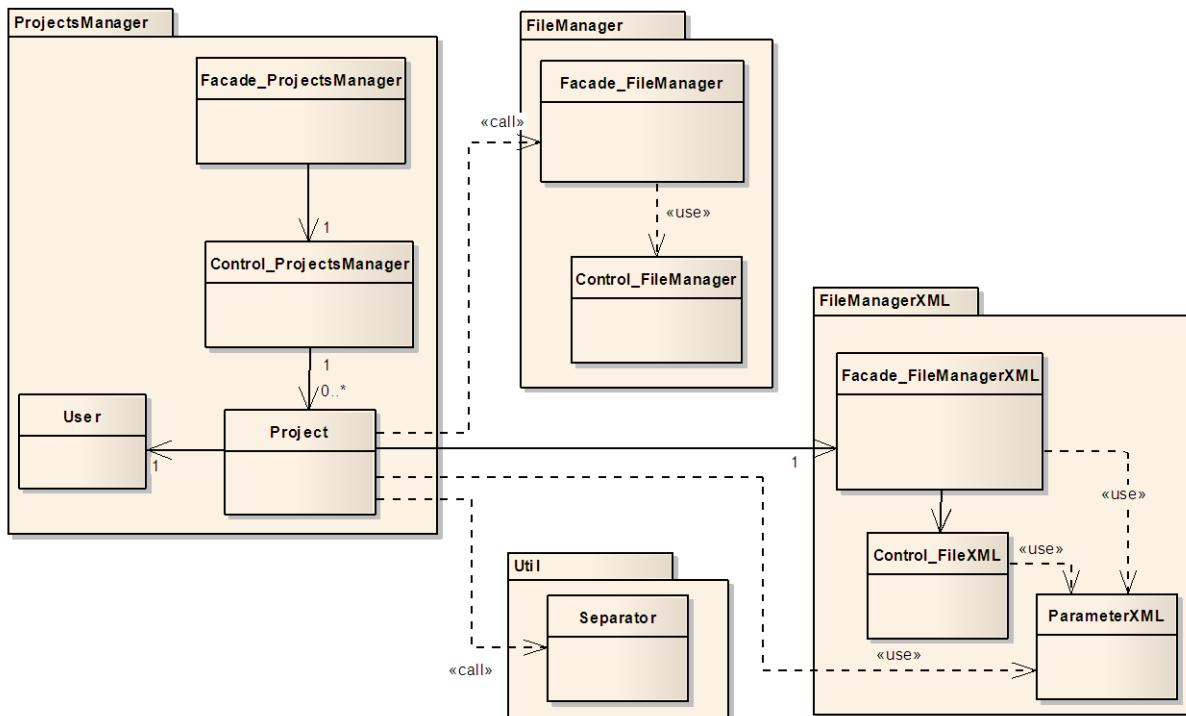


Diagrama 2. Diagrama de Clases de la Iteración 1

- **Pruebas de Unidad**

Para las pruebas unitarias se usó JUnit como herramienta para llevar a cabo las pruebas de software, es un complemento que se integra fácilmente a Netbeans. Para esta iteración se tendrán en cuenta las siguientes operaciones:

- Crear un proyecto al usuario cas@ llamado “PruebaTest2”.
- Cargar un proyecto al usuario cas@ llamado “sepulveda”.
- Renombrar el proyecto “creadoParaRenombrar” al usuario cas@, y cambiado por “renombrado”.
- Cerrarle un proyecto al usuario cas@ llamado “sepulveda”.
- Eliminarle un proyecto al usuario cas@ llamado “creadoParaEliminar”.

Resultado de la Prueba de Unidad

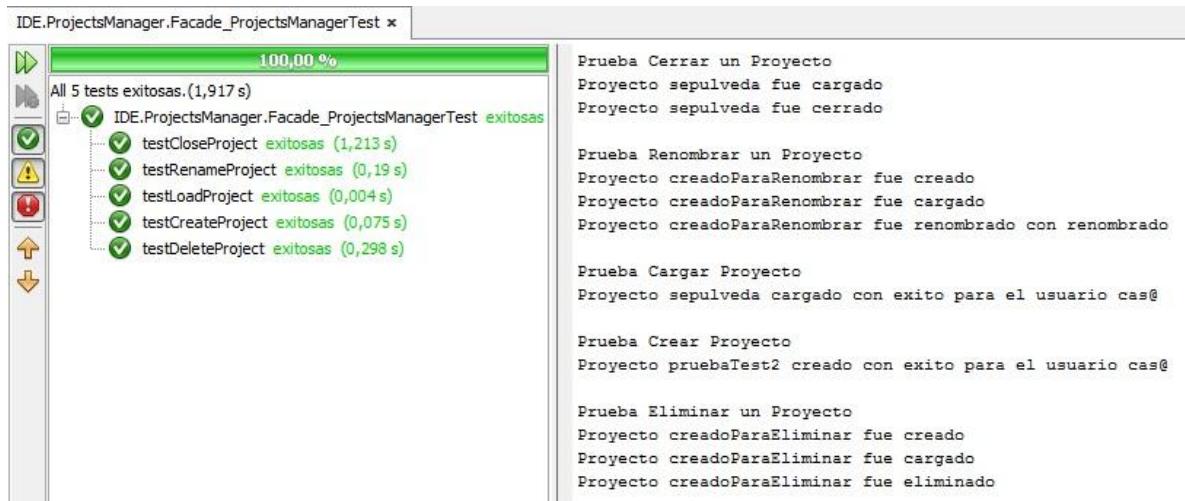


Figura 30. Pruebas de Unidad de Iteración 1

8.6.2. Iteración 2

- **Historias de Usuario Abarcadas**

		Historia de Usuario
Número:	H10	Usuario: Programador
Nombre de la Historia: Gestión de Ficheros		
Prioridad en Negocio (Alta/Media/Baja):	Alta	Estimación: 3
Descripción: Un programador autor o con privilegios de lectura/escritura sobre un proyecto, podrá: crear un tipo de fichero, agregar al proyecto una clase, una librería o cualquier fichero con extensión permitida, abrir una clase o archivo de texto plano en un editor, renombrar un fichero y eliminarlo.		
Observaciones: Un fichero puede ser una clase, un paquete, una librería, archivos planos o archivos con una extensión permitida para la aplicación. Para realizar operaciones con ficheros debe existir al menos un proyecto abierto en la IDE. Las operaciones efectuadas sobre los ficheros en un		

proyecto deben ser notificadas.

Actividades:

- Definir las extensiones de los archivos que la aplicación permitirá cargar a un proyecto. Es necesario restringir ciertas extensiones con fines de seguridad.
- Verificar que el programador que desee gestionar ficheros sea el autor del proyecto, o es un programador al cual se le ha compartido, con privilegios de lectura/escritura.
- Diseñar e implementar la gestión de cada operación (crear, renombrar, abrir, y eliminar) de un fichero, ya sea clase, paquete, librería o archivo de extensión permitida, a través de archivos de control en formato XML. Todas las operaciones efectuadas sobre un fichero implican actualizar el archivo XML que corresponda.
- Notificar los cambios realizados por una operación hecha sobre un fichero, a los programadores a los cuales les ha sido compartido el proyecto y estén visualizándolo en la IDE.
- Crear un paquete.
- Renombrar un paquete.
- Eliminar un paquete.
- Crear una clase. Esta operación puede ser realizada sobre un paquete creado en el proyecto, o en el paquete predeterminado del proyecto.
- Abrir una clase en el editor.
- Renombrar una clase.
- Eliminar una clase.
- Crear un archivo de texto plano.
- Abrir un archivo de texto plano en el editor.
- Renombrar un archivo de texto plano.
- Eliminar un archivo de texto plano.
- Cargar una clase externa.
- Cargar una librería externa.
- Cargar un archivo de extensión permitida.
- Renombrar una librería.
- Eliminar una librería.
- Guardar cambios respecto a la gestión de ficheros automáticamente y en tiempo real, es decir si el proyecto está compartido y lo están visualizando concurrentemente, los cambios se reflejan de inmediato en todos los sitios.

Tabla 20. Descripción de la historia de Usuario H10

- **Diagrama de Clases de la Iteración 2**

Para la Iteración 2 fue necesario crear dos clases que permitirían crear las instancias que correspondieran a las clases y a los ficheros permitidos por la aplicación. Las clases adicionadas son:

- Class.java
- File.java

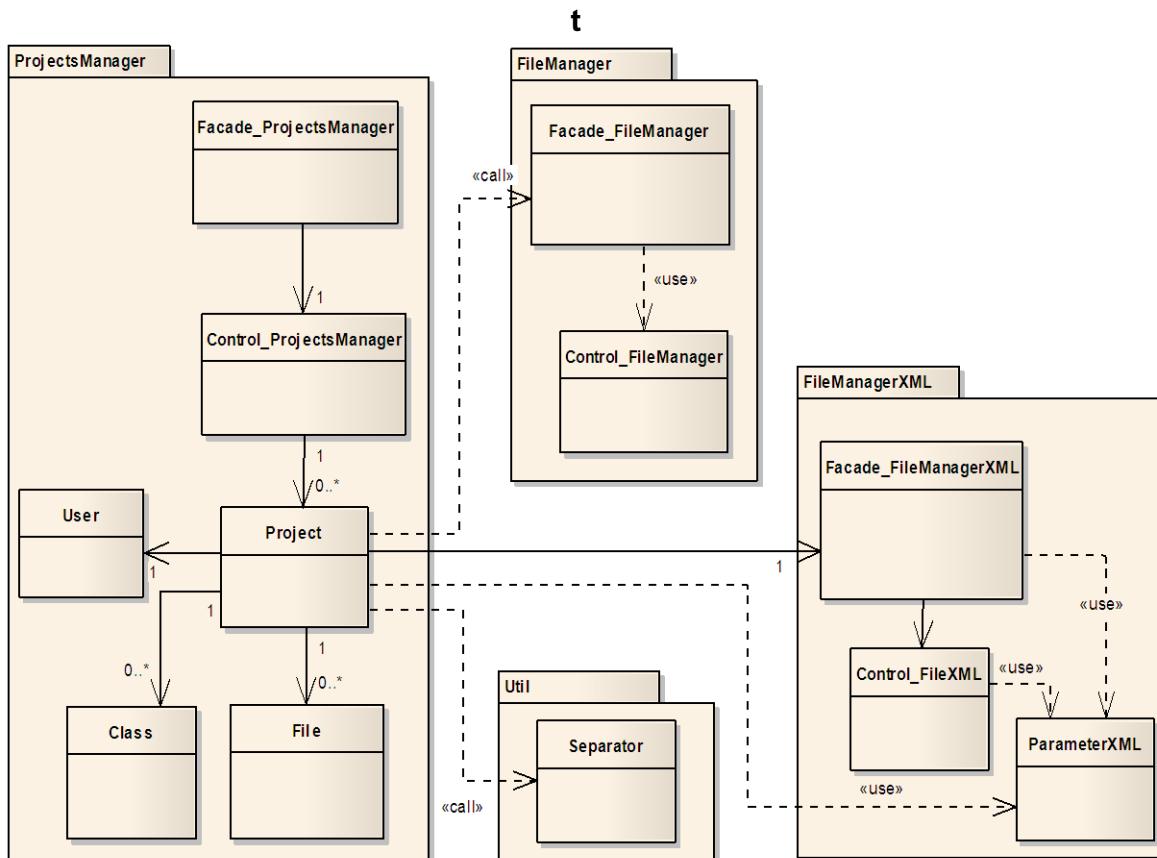


Diagrama 3. Diagrama de Clases de la Iteración 2

- **Pruebas de Unidad**

Las pruebas de unidad que corresponde para esta iteración, comprende la siguiente funcionalidad:

- `testAddPackage()`: Agregar un Paquete “Mundo” a un proyecto.
- `testAddClass()`: Agregar una clase “Principal.java” al paquete “Mundo”.

- `testAddClassExisting()`: Agregar una clase existen en el cliente “Gato.java” al paquete “Mundo”.
- `testAddNewFile()`: Agregar un archivo nuevo “Archivo.txt” al paquete “Mundo”.
- `testAddFile()`: Agregar un archivo “Koala.jpg” que no sea un ejecutable (exe, sh, bat) al paquete “Mundo”.
- `testAddLibJAR()`: Agregar una librería JAR “cos.jar” a una ubicación establecida para las librerías llamada libs incluida en el directorio del proyecto.
- `testRenamePackage()`: Renombrar el paquete “GUI” por “PaqueteRenombrado”.
- `testRenameClass()`: Renombrar la clase “Principal.java” por “Renombrada.java”.
- `testRenameFile()`: Renombrar el archivo “Archivo.txt” por “RenombradoArchivo.txt”.
- `testRenameLib()`: Renombra la librería “cos.jar” por “nuevo.jar”.
- `testDeletePackage()`: Elimina el paquete “Mundo”.
- `testDeleteClass()`: Elimina la clase “Principal.java”
- `testDeleteFile()`: Elimina el archivo “Archivo.txt”.
- `testDeleteLib()`: Elimina la Librería “cos.jar”.

Resultado de Prueba de Unidad

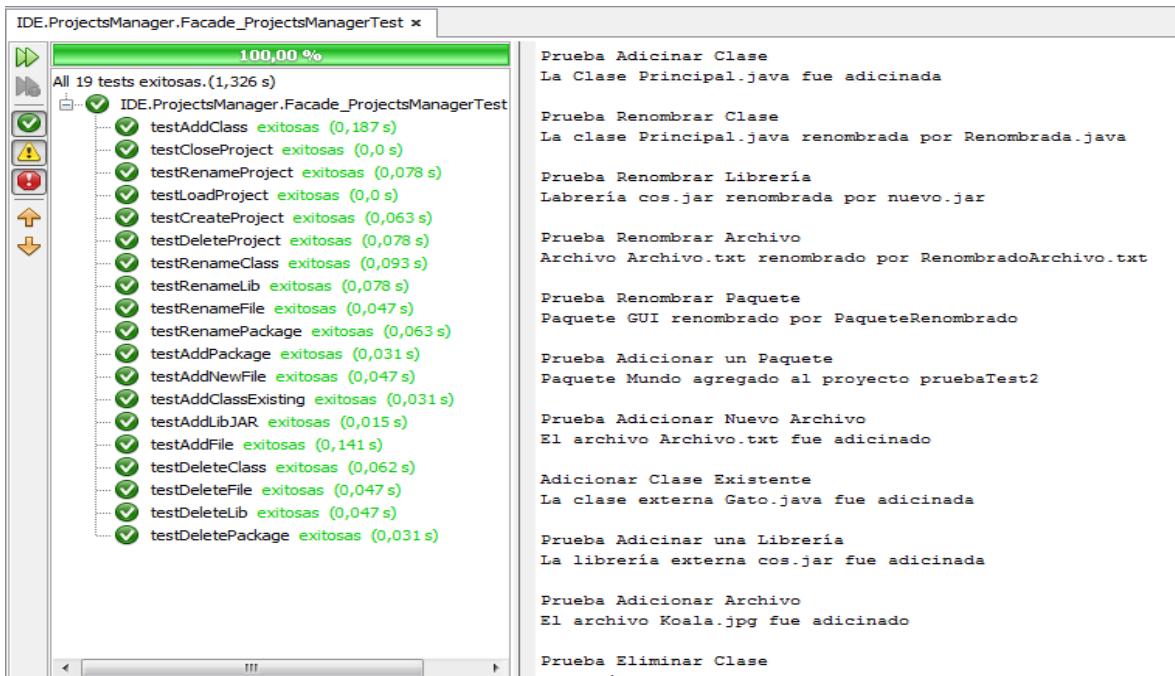


Figura 31. Resultado de Prueba de Unidad Iteración 2

8.6.3. Iteración 3

- **Historias de Usuario Abarcadas**

		Historia de Usuario
Número:	H1	Usuario: Programador
Nombre de la Historia: Compilar Proyecto.		
Prioridad en Negocio (Alta/Media/Baja):	Alta	Estimación: 3
Descripción: El programador podrá compilar un proyecto (creado o compartido) abierto desde la IDE para examinar si tiene algún error, excepción o advertencia. La aplicación tiene que notificarle respuesta de compilación al programador, ya sea satisfactoria o denegada, que en este caso debe presentarle los errores, excepciones o advertencias ocurridas.		
Observaciones: Pensando en la facilidad de uso de la IDE por parte del programador, es importante tener en cuenta que una vez la operación de compilar emita alguna respuesta no satisfactoria, se describa la clase y la línea donde ocurrió el error, y además brindarle un acceso directo al punto exacto donde ocurrió.		
Actividades: <ul style="list-style-type: none">– Investigar acerca del compilador de Java, su comportamiento y funcionamiento.– Analizar la utilidad <i>javac</i> de la JDK, la sintaxis de sus comandos, los parámetros de entrada y las respuestas de la compilación.– Hacer pruebas con la librería disponible en la API de Java llamada JavaCompiler del paquete javax.tools.– Estudiar la sintaxis de compilación, las posibles opciones y la ubicación de los archivos fuentes a compilar.– Implementar un programa que compile clases java.– Crear un componente independiente que se encargue exclusivamente de la compilación de archivos fuente.– Integrar el componente compilador a la aplicación.– Hacer pruebas de compilación con los ficheros de un proyecto de la aplicación.		

Tabla 21. Descripción de la Historia de Usuario H1

- **Diagrama de Clases de la Iteración 3**

Para la iteración 3 fue necesario implementar un componente que se encargara exclusivamente a compilar los archivos fuentes de un proyecto. Para la implementación de este componente llamado compiler se creó un paquete que contiene las clases Facade_Compiler y su respectivo Control_Compiler. A continuación se presenta el diagrama de clases que surge para la iteración 3.

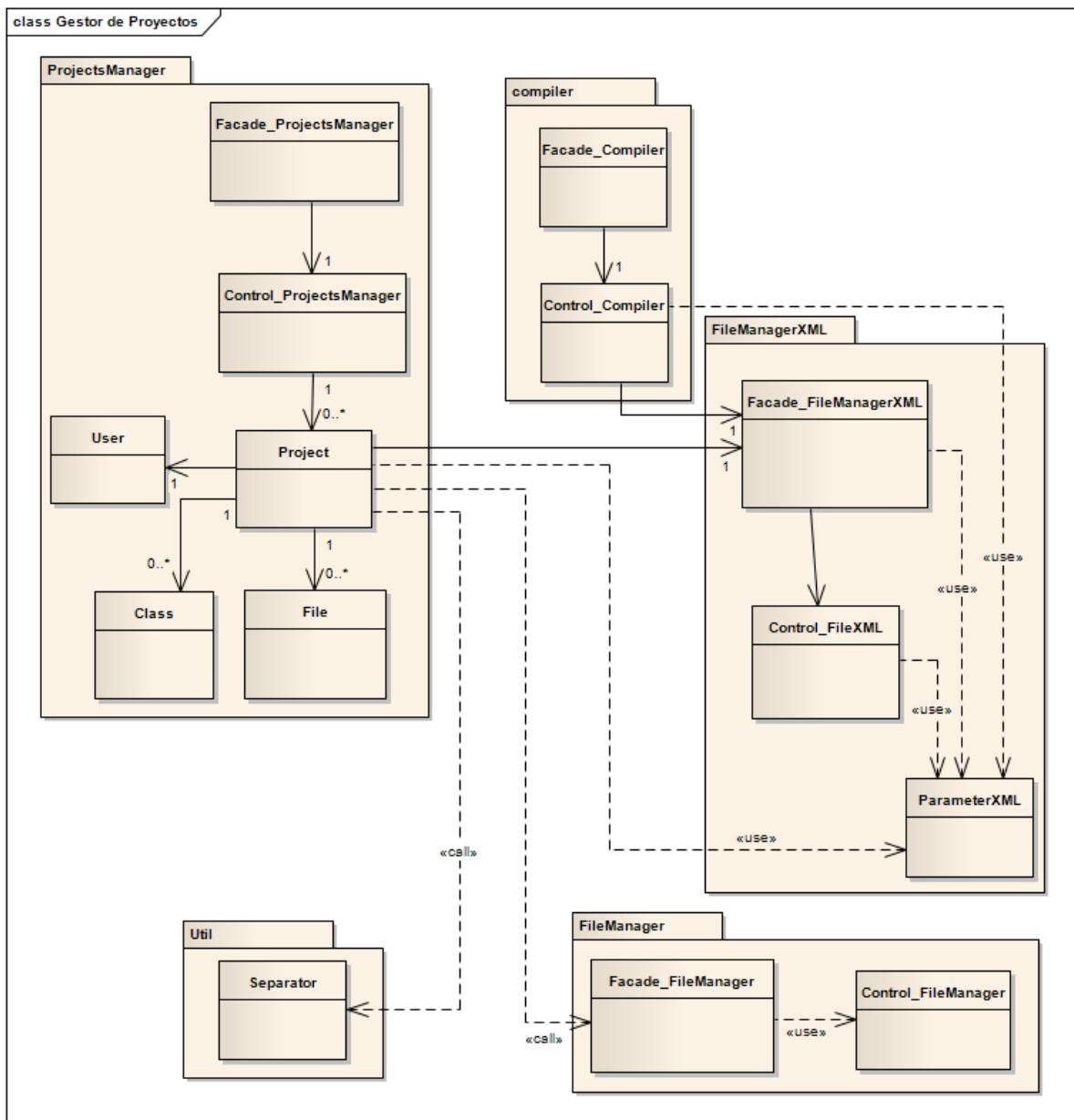


Diagrama 4. Diagrama de Clases Iteración 3

- **Prueba de Unidad**

Para el nuevo componente, el cual cumple únicamente con compilar proyectos solo requiere ciertas rutas que correspondan al proyecto de un usuario, la rutas de las librerías que usa y la ruta que contendrá los archivos .class que podría construir el proceso de compilación, siempre y cuando este sea satisfactorio.

testCompileProject(): Método que enviará a compilar un proyecto “nuevo”, en él se creó un paquete “Mundo” y en este, una clase “Principal.java”. El resultado de esta prueba es comparado con un resultado previsto para conocer si la operación es exitosa.

Resultado de la Prueba

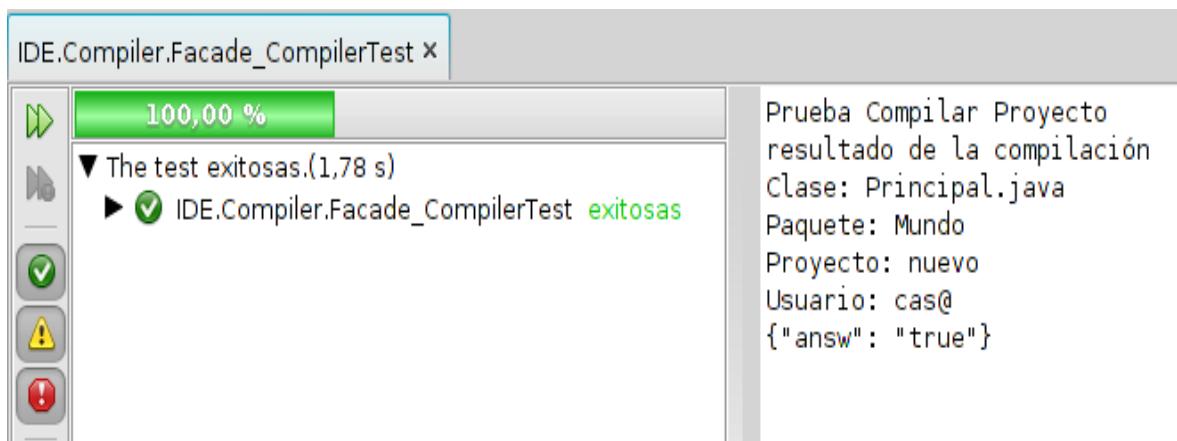


Figura 32. Resultado de Prueba de Iteración 3.

8.6.4. Iteración 4

- **Historias de Usuario Abarcadas**

Historia de Usuario	
Número: H2	Usuario: Programador
Nombre de la Historia: Construir Archivo JAR	
Prioridad en Negocio (Alta/Media/Baja): Alta	Estimación: 1

Descripción: La aplicación debe permitirle al programador construir un archivo JAR del proyecto que desee, siempre y cuando esté abierto en la IDE.

Observaciones: Para construir el archivo JAR, el proyecto se debe compilar, y si esta operación es exitosa, procede a la construcción del JAR. En caso de ocurrir alguna eventualidad que impida la construcción del archivo JAR se debe notificar al programador lo sucedido.

Actividades:

- Investigar cómo llevar a cabo las operaciones básicas con ficheros JAR, por ejemplo: crear, ver el contenido, extraer el contenido, modificar archivos manifiesto, para firmar digitalmente un archivo JAR, ejecutar una aplicación o un applet empaquetada en un archivo JAR.
- Analizar la utilidad *jar* disponible en la JDK, la sintaxis de sus comandos, los parámetros de entrada y los ficheros generados.
- Implementar ejemplos con la librería disponible en la API de Java llamada jar del paquete java.util.
- Estudiar la sintaxis de la operaciones con ficheros JAR, las posibles opciones y los resultados generados.
- Implementar un programa que construya archivos JAR.
- Crear un componente independiente que se encargue exclusivamente de la construcción de archivos JAR.
- Integrar el componente constructor de JAR a la aplicación.

Tabla 22. Descripción de la Historia de Usuario H2

• **Diagrama de Clases para H2**

Para la iteración 4, exactamente a la Historia de Usuario 2, fue necesario implementar un componente que se encargara exclusivamente de construir un archivo Jar de un proyecto. Para la implementación de este componente se creó un paquete llamado “Compressor” que contiene las clases Facade_Compressor y su respectivo Control_Compressor. A continuación se presenta el diagrama de clases que surge para la iteración 4 Historia de Usuario 2.

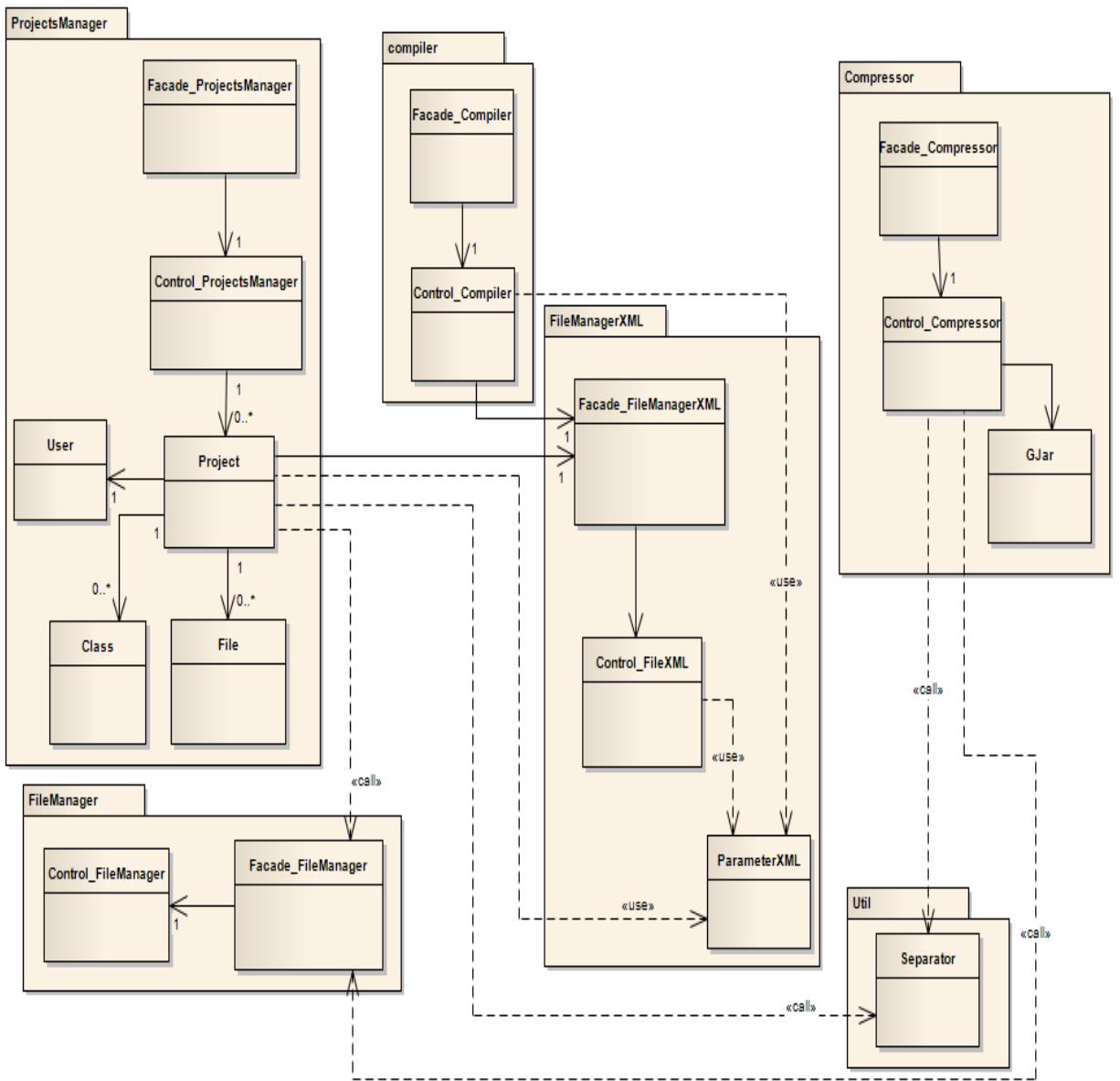


Diagrama 5. Diagrama de Clases de la Iteración 4 - H2

- **Prueba de Unidad para H2**

La prueba de unidad para la historia de usuario 2 corresponde a generar un archivo JAR que comprima los archivos involucrados en un proyecto llamado “nuevo”.

Resultado de la Prueba

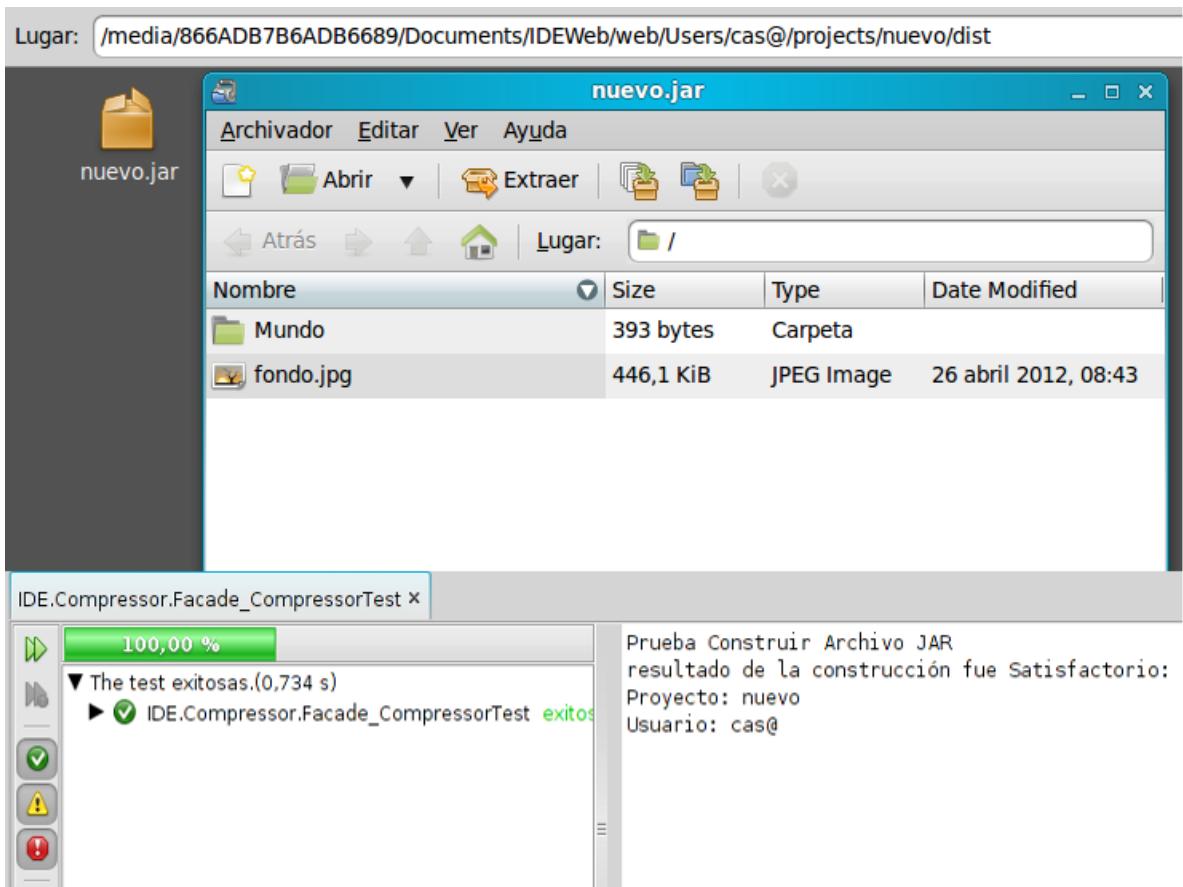


Figura 33. Resultado de Prueba de Unidad H2

Historia de Usuario	
Número: H3	Usuario: Programador
Nombre de la Historia: Ejecutar Proyecto	
Prioridad en Negocio (Alta/Media/Baja): Alta	Estimación: 2
Descripción: El programador podrá ejecutar un proyecto que tenga abierto en la IDE. Si esta operación no se lleva acabo exitosamente la aplicación debe notificarle lo que sucedió.	
Observaciones: Para llevar a cabo la ejecución, debe estar especificada en	

las propiedades del proyecto, la clase principal que contiene el método main que lanzará la ejecución.

Actividades:

- Investigar sobre el ambiente de ejecución de una aplicación sobre la plataforma de Java, identificando qué requiere, cómo procesa y qué produce la operación de ejecutar una aplicación Java.
- Determinar la secuencia de operaciones que deben efectuarse para lograr ejecutar una aplicación.
- Analizar la utilidad *java* disponible en la JDK, la sintaxis de sus comandos, los parámetros de entrada y los resultados de la operación.
- Investigar alternativas de ejecución de aplicaciones Java.
- Identificar las distintas restricciones de seguridad con respecto a la ejecución de aplicaciones Java, que los navegadores implementan para proteger a los usuarios de programas dañinos.
- Investigar la manera de evitar dichas restricciones.
- Determinar la manera más adecuada, eficiente y segura de ejecutar un programa hecho a través de la web.
- Implementar el lanzamiento de la ejecución de un programa escrito en la IDE, de manera transparente para el programador, es decir, la ejecución debe efectuarse con un simple clic a un ícono, y a partir de allí la aplicación hará lo necesario para que el programa sea ejecutado.
- Crear un componente independiente que se encargue exclusivamente de la ejecución de programas Java.
- Integrar el componente a la aplicación.

Tabla 23. Descripción de la Historia de Usuario H3

• **Diagrama de Clases para H3**

Para la desarrollo de la historia de usuario 3 fue necesario integrar los anteriores componentes creados (*ProjectsManager*, *Compiler* y *Compressor*), y construir uno que permitiera crear un certificado de autenticación para firmar el archivo JAR ejecutable, con el objetivo de acceder a la máquina virtual del programador que desea realizar la ejecución. La integración de los componentes permitió generar uno nuevo, que encapsula los servicios de los demás, este componente fue llamado “IDE”. La siguiente figura muestra el diagrama de clases resultante del desarrollo de la historia de usuario 3.

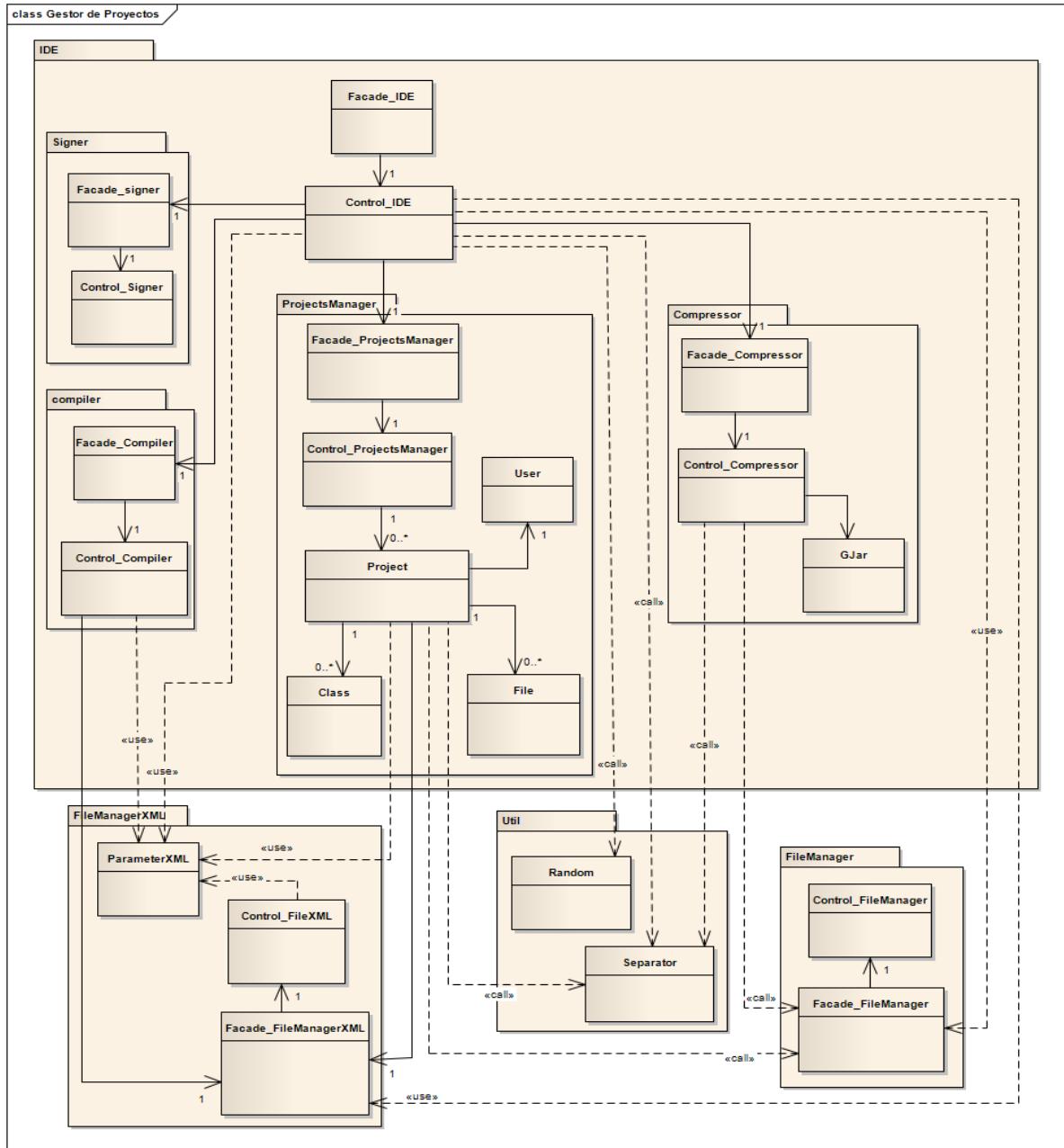


Diagrama 6. Diagrama de Clases de la Iteración 4 - H3.

• Prueba de Unidad para H3

La prueba de unidad para esta historia de usuario consiste en evaluar el comportamiento del algoritmo con respecto a dos operaciones:

- Asignar Clase Principal al Proyecto: “`testSetMainClass()`” .
- Ejecutar un proyecto: “`testExecuteProjectWEB()`”

Resultado de la Prueba

El resultado de la prueba de las dos operaciones implicadas en esta historia de usuario fue satisfactorio, y como resultado fue creado en un directorio llamado “execute” dispuesto en el proyecto, un archivo jar ejecutable que empaqueta todos los ficheros necesarios para la ejecución del mismo. La siguiente figura muestra el resultado obtenido.

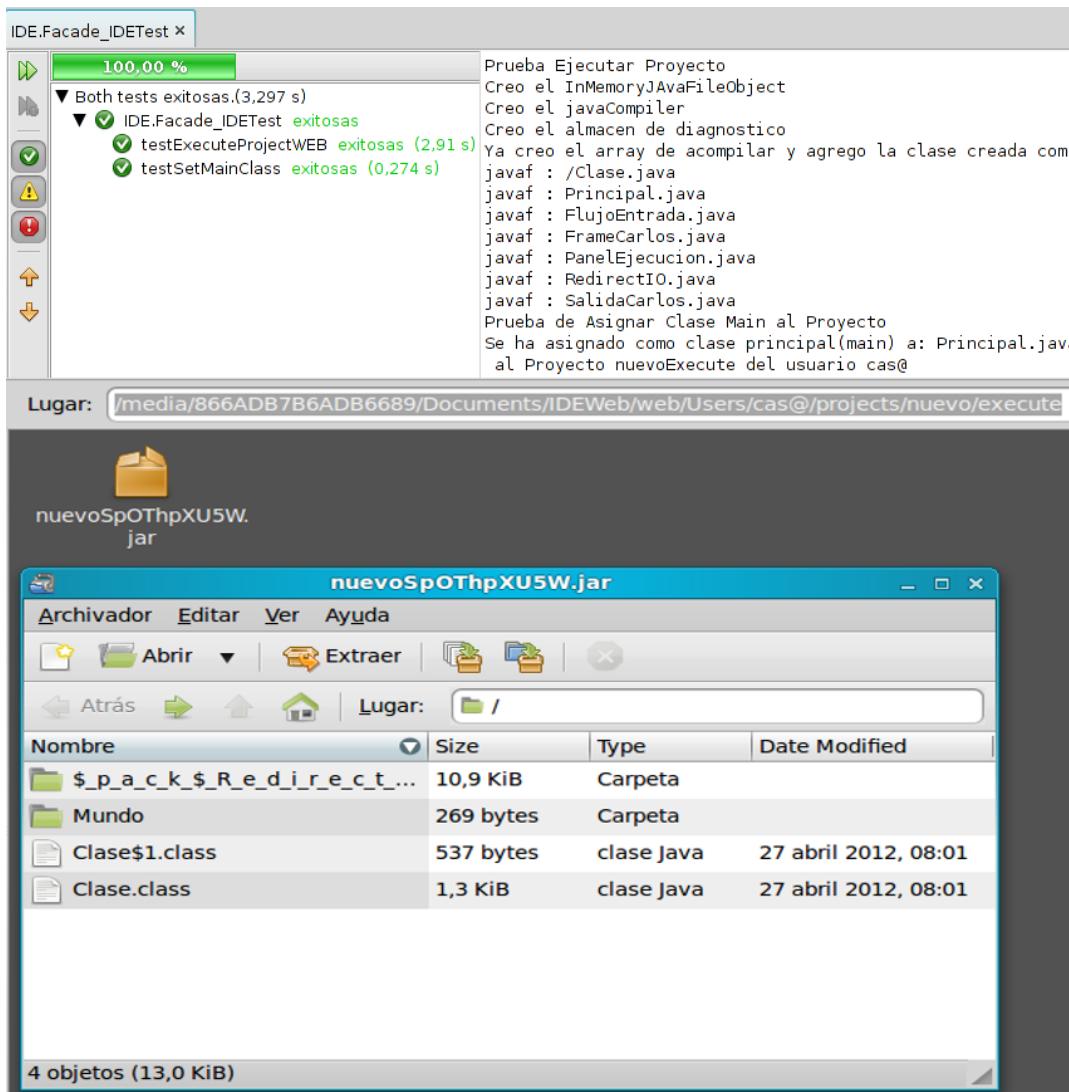


Figura 34. Resultado de prueba de Unidad H3

8.6.5. Iteración 5

- Historias de Usuario Abarcadas para

Historia de Usuario		
Número: H12	Usuario: Programador	
Nombre de la Historia: Entorno Gráfico		
Prioridad en Negocio (Alta/Media/Baja):	Media	Estimación: 3
Descripción: Es necesario que el entorno gráfico de la aplicación sea similar a un ambiente de aplicación Desktop. Por lo tanto la vista para IDE debe ser similar y adaptarse al factor común gráfico que muestra la figura 18 de la sección 4.3.2. Se requiere que el programador cuente con elementos gráficos como: un árbol de proyectos, editor con resaltador de sintaxis y formateo de código, una consola de salida para mostrar los resultados de compilación, entre otros elementos.		
Observaciones: El resaltador de sintaxis del editor de la IDE debe identificar y resaltar los componentes sintácticos del lenguaje de programación Java. Cada componente sintáctico debe corresponder a un color que los identifique, y esta operación debe llevarse a cabo durante la edición, de manera que los cambios de los colores de los caracteres escritos se reflejen de inmediato.		
Actividades: <ul style="list-style-type: none">– Buscar librerías escritas en JavaScript que permitan obtener elementos gráficos similares a las de una aplicación desktop.– Buscar, ajustar e integrar componentes que permitan vistas interactivas de árboles de directorios.– Buscar, ajustar e integrar componentes que permitan vistas interactivas de pestañas de paneles. Estos paneles posteriormente contendrán un editor que corresponda al contenido de una clase o archivo plano que haya sido abierto e incluso podría presentar un lienzo para construir interfaces gráficas de usuario.– Buscar componentes desarrollados por terceros que permitan resaltar sintaxis de Java.– Realizar ejemplos con los componentes encontrados, para identificar las características y funcionalidades que puedan ser útiles para el propósito		

- de la IDE.
- Analizar la estructura HTML generado por cada componente, pensando en cuál de estos será el más apropiado para la integración de la edición concurrente.
 - Estudiar la documentación y revisar el código fuente del componente con el fin identificar las funcionalidades con las cuales cuenta, y así lograr adaptarlo a lo que se necesita.
 - Implementar un editor acoplando los dos componentes (pestañas, árbol de proyectos), lo cual requiere efectuar algunos ajustes que permitan que operen en conjunto generando un nuevo componente.
 - Implementar la operación de abrir una clase.
 - Ajustar la operación de cerrar una pestaña o clase, ya que esto implica tener en cuenta tanto el componente de pestañas como el del árbol de proyectos, para reflejar el cambio.
 - Implementar las funciones básicas para un editor: deshacer, rehacer, seleccionar todo, buscar, buscar y reemplazar, aumentar el tamaño de la fuente.
 - Verificar que si una clase ha sido abierta en el editor, e intentan abrirla de nuevo, no lleve a cabo nuevamente la operación de abrirla, de manera que solo le dé el foco a la pestaña que la contiene. Esto es necesario para evitar rutinas innecesarias.
 - Implementar el diseño de los elementos gráficos para el editor, pero es necesario, que los colores que se utilicen puedan ser modificados con facilidad por motivos de ajustes institucionales.
 - Es necesario que el editor sea un componente independiente, de manera que acoplarse fácilmente con el resto de la aplicación, y que además pueda ser reutilizado.

Tabla 24. Descripción de la Historia de Usuario H12

- **Modelo JavaScript del entorno gráfico**

Para el desarrollo de la interface gráfica de usuario y su comportamiento se tuvo en cuenta JavaScript como lenguaje adecuado para esta historia de usuario. Javascript dejo de ser aquel lenguaje usado para validar formularios, y tomó importancia en el desarrollo de software para la web, de manera que muchas tecnologías, componentes y librerías han surgido. Para el escenario gráfico del proyecto se tomaron en cuenta ciertas librerías JavaScript que permiten crear ambientes gráficos de escritorio, como es el caso de DHTMLX SUITE (www.dhtmlx.com) , y para el editor se acogió una librería JavaScript llamada ace (www.ace.ajax.org) del cual se logró tener colaboración del grupo Ace Internals Dev

(<http://groups.google.com/group/ace-internals>) para llevar a cabo ciertas modificaciones al código fuente de la librería. El resultado del manejo del entorno gráfico se muestra en la siguiente figura:

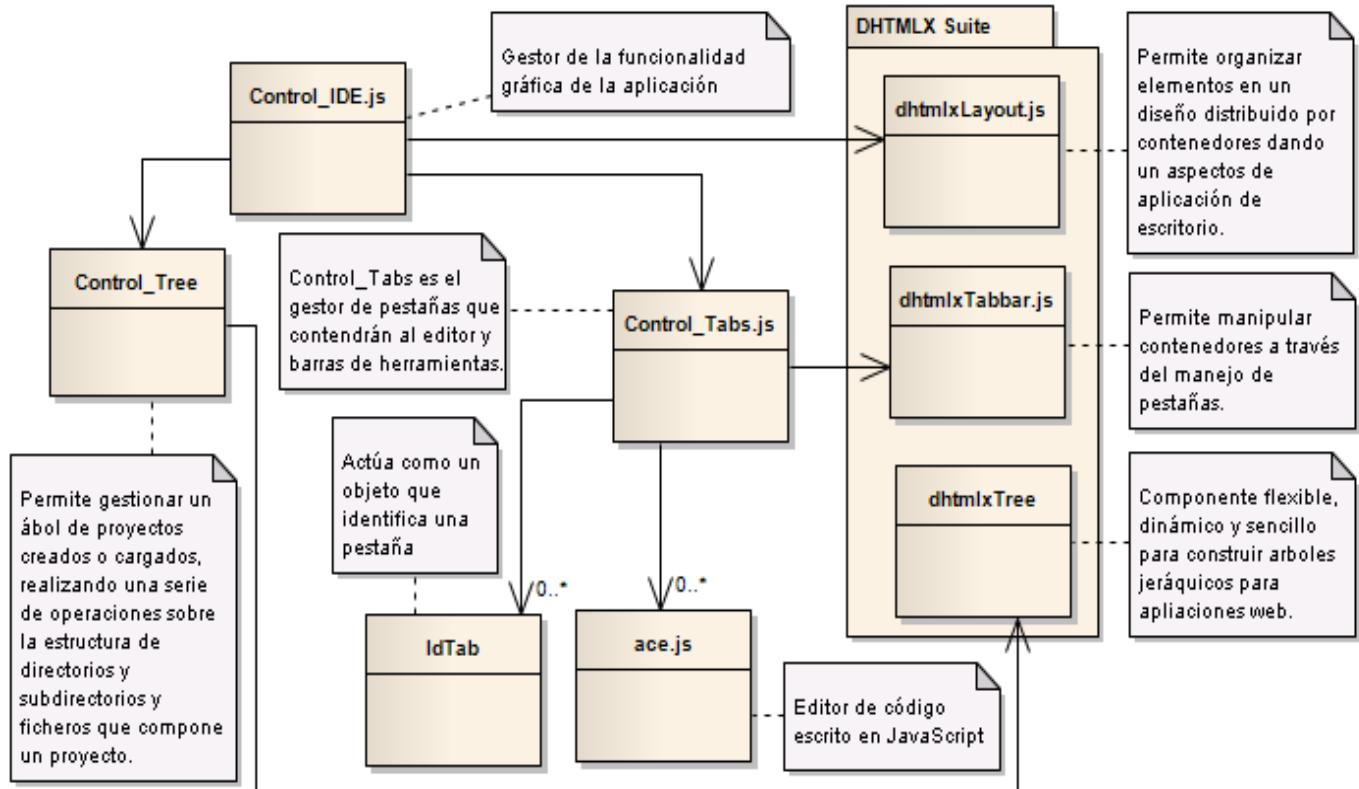


Diagrama 7. Diagrama JavaScript Entorno Gráfico

Es importante tener en cuenta que este diagrama no debe ser visto como un diagrama de clases como tal, pero se sabe que con JavaScript se puede implementar la orientación a objetos. Con la implementación de archivos JavaScript (Control_IDE, Control_Tree, Control_Tabs y IdTab), junto con la inclusión y adaptación de la librería ace.js y tres de los componentes de la librería DHTMLX Suite en su versión libre, se logró manejar e integrar el comportamiento y funcionalidad del entorno gráfico, como se muestra en la figura 35

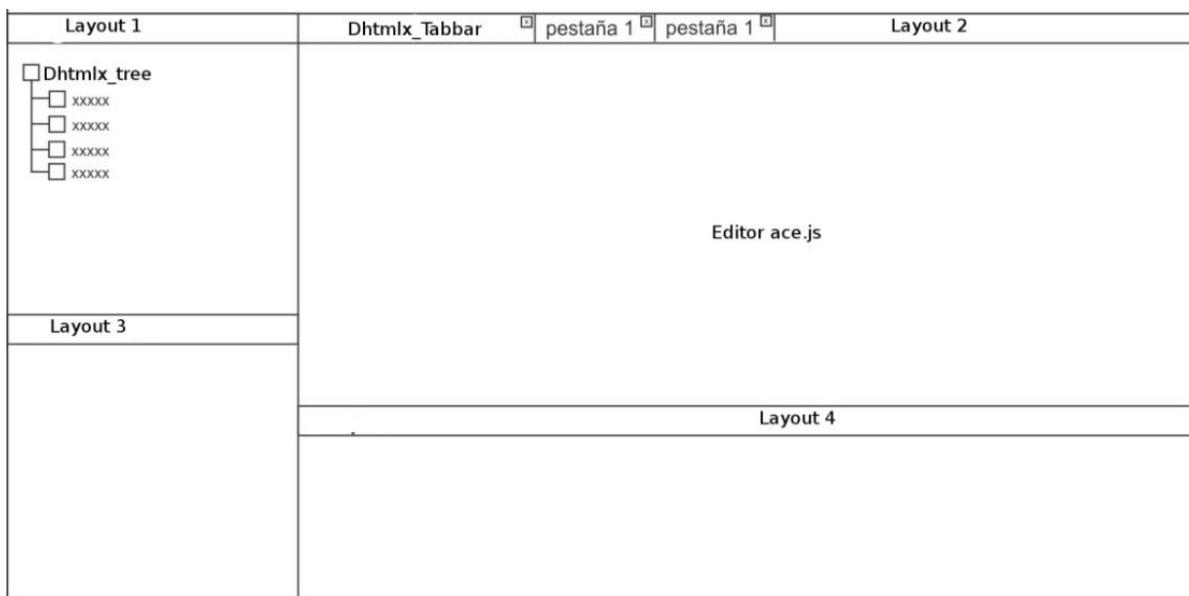


Figura 35. Composición gráfica de la IDE

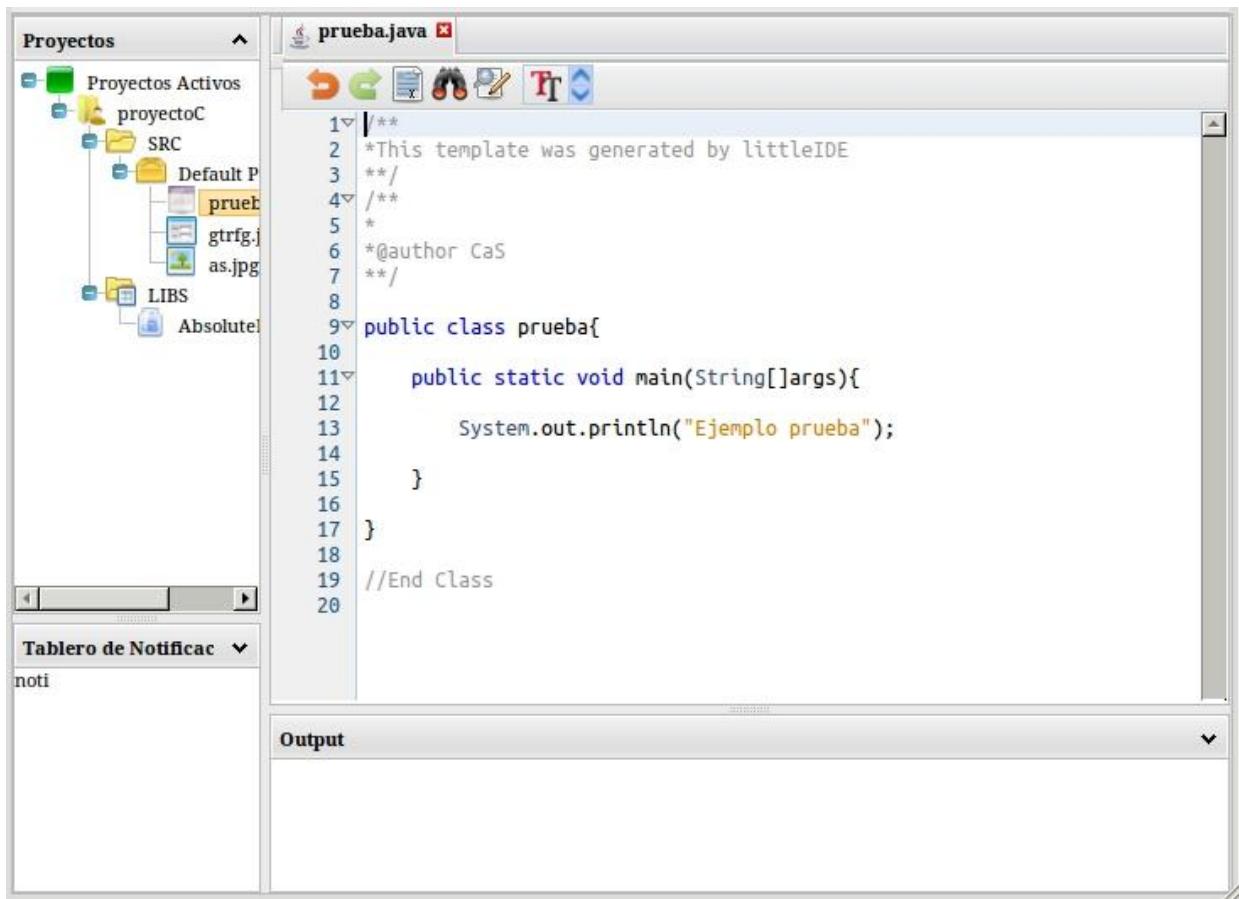


Figura 36. Resultado del Entorno Gráfico

8.6.6. Iteración 6

- Historias de Usuario Abarcadas

Historia de Usuario	
Número: H4	Usuario: Visitante
Nombre de la Historia: Crear Cuenta de Programador	
Prioridad en Negocio (Alta/Media/Baja):	Media
Descripción: El visitante del portal podrá crear una cuenta de programador, ingresando su nombre, correo electrónico, y contraseña. Ahora, si el visitante interno olvida o han descubierto su contraseña, la aplicación debe permitir restablecerla, permitiéndole una modificación.	
Observaciones: El visitante no debe ingresar a la aplicación hasta que no active su cuenta, para llevar a cabo esta operación debe dirigirse al correo electrónico suministrado a la aplicación y abrir un correo enviado por la aplicación con un link de activación. La aplicación debe implementar una estrategia de seguridad que gestione las cuentas de programadores, para posibles ataques de vulnerabilidad de acceder a sus datos privados.	
Actividades: <ul style="list-style-type: none">– Implementar una rutina que permita registrar temporalmente una cuenta de programador, verificando que no exista en el contexto de la aplicación, y a espera de que sea activada. Esta cuenta temporal debe tener un identificador que permita validarla cuando se pretenda activar.– Implementar una rutina que una vez se haya creado una cuenta temporal, envíe un mensaje al correo electrónico registrado, que contenga un link de activación.– Implementar una rutina que verifique la validez del link de activación, y que posteriormente saque la cuenta del estado temporal y la active, a partir de allí el nuevo programador puede ingresar al sistema.– Implementar una rutina que permita restablecer la contraseña de la cuenta de un programador a través del correo electrónico registrado.– Es necesario encriptar la contraseña registrada, e implementar una rutina que gestione y verifique las contraseñas encriptadas para la seguridad de acceso a la aplicación.	

Tabla 25. Descripción de la Historia de Usuario H4

- **Diagrama de Clases de H4**

Para esta historia de usuario se creó un paquete llamado *WebSite* que permite llevar a cabo la gestión de las cuentas de programador. Este paquete es independiente al paquete *IDE*, pero comparte clases utilitarias y anexa una al paquete *Util*, llamada *SendMail* que encargará de envío de correos electrónicos. Ahora, con respecto a la encriptación de contraseña se creó un nuevo paquete utilitario llamado *Encrypter* que se encargará de las operaciones de encriptado y desencriptado. El diagrama 8 muestra el diagrama de clases que resulta de esta iteración.

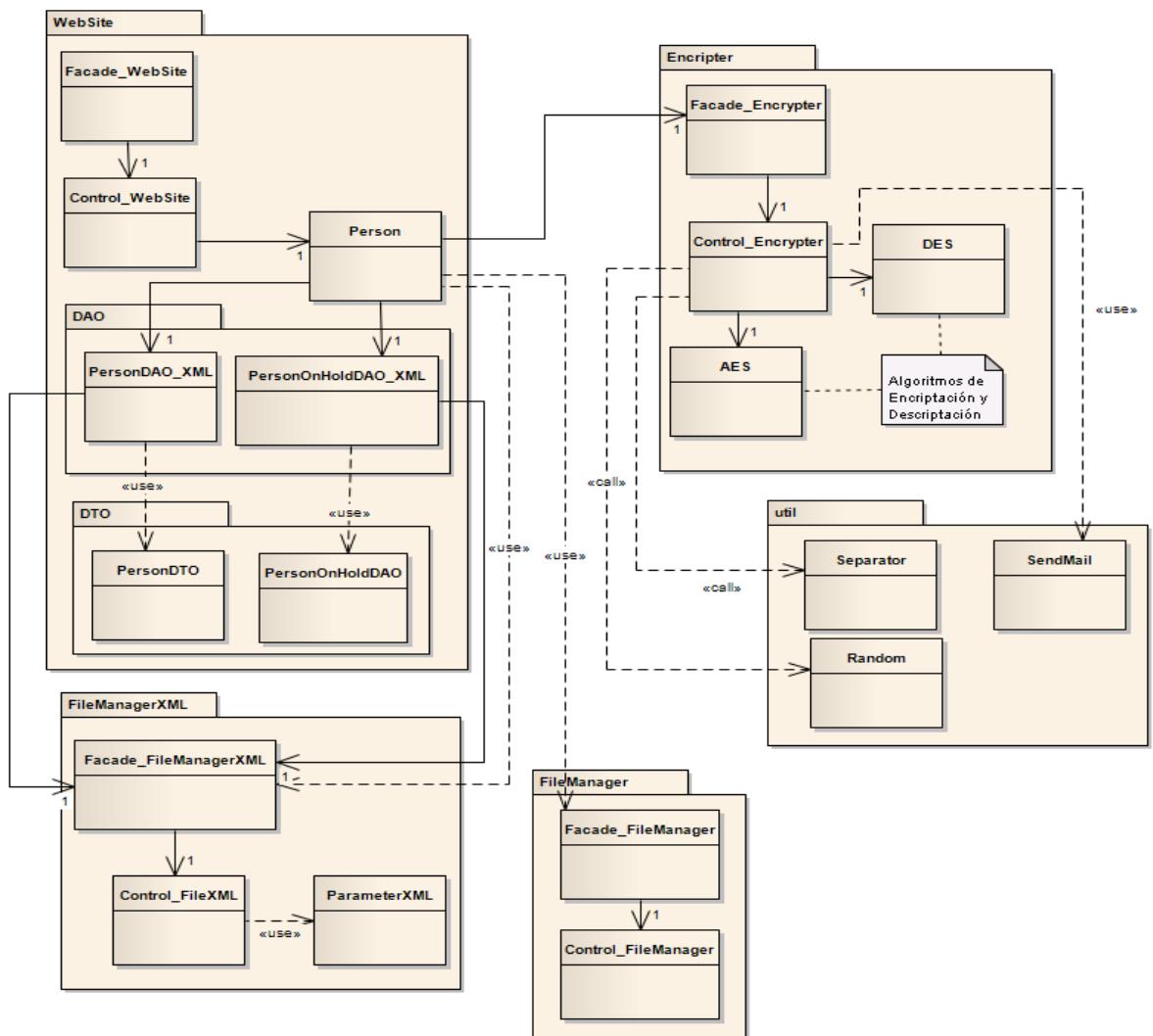


Diagrama 8. Diagrama de Clases de Iteración 6 H4

- **Prueba de Unidad para H4**

La prueba de unidad que corresponde a esta historia de usuario, evalúa las siguientes operaciones:

- *testRegisterProgrammer()*: Test que permite crear un registro temporal a espera de activación, por parte del visitante. El algoritmo consisten en procesar los datos ingresados en un formulario, encriptar la contraseña, almacenar un usuario en un archivo XML de temporales y posteriormente enviar un mensaje que contenga un link de activación, al correo electrónico ingresado.
- *testCreateProgrammer()*: Test que crea una cuenta de programador. Esta operación es activada cuando el visitante desde la bandeja de entrada de su correo electrónico, da clic en el link de activación. Posterior a esto, a través de un identificador, busca, obtiene y elimina el usuario en el XML de temporales, toma sus datos, los almacena en un XML de programadores y crea los respectivos directorios para el Programador.
- *testRestorePassUser()*: Test que permite restaurar la contraseña de un programador.

- Resultado de la Prueba

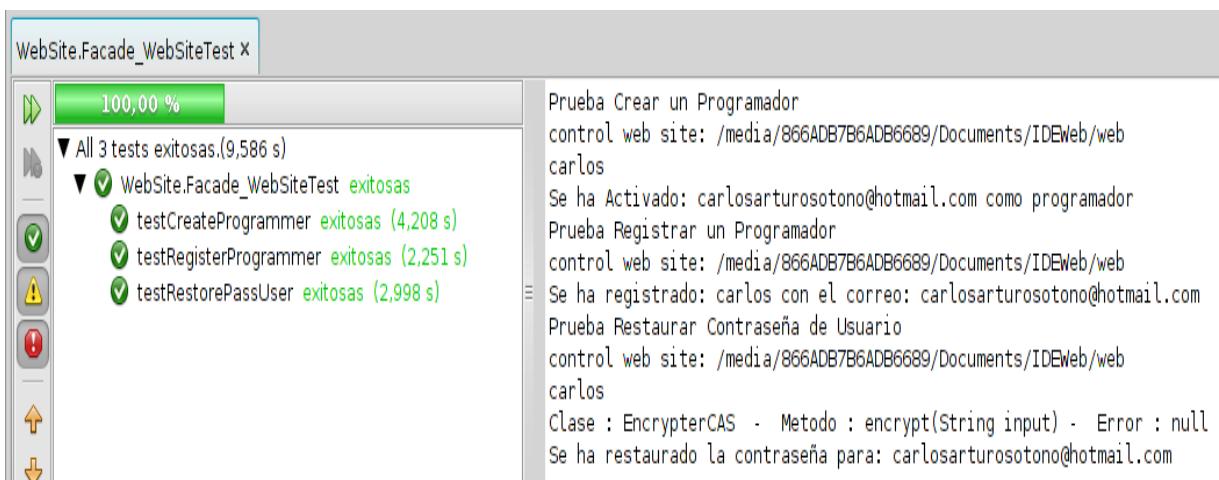


Figura 37. Resultado Prueba de Unidad H4

Historia de Usuario	
Número: H5	Usuario: Visitante Interno
Nombre de la Historia: Gestionar Sesiones	
Prioridad en Negocio (Alta/Media/Baja): Media	Estimación: 1
<p>Descripción: El visitante interno podrá iniciar sesión en la aplicación, ingresando su correo electrónico y su contraseña en un formulario. La aplicación debe permitirle al visitante interno cerrar su sesión, ya sea a través de un clic, o al cerrar el navegador.</p>	
<p>Observaciones: La aplicación debe controlar que dos visitantes no usen la aplicación al mismo tiempo con la misma cuenta.</p>	
<p>Actividades:</p> <ul style="list-style-type: none"> – Implementar una rutina que tome un correo y contraseña ingresados en un formulario, los verifique en el contexto de la aplicación, desencriptando la contraseña almacenada y comparándola con la que se ingresó. Si la verificación es válida se crea una sesión. – Implementar un gestor de sesiones encargado de mantener una sesión activa por cuenta, es decir, que no permita concurrencia de sesiones en una misma cuenta. – Adicionar un atributo al archivo XML de las cuentas de programador, que permita almacenar un identificador que corresponda a la sesión activa. Si este identificador cambia mientras el programador este en la aplicación, indica que se ha iniciado sesión en otro lugar. Cuando el programador intente hacer una nueva operación dentro de la aplicación, se debe verificar que el identificador no haya cambiado; ahora, si esto sucede, su sesión termina y es retornado al index. – Implementar rutinas que permitan cerrar una sesión a través de un enlace, también es necesario tener en cuenta que si el navegador es cerrado y aún existen sesiones activas en la aplicación, deben ser terminadas. 	

Tabla 26. Descripción de la Historia de Usuario H5

- **Diagrama de Clases para H5**

El Diagrama de clases para esta historia de usuario se mantiene, pero se han agregado al contexto del paquete `WebSite` unos métodos que permitirán gestionar las sesiones:

- `isValidPerson()`: Permite verificar si existe en la persistencia de la aplicación, una persona que en este caso es un posible programador. Si existe retornará `true` de lo contrario `false`.
- `isValidAdmin()`: Cumple la misma función que `isValidPerson()` pero para el Administrador del sistema.
- `openSession()`: Permite crear una sesión con un identificador para posteriores validación de seguridad.
- `closeSession()`: Permite terminar la sesión de un programador o el administrador.
- `isValidSession()`: Permite verificar que el identificador almacenada de la sesión coincide en un momento dado con el identificador de la sesión activa, esto permite mantener una sesión activa por cuenta, para evitar que exista más de una sesión por programador al tiempo.

- **Prueba de Unidad H5**

La prueba de unidad para esta historia de usuario es evaluar el funcionamiento de las nuevas operaciones agregadas.

Resultado de la Prueba

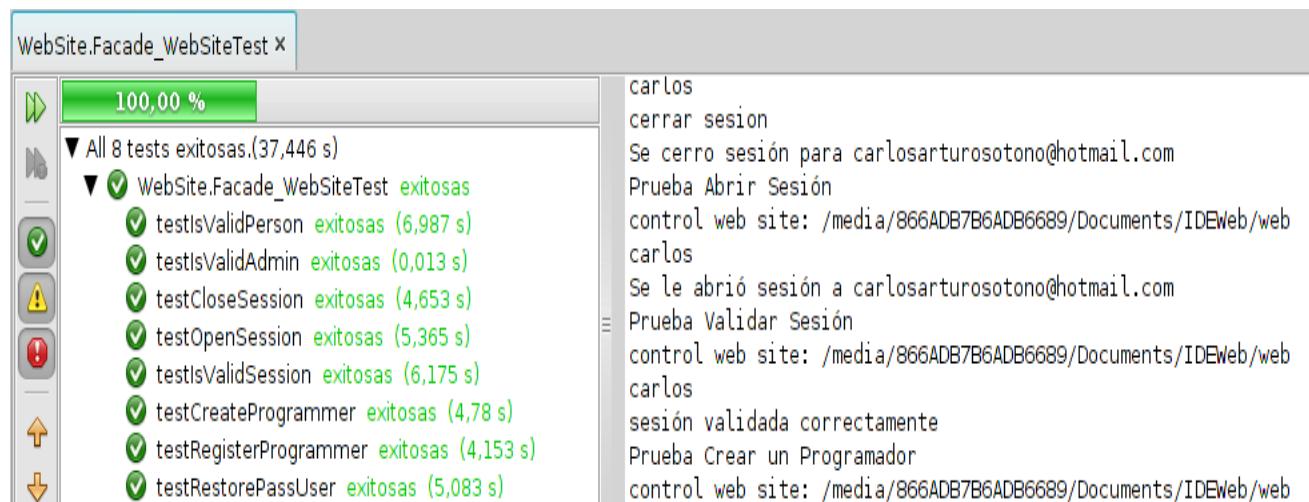


Figura 38. Resultado de Prueba de Unidad H5

8.6.7. Iteración 7

- Historias de Usuario Abarcadas

Historia de Usuario	
Número: H18	Usuario: Administrador del Sistema
Nombre de la Historia: Consultar Accesos a la Aplicación	
Prioridad en Negocio (Alta/Media/Baja): Media	Estimación: 1
Descripción: La aplicación debe mantener un registro de ingreso de los programadores, que permita consultar los accesos para que el administrador pueda disponer de ellos. Esto le servirá al administrador para tomar decisiones sobre un usuario, partiendo del tiempo en el cual el programador no ha usado la aplicación, posiblemente considerándolo como no uso de la cuenta.	
Observaciones: No está estipulado el periodo de inactividad para considerar abandono de cuenta. Esto está a voluntad del Administrador del sistema, quién tendrá la facultad de decidir si una cuenta está abandonada.	
Actividades: <ul style="list-style-type: none">– Almacenar para cada programador la fecha del último ingreso. Esta se almacena una vez que el programador ha iniciado sesión y ha ingresado a la aplicación.– Implementar la manera en que el administrador pueda consultar los últimos ingresos a la aplicación antes de determinada fecha.– Implementar las rutinas que correspondan para consultar y construir un listado.	

Tabla 27. Descripción de la Historia de Usuario H18

- Diagrama de Clases H18

Para esta iteración se mantiene el diagrama de clases del paquete *WebSite*, pero se agrega al contexto del paquete un método llamado *getProgrammerByEntry()*, que permitirá obtener los usuarios que hayan ingresado antes e igual a una fecha determinada. En esta iteración se integran los dos grandes paquetes desarrollados hasta el momento (*IDE* y *WebSite*), con un paquete llamado *WEBAplication* que integra los servicios

de cada uno, para ofrecerlos a la capa vista de la aplicación. El diagrama 9 muestra el diagrama de clases resultado de la presente historia de usuario.

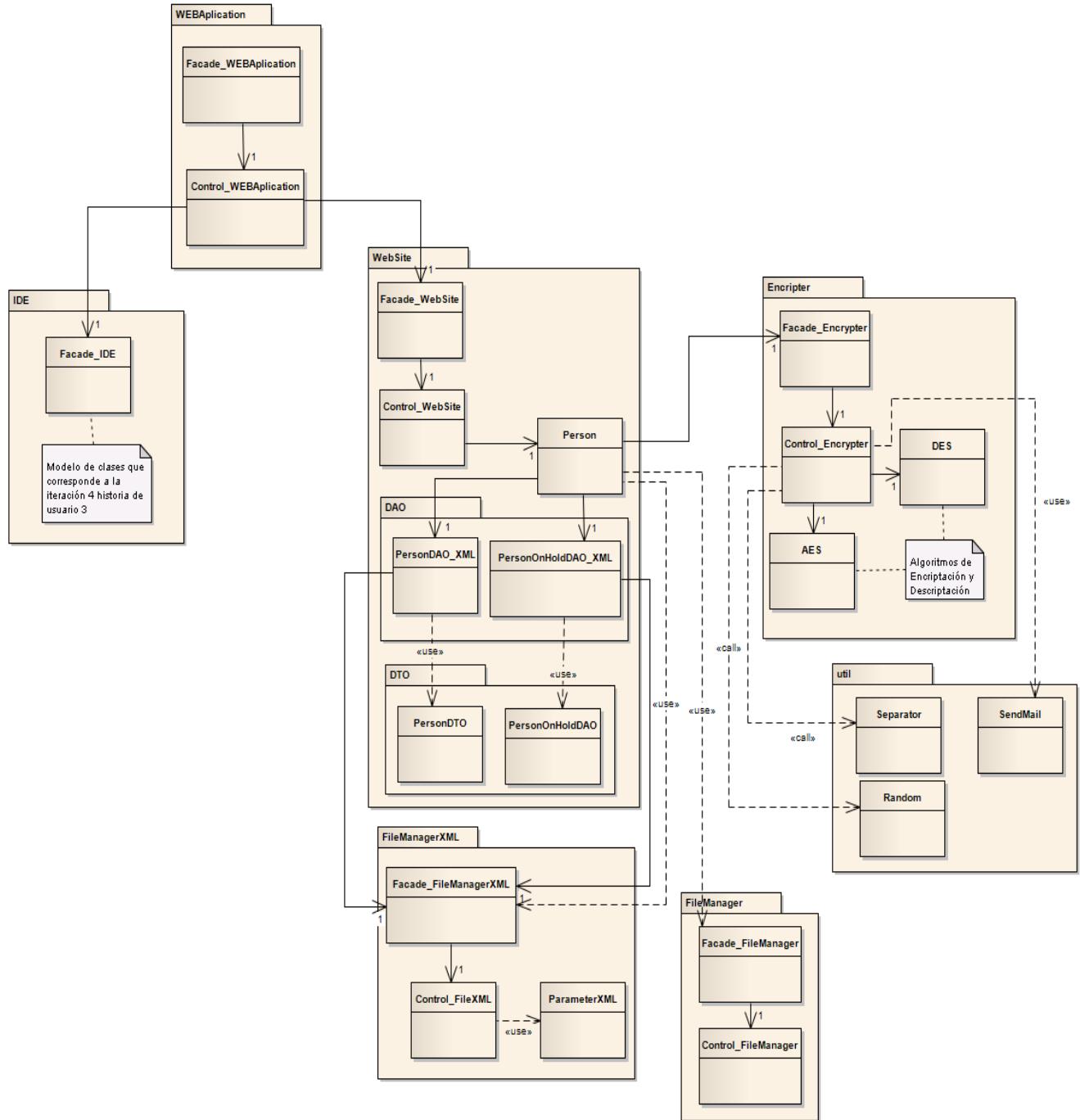


Diagrama 9. Diagrama de Clases para H18

– Prueba de Unidad H18

Para llevar a cabo la prueba de unidad de esta iteración se tuvo en cuenta el método agregado (*testGetProgrammerByEntry()*) y el resultado lo refleja la figura 39

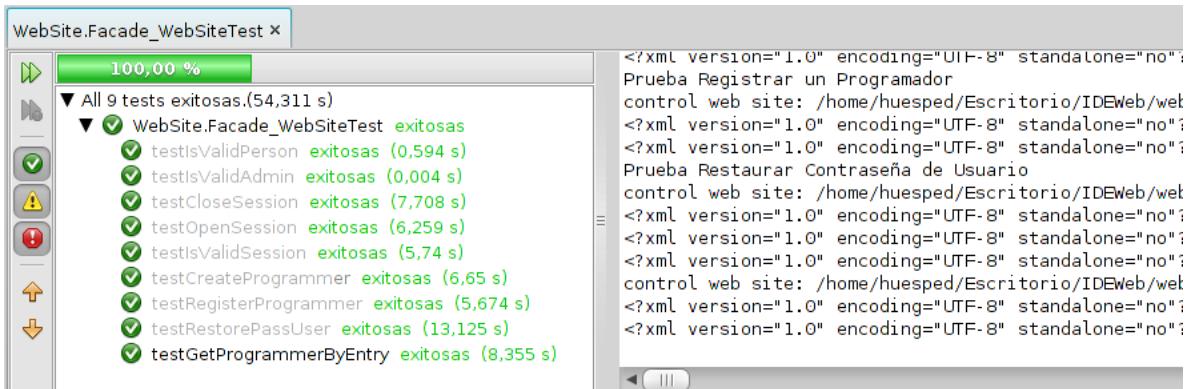


Figura 39. Resultado de Prueba de Unidad para H18.

Historia de Usuario	
Número: H19	Usuario: Administrador del Sistema
Nombre de la Historia: Gestionar Programadores	
Prioridad en Negocio (Alta/Media/Baja):	Media
Estimación: 1	
Descripción: La aplicación debe brindarle al administrador del sistema, un módulo de gestión de programadores que le permita buscar programadores, listarlos, ver sus perfiles, eliminarlos y ver sus proyectos.	
Observaciones: Eliminar un programador implica borrar cualquier rastro de el del contexto de la aplicación.	
Actividades: <ul style="list-style-type: none"> – Consultar Programadores ingresando el correo electrónico, y construir un listado. – Implementar la rutina que permitirá eliminar un programador del contexto de la aplicación. – Ver y eliminar los proyectos de un programador. 	

- Gestionar los proyectos de toda la aplicación.
- Consultar los proyectos de la aplicación y eliminarlos si así lo requiere.
- Implementar una rutina que permita editar la contraseña del administrador.
-

Tabla 28. Descripción de la Historia de Usuario H19

– Diagrama de Clases para H19

La estructura del diagrama de clases para la historia de usuario 19, no varía con respecto a la anterior, pero se agregan los siguientes métodos al contexto del paquete *WEBApplication*:

- *getProgrammers()*: Obtiene los programadores activos de la aplicación, es decir, aquellos que ya han activado su cuenta.
- *getProgrammersOnHold()*: Obtiene los programadores que están en un estado temporal a espera de que la cuenta se active.
- *deleteProgrammer()*: Permite eliminar un programador a partir de su correo electrónico, esto implica eliminar el directorio, subdirectorios y archivos dispuestos para él.
- *deleteProgrammerOnHold()*: Permite eliminar cierto visitante que se ha registrado temporalmente en la aplicación y aún se encuentra en espera de activación.
- *getProjects()*: Obtiene todos los proyectos de la aplicación.
- *getProjectsShared()*: Obtiene los proyectos compartidos de la aplicación.
- *GetProgrammerProjectsXML_Filtered()*: Obtiene los proyectos de un programador, el método permite especificar filtros como: “mis proyectos”, “proyectos que comparto”, “proyectos que me comparten” y “todos los proyectos”.

• Prueba de Unidad para H19

La prueba de unidad para esta historia de usuario corresponde a validar el funcionamiento de los métodos anteriormente mencionados.

Resultado de la Prueba

La figura 40 muestra el resultado de la prueba de unidad realizada a los métodos que corresponden a la gestión de programador.

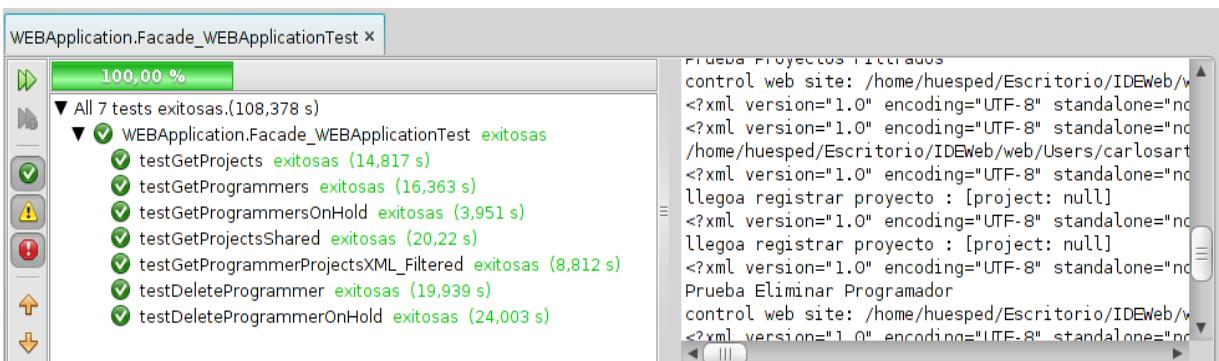


Figura 40. Resultado de Prueba de Unidad para H19.

8.6.8. Iteración 8

- Historias de Usuario Abarcadas

Historia de Usuario	
Número: H6	Usuario: Programador
Nombre de la Historia: Editar Perfil de Programador	
Prioridad en Negocio (Alta/Media/Baja): Baja	Estimación: 1
Descripción: A el Programador se le va permitir gestionar su imagen personal y contraseña de ingreso a la aplicacion.	
Observaciones: El correo electronico con el cual se registra el programador, no puede ser modificado debido a que es la llave primaria del usuario ante la aplicación.	
Actividades: <ul style="list-style-type: none"> – Implementar un método que consulte los datos que correspondan al programador. – Implementar la manera de cambiar el avatar, cargando la foto que desee el programador, redimensionarla si es necesario y almacenarla. – Implementar una rutina que le permita al usuario cambiar su contraseña, si lo desea. 	

Tabla 29. Descripción de la Historia de Usuario H6

- Diagrama de Clases para H6

La actual historia de usuario presenta las mismas clases que se observan en el diagrama 9, pero con la incorporación de los siguientes métodos al contexto del paquete *WEBApplication*:

- *restorePassUser(String id, String np)*: Este método permite identificar el programador con sesión activa, para el cambio de contraseña, mientras que *np* contiene el dato de la nueva contraseña, que posteriormente pasara por una rutina que la encripta y luego se edita el valor en el correspondiente archivo xml.
- *uploadPhoto(String user, HttpServletRequest request, String field, String[] extension, String[] noExtension)*: Este método permite actualizar la imagen avatar del programador. *User* indica a quién se le va a realizar la operación, *request* objeto que lleva la información de la petición multiparte, *field* indica el campo que corresponde a la imagen, *extension* indica las extensiones de archivo válidos y *noExtension* indica todas las extensiones no válidas.

• Prueba de Unidad para H6

La prueba de unidad para esta historia de usuario corresponde a validar el funcionamiento de los métodos que fueron agregados.

Resultado de la Prueba

La figura 41 muestra el resultado de la prueba de unidad realizada a las funcionalidades agregadas, que corresponden a la edición del perfil del programador.

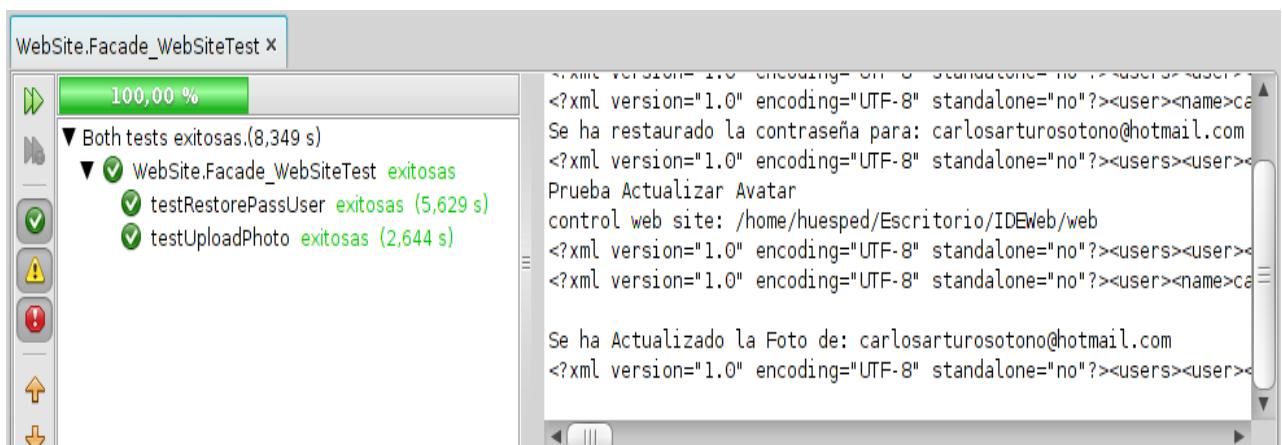


Figura 41. Resultado de prueba de Unidad para H6

Historia de Usuario		
Número: H8	Usuario: Programador	
Nombre de la Historia: Compartir Proyecto		
Prioridad en Negocio (Alta/Media/Baja):	Baja	Estimación: 1
<p>Descripción: Un programador podrá compartir un proyecto con otros programadores, siempre y cuando sea el propietario. El programador que comparte asigna los privilegios que los demás tendrán sobre el proyecto (solo lectura o lectura/escritura). Si un proyecto es compartido con privilegios de lectura/escritura, todos los programadores que tengan acceso de este tipo, podrán realizar modificaciones de manera concurrente, en caso de tener privilegios de solo lectura, es posible hacer uso de la concurrencia pero solo en la visualización de cambios.</p>		
<p>Observaciones: El proceso de dejar de compartir puede ser realizado tanto por el autor, como el programador a quien se le compartió, es decir, el autor decide a quién o quienes dejar de compartir, o un programador al cual se le ha compartido un proyecto podrá decidir abandonarlo.</p>		
<p>Actividades:</p> <ul style="list-style-type: none"> – Implementar una rutina que permita compartir un proyecto a un programador a través de su correo electrónico, y asignarle privilegios. – Actualizar los archivos de configuración XML que correspondan a proyectos tanto del autor o propietario, como el programador al cual se le comparte. 		

Tabla 30. Descripción de la Historia de Usuario H8

– **Diagrama de Clases para H8**

La estructura del diagrama de clases para la historia de usuario 8, mantiene las mismas clases de la Figura 55, pero se agregan los siguientes métodos al contexto del paquete *WEBAplication* que involucra el gestor de proyectos:

- `shareProject(String name, String owner, String user, String type)`: Este método es invocado para cada programador a quién se le va a compartir el proyecto. Los parámetros corresponden a: `name` nombre del proyecto, `owner` programador dueño del proyecto, `user` es el programador a quien se le compartirá y `type` es el privilegio otorgado al programador sobre ese proyecto, este último puede ser Escritura/Lectura o solo Lectura.

Esta historia de usuario podrá ser llevada a cabo desde dos vistas diferentes, la primera corresponde a la vista inicial o home del programador, visualizada inmediatamente después del proceso de satisfactorio de `login`, y la segunda hace referencia a la vista correspondiente a la `IDE`, la cual permite compartir un proyecto desde el árbol de proyectos.

- **Prueba de Unidad para H8**

La prueba de unidad para esta historia de usuario corresponde a validar el funcionamiento del método que fue agregado.

Resultado de la Prueba

La figura 42 muestra el resultado de la prueba de unidad realizada a la funcionalidad agregada, que corresponden a compartir un proyecto. En este caso el programador `email@email.com` crea un proyecto llamado “proyecto”, y se lo comparte al programador “`email@email.com1`” con privilegios de Escritura.

```

WEBApplication.Facade_WEBApplicationTest x
100,00 %
e test exitosas.(8,644 s)
    WEBApplication.Facade_WEBApplicationTest exitosas
        testShareProject exitosas (8,466 s)

<?xml version="1.0" encoding="UTF-8" standalone="no" 
is valid person persona
login-contraseña-9gc+xKNZ1bQFQ7AUmK04qLy5TaFmmdkzKXZ
/home/huesped/Escritorio/IDEWeb/web/Users/email@email.com
<?xml version="1.0" encoding="UTF-8" standalone="no" 
llegoa registrar proyecto : [project: null]
<?xml version="1.0" encoding="UTF-8" standalone="no" 
llegoa registrar proyecto : [project: null]
<?xml version="1.0" encoding="UTF-8" standalone="no" 
Proyecto proyecto ha sido compartido
a email@email.com1 con privilegio de Escritura

```

Figura 42. Resultado de Prueba de Unidad para H8.

Historia de Usuario		
Número: H9	Usuario: Programador	
Nombre de la Historia: Descargar Proyecto		
Prioridad en Negocio (Alta/Media/Baja): Baja		Estimación: 1
<p>Descripción: El programador cuenta con dos opciones de descarga de proyectos que haya creado o que tenga compartido. Una corresponde a descargar el ejecutable del proyecto, y la otra a la descarga de todo el proyecto, después de esto se iniciará la descarga.</p>		
<p>Observaciones: Para obtener el ejecutable del proyecto la aplicación proporcionará un archivo JAR con su fichero mafiesto editado. Y Para obtener el contenido del proyecto, la aplicación empaquetará lo que corresponda, en un archivo formato ZIP.</p>		
<p>Actividades</p> <ul style="list-style-type: none"> – Implementar una rutina que permita descargar el proyecto en formato ZIP . – Implementar rutina que permita descargar el ejecutable del proyecto. 		

Tabla 31. Descripción de la Historia de Usuario H9

– Diagrama de Clases para H9

La estructura del diagrama de clases para la historia de usuario 9, presenta las mismas clases que se observan en la Figura 55, pero se agregan los siguientes métodos al contexto del paquete *WEBAplication* que involucra el gestor de proyectos:

- *compressExecutableProject(String name, String owner)*: Este método permite comprimir en un fichero formato ZIP, el fichero .JAR correspondiente al ejecutable del proyecto, junto con un directorio *libs* que contiene las librerías importadas por el programador para el correcto funcionamiento del proyecto.
- *compressProject(String name, String owner)*: A diferencia del anterior método, este permite comprimir en un fichero formato ZIP, todos los

ficheros pertenecientes a un proyecto, con excepción de aquellos que son utilizados por la aplicación para procesos de configuración y correcto funcionamiento(directorios *config*, *buildWEB*, *execute*). Los parámetros *name* y *owner* corresponde al nombre del proyecto y propietario del mismo, respectivamente.

- **Prueba de Unidad para H9**

La prueba de unidad para esta historia de usuario corresponde a validar el funcionamiento de los métodos agregados.

Resultado de la Prueba

La figura 43 muestra el resultado de la prueba de unidad realizada a las funcionalidades agregadas, que corresponden a descargar un proyecto, ya sea ejecutable o el directorio. En este caso el programador email@email.com crea un proyecto llamado “proyecto”, y lo desea descargar como un ejecutable; el programador email2@email2.com crea un proyecto con el mismo nombre que el anterior y desea descargar el directorio completo del mismo.

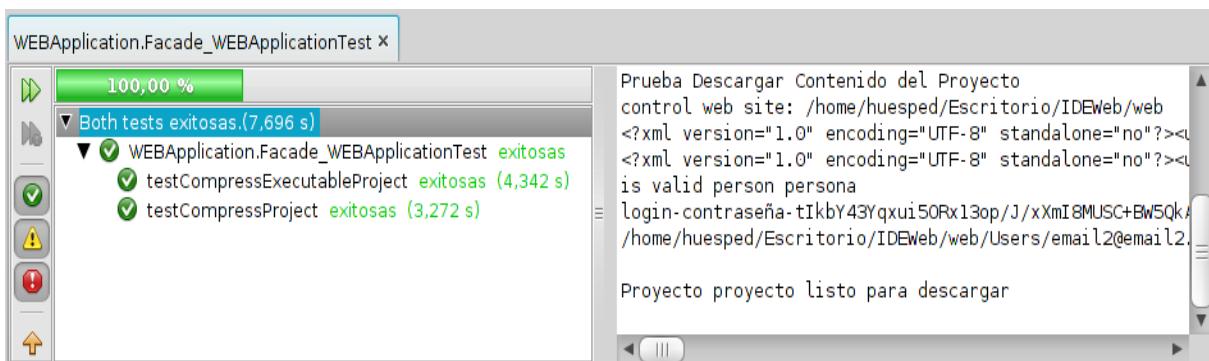


Figura 43. Resultado de Prueba de Unidad para H9.

8.6.9. Iteración 9

Historia de Usuario	
Número: H14	Usuario: Programador
Nombre de la Historia: Edición Concurrente	
Prioridad en Negocio (Alta/Media/Baja): Baja	Estimación: 3

Descripción: La aplicación debe permitir la edición colaborativa, a través de la participación de dos o más programadores editando en un proyecto en tiempo real.

Observaciones: La edición concurrente es permitida siempre y cuando esten compartiendo un proyecto y los participantes excepto el propietario deben tener privilegios de escritura.

Actividades

- Investigar y analizar sobre Transformación Operacional (OT).
- Buscar tecnologías y algoritmos que implementen OT.
- Analizar, estudiar, y probar implementaciones encontradas.
- Sobrescribir y adaptar la tecnología a los componentes a la aplicación.
- Hacer pruebas de concurrencia, editando en tiempo real.
- Acoplar OT con el editor.

Tabla 32. Descripción de la Historia de Usuario H14

– Diagrama de Clases para H14

El resultado de esta iteración es un valor agregado a la interacción del programador con la IDE. La Edición concurrente es una funcionalidad dinámica y colaborativa, donde dos o más programadores que comparten un proyecto con privilegios de escritura, pueden estar editando en una mismo código fuente o diferentes en tiempo real, sin que esto implique actualización de versiones, para que se reflejen los cambios para todos los programadores. A diferencia de un desarrollo de subversiones, la edición concurrente actualiza en todo momento los cambios que se efectúen en el proyecto, para que todos los programadores tengan siempre una versión actualizada (esto es totalmente automático y transparente para el programador). Para que esto fuese posible, fue necesario integrar: una implementación de Transformación Operacional (OT) llamada *ShareJS* (*más adelante presentado en detalle*), un editor con resaltador de sintaxis llamado ACE, servidor NodeJS y Socket.io. A raíz de que dicha interacción es implementada en Javascript, a continuación se presenta de manera genérica el flujo general de dicho proceso.

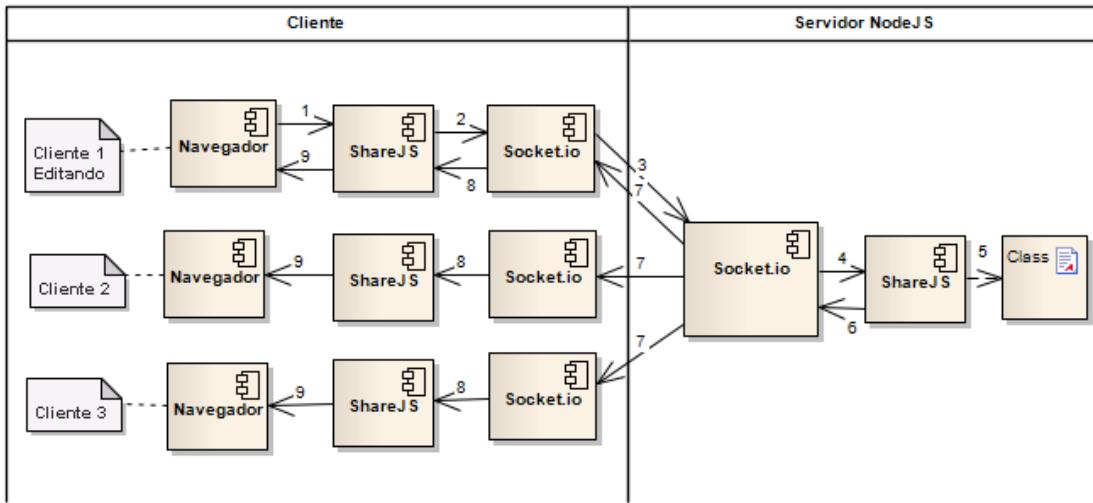


Diagrama 10. Flujo General de Edición Concurrente.

El proceso consiste en recibir e inmediatamente actualizar a todos el cambio. El diagrama 10 muestra 3 clientes que participarán en el proceso, el primero de ellos es quién ha editado, por lo tanto ShareJS tomará el control del evento y a través de socket.io todo en el lado del cliente notificará al socket.io del lado del servidor, quién le dirá a ShareJS que hay una nueva versión de la clase, y él atenderá controladamente las peticiones de cada edición , escribirá en el archivo de la clase, y devolverá la versión al socket.io del servidor quien se encargará de hacer un *broadcast* a todos los socket.io del lado cliente que este involucrados con dicho archivo, de manera que estos recibirán la respuesta y la pasarán al ShareJS, quien hará los cambios en la vista.

- **Prueba de Unidad para H9**

Para llevar a cabo la prueba de unidad para esta historia es necesario implementar una lista de chequeo, en donde se describirán una serie de funciones con respecto a la edición concurrente, con su respectiva caja de chequeo que representa el cumplimiento de dicha operación.

H9. Edición Concurrente	
Acción	Cumple
Verificar los privilegios de los programadores sobre el proyecto, para identificar quienes pueden usar la edición concurrente.	✓
Dos o más programadores editan en una clase en una misma línea.	✓
Dos o más programadores editan en una clase en diferentes líneas.	✓
Dos o más programadores ven en tiempo real los cambios que se	✓

realizan en el código fuente de la clase.	
Dos o más programadores editan en diferentes clases.	✓
Un programador podrá manipular una clase GUI en el constructor de interfaces gráficas, mientras que otro editará el código fuente, este último observa los cambios de los componentes.	✓
Dos o más programadores manipulan componentes en el lienzo y los cambios en el código fuente se efectuarán concurrentemente.	✓
Los programadores que están editando mantienen la misma versión del documento en todo momento	
Un programador con privilegios de solo lectura asociado a un proyecto donde hay dos o más programadores con privilegios de escritura editando en tiempo real, observa los cambios en todo momento.	✓
Un programador abre el código fuente de una clase o interface que en ese momento están editando, recibe la versión actual que ShareJS tiene en ese instante.	✓

Tabla 33. Lista de Chequeo de prueba de Unidad para H9

8.6.10. Iteración 10

Historia de Usuario	
Número: H17	Usuario: Programador
Nombre de la Historia: Constructor de Interfaces Gráficas de Usuario	
Prioridad en Negocio (Alta/Media/Baja): Baja	Estimación: 2
Descripción: La aplicación debe contar con una herramienta que le permita al programador diseñar interfaces gráficas para sus programas, a través de un lienzo y una paleta de componentes gráficos que puedan ser arrastrados y organizados. Tal herramienta se integrara con las características colaborativas de la aplicación.	
Observaciones: El constructor de interfaces gráficas es limitado en componentes gráficos, ya que solo contempla JButton, JTextArea, JLabel y JTextField como componentes arrastrables. En futuras versiones es recomendable que la paleta de componentes sea mas robusta.	
Actividades <ul style="list-style-type: none"> – Maquetar el área dispuesta para frame o lienzo y para la paleta de 	

componentes.

- Aplicar estilos y scripts a los elementos que hacen parte del maquetado, con el propósito de darle apariencia componentes gráficos de Java, y para visualizar efectos de movimiento de arrastrar y soltar, organización a través de teclas cursoras, y redimensión.
- Se debe tener en cuenta que en el editor se podrá visualizar no solo código fuente, también debe permitir embeber el constructor de interfaces gráficas, y permitir además intercambiar entre el diseño gráfico y el código fuente en la misma pestaña.
- Los componentes de la paleta debe manipularse a través de coordenadas y medidas de ancho y alto con respecto al lienzo, pensando siempre en las posiciones que pueden ocupar dentro de este.
- Implementar restricciones de organización de los componentes dispuestos en el lienzo, es decir, es importante que los elementos gráficos ubicados en el lienzo, no queden uno sobre otros en ninguna posible intersección de líneas.
- Desde el mismo espacio de trabajo es importante que el programador pueda cambiar entre el diseño gráfico y el código fuente correspondiente al JFrame generado y viceversa.
- En el código fuente generado dinámicamente, es muy importante bloquear la edición del código que corresponda estrictamente al entorno gráfico, es decir, negarle la edición de ciertos segmentos de código fuente al programador para evitar daños en la interfaz gráfica.
- Implementar rutinas que permitan utilizar el constructor de interfaces gráficas en forma concurrentes entre los programadores que comparten el proyecto con privilegios de escritura.
- Es muy importante controlar los cambios producidos durante una edición concurrente, porque se puede presentar casos que mientras un programador se encuentre editando el diseño, otro este editando el código fuente.
- Si un programador crea componentes manualmente desde el código fuente, estos no serán visualizados en la vista de diseño del constructor, ya que esta función es unidireccional, pero cuando el JFrame generado se ejecute, este componente creado si será visible.

Tabla 34. Descripción de la Historia de Usuario H17

- **Diagrama de Clases para H17**

El desarrollo de esta iteración está basada en su totalidad en el lado del cliente e implementado totalmente en Javascript. La idea básica fue crear un constructor de interfaces gráficas, partiendo de una paleta de componentes

que puedan ser arrastrados y ubicados en un lienzo (JFrame en Java) como lo deseen, siempre y cuando estos no se intercepten. En el momento que el programador desea crear una interface gráfica, internamente se creará un lienzo y el código fuente en el editor que esto implica; a su vez cuando un componente es arrastrado hacia el lienzo, este se ubica en una coordenada con respecto a este y con una dimensión predeterminada según sea. Una vez ubicado el componente se generará el código fuente que corresponda en el editor. Cabe destacar que este componente puede ser usado concurrentemente en tiempo real por varios programadores. A continuación se muestra los scripts que intervienen en esta iteración, simulando un modelo de clases. Este diagrama se anexa al diagrama de la historia de usuario 12 del entorno gráfico.

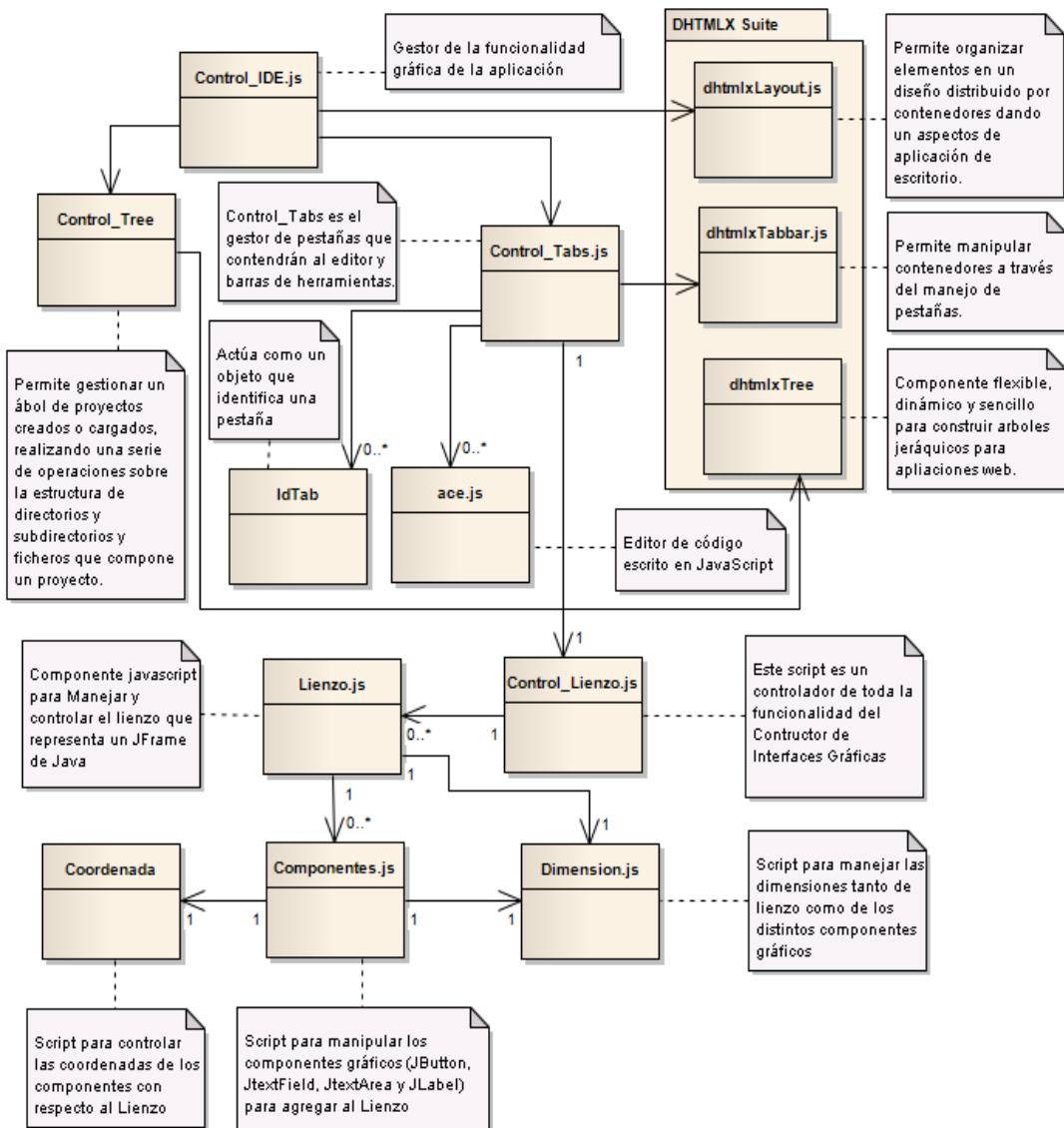


Diagrama 11. Diagrama Javascript de Constructor de Interfaces Gráficas

- **Prueba de Unidad para H17**

H17. Constructor de Interfaces Gráficas de Usuario	
Acción	Cumple
Crear una clase GUI	✓
Generar la vista del constructor de interfaces y la paleta de componentes	✓
Generar el código fuente del lienzo	✓
Agregar Componente al lienzo arrastrado de la paleta.	✓
Generar el código fuente correspondiente al componente agregado.	✓
Organizar los componentes en el lienzo arrastrando con el mouse o con las teclas cursoras.	✓
Cambiar las propiedades del componente, ya sea cambiar el nombre la variable, el del value o agregar un actionPerformed.	✓
Redimensionar Componentes, ya sea el lienzo o los componentes que estén en él.	✓
Restringir que un componente, se posicione sobre otro o que se intersecte en algún punto.	✓
El lienzo no puede ser redimensionado a un tamaño que no aborde los componentes que tenga posicionados en él.	✓
No permitir redimensionar un componente, si su dimensión alcanza a intersectar a otro.	✓
El constructor de interface gráficas permite la manipulación de dos o más programadores en tiempo real usando socket.io	✓

Tabla 35. Lista de Chequeo de Prueba de Unidad para H9

A continuación se presentan algunas capturas de pantalla de las operaciones realizadas sobre el constructor de interfaces de la IDE.



Figura 44. Vista Lienzo y Paleta de Componentes

The screenshot shows the NetBeans IDE interface. On the left is the 'Proyectos' (Projects) panel, which lists a project named 'asdasd' containing a 'SRC' folder with 'carlos' and 'clase1.java', 'clase2.java', and 'Principal.java'. Below 'SRC' is a 'LIBS' folder with 'AbsoluteLayout.jar'. The main window on the right is titled 'Principal.java' and displays the following Java code:

```

5 package carlos;
6 /**
7 * @author acavaelusuarios
8 */
9
10 public class Principal extends javax.swing.JFrame {
11
12     public Principal(){
13
14         init();
15     }
16
17     private void init(){
18
19         this.setSize(new java.awt.Dimension(300,300));
20         this.setPreferredSize(new java.awt.Dimension(300,300));
21         setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
22         getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
23
24         ...
25

```

A red rounded rectangle highlights the entire code area. To the right of the code, the text 'Código Fuente Generado para el lienzo' is displayed in red.

Figura 45. Vista de Código Fuente Generado del Lienzo

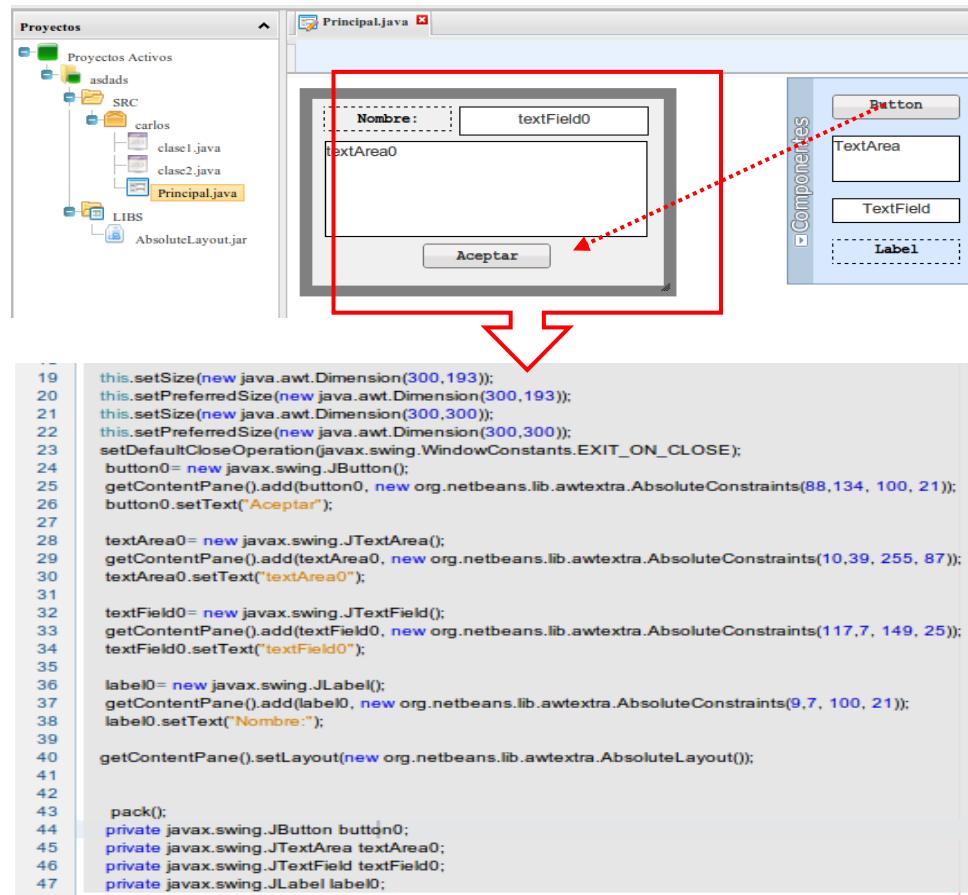


Figura 46. Vista Agregar Componentes al Lienzo

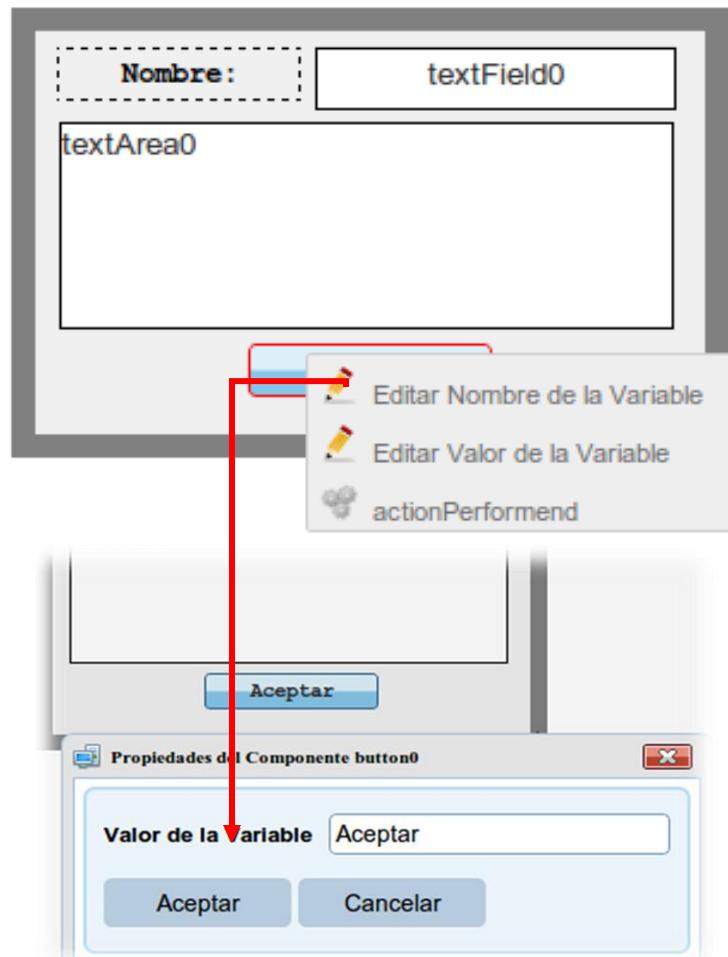


Figura 47. Vista de Propiedades de un Componente.

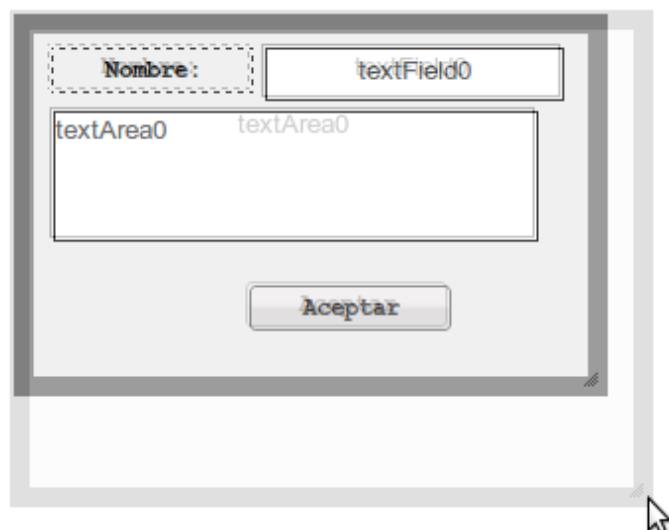


Figura 48. Vista de Redimensionar Lienzo

8.6.11. Iteración 11

Historia de Usuario	
Número: H11	Usuario: Programador
Nombre de la Historia: Mover Fichero	
Prioridad en Negocio (Alta/Media/Baja): Baja	Estimación: 1
<p>Descripción: El propietario del proyecto o programador con privilegios de lectura/escritura, podrá mover ficheros entre paquetes, e incluso entre proyectos abiertos. Esta operación debe hacerse solo en movimientos permitidos y visualizar los cambios. Los programadores con privilegios de solo lectura podrán copiar un fichero hacia otro proyecto.</p>	
<p>Observaciones: Esta operación cobra mucha importancia en la vista, ya que se debe visualizar un efecto de arrastrar y soltar.</p>	
<p>Actividades:</p> <ul style="list-style-type: none"> – Buscar un componente gráfico que permita visualizar el efecto de arrastrar y soltar un elemento en un árbol de archivos, adaptarlo y acoplarlo. – Implementar las rutinas de copiar y pegar los ficheros que correspondan de acuerdo a la petición realizada a través del componente gráfico. – Verificar que estas operaciones sean permitidas solo por el propietario o por el programador que le han compartido con privilegios de escritura. – Mover una clase. – Mover un archivo de texto plano. – Mover un paquete, ya sea dentro del mismo proyecto, o entre proyectos. Es importante tener en cuenta que en el paquete nombrado como “<i>Default Package</i>” no se puede realizar esta operación. 	

Tabla 36. Descripción de la Historia de Usuario H11

– Diagrama de Clases para H11

En esta historia de usuario se desea implementar rutinas que permitan mover los diferentes tipos de ficheros entre paquetes del mismo proyecto, o

incluso entre proyectos. La estructura del diagrama de clases para la historia de usuario H11, presenta las mismas clases de la Figura 47, pero se agregan los siguientes métodos al contexto del paquete *IDE* que involucra el gestor de proyectos.

- *moveClass(String name1, String owner1, String nPackage1, String nameC, String name2, String owner2, String nPackage2, boolean cut)*: Este método básicamente permite mover una clase desde determinado paquete hasta un paquete destino.
- *moveGUIClass(String name1, String owner1, String nPackage1, String nameC, String name2, String owner2, String nPackage2, boolean cut)*:Este método permite mover una clase que corresponde a una interface gráfica.
- *moveFile(String name1, String owner1, String nPackage1, String nameC, String name2, String owner2, String nPackage2, boolean cut)*: Este método permite mover ficheros permitidos en la aplicación.
- *moveLibarie(String name1, String owner, String nameL, String name2, String owner2, boolean cut)*: Este método permite mover librerías desde diferentes proyectos, pero solo en los correspondientes directorios dispuestas para estas.

Prueba de Unidad para H11

La prueba de unidad para esta historia de usuario corresponde a validar el funcionamiento de los métodos agregados.

Resultado de la Prueba

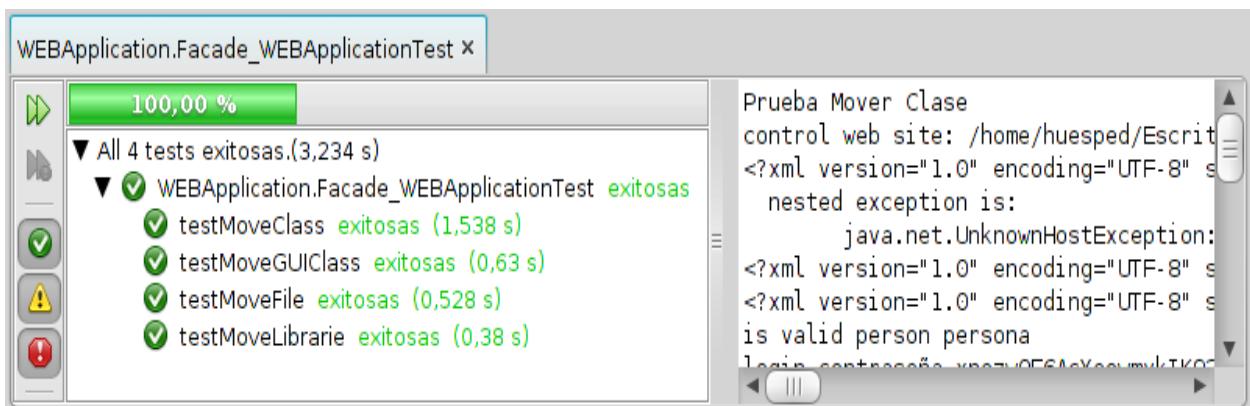


Figura 49. Resultado de Prueba de Unidad para H11

Historia de Usuario		
Número: H13	Usuario: Programador	
Nombre de la Historia: Generar Código Fuente		
Prioridad en Negocio (Alta/Media/Baja):	Baja	Estimación: 2
<p>Descripción: La aplicación debe ofrecerle al usuario una operación automática que le genere código fuente, con el propósito de ahorrarse tiempo de trabajo. Esto puede implementarse cuando se crea una clase, escribiéndole una plantilla básica de su estructura. También en el constructor de interfaces gráficas de usuario, cada vez que agregue un componente al lienzo, debe generarse el código fuente que involucre ese elemento agregado (Ver anexo XXX para las plantillas y código generado por la app).</p>		
<p>Observaciones: Esta operación está dada para programadores propietarios o con privilegios de escritura.</p>		
<p>Actividades:</p> <ul style="list-style-type: none"> – Implementar una rutina que permita que una vez el programador cree una clase o una interfaz gráfica de usuario, genere automáticamente código fuente que corresponda a la estructura básica de una clase o un JFrame con el nombre asignado. – Con respecto a los componentes del constructor de interfaces se debe implementar una rutina que permita generar el código fuente que corresponda a la declaración, inicialización y configuración del componente ubicado en el lienzo. – Crear rutinas que permitan actualizar conforme el usuario realiza modificaciones gráficamente en la estructura de un lienzo. – Controlar las implicaciones en términos de concurrencia (inconsistencias) que puedan tener las modificaciones realizadas mediante el editor gráfico. 		

Tabla 37. Descripción de la Historia de Usuario H13

– **Diagrama de Clases para H13**

Esta historia de usuario involucra una funcionalidad que para el programador es útil, ya que le evita escribir código que no involucra lógica de negocio. En este caso la IDE permitirá generar el código fuente base de una clase normal o de una GUI una vez ha sido creada, permitiéndole a nivel gráfico observar

una nueva pestaña donde el editor contiene dicho código generado. El desarrollo de esta funcionalidad fue escrita en Javascript y se ejecuta por ejemplo, cuando el programador desea crear una clase, o cuando se está creando una interface gráfica, cada componente agregado al lienzo implica generar código fuente.

En esta historia de usuario se mantiene el modelo de scripts resultado de H17, y se las siguientes funciones:

En Control_IDE.js se agregaron:

- *agregarClase(njava)*: Esta función permite crear una clase Java y generar un código fuente plantilla para la misma.
- *agregarClaseGUI(nJava)*: Crea una clase interface gráfica de usuario, y genera el código fuente plantilla de un JFrame.

En Lienzo.js se agregaron:

- *createComponent*: Esta función se encarga de identificar qué tipo de componente gráfico fue agregado al lienzo, y así define el código fuente a generar.
- *moveComponent*: Consiste en mover un componente en el lienzo, de manera las coordenadas con respecto a su contenedor cambiará, provocando así un cambio en el código fuente de dicho componente.
- *updateCodeComponent*: Esta función se encarga de actualizar el código fuente de determinado componente, usando sus atributos para modificar las líneas que correspondan.
- *resizeComponent*: Es una función que permitirá cambiar la dimensión del componente, haciendo necesario que el código fuente sea modificado.
- *updateName*: Función que permite modificar el nombre de la variable del componente, esto acarrea la modificación del código generado.
- *updateValue*: Función que modifica el value del componente. El Value es la propiedad del componente visible en la interface gráfica, por ejemplo, el texto visible en un botón. Este cambio implica modificar el código fuente.
- *deleteComponent*: Función para eliminar el componente gráfico y su código fuente generado en el JFrame.
- *createAction*: Esta función es usada para el componente JButton, ya que permite agregar una acción después de un evento de pulsado. Esta acción se debe ver reflejada en código, por lo tanto debe ser generado.

Prueba de Unidad para H13

La prueba de unidad para esta historia de usuario corresponde a validar el funcionamiento de los métodos agregados.

.Resultado de la Prueba

H13. Generar Código Fuente	
Acción	Cumple
Generar código fuente plantilla al crear una clase	✓
Crear código fuente plantilla al crear una clase GUI	✓
Generar código fuente de la declaración e inicialización de un JButton agregado al lienzo	✓
Generar código fuente de la declaración e inicialización de un JTextArea agregado al lienzo	✓
Generar código fuente de la declaración e inicialización de un JTextField agregado al lienzo	✓
Generar código fuente de la declaración e inicialización de un JLabel agregado al lienzo	✓
Generar código fuente de la declaración de un actionPerformed de un JButton.	✓
Actualizar el código fuente de un componente o lienzo cuando es redimensionado.	✓
Actualizar el código fuente de un componente cuando es repositionado	✓
Actualizar el código fuente de un componente cuando el nombre de la variable ha sido modificado	✓
Actualizar el código fuente de un componente cuando el nombre del value ha sido modificado	✓

Tabla 38. Lista de Chequeo de la Prueba de Unidad para H13.

8.6.12. Iteración 12

Historia de Usuario	
Número: H15	Usuario: Programador
Nombre de la Historia: Notificación de Operaciones	
Prioridad en Negocio (Alta/Media/Baja): Baja	Estimación: 1
Descripción: La aplicación debe notificar las operaciones que los	

programadores están efectuando sobre un proyecto compartido y que se encuentran trabajando concurrentemente. Esto es necesario para mantener al tanto a todos los programadores de lo que sucede con el proyecto en tiempo real. Por ejemplo, si un programador con privilegios de lectura/escritura elimina una clase, todos deben ser notificados para evitar confusiones.

Observaciones: Estas funciones están estrechamente ligadas a la concurrencia, ya que los programadores deben ser informados sobre los cambios que están ocurriendo en el proyecto en el que trabajan.

Actividades:

- Analizar y estudiar Socket.io, el cual se convierte en una tecnología que proporciona canales de comunicación bidireccionales sobre todos los navegadores y también en dispositivos móviles.
- Definir las operaciones sobre la aplicación, que ameritan ser notificadas a los programadores que están compartiendo en un determinado momento.
- Implementar rutinas que verifiquen los programadores que se encuentran en concurrencia para enviarles los respectivos mensajes de las operaciones llevadas a cabo.
- Implementar un tablero de notificaciones en la IDE que le ofrezca al programador la información de los operaciones que se está presentando, quién las está efectuando y qué sucedió. Esto es necesario para evitar confusiones de cambios inesperados.

Tabla 39. Descripción de la Historia de Usuario H15

– **Diagrama de Clases para H15**

La notificación de operaciones es una funcionalidad de valor agregado, y consiste en mantener informado al programador de ciertas cosas que suceden con determinado proyecto, donde él puede ser el dueño o un colaborador. Por ejemplo, dos programadores están editando en un proyecto y uno de ellos, crea una clase, entonces se envía una notificación al otro programador de lo que sucedió para mantenerlo al tanto, esto evitaría confusiones de ver cambios sin explicación. A continuación se presenta un flujo genérico de lo que sucede cuando un programador entra en la IDE y abre un proyecto que está compartido con otros programadores.

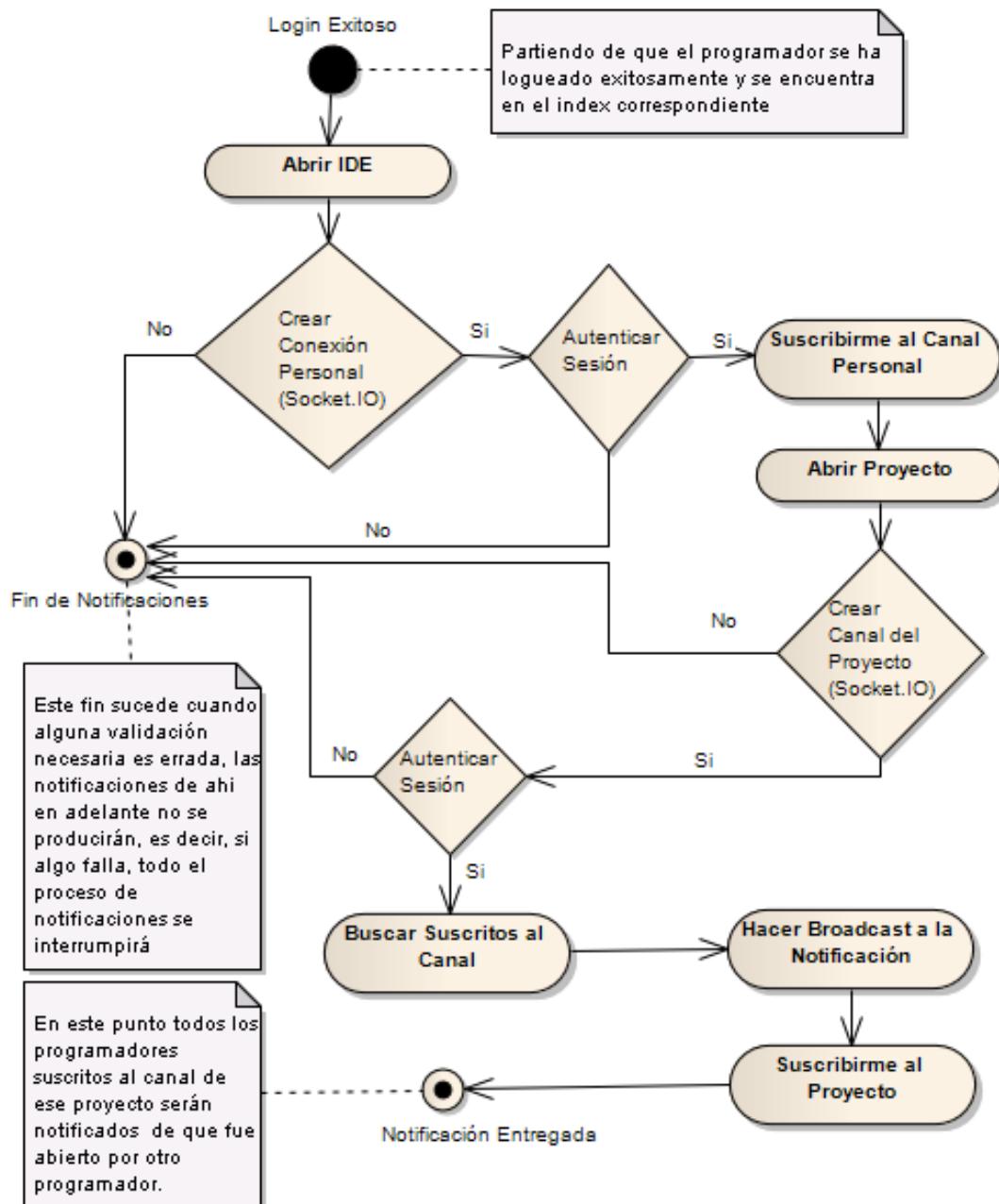


Diagrama 12. Flujo Genérico de Notificación de Operaciones.

Prueba de Unidad para H15

La prueba de unidad para esta historia de usuario corresponde realizar una lista de chequeo, corroborando el cumplimiento de las distintas operaciones que son notificadas.

Resultado de la Prueba

H15.Notificación de Operaciones	
Acción	Cumple
Notificar que un proyecto fue abierto	✓
Notificar que el nombre del proyecto ha cambiado	✓
Notificar que un proyecto fue compartido	✓
Notificar que un proyecto fue eliminado	✓
Notificar el cambio de privilegios de un programador sobre un proyecto.	✓
Notificar que un programador dejó de compartir un proyecto	✓
Notificar que un paquete fue creado	✓
Notificar que un paquete fue renombrado	✓
Notificar que un paquete fue movido	✓
Notificar que un paquete fue eliminado	✓
Notificar que una clase fue creada	✓
Notificar que una clase fue renombrada	✓
Notificar que una clase fue movida	✓
Notificar que una clase fue eliminada	✓
Notificar que una clase GUI fue creada	✓
Notificar que una clase GUI fue renombrada	✓
Notificar que una clase GUI fue movida	✓
Notificar que una clase GUI fue eliminada	✓
Notificar que una librería fue agregada	✓
Notificar que una librería fue renombrada	✓
Notificar que una librería fue movida al paquete librerías de otro proyecto	✓
Notificar que una librería fue eliminada	✓
Notificar que un fichero fue agregado	✓
Notificar que un fichero fue renombrado	✓
Notificar que un fichero fue movido	✓
Notificar que un fichero fue eliminado	✓
Notificar que la sesión caduco porque otro usuario hizo login con la misma cuenta	✓
Notificar cuando el administrador ha eliminado un usuario.	✓

Tabla 40. Lista de Chequeo de la Prueba de Unidad de H15.

Historia de Usuario		
Número: H16	Usuario: Programador	
Nombre de la Historia: Chat		
Prioridad en Negocio (Alta/Media/Baja):	Baja	Estimación: 1
Descripción: La aplicación debe ofrecerles un chat por proyecto, a los programadores que trabajan colaborativamente, con el propósito de que cuenten con un medio de comunicación en tiempo real, entre los que estén trabajando en el proyecto compartido.		
Observaciones: La ventanas de chat corresponden a un proyecto que está abierto en la IDE.		
Actividades: <ul style="list-style-type: none"> – Implementar rutinas que permitan acoplar el componente para el chat con la tecnología socket.io, con el fin de reutilizar el canal de notificaciones para transmitir los mensajes enviados. – Hacerle pruebas funcionales y refinar. 		

Tabla 41. Descripción de la Historia de Usuario H16

– **Diagrama de Clases para H16**

El chat consiste en proporcionar un medio de interacción a los programadores que trabajan en un proyecto dado, esto quiere decir, que un chat está asociado a un único proyecto, de manera que un programador puede tener tantos chat como proyectos tenga abiertos en la IDE. En esta iteración no intervienen clases, pero si dos componentes:

- Chat.js
- Socket.io.js

A continuación se presenta un diagrama de actividades el funcionamiento a grandes rasgos del uso del chat para un proyecto. Se parte de que el programador ha iniciado sesión, abrió la IDE y ha abierto un proyecto, por lo tanto ya el programador se ha inscrito al canal de dicho proyecto, tal como lo muestra el flujo genérico de la historia de usuario H15 anteriormente presentada.

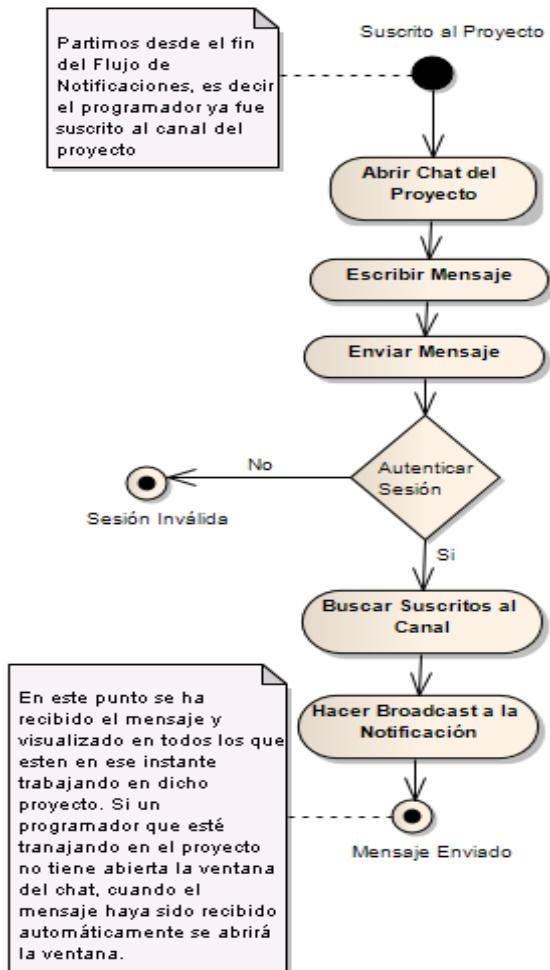


Diagrama 13. Flujo Genérico del Chat para H16

Prueba de Unidad para H16

La prueba de unidad para esta historia de usuario corresponde verificar el funcionamiento del componente chat dispuesto para un proyecto compartido por programadores.

H16. Chat	
Acción	Cumple
La ventana del chat se abrió satisfactoriamente.	✓
El mensaje redactado fue enviado	✓
El mensaje fue recibido por los programadores que están trabajando en dicho proyecto.	✓
Se da una alerta gráfica cuando hay un mensaje nuevo en el chat.	✓
Al cerrar el proyecto la ventana del chat se cierra.	✓

Tabla 42. Lista de Chequeo para H16

9. IMPLEMENTACIÓN

9.1. ARQUITECTURA

Arquitectura es la organización fundamental de un sistema descrita en sus componentes, su relación entre ellos y con el ambiente, y los principios que guían su diseño y evolución.⁹³

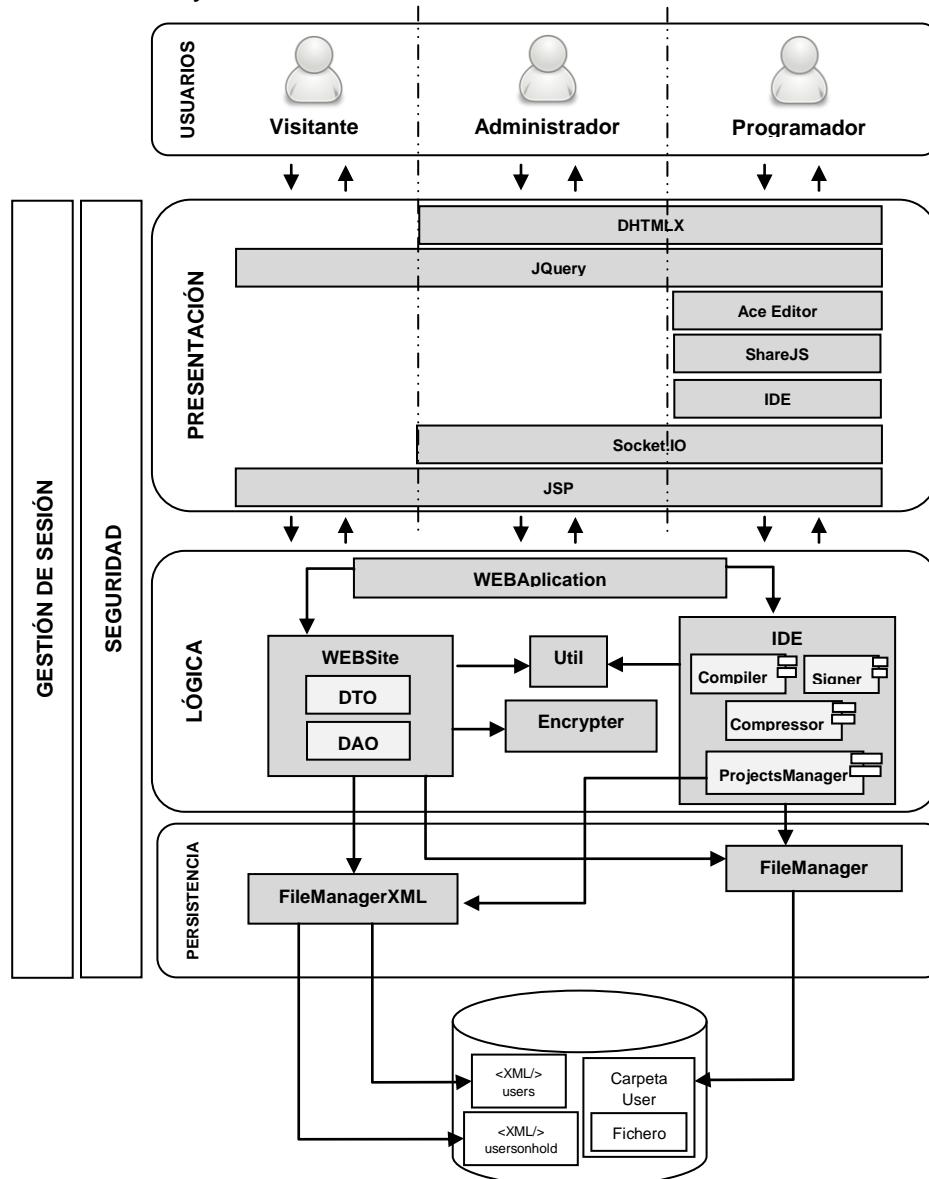


Diagrama 14. Arquitectura de la IDE

⁹³ IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Sponsor: Software Engineering Standards Committee of the IEEE Computer Society. IEEE Std 1471-2000, Approved 21 September 2000

El diagrama 14 representa la arquitectura de la IDE, donde se determina la organización en capas, la relación de los respectivos componentes, paquetes e interacción con la persistencia de los datos. Los usuarios de la aplicación interactúan directamente con los diferentes componentes Javascript de la capa de presentación, a través del uso de su navegador. En la figura se puede percibir que el programador es el único usuario de la aplicación que usará todos los componentes dispuestos, a diferencia del administrador que solo tendrá acceso a vistas de administración y no tendrá funcionalidad técnica, y aún más limitado es el caso del visitante que solo podrá disponer de una página index, donde encontrará un formulario para loguearse, uno para registrarse, un link de recordatorio de contraseña y links de información. La capa de lógica contiene la distribución y organización de los componentes y módulos de la aplicación, de manera que toda comunicación con la capa de presentación se hará a través de WEBApplication, quién identifica el rol del usuario y direccionará a los respectivos módulos que requieran. El manejo de la persistencia de la aplicación está orientada a directorios y documentos, de manera que para la gestión y almacenamiento de la información de los usuarios se llevará a cabo a través de documentos XML, y cada programador tendrá un directorio personal que contendrá sus proyectos, su imagen de avatar, una carpeta temporal que tendrá utilidad cuando se desea ejecutar un proyecto o cuando se desea cargar algún fichero a la aplicación, y tendrá dos archivos XML de configuración, uno para almacenar la información de los proyectos y el otro para la información del perfil, sesión activa y último acceso. Por último se tienen dos ejes transversales que corresponden a la seguridad y la gestión de sesión, que le brinda un blindaje a accesos a la aplicación no autorizados y no permitir que exista más de una sesión activa por cuenta de programador.

9.2. PATRONES

“Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, y describe la esencia de la solución a ese problema,

*de tal modo que pueda utilizarse esta solución un millón de veces más, sin siquiera hacerlo de la misma manera dos veces*⁹⁴

En la implementación del proyecto se consideraron ciertos patrones de acuerdo a las necesidades del mismo, de manera que fuese posible obtener un producto mantenible y estructurado para lograr nuevas versiones sin mayor esfuerzo.

A continuación se mencionan los patrones que intervienen en el desarrollo según su clasificación:

- **Patrones Estructurales:** Básicamente se encargan de mostrar cómo combinar clases y objetos para formar estructuras más grandes:
 - Facade: proporciona un mediador simplificado para un conjunto de funcionalidades de un o unos subsistemas, proporcionando una interface de alto nivel para facilitar el uso de estos. En el proyecto se identifica claramente algunos facade que ofrecen acceso a cierta funcionalidad del algún componente o subsistema, por ejemplo, ProjetsManager tiene un facade que ofrece funciones como crear proyectos, renombrarlos, entre otros, si mostrar el detalle de cómo está implementado.
- **Patrones Creacionales o de Instanciación:** Estos patrones abstraen el proceso de instanciación de objetos, ayudando a que el sistema sea independiente de cómo se crean, componen y representan sus objetos.
 - Singleton: Este patrón se asegura de que cierta clase sea instanciada una sola vez en todo el hilo de ejecución, ofreciendo un único punto de acceso global a la misma. En el proyecto este patrón se ve reflejado en el componente FileManager, el cual mantiene un acceso único y global a la funcionalidad sobre los ficheros que intervienen en un proyecto (paquetes, clases, archivos).

⁹⁴ Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language*. Oxford University Press, New York, 1977.

- **Patrones de Arquitectura:** Básicamente estos patrones distribuyen responsabilidades, focalizándose en el control del sistema. Definen la estructura básica del sistema y podrían estar relacionados con otros patrones.
 - MVC: Modelo-Vista-Controlador se encarga de separar la lógica de negocio de la interfaz de usuario, facilitando la funcionalidad, mantenibilidad y escalabilidad del sistema que se construyen bajo sus reglas. En el proyecto este patrón trabaja junto al patrón facade para mantener fronteras entre las vistas, los diferentes facades, controles y el modelo, consiguiendo así que cuando una petición es realizada, el control de mayor nivel decidirá a cuál facade invocar al punto de llevar al modelo para que procese y responda al control que hará la respectivas operaciones para entregar una respuesta a la vista que finalmente entregará al usuario.
- **Patrones de Acceso a Datos:** Estos patrones determinan la manera de cómo interactuar con una fuente de datos.
 - *Data Access Object (DAO)* y *Data Transfer Object (DTO)*: Estos patrones de diseño proveen una forma totalmente abstracta del acceso a datos. El acceso al modelo de datos viene dado a través de una serie de métodos que acceden a los datos y devuelven objetos construidos a partir de estos datos consultados. El DAO es particular al motor o la fuente de los datos, mientras que el DTO seguirá siendo clases plantilla sin depender de la fuente de los datos. El proyecto este patrón se ve reflejado en el acceso a la fuente de datos que contienen a los usuarios del sistema. En este caso este patrón estará incluido en el modelo del patrón arquitectónico descrito anteriormente.

9.3. PERSISTENCIA DE DATOS, ARCHIVOS DE INDEXACIÓN Y CONFIGURACIÓN

Desde el inicio del proyecto se tuvo en consideración que el modelo de persistencia para la aplicación debía asemejarse a la manera como las IDEs convencionales lo hacen, y esto es a través de directorios, ficheros y

archivos de configuración, de manera que una base de datos no se ajustaba a esta estrategia.

El modelo de persistencia en el proyecto está dado de dos formas:

1. Persistencia de datos del usuario.

Se desarrolló una capa de persistencia usando el patrón DAO y DTO, para el acceso exclusivo a los datos que corresponden a un usuario. Es muy importante mencionar que una ventaja de usar DAO es lograr una independencia de la lógica con la fuente o motor de datos, esto quiere decir, que si en un momento dado se decide cambiar de DBMS (*Data Base Manager System*), es tan sencillo como implementar un nuevo DAO para dicha fuente de datos, reemplazar el existente y perfectamente la aplicación no se enteraría que el motor o fuente de datos fue reemplazado.

2. Persistencia de datos de configuración

Los datos de configuración no son propiamente un modelo de datos convencional, sino la manera por la cual se simula el comportamiento de una IDE de ambiente desktop. En la capa de persistencia se tiene un manejador que se encarga de la gestión de los directorios, clases, paquetes, archivos de extensión permitida, y además algo muy importante es la creación de archivos xml de configuración, que al consultarlos permiten conocer qué proyectos existen, qué paquetes, clases, librerías y archivos tiene cada uno, quién es el propietario del proyecto, si es compartido o no y a quienes se le comparte, perfiles de usuario. Estos archivos le permiten a la IDE conocer el contenido de su espacio de trabajo.

Ahora es importante justificar la elección de archivos XML como fuente de datos, y por qué no se usó algún DBMS.

Primero y como se menciona en el primer párrafo de este apartado la intención inicial fue simular una IDE en un ambiente de escritorio, de manera que para conseguirlo era necesario crear carpetas, archivos en un espacio de trabajo dispuesta para cada programador, y crear la estrategia para que la aplicación conozca el contenido de dicho espacio. Es aquí donde

intervienen los archivos XML, para mantener un indexador del contenido e información de configuración de cada proyecto. Los archivos XML son ampliamente usados por las aplicaciones para interpretar información importante y vital para su efectiva ejecución y en nuestro caso es usado para interpretar las configuraciones y contenido de los proyectos y sus ficheros.

Segundo argumento por el cual se usa XML como fuente de datos es por su universabilidad, es decir, en gran parte de aplicaciones no solo web usan XML para la transferencia, configuración, migración, compatibilidad e interpretación de su información.

Tercer argumento es que a raíz de que las tablas que implicarían la creación, consulta, actualización y eliminación de usuarios serían solo dos, por lo tanto la adquisición de un hosting con motor de base de datos no sería viable.

Cuarto argumento es que al ser archivos XML, habría gran portabilidad ya que la aplicación no dependería de un motor de base de datos ni de backups, y la configuración del servidor sería más sencilla, y la portabilidad de los datos implicaría solo copiar y pegar los archivos xml.

Basados en los anteriores argumentos se decide que la aplicación establecería su persistencia a través de documentos XML, como fuente de datos para la gestión de usuarios y proyectos, además de directorios y ficheros.

A continuación se describe las estructuras de los documentos XML usados para las dos formas de persistencia de la aplicación y el significado de cada etiqueta y elemento.

9.3.1. Usuarios En Espera

Usuarios en espera es un archivo XML usado como técnica para evitar que sean creadas cuentas de programador sin control. Básicamente el proceso consiste en:

- Un visitante entra a la página index de la aplicación y presiona clic en el vínculo de registrarse, y llenará los datos que corresponda, entre ellos el correo electrónico.

- Al dar clic en aceptar, internamente se enviará un link de activación al correo registrado. Eso obliga a que sean datos veraces.
- Se almacena el posible programador en este archivo, a espera de ser activado.

El archivo XML llamado “usersonhold.xml” está compuesto como lo muestra la siguiente figura.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <users>
3   <user>
4     <email>charlesandrew621@hotmail.com</email>
5     <password>wm64To+7+kbmC4MCjK+OVQ==</password>
6     <type>programmer</type>
7     <name>carlos</name>
8     <profile/>
9     <id>k2rxGdc5RQVNbKmyg2cajWuZpG3hqSuiZGYpRjmrSlrgwPXZkN</id>
10    </user>
11  </users>
```

Figura 50. XML Usuarios En Espera.

Etiqueta	Descripción
<users>	Esta etiqueta raíz es usada para contener a todos los usuarios temporales.
<user>	Esta etiqueta representa y enmarca a un usuario temporal. En el documento habrá tantos elementos user como usuarios se hayan registrado.
<email>	Corresponde al correo electrónico registrado para dicho usuario temporal.
<password>	Corresponde a la contraseña encriptada registrada por el usuario temporal.

<type>	Indica que tipo de usuario es en el contexto de la aplicación. Se coloca este atributo pensando en que en futuras versiones aparezcan nuevos roles.
<profile>	Esta etiqueta almacenará la información acerca del usuario siempre y cuando este la edite.
<id>	Corresponde a una serie de caracteres randómicos que servirán para verificar la validez del enlace de activación.

9.3.2. Usuarios Activos

Usuarios es un archivo XML que almacena los usuarios activos del sistema, estos pueden ser programadores o el administrador. Este archivo es alimentado de la siguiente manera:

- El visitante desde su cuenta email, abre desde su bandeja de entrada el correo electrónico enviado por la aplicación, y da clic en el enlace de activación.
- El visitante es redireccionado a una página de activación.
- De manera transparente al usuario, la aplicación consulta sobre el archivo de usuarios temporales la existencia y validez de dicho enlace comparando con el dato almacenado en la etiqueta <id>.
- Si la validación es exitosa, la aplicación remueve ese usuario temporal, y lo almacena en el XML de usuarios activos.

El archivo XML llamado “users.xml” está compuesto como lo muestra la siguiente figura:

```

1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <users>
3      <user>
4          <email>wilsonyesidrc@ufps.edu.co</email>
5          <password>+4D21cdZMpH2rRgVtReE1Q==</password>
6          <type>programmer</type>
7          <session>85EBBD0BA7221C7F2C110JUEDC3ED8BC</session>
8      </user>
9      <user>
10         <email>admin</email>
11         <password>W941LL7jTPR4FvSUuJBs9Q==</password>
12         <type>administrator</type>
13         <session/>
14     </user>
15 </users>

```

Figura 51. Archivo XML de Usuarios Activos

Etiqueta	Descripción
<users>	Esta etiqueta es usada para contener a todos los usuarios activos.
<user>	Esta etiqueta representa y enmarca a un usuario activo. Habrá tantos elementos user como usuarios hayan sido activos.
<email>	Corresponde al correo electrónico registrado para dicho usuario activo.
<password>	Corresponde a la contraseña encriptada registrada por el usuario.
<type>	Etiqueta que indica si el usuario activo es un programador o es el administrador. Se coloca este atributo pensando en que en futuras versiones aparezcan nuevos roles.
<session>	Esta etiqueta es fundamental para mantener control de la sesión, y así evitar que existan diferentes sesiones en un instante dado para una misma cuenta. Esta etiqueta se modifica cada vez que el programador acceda a la aplicación, asignándole un valor randómico.

9.3.3. Directorio de un Programador en el Servidor

Cuando un programador ha sido registrado, y en el instante que es activado, la aplicación crea en el servidor un directorio solo para él. Allí se almacenarán sus proyectos, su imagen de avatar, sus archivos de configuración, tal y como lo muestra la siguiente figura:

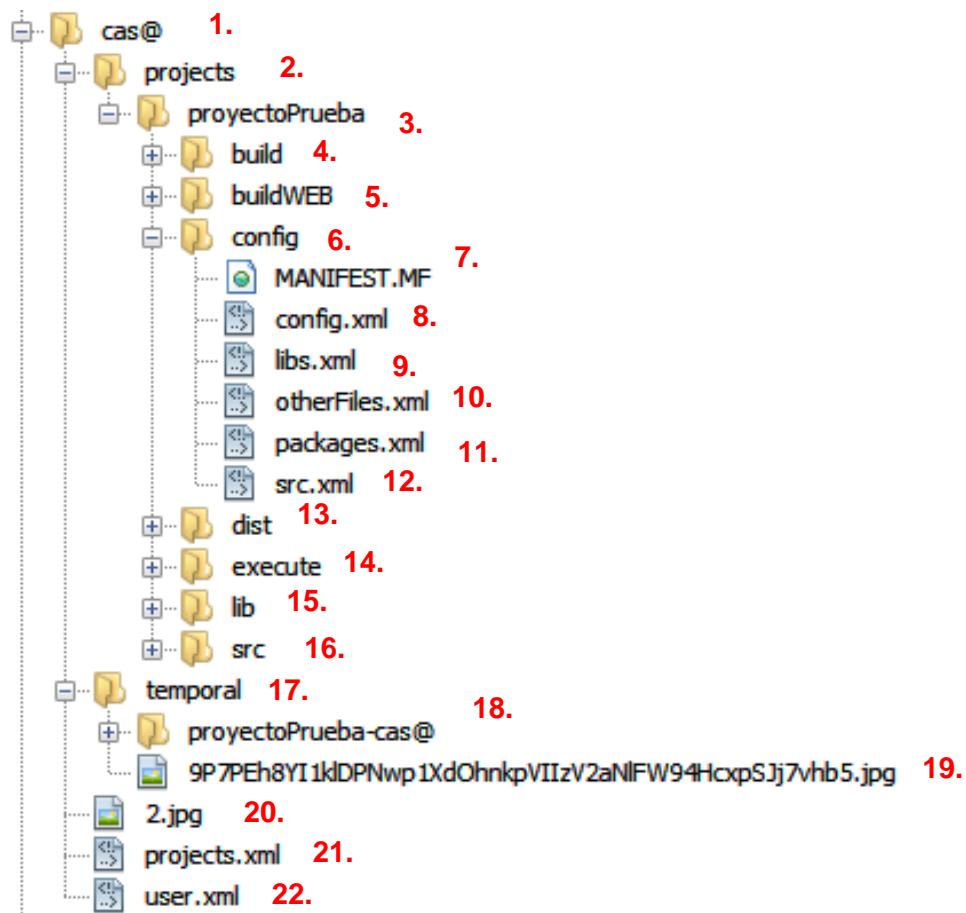


Figura 52. Estructura de Carpeta de un Programador en el Servidor

Descripción

1. Directorio raíz del programador dispuesto en el servidor.
2. Directorio que contendrá los proyectos de dicho programador. Este directorio será el “workspace” o espacio de trabajo del programador.
3. Directorio de un proyecto llamado “proyectoPrueba”

- 4.** Directorio que contendrá los archivos compilados del proyecto (.class).
- 5.** Directorio que será útil para la ejecución web, de manera que además de contener los archivos compilados del proyecto, tendrá otros archivos utilitarios que corresponde a una consola de entrada y salida.
- 6.** Directorio que contiene archivos xml de configuración e indexación.
- 7.** Archivo especial que servirá para contener información sobre los ficheros que se empaquetarán cuando se construya el archivo ejecutable del proyecto.
- 8.** Este fichero XML tiene información referente al proyecto.
- 9.** Fichero XML que contiene información de las librerías usadas en el proyecto.
- 10.** Fichero XML que tendrá información acerca de aquellos ficheros con extensión permitida, que harán parte del proyecto. (imágenes, archivos planos, etc...).
- 11.** Fichero XML que almacenará información de los diferentes paquetes que contienen clases, interfaces gráficas, y otros ficheros.
- 12.** Fichero XML que contiene detalles de todas las clases del proyecto.
- 13.** Directorio que contendrá el archivo JAR construido para el proyecto.
- 14.** Directorio que contendrá un archivo JAR que empaqueta lo que contiene el directorio buildWEB, y además tiene un archivo JSP que tendrá un applet con dicho JAR embebido para lanzar la ejecución del proyecto en el cliente.
- 15.** Directorio que contiene todas aquellas librerías que el programador ha usado en su proyecto.
- 16.** Directorio que contiene la estructura de paquetes y archivos fuente del proyecto.
- 17.** Este directorio cumple con dos funciones:
 - Es usada por el subsistema WebSite para ubicar un fichero cuando es cargado desde el cliente (imagen, clase, una librería, un archivo plano, etc...), para que posteriormente el subsistema IDE lo tome lo renombre y le dé la ubicación que corresponda.
 - La otra función que cumple es bastante importante, permitiendo almacenar una copia del proyecto tomada en el instante que el programador desea ejecutarlo, para evitar así que mientras se ejecute, sufra cambios debido a que otro programador siguió editando. Más adelante se explicará con mayor detalle el proceso de ejecución.

18. Almacena la copia del proyecto que se quiere ejecutar.
19. Corresponde a un fichero que se ha cargado desde el cliente, al cual se le ha asignado un nombre randómico para evitar conflictos de duplicidad.
20. Imagen del perfil del programador.
21. Fichero XML que almacena información de los proyectos del programador, dando detalles del nombre, propietario, si es compartido o no y los privilegios que tiene sobre el mismo.
22. Fichero XML que almacena información del perfil del programador.

En la figura 52 se presentaba la estructura de directorios de un programador en el servidor, y en ella se puede observar que existe un “workspace” o espacio de trabajo llamado “projects” donde la IDE podrá administrar los proyectos de dicho programador. El espacio de trabajo contendrá tantos proyectos como el programador haya creado, y cada uno de esos proyectos tiene un directorio importante llamado “config”, donde se encontrarán una serie de ficheros XML que permitirán obtener información vital del proyecto. En los siguientes apartados de esta sección se darán detalle de la estructura interna de estos ficheros.

9.3.4. Fichero XML de Configuración del Proyecto

El fichero config.xml contiene toda la información descriptiva del proyecto. Si se desea conocer el propietario, la fecha de creación, a qué programadores se les está compartiendo el proyecto, este fichero proporcionará estos datos. A continuación se muestra como está estructura este fichero.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <structure>
3      <name>proyecto</name>
4      <date>4/4/112</date>
5      <owner>cas@</owner>
6      <mainClass name="principal.java" package="Mundo"/>
7      <users>
8          <user>
9              <email>wil@</email>
10             <type>Write</type>
11         </user>
12     </users>
13 </structure>
```

Figura 53. Estructura Interna del Fichero config.xml

Etiqueta	Descripción
<name>	Esta etiqueta corresponde al nombre asignado al proyecto.
<date>	Esta etiqueta contiene la fecha en la que fue creado el proyecto.
<owner>	Owner indica el email del programador que creó el proyecto.
<mainClass>	Esta etiqueta indica la clase Java que contiene el método main y el paquete donde ésta se encuentra.
<users>	Enmarca todos los programadores que participan en el proyecto.
<user>	Corresponde a un programador que participa en el proyecto.
<email>	Email que identifica a un programador.
<type>	Esta etiqueta indica los privilegios que dicho programador tiene sobre el proyecto, este puede ser solo lectura (<i>Read</i>) o escritura y lectura (<i>Write</i>).

9.3.5. Fichero XML de Librerías

El fichero libs.xml contiene el índice de las librerías que estaría usando un proyecto. Cada vez que un proyecto es creado, por defecto se cargará y firmará una librería llamada AbsoluteLayout.jar, en el directorio lib del proyecto y a su vez se registrará en el fichero XML. Esta librería está incluida en el set de librerías de netbeans, y es usada como distribuidor de posiciones absolutas de los componentes gráficos que son agregados en una JFrame. AbsoluteLayout es cargada por defecto, debido a que es muy probable que un programador incluya en su proyecto interfaces gráficas, y es muy útil porque una vez un componente es ubicado en el lienzo, se toma las coordenadas y las dimensiones y se traduce fácilmente a código java, ya que el JFrame ya tiene configurado como gestor de distribución a esta librería, por lo tanto en la inicialización del componente solo se le dan los mismos parámetros. A continuación se muestra la estructura del fichero libs.xml.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <structure>
3      <lib name="AbsoluteLayout.jar"/>
4  </structure>

```

Figura 54. Estructura Interna del Fichero libs.xml

Etiqueta	Descripción
<lib>	Etiqueta que indexa las librerías del proyecto a través de su nombre y extensión.

9.3.6. Fichero XML para Otros Ficheros

El fichero otherFiles.xml permite mantener un índice de los diferentes archivos que no corresponden a una clase o una librería, indicando su nombre, su ubicación y el tipo de fichero. A continuación se presenta la estructura del fichero.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <structure>
3      <file name="imagenPrueba.jpg" package="images" path="images/imagenPrueba.jpg" type="image"/>
4      <file name="usuarios.txt" package="users" path="users/usuarios.txt" type="document"/>
5  </structure>

```

Figura 55. Estructura Interna del Fichero otherFiles.xml

Etiqueta	Descripción
<file>	Etiqueta que corresponde a un fichero o archivo que forma parte del proyecto. Esta etiqueta tiene atributos como nombre, el paquete en que se encuentra, la ruta relativa, y el tipo de fichero que es.

9.3.7. Fichero XML para un Paquetes

El fichero packages.xml que es usado como índice para que al consultar se pueda conocer la distribución de paquetes en el proyecto. Si en la estructura existe más de un nivel de paquetes, se presentarán desde el padre, seguido de punto hasta el subnivel que corresponda, es decir, existe un paquete Control que a su vez contiene un paquete oficina, este elemento en el XML será presentado así: "Control.oficina" tal y como lo muestra la siguiente figura.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <structure>
3      <package name="GUI"/>
4      <package name="Control"/>
5      <package name="Control.oficina"/>
6      <package name="DAO"/>
7      <package name="DTO"/>
8  </structure>

```

Figura 56. Estructura Interna del Fichero packages.xml

Etiqueta	Descripción
<package>	Esta etiqueta corresponde a un paquete y contiene como atributo el nombre del mismo, o el nombre a través de los niveles en que se encuentre.

9.3.8. Fichero XML para los Archivos Fuente

El fichero src.xml representa el índice de las diferentes clases del proyecto, de manera que pueda servir para tener una vista previa del contenido de archivos fuentes en el proyecto. Cada elemento en este fichero XML puede proporcionar información acerca del nombre, el paquete, ubicación y el tipo de clase (clase convencional, o clase GUI). A continuación se muestra la estructura interna de este fichero.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <structure>
3      <class name="Persona.java" package="Mundo" path="Mundo/Persona.java" type="class"/>
4      <class name="Registrar.java" package="GUI" path="GUI/Registrar.java" type="gui"/>
5  </structure>

```

Figura 57. Estructura Interna del Fichero src.xml

Etiqueta	Descripción
<class>	Esta etiqueta corresponde a una clase. Contiene atributos como el nombre, el paquete en el que se encuentra, la ruta relativa y el tipo ya sea clase o una interface gráfica.

9.3.9. Fichero XML para Proyectos

El fichero projects.xml indexa los diferentes proyectos que han sido creados por el programador, y aquellos que le han sido compartidos ya sea con privilegios de solo lectura o de escritura. A continuación se presenta un ejemplo de la estructura interna de los proyectos de un programador, y se puede apreciar que tiene tres proyectos. El primero ha sido creado por él y no lo ha compartido, el segundo es un proyecto creado por el programador *wilsonyesidrc@ufps.edu.co* y le fue compartido con privilegios de solo lectura, y por último el programador *andres@ufps.edu.co* creó un proyecto y se lo compartió con privilegios de escritura.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <projects>
3      <project>
4          <name>proyecto1</name>
5          <owner>cas@</owner>
6          <type>Write</type>
7          <shared>no</shared>
8      </project>
9      <project>
10         <name>proyecto2</name>
11         <owner>wilsonyesidrc@ufps.edu.co</owner>
12         <type>Read</type>
13         <shared>yes</shared>
14     </project>
15     <project>
16         <name>proyecto3</name>
17         <owner>andres@ufps.edu.co</owner>
18         <type>Write</type>
19         <shared>yes</shared>
20     </project>
21 </projects>
```

Figura 58. Estructura Interna del Fichero projects.xml

Etiqueta	Descripción
<projects>	Esta etiqueta enmarca todos los proyectos que tiene el programador.
<project>	Corresponde a un proyecto.
<name>	Esta etiqueta indica el nombre que le fue asignado al proyecto.
<owner>	En esta etiqueta estará almacenado el email del programador que creó el proyecto.
<type>	Esta etiqueta indica el tipo de privilegio que dicho programador tiene sobre el proyecto.

<shared> Esta etiqueta indica si el proyecto es compartido o no.

9.3.10. Fichero XML para el Programador

El fichero user.xml permite obtener información acerca del programador. Esta puede ser útil para consultar sus datos, la fecha en que fue activado en la aplicación, además de la última fecha de ingreso. Este fichero está pensado como información de perfil, eso lo diferencia del fichero users.xml descrito en 9.3.2.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <user>
3   <name>Carlos Sepulveda</name>
4   <email>cas@</email>
5   <profile>Carlos Sepulveda</profile>
6   <photo>9P7PEh8YI1k1DPNwp1Xd0hnkpVIIzV2aN1FW94HcxpSJj7vhb5.jpg</photo>
7   <dateMembership>9/11/111</dateMembership>
8   <lastEntry>4/5/2012</lastEntry>
9 </user>
```

Figura 59. Estructura Interna del Fichero user.xml

Etiqueta	Descripción
<user>	Etiqueta que corresponde al programador.
<name>	Etiqueta que almacena el nombre del programador.
<email>	Etiqueta que corresponde al correo electrónico del programador.
<profile>	Etiqueta que puede almacenar alguna descripción del programador.
<photo>	Esta etiqueta contiene el nombre de la imagen que corresponda a su avatar.
<dateMembership>	Etiqueta que almacena la fecha en la cual el programador se activó en la aplicación.
<lastEntry>	En esta etiqueta se almacena la fecha de último ingreso a la aplicación.

9.3.11. Fichero XML Consola WEB

Debido a que la ejecución del proyecto se efectuará en el cliente, es necesario proporcionarle una consola de entrada y salida, en caso de que el proyecto no tenga interfaces gráficas para la entrada y salida de datos. Por ejemplo si el proyecto es el típico “Hola Mundo” que posee un System.out.println, es necesario brindarle esa consola que pueda mostrarle esos mensajes, e incluso obtener datos por teclado. La estructura interna del XML indexa aquellas clases que intervienen en la consola WEB.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <structure>
3      <class path="$_p_a_c_k$_R_e_d_i_r_e_c_t$_p_a_c_k$/FlujoEntrada.java"/>
4      <class path="$_p_a_c_k$_R_e_d_i_r_e_c_t$_p_a_c_k$/FrameCarlos.java"/>
5      <class path="$_p_a_c_k$_R_e_d_i_r_e_c_t$_p_a_c_k$/PanelEjecucion.java"/>
6      <class path="$_p_a_c_k$_R_e_d_i_r_e_c_t$_p_a_c_k$/RedirectIO.java"/>
7      <class path="$_p_a_c_k$_R_e_d_i_r_e_c_t$_p_a_c_k$/SalidaCarlos.java"/>
8  </structure>
```

Figura 60. Fichero XML Indexador de Clases de Consola WEB.

Etiqueta	Descripción
<class>	Etiqueta que representa una clase utilitaria, que tiene como parámetro la ruta relativa de su ubicación

9.4. PROPERTIES

Los ficheros properties son simplemente archivos que permiten almacenar variables de configuración para nuestra aplicación, y corresponden a líneas de par clave – valor. Estos archivos son accedidos a través de una clase de Java y usualmente se usan para almacenar datos configurables, de manera que al ser modificados, la aplicación no se entera de dicha reconfiguración, ella simplemente seguirá leyendo este fichero sin importar que los valores de las claves cambien.

Para la aplicación fue necesario usar ficheros properties para darle un grado de mantenibilidad, de manera que a continuación se describirán los distintos ficheros usados.

9.4.1. DBXMLProperties.properties

Este fichero permite almacenar variables acerca de la persistencia de los usuarios en la aplicación, ya sean aquellos en espera o los activos, además de la ubicación de los documentos XML que los almacenan. En caso de que la ubicación de los XML en el servidor cambie, solo hay que modificar la variable location, de igual manera con las variables que correspondan si los nombres de los archivos cambian.

```
4 #direccion de la bd
5 location=DB
6 #archivo en el cual se almacenan los usuarios registrados
7 users=users.xml
8 #archivo en el cual se almacenan los usuarios en espera
9 useronhold=useronhold.xml
```

Figura 61. Property para la Persistencia de Usuarios

9.4.2. ExtensionFileUploadProperties.properties

Este fichero cobra importancia cuan se desea cargar un fichero a un proyecto. En este se puede percibir las variables dispuestas para los tipos de ficheros que podría aceptar la aplicación e incluso aquellas que no son permitidas. En caso que se desee agregar una nueva extensión de fichero o excluir alguna que ya existan solo correspondería a modificar la respectiva variable.

```
4 #Extensiones permitidas para subir ficheros java
5 java=java
6 #Extensiones permitidas para subir fotos
7 photo=png,jpg,jpeg,gif
8 #Extensiones no permitidas para subir archivos
9 noFile=exe,sh,bat
10 #Extensiones permitidas para librerias java
11 javaLibraries=jar
```

Figura 62. Property de Configuración de Extensión de Ficheros a Cargar

9.4.3. KeyEncryptProperties.properties

Este fichero es importante para los algoritmos de encriptado y desencriptado, ya que parte de sus algoritmos necesitan de una llave tanto para encriptar como para desencriptar de manera que fue necesario generar dichas llaves y almacenarlas en dos archivos: AES.hty y DES.hty. Si en algún momento fuera necesario cambiar la ubicación de los archivos o los nombres de los mismos, solo bastaría con modificar las variables que correspondan en el property.

```
4 #ubicacion de los archivos correspondientes
5 #a los Key de encriptacion
6 path=KeyCripter
7 #nombre del archivo correspondiente a la key AES
8 AES=AES.hty
9 #nombre del archivo correspondiente a la key DES
10 DES=DES.hty
```

Figura 63. Property de Llaves para Encriptar y Desencriptar.

9.4.4. MatchFolderProperties.properties

Fichero que indica la ruta en donde se encuentra un archivo que cobra importancia en la gestión de las sesiones activas, este archivo plano mantiene una cadena string que contiene al programador y un identificador de la sesión activa. Este fichero es llamado “archivo de emparejamiento” y es un puente informativo de la sesión asignada al programador entre Java y NodeJS. De manera que NodeJS pueda tener información acerca de las sesiones activas en un momento dado en la aplicación y así a través de un servidor de notificaciones puedan gestionar que solo exista una sesión activa por programador. Por ejemplo si un programador se loguea en la aplicación pasará por un proceso de validación, y si es exitoso se toma la fecha del instante y se almacena como última entrada, y además se modificará el elemento de la sesión en el xml de usuarios activos, y cuando este proceso termina procede a escribir el usuario y el identificador de la sesión en el archivo de emparejamiento. El fichero property almacena la ruta donde se encuentra dicho archivo para que sea alcanzado por NodeJS cuando verifica las sesiones. Este archivo de emparejamiento fue necesario para evitar que NodeJS tuviera un acceso directo con el xml de usuarios activos, violando el

patrón DAO, y posiblemente creando un “cuello de botella” en la seguridad. Más adelante se explicará con mayor detalle estas consideraciones.

```
4 #Ruta relativa del archivo de emparejamiento con NodeJS  
5 path=MatchFolder/sessionActives.txt
```

Figura 64. Fichero Property para el Archivo de Emparejamiento

9.4.5. RegisterUserProperties.properties

Este fichero permite almacenar una variable que contiene una ruta relativa del JSP que se encargará del proceso de activación del usuario temporal. Si fuese necesario cambiar el JSP que atenderá este proceso, solo se modificaría el valor de la variable, y cuando el visitante desde su correo electrónico abre el enlace de activación enviado, será redirigido a la página que indique esta variable.

```
4 #pagina hacia donde se enviara la persona que desea activar una cuenta  
5 page=IDEWeb/IDE/activarProgramador.jsp
```

Figura 65. Fichero Property para la Activación de Usuarios Temporales.

9.4.6. SignerProperties.properties

Este fichero es importante para la configuración del firmador que se encargará de dar un certificado de autenticidad al cliente o programador en este caso, de que el applet que lanzará la ejecución del proyecto usará recursos de la máquina del cliente, específicamente la máquina virtual. En la sección 6.8.2. muestran una forma de lograr una ejecución remota a través de un applet que incluye archivos JAR firmados en un navegador, y que para esto es necesario crear un almacén de claves, con password y un alias que permita crear un certificado de autenticidad. El fichero almacena estos parámetros, y pueden ser modificados si se amerita, y la aplicación no sufrirá ningún daño por el cambio.

```
4 #contraseña del almacen de claves
5 storepass=passKEYSTOREpass
6 #contraseña del archivo signado
7 keypass=passIDEpass
8 #alias del firmado
9 alias=UFPS
```

Figura 66. Fichero Property para el Firmador de Applets.

9.4.7. UserLibrariesProperties.properties

Este fichero *property* tiene una variable que almacena el nombre del directorio que contendrá aquellas librerías que por defecto serán cargadas en un proyecto al crearlo.

```
2 #Folder donde se ubican la librerias disponibles para el usuario
3 location=UserLibraries
```

Figura 67. Fichero Property de Librerías por Defecto.

9.4.8. UsersProperties.properties

Fichero *Property* que indica el nombre del directorio que contiene todos los directorios de los diferentes programadores en el servidor.

```
4 #ubicacion relativa de los proyectos
5 path=Users
```

Figura 68. Fichero Property del Directorio de Usuarios en el Servidor.

9.4.9. WEBConsoleLibrarieProperties.properties

Property que posee variables que indica el nombre del directorio en donde se encuentran las clases que conforman la consola WEB de ejecución, y el archivo xml que las indexa.

```

4 #ubicacion relativa de las clases de consola web
5 path=WEBConsoleLibrerie
6 #nombre del xml que contiene las rutas de las
7 #clases pertenecientes a la consola web
8 nameXML=lwc.xml

```

Figura 69. Fichero Property Consola WEB.

9.4.10. WebSiteProperties.properties

Este fichero tiene mucha importancia ya que contiene datos de configuración vital para la aplicación. Si por circunstancias se debe cambiar el servidor se modifica la variable que corresponde, ya que este valor es usado para la concurrencia y notificaciones, además el correo es un dato que puede variar mucho por lo tanto también era necesario incluirlo, ya que si este el envío de correos no funcionaría.

```

4 #Direccion del servidor de la aplicacion
5 server=sandbox1.ufps.edu.co:8080
6
7 #foto de sera copiada por default en cada carpeta de usuario
8 photoDefaultUser=Images/SupportWindow/userProgrammer.png
9 #Email de la aplicacion
10 email= Correo Electrónico de la Aplicación
11 #Password del correo
12 passEmail= Contraseña del Correo

```

Figura 70. Fichero Property de la Aplicación.

9.5. HERRAMIENTAS DE DESARROLLO

La idea de desarrollar una IDE online fue pensada inicialmente como un aplicación web con las funciones básicas de toda IDE, por lo tanto como mínimo debía tener un editor para escribir las clases, funciones de compilación, ejecución, y la posibilidad de crear proyectos, crear paquetes y clases. Mientras el desarrollo del proyecto avanzaba, se presentaron oportunidades de agregar o potenciar funciones, ya que inicialmente no eran posibles por la falta de conocimiento técnico.

Afrontar una historia de usuario implica un análisis, un diseño preliminar, y en algunos casos alguna investigación de herramientas disponibles que

podrían aportar ideas de implementación, de manera que se obtenga una visión de cómo se debería implementar. En algunas historias de usuario fue necesario investigar qué librerías o herramientas existían en la web que podrían ser adaptadas al desarrollo, con el fin de reutilizar trabajos de terceros, siempre y cuando estos sean libres y dando el respectivo crédito al autor. A continuación mencionaremos qué herramientas fueron usadas en el proyecto, y cuál fue el proceso por el cual se llegaron a ellas.

9.5.1. DHTMLX

DHTMLX es una librería JavaScript que provee un completo conjunto de componentes de interfaces de usuario accionados por AJAX para su interacción entre el cliente y el servidor. Esta librería permite construir aplicaciones web de nivel empresarial con interfaces limpias, de alto rendimiento, que proporcionan una rica experiencia de usuario.⁹² En varias ocasiones se ha mencionado que el objetivo gráfico de la aplicación es que el usuario interactúe con ella, de igual forma como lo hace en una aplicación en ambientes *desktop*. Pero la implementación de esta consideración en la Web implica un trabajo bastante arduo, ya que es necesario simular un escenario de explorador de directorios, una interface gráfica de usuario de una IDE convencional, ventanas, menús, barra de herramientas, árboles de directorios, paleta de componentes, entre otras. Ante esta necesidad y la gran cantidad de horas que implicaría implementar estos componentes desde cero, se recurre a realizar una búsqueda en la web de librerías JavaScript que permitieran alivianar esta implementación. Durante esta búsqueda se encontraron algunas implementaciones hechas en JSF (*Java Server Faces*), y otras a través de applets, pero estas no tendrían utilidad para el ambiente e interacción de la aplicación. Pero conforme avanzó la búsqueda surgieron ciertas librerías que fueron analizadas para determinar cuál de ellas era la más adecuada para el desarrollo de la aplicación. Algunas de estas librerías tienen de demos disponibles para mostrar sus distintas funcionalidades, y entre las analizadas se encuentran:

- Vaadin: <http://demo.vaadin.com/sampler>

Es una Librería o framework en Java libre, que permiten construir aplicaciones web modernas de gran aspecto y buen rendimiento. Esta librería es usada como plugin en Eclipse, y las interfaces gráficas son programadas, con los respectivos componentes y eventos. En tiempo de

ejecución la carga de la construcción de la interface gráfica está dirigida al servidor, lo cual es un punto que desfavorece para la implementación de un proyecto que puede tener alta congestión.

- LivePipe UI: <http://livepipe.net/>

Es una suite de componentes gráficos y controles de alta calidad para construir aplicaciones web 2.0 usando el *Framework Prototype de JavaScript*. Cada control ha sido probado, es altamente extensible, totalmente documentado. LivePipe introduce un mecanismo para crear y observar eventos sobre cualquier objeto, no justamente elementos DOM, permitiendo obtener un control minucioso de la interface de usuario. LivePipe no cuenta con un manejador de paneles o *layouts*.

- Alloy User Interface: <http://alloy.liferay.com/demos.php>

Alloy es un meta-framework de interfaces de usuario que provee una consistente y simple API para construir aplicaciones web a través de los tres niveles de un navegador: estructura, estilo y comportamiento. En esto se incorpora tres lenguajes de diseño: HTML, CSS y Javascript. Al igual que LivePipe, Alloy cuenta con una serie de componentes gráficos o *widgets* con el desarrollador cuenta para construir sus interfaces gráficas de usuario, pero no cuenta con un componente de *layout* para la distribución de contenidos.

- jQuery Tools: <http://jquerytools.org/>

jQuery Tools es una colección de los más importantes componentes de interface gráfica de usuario para la construcción de modernos sitios web. JQuery Tools es muy limitado en componentes gráficos convirtiéndose esto en su desventaja frente a HTMLX.

- jQuery User Interface: <http://jqueryui.com/>

jQuery UI provee abstracciones de bajo nivel de interacción y animación, efectos avanzados, componentes gráficos o *widgets* con temas de diseño, construidas sobre la librería JavaScript JQuery, y se puede usar para crear aplicaciones web altamente interactivas. Esta librería es una de las más interactivas y compatibles pero no cuenta con un widgets que se comporte como un layout.

- JxLib: <http://jxlib.org/>

JxLib es una librería JavaScript para la creación de interfaces gráficas de usuario en la web basada sobre la librería MooTools. A pesar de que es una de las librerías más completas, hablando de componentes gráficos, está no está basada en JQuery.

- Sigma Visual –AJAX GUI visual builder:
<http://www.linb.net/VisualJS/UIBuilder.html>

Sigma Visual – AJAX GUI visual builder es una aplicación web que permite crear visualmente interfaces gráficas de usuario, con componentes arrastrables a un lienzo de diseño, que transparente al usuario traduce a javascript y puede ser ejecutado a través PHP. Esta aplicación fue útil para empezar a tener una idea sobre el constructor de interfaces gráficas de la IDE.

9.5.2. Ace Editor

Es un editor de texto para lenguajes de programación escrito en JavaScript, coincide y extiende las características, la usabilidad y el rendimiento de los editores nativos que existen.⁹¹

Una de las partes o componentes fundamentales de una IDE, sea cual sea el lenguaje que soporte, es el editor de código. Algunas veces se escucha decir: “yo programo en un block de notas, compilo y ejecuto por consola y no necesito una IDE”, esto es un comentario válido y real, de hecho tiene razón, no necesitamos una IDE para programar, pero si hiciéramos las siguientes preguntas: ¿es cómodo para usted programar en un editor de texto plano?, ¿se le facilita la lectura y comprensión del código en un editor de texto plano?, ¿es sencillo determinar dónde se cierra o se abre una llave, dónde termina un método?, creerían entonces que las respuestas serán a favor de la programación en texto plano?. Entonces es aquí donde cobra suficiente importancia un componente que le brinde al programador un entorno más cómodo, interactivo y dinámico cuando programa.

Algunos editores tienen más características que otros, pero un editor por más sencillo que fuese debe al menos tener la propiedad de resaltar la

sintaxis del lenguaje a través de colores de fuente de acuerdo a criterios definidos.

Para el proyecto era muy importante ofrecer un editor de código que estuviese a la par con los editores de las IDE-Desktop más conocidas, de manera que era indispensable un componente que llenara las expectativas de un programador cuando escribe código.

En la búsqueda de componentes desarrollados por terceros, se seleccionó Ace Editor por su dinamismo, la capacidad de configurarse y personalizarse de acuerdo a las necesidades, al soporte de muchos lenguajes, a sus características de desempeño y usabilidad basadas en la ejecución de un demo disponible en: <http://ace.ajax.org/build/kitchen-sink.html>. Otro aspecto importante que se tuvo en cuenta sobre este componente, fue su documentación y los recursos de ayuda que le ofrecen al usuario donde pueden expresar sus dudas para lograr conseguir a través de estos, una orientación y soporte. Pero no solo se analizó este componente. Los siguientes componentes fueron analizados de igual forma como se hizo con Ace Editor:

- CodeMirror.js <http://codemirror.net/>

Es un componente JavaScript que provee de un editor de código en un navegador. Cuando un lenguaje en específico se encuentra entre los modos de resaltado código del componente, este se encarga de asignar colores al código, y opcionalmente ayuda con espacios de sangría. CodeMirror cuenta con una rica API de programación y un sistema de temas a través de CSS disponible para la personalización del componente de acuerdo a la aplicación que lo implemente, además de la posibilidad de extender su funcionalidad.

- CodePress <http://codepress.sourceforge.net/>

Está basado en un editor de código fuente con resaltador de sintaxis escrito en JavaScript, el cual asigna color al texto en tiempo real mientras se teclea en el navegador. El inconveniente de este componente es su limitado soporte de lenguajes, de manera que se descartó de inmediato.

9.5.3. NodeJS

Este componente al igual que los siguientes surgió una vez se inició la investigación acerca de la posibilidad de integrar la edición colaborativa en tiempo real, y que todos los cambios que se efectúen en un proyecto sean notificados a todos los programadores que lo comparten. NodeJS es la plataforma que permite que la concurrencia y las notificaciones puedan ejecutarse. Analógicamente NodeJS cumple la función de la máquina virtual de Java, y los componentes ShareJS y Socket.IO actúan como archivos JAR; esta analogía permite entender de alguna manera cuál es el rol que cumple NodeJS en el servidor con los componentes ya mencionados que corren sobre él.

9.5.4. Socket.IO

Socket.IO es el componente JavaScript que permite establecer canales de comunicación entre navegadores web y el servidor, donde se encarga de decidir el mecanismo de transporte que esté disponible en el navegador para establecer los canales de comunicación full dúplex, por el cual transmitan mensajes en tiempo real.

9.5.5. ShareJS

Un valor agregado que a la aplicación le ha sido otorgado es la posibilidad de que un grupo de programadores que trabajen en un proyecto compartido en la aplicación, puedan editar simultáneamente sobre una clase, o en clases diferentes, pero siempre manteniendo una única copia del proyecto en una misma versión para todos los participantes. Para llevar a cabo lo anterior fue necesario iniciar un búsquedas de todo aquello que aportará en la implementación o integración de dicha funcionalidad. Durante la búsqueda se encontraron algunos protocolos e incluso componentes desarrollados que implementaban edición colaborativa en tiempo real, de los cuales se analizaron, se realizaron algunas pruebas a algunos demos, para posteriormente escoger el más adecuado para la integración con la aplicación. A continuación será presentado el resultado de la consulta:

- **Jinfinote:** Es una implementación en Javascript de un protocolo llamado *infinote*⁹⁵, que básicamente organiza toda la comunicación en grupos, donde cada uno está conformado por usuarios que envían mensajes unos con otros entre el grupo. Proporciona un estándar abierto para el desarrollo de software que soporte edición en tiempo real de archivos de texto. Esta implementación se divide en 4 archivos de Javascript:⁹⁶
 - *Operations.js*: contiene las operaciones básicas para textos como por ejemplo insertar, eliminar, actualizar, dividir, restaurar, e incluso un tipo de operación llamado NoOp que simplemente no refiere nada.
 - *Request.js*: Define las peticiones de hacer, deshacer y rehacer.
 - *State.js*: Define las clases de estado y vector, que ayudan al control de las incidencias que cada usuario genera.
 - *Text.js*: Este Javascript es necesario para mantener un registro de qué usuario ha escrito en qué parte.
- **Etherpad:** Permite editar documentos colaborativamente en tiempo real. Cuenta con dos versiones una llamada *Etherpad* que tiene un tamaño de 30 MB y consume 257 MB de memoria una vez arranca; y una versión ligera llamada *Etherpad Lite* que está totalmente soportada sobre NodeJS, y su tamaño es de 1.5 MB y consume 16 MB de memoria en su arranque⁹⁷.
- **Fitzgen – Operational Transformation:** Implementación Javascript del Transformación operacional. <https://github.com/fitzgen/operational-transformation>
- **Google Wave Protocol:** Protocolo definido por Google en el desarrollo de su aplicación Google Wave. En él se explica al detalle la manera cómo opera la edición colaborativa en tiempo real⁹⁸.
- **ShareJS:** Es básicamente un librería de transformación operacional para NodeJS y navegadores, implementada en Javascript. Permite integrar la edición concurrente con facilidad en una aplicación. Además usa a Socket.IO para el transporte de los mensajes que cada

⁹⁵ Protocolo *Infinote*, Comunicación por Grupos <http://infinote.org/Protocol/Groups/>

⁹⁶ Implementación en Javascript del Protocolo *Infinote*. <https://github.com/sveith/jinfinote>

⁹⁷ Etherpad. Implementación de edición colaborativa en tiempo real. <http://etherpad.org/>

⁹⁸ Protocolo Google Wave OT. <http://www.waveprotocol.org/whitepapers/operational-transform>

usuario emitiría cada vez que edita un carácter. Básicamente opera a través de operaciones que indican insertar el siguiente texto en la posición x, y al igual que la *subversion* maneja versionamiento. Esta librería se asegura que en todo momento los diferentes usuarios que están editando tengan el mismo documento.

Después de revisar algunos demos disponibles en repositorios, y algunas pruebas locales, se determinó que el componente más adecuado para integrar a la aplicación es *ShareJS*, ya que en su implementación usan *Socket.IO* (también usado en las notificaciones) para la comunicación entre usuarios donde se transmiten en tiempo real cada cambio que se efectúa. Un criterio que facilitó la decisión de usar *ShareJS* para la concurrencia, fue gracias a que en la página oficial del autor, en su sección de demos se pública la integración de este componente con un editor de resaltado de código fuente llamado *ace editor* ([9.5.2. Ace Editor](#)), siendo este último el componente para integrar a la IDE en el editor de clases. En la sección [8.1.3. Familiarización Con Tecnologías, Herramientas y Prácticas](#) se presenta *ShareJS* en uno de los puntos allí descritos.

9.6. ESPECIFICACIÓN DE FUNCIONALIDAD DE LA APLICACIÓN

Inicialmente el desarrollo de la aplicación estaba orientado a obtener una funcionalidad básica, de manera que el usuario tuviera a su disposición una herramienta en la cual pudiese desarrollar pequeños programas y sencillas rutinas de programación, pero en el transcurso de las iteraciones se logró darle valor agregado a la aplicación funcionalmente hablando.

En las doce iteraciones descritas en el apartado [8.6](#), se puede ver la evolución en la construcción de la aplicación, destacando funciones que no se encuentra entre las básicas de cualquier IDE, como la edición concurrente entre programadores sobre un proyecto, el constructor de interfaces gráficas y la notificación de las operaciones que ocurren sobre un proyecto a todos los programadores que participan en este.

A continuación es necesario describir de manera conceptual las principales funciones de la aplicación, soportándonos en una serie de diagramas que expliquen claramente al lector del comportamiento de cada una.

9.6.1. Registro y Activación de Programador

Registrar y activar un nuevo programador es un proceso que se lleva a cabo en dos escenarios, con el fin de evitar la creación de cuentas de programador de manera descontrolada. El proceso básicamente consiste en hacer un registro temporal, y esperar una activación por parte del usuario a través de un enlace enviado a su correo electrónico registrado. Solicitar el email del usuario y enviar un link de activación a este, es una manera de asegurar que no se creen cuentas sin intención de uso, de manera que si desean crear una, deben ingresar un email válido, pero esta no será una cuenta activa para aplicación hasta que haya sido activada. Si la intención fuese crear cuentas de manera automática desde cualquier procedimiento, solo podrían conseguir llenar de registros “basura” el archivo *useronhold.xml*, que simplemente almacena posibles programadores y esto no tendría ningún impacto directo en la funcionalidad, ya que solo se tiene en cuenta los programadores que estén almacenados en el archivo *users.xml* que representan los programadores activos. En los siguientes diagramas de actividades se muestra la manera en la cual se lleva a cabo estas operaciones:

9.6.1.1. Registro Temporal de Usuario

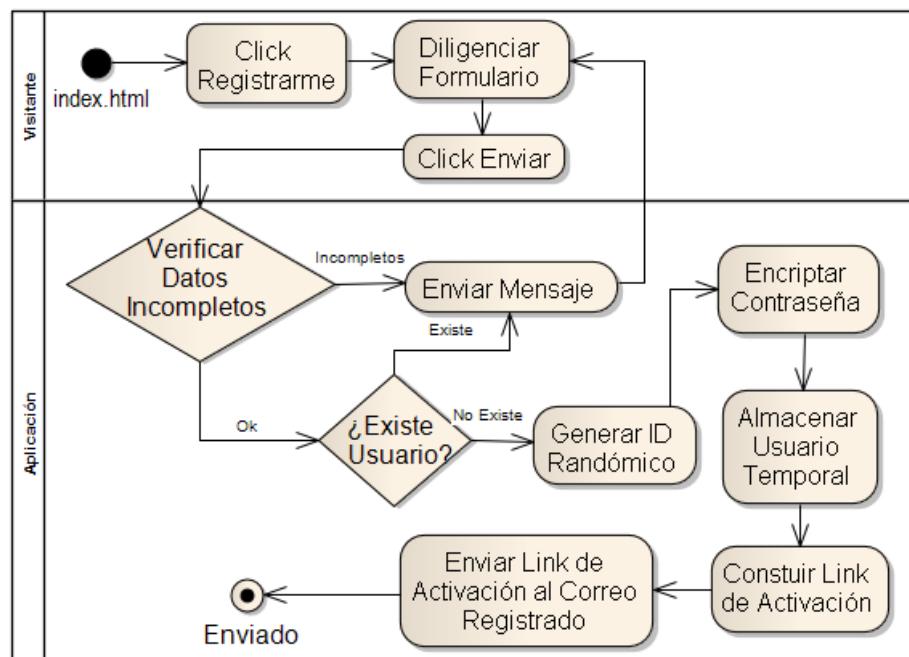


Diagrama 15. Descripción del Proceso Registrar un Usuario Temporal

Especificación

1. El visitante ingresa a la página index.html de la aplicación, en la cual encontrará un formulario de *login*, desde donde podrá observar un botón de “Registrarme”.
2. El visitante pulsa el botón Registrarme, el cual se encarga de mostrar un formulario de registro.
3. El visitante diligencia los respectivos campos del formulario
4. El visitante pulsa el botón que envía la petición.
5. La aplicación se encarga de verificar si los campos están diligenciados. Si algún campo está incompleto, se notifica del error al usuario. Si los campos fueron diligenciados correctamente procede a invocar a *registrarProgramador.jsp* y envía los respectivos parámetros que corresponden a los datos que el usuario ingresó.
6. A nivel de lógica, específicamente en el *Control_WebSite*, se verifica que no exista un usuario registrado con el correo ingresado, tanto en los archivos de persistencia de usuarios *useronhold.xml* y *users.xml*. Si existe se devuelve como respuesta un false hasta la vista y allí envía la notificación al visitante.
7. Si en la persistencia de los usuarios no existe un registro que coincida con el correo diligenciado, procede a generar una cadena randómica (cadena de 50 caracteres con posibilidad de que cada carácter tome un valor de 62 posibles y los caracteres se pueden repetir), que servirá de verificador para cuando el usuario abra el enlace de activación. Este id randómico quedará almacenado en la persistencia de usuario temporal para compararlo con el id que llega en los parámetros del enlace de activación. La probabilidad de que un id coincida viene dada por: uno sobre una permutación con repeticiones de 62 valores posibles elevado a la longitud de la cadena que es 50, definido así:

$$\frac{1}{62^{50}} = 2.401129178e^{-90}$$

Entonces, dada esta probabilidad, la intención de que alguna persona intente a propósito coincidir con una id, queda reducida a casi nula.

8. Se encripta la contraseña suministrada para darle un grado de seguridad. Esto implica que la persistencia no conoce la contraseña real del usuario.
9. Una vez encriptada la contraseña, se procede almacenar el usuario en el archivo *useronhold.xml*, con los valores diligenciados en el formulario, además de la id generada y la contraseña encriptada.
10. El usuario ha sido almacenado, ahora se debe construir el enlace que permitirá activar al programador. En este enlace, se embeben datos el email del usuario y la id generada.
11. Por último se envía un email que contiene dicho enlace de activación a la cuenta de correo suministrada.

9.6.1.2. Activar Programador

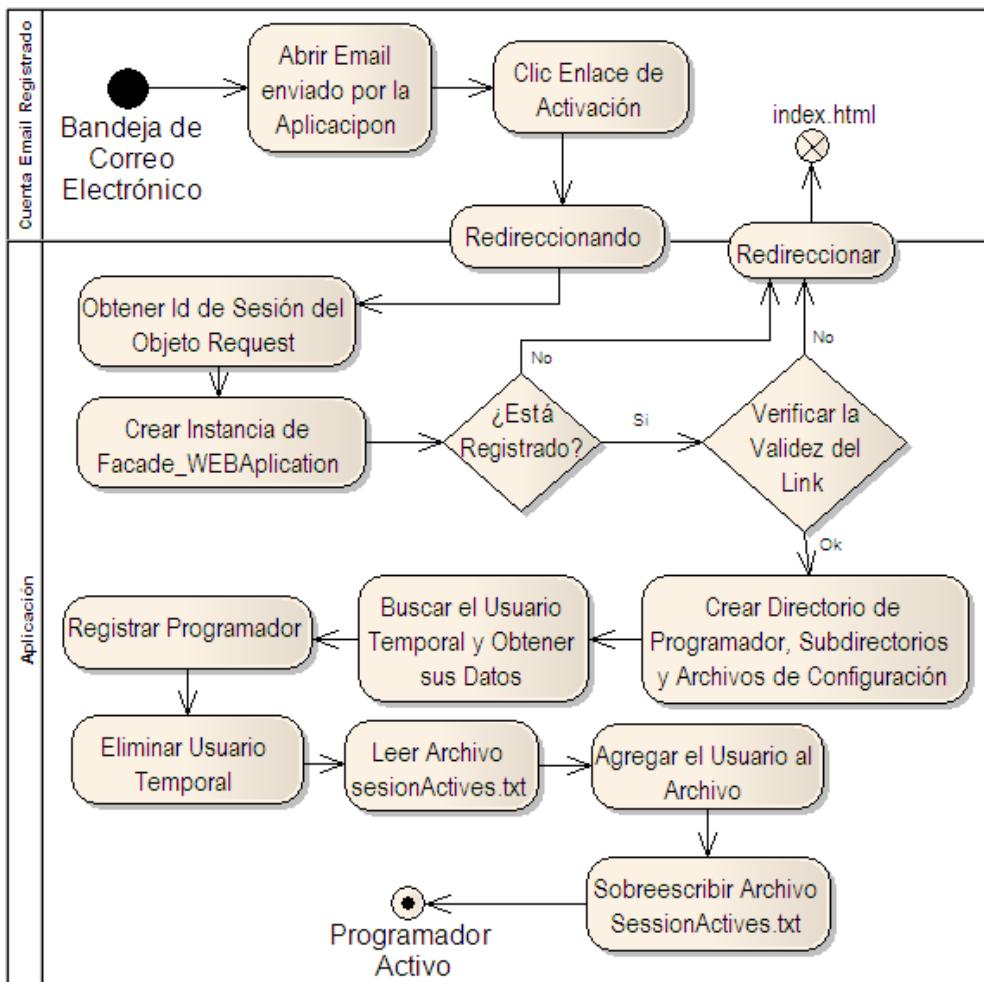


Diagrama 16. Descripción del Proceso Activar Programador

Especificación

1. El usuario que se ha registrado previamente, deberá ir a su bandeja de entrada de su cuenta de correo registrado.
2. Buscará y abrirá el email enviado por la aplicación, en el cual estará un link de activación.
3. El usuario dará clic en dicho enlace, que contiene como parámetros, el email y un id randómico.
4. Al dar clic se efectuará un redireccionamiento desde la bandeja de entrada de la cuenta de correo del usuario, a una página de activación de la aplicación.
5. En este punto se obtiene un identificador de la sesión que está contenida en el objeto *request* de la petición.
6. Se procede a crear una instancia del *Facade_WEAApplication* y se le envía dicho identificador de sesión como parámetro.
7. A través de la anterior instancia, se verifica si en el archivo *useronhold.xml* existe el usuario que viene como parámetro desde el enlace. Si este no está se redirecciona al *index.html* y termina el flujo. Si está se devuelve un objeto *Document* que embebe los datos obtenidos del archivo xml.
8. Se lleva a cabo una verificación de la validez del link proveniente, a través de la comparación de la id que viene como parámetro, con aquella id que se asoció y almacenó cuando el usuario fue registrado. Esto permite que, si aún conocieran la estructura del link e intentaran enviar uno a propósito, este no sería válido ya que es muy pero muy poco probable que dieran con una id registrada con el email enviado, tal y como lo indica el cálculo efectuado en la anterior especificación. Si el link no es válido, es decir, el email y la id tomados del link no tienen correspondencia con algún registro del archivo *useronhold.xml* se redireccionará al *index* de la aplicación. Es necesario aclarar que un link solo es válido una vez, de manera que si el usuario ya fue activo, el link dejará de ser válido.
9. Si el link fue válido para aplicación, esta procederá a crear un directorio en el servidor para el nuevo programador, unos subdirectorios y una serie de archivos de configuración para el nuevo programador. En este directorio estarán archivos xml correspondientes a la información del programador, a un indexador de proyectos, y otros directorios más, tal y como se describe en el apartado [9.3.3](#).

10. Se procede a tomar los datos del usuario registrado en *useronhold.xml*.
11. Se registra el programador en *users.xml* para que la aplicación lo asocie como una cuenta activa.
12. Se elimina el registro del usuario temporal, ya que este ya fue activado. Esta es la razón por la cual el link solo es válido una vez, ya que por segunda vez, el validador no encontrará ningún registro asociado a los datos que provienen del enlace.
13. Se lee el archivo plano *sessionActives.txt* que cobrará importancia cuando se gestione la validez de la sesión en un momento dado.
14. Se agrega el nuevo Programador a dicho archivo con un identificador de sesión arbitrario por el momento. El archivo contiene elementos que están formados por el correo electrónico del usuario y un identificador de sesión separados por dos puntos, y cada pareja usuario-id es separado por un punto y coma, como lo muestra el siguiente ejemplo:
pepito@pepito.com:x3urjjdHGj;pepito2@pepito2.com:tYt6uuoin;
15. Por último se vuelve a escribir el archivo para que los cambios queden efectuados, y por lo tanto el proceso de activación de un programador culmina.

9.6.2. Encriptar Contraseña

Con el único propósito de brindar seguridad con respecto a la gestión de contraseña, se decide usar un algoritmo de encriptado para tomar la contraseña ingresada por el visitante al registrarse, y aplicarle una codificación para que la persistencia no conozca la clave real. El encriptado de la contraseña evita que si por alguna circunstancia de ataque a la aplicación logran tener acceso al archivo *users.xml*, no pueden tener una contraseña válida para ingresar en la pantalla del login de la aplicación.

9.6.3. Login y Gestión de Sesión

El proceso de loguearse en la aplicación ya sea un programador o el administrador cobra un papel importante, primero porque es la llave de entrada de la aplicación, y segundo porque es a partir de este proceso que inicia el control de las sesiones. Ahora, con respecto a las sesiones, es necesario tener un control estricto debido a que se pueden presentar

circunstancias en las que invoquen algún *jsp* sin haberse logueado, o se puede presentar que un programador haya iniciado sesión, y otra persona se loguea con la misma cuenta.

9.6.3.1. Login

El inicio de sesión o login para el programador exige como datos requeridos el correo electrónico registrado y una contraseña. En el caso del administrador exigirá un nombre de usuario o correo electrónico según se haya establecido y una contraseña. El diagrama 17 describe el proceso de inicio de sesión:

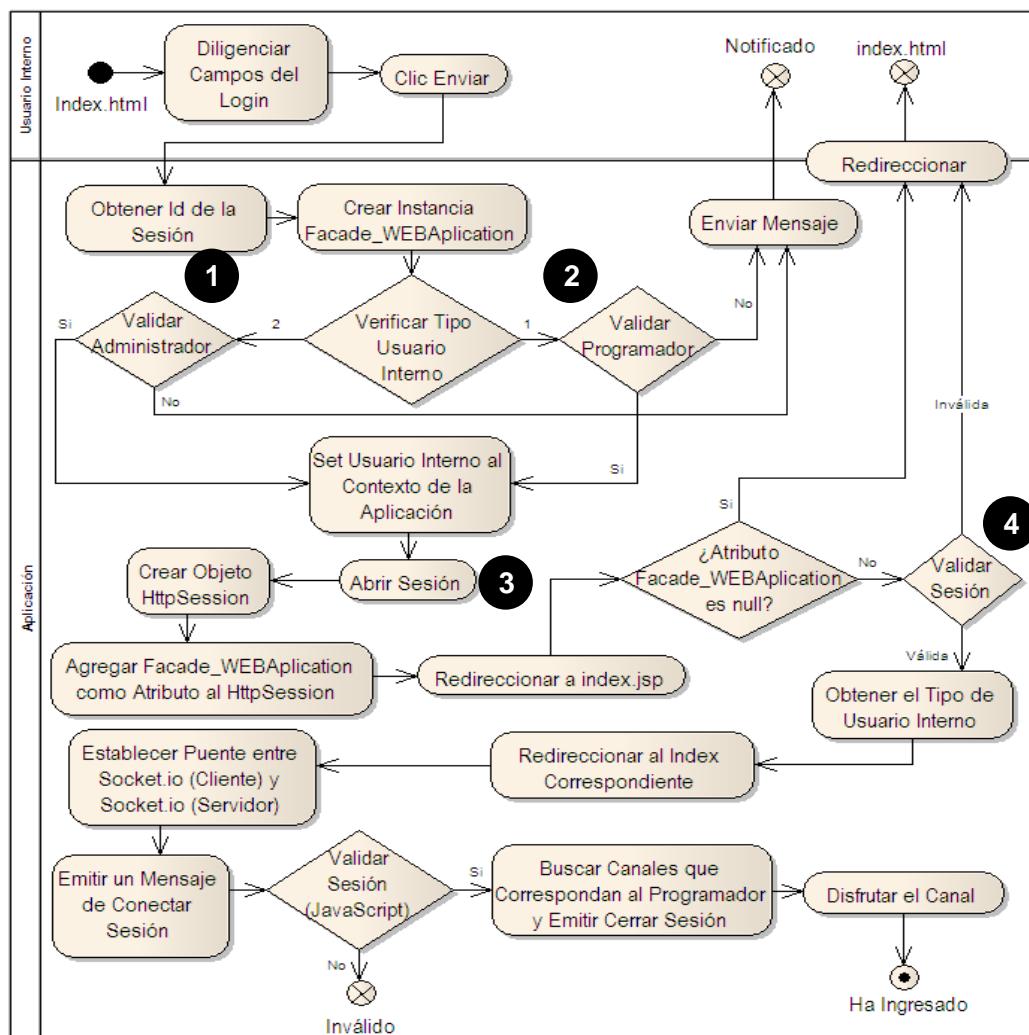


Diagrama 17. Descripción del Proceso de Inicio de Sesión

Especificación

1. El Programador o Administrador se encuentra en el index de la aplicación y desea iniciar sesión.
2. El usuario Interno diligencia los campos requeridos para el inicio de sesión.
3. Pulsa clic en enviar.
4. La aplicación obtiene la id de la sesión del objeto request.
5. Se crea una instancia del Facade_WEBApplication incluyendo la id de la sesión. Este objeto Facade es el que interactúa directamente con la vista, a partir de él empieza toda la lógica del negocio.
6. La aplicación identifica que tipo de usuario interno está realizando la petición, ya que se le envía dos valores posibles, 1 si es un programador y 2 si es el administrador.
7. Si la opción fue 1 se procede a validar el programador. Si el tipo es 2 se debe verificar al administrador (ver especificación 1 y 2 más adelante). Si la validación falla, sea cual sea el camino que tomó el filtro, se debe enviar un mensaje al usuario interno de la falla en la validación, y allí terminará el flujo.
8. Si la validación que corresponde es válida se define en el contexto de la aplicación qué tipo de usuario interno iniciará sesión, y a continuación se debe hacer un set de usuario al contexto de la aplicación, esto permitirá que de ahí en adelante la aplicación conozca el usuario y el tipo al cual pertenece. Por ejemplo, si el programador con correo electrónico *pepito@pepito.com* quiere iniciar sesión, la aplicación identifica que es un programador y se procede a verificar si es un programador activo, si esto es correcto, se le debe decir al contexto de la aplicación que es un programador y que es el usuario *pepito@pepito.com*.
9. Una vez hecho lo anterior se procede a abrir la sesión (Ver especificación de 3 más adelante).
10. Si la sesión fue abierta, se continúa creando un objeto de tipo HttpSession inicializado con la sesión del request, esto nos permitirá tener un control sobre dicha sesión, logrando crear en él atributos que almacenen información relevante para el flujo de la aplicación.
11. Ahora al objeto HttpSession se le agrega como atributo el objeto Facade_WEBApplication creado anteriormente. Este objeto sesión queda cargado en memoria durante toda la sesión de dicho usuario

- interno, y en todo momento se tiene acceso a los atributos que a él se le agreguen.
12. Después de agregar el atributo se realiza un redireccionamiento a *index.jsp*.
 13. Al llegar allí se verifica el estado del atributo que fue agregado, ya que se puede presentar de que alguien quisiese entrar a la aplicación a través de un link directo al *index.jsp* sin haber pasado por el proceso de inicio sesión. En caso de que lo anterior suceda y como se pasó por alto el inicio de sesión, la verificación del estado del atributo facade será *null* y se procederá a redireccionar al *index.html*.
 14. Ahora si el atributo no es *null*, se debe verificar la sesión y esto es para comprobar si alguien ha iniciado sesión con la misma cuenta, mientras el usuario interno tenga una sesión activa. Esto controla que solo exista una sesión activa por cuenta de usuario interno. (Ver especificación de 4 más adelante). Si la sesión es inválida se redirecciona al *index.html*.
 15. Si la sesión es válida, se debe obtener el tipo de usuario que fue definido en el paso 8.
 16. Dependiendo del tipo de usuario interno se determina si se redirecciona al *indexProgramador.jsp* o *indexAdministrador.jsp*.

(Nota: los siguientes pasos corresponden a la integración del componente *socket.io* para el gestor de notificaciones, y para el caso del control de la sesión que será descrito en detalle en 9.6.6. Notificaciones en Tiempo Real)

17. Una vez termine la redirección al *index* que corresponda, se establece un puente de conexión entre el *socket.io* instanciado en el cliente, y el *socket.io* alojado en el servidor. A través de este puente se establecerán canales por los cuales pasarán todas las notificaciones que se presente en la aplicación.
18. El cliente emite un mensaje al servidor de realizar una conexión de sesión, esto básicamente indica que establezca un canal entre el servidor y el cliente. En esta petición se envía un objeto JSON donde los atributos corresponden al usuario y el identificador de la sesión.
19. Teniendo en cuenta que este componente se ejecuta en el cliente y en un servidor virtual que escucha por cierto puerto, y que no tiene ninguna interacción con el servidor que aloja a Java, se debe tener una validación de sesión a nivel de *JavaScript* y lo hace consultando un archivo plano *sessionActives.txt* el cual es usado cuando el

programador es activado y también en abrir sesión. En este archivo se encuentran todos los programadores y su identificador de sesión activa. Si la validación encuentra que el identificador de sesión ya no corresponde con la que se encuentra cargado en el contexto de la aplicación, no permite realizar ninguna operación.

20. Es importante mencionar antes de describir el siguiente paso, que un canal entre el cliente y el servidor se identifica con el objeto JSON recibido. Ahora si el programador es válido, procede a enviar un mensaje que ordena cerrar sesión a aquellos canales que concuerden con el usuario. Esto toma mucha importancia cuando un programador ya ha ingresado en la aplicación y ya se le ha asignado un canal con el servidor, entonces si una nueva persona inicia sesión con la misma cuenta y llega a este punto del proceso, la aplicación enviará la orden de cerrarle la sesión al programador que se encontraba en la aplicación.
21. Si la aplicación encontró un canal lo cierra, y le asigna un nuevo para aquel programador que ha iniciado sesión.

9.6.3.2. Validación de Usuarios Internos

① y ② corresponden a un proceso de verificación de usuarios internos de la aplicación, donde básicamente coinciden, en la consulta del usuario interno con el correo electrónico que ingresaron en el formulario de inicio de sesión, y si este existe, proceden a comparar la contraseña del usuario encontrado con la que digitaron. Este proceso es similar para ambos tipos de usuarios internos, a excepción de que para el administrador hay una validación más, pero aun así se describirá el proceso en el mismo diagrama y en la especificación se dará el detalle para el caso del administrador. El diagrama 18 describe el proceso.

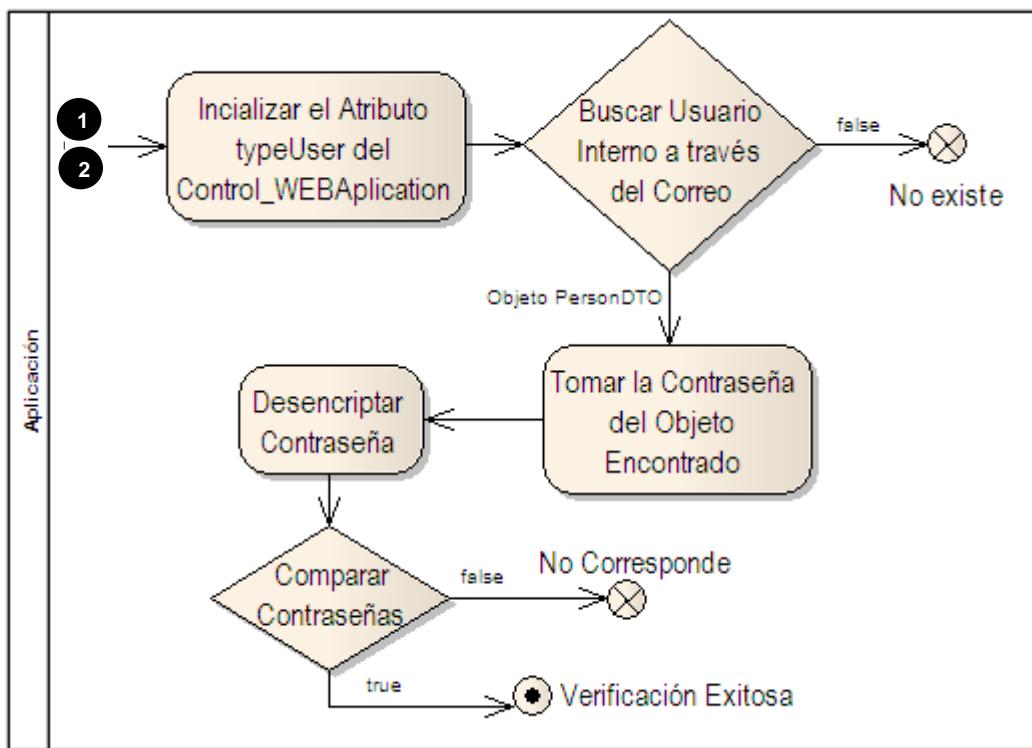


Diagrama 18. Descripción del Proceso de Validación de un Usuario Interno

Especificación

1. El proceso de verificación de un usuario interno inicia en el instante que algún flujo requiere de dicha validación. En este caso la validación está dada cuando en el proceso de iniciar sesión se solicita una verificación de un programador o el administrador.
2. Realizada la petición de la validación, y raíz de que de antemano ya se conoce el tipo de usuario interno, se define en el *Control_WEBAplication* el tipo de usuario que va a ser validado.
3. Se procede a realizar una búsqueda en el archivo *users.xml* usando el correo electrónico (tomado del formulario de iniciar sesión) como parámetro de búsqueda. Si la búsqueda no arroja resultado, se devuelve un false y terminará el flujo. Si se encuentra un registro que corresponda al correo electrónico, se devuelve un objeto *PersonDTO* que contiene como atributos: el correo electrónico, la contraseña encriptada y el tipo de usuario interno.
4. Del objeto producto del resultado se toma la contraseña encriptada.
5. Se procede a desencriptar dicha contraseña.

6. Por último se toma la contraseña ingresada en el formulario y se compara con la contraseña tomada del objeto y anteriormente fue desencriptada. Si la comparación no concuerda, se retorna un false y termina el flujo. Ahora, si la comparación concuerda se retorna un true y el proceso finaliza.

9.6.3.3. Abrir Sesión

Abrir la Sesión ③ corresponde a un proceso que tiene mucha importancia para gestionar la validez de una sesión en un momento determinado. Este es un mecanismo que ayuda a la gestión de sesiones activas de programador. La aplicación en todo momento está verificando si la sesión es válida antes de efectuar una acción. Para llevar a cabo las verificaciones es necesario almacenar el identificador de la sesión, con el cual se hará la comparación y determinar si es válida o no. El diagrama 19 describe este proceso:

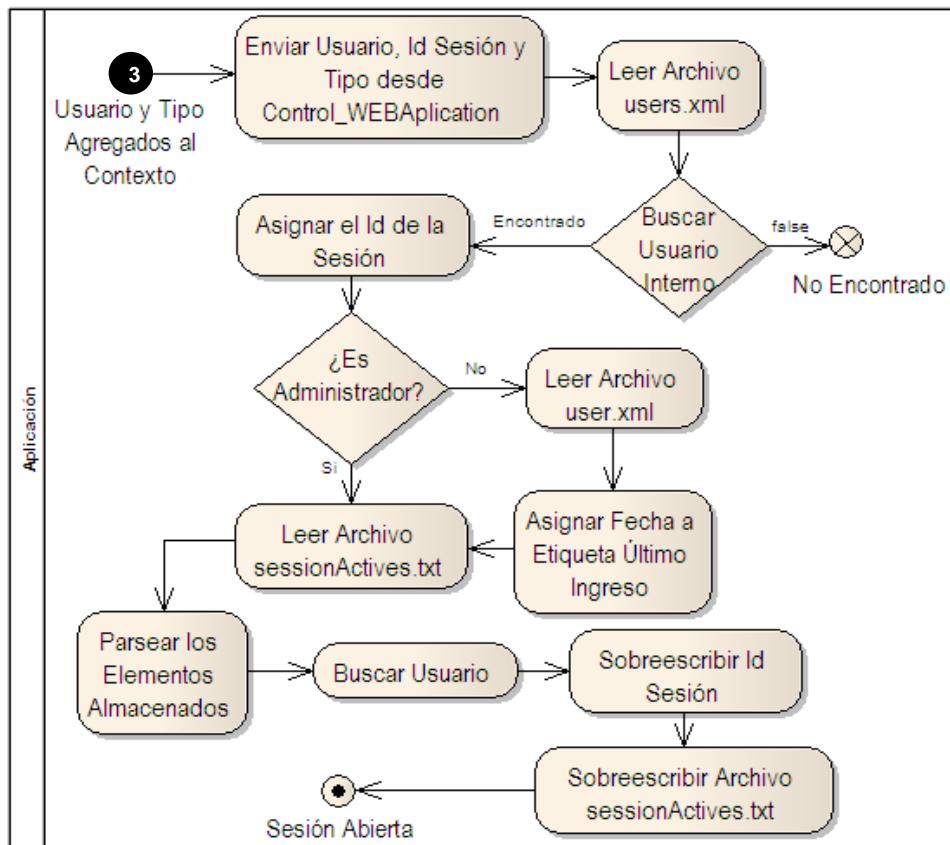


Diagrama 19. Descripción del Proceso Abrir Sesión

Especificación

1. Abrir Sesión es un proceso que hace parte del flujo Iniciar Sesión, y es activado una vez que se han definido en el contexto de la aplicación, el usuario y el tipo al que pertenece, después de la respectiva validación del usuario interno.
2. En el *Control_WEBApplication* ya se encuentra contextualizados o inicializados los atributos *user*, *idSession* y *typeUser* en lo que lleva ejecutado hasta el momento el flujo de iniciar sesión. Estos valores son enviados como parámetros en la solicitud de abrir sesión.
3. Se lee el archivo *users.xml*.
4. Se recorre cada registro que contiene este archivo en búsqueda del usuario interno activo, a través del correo que fue enviado desde el *Control_WEBApplication*. Si no se encuentra el usuario retornará un false y terminará el flujo.
5. Si el usuario fue encontrado, se le asigna a su elemento *session*, el valor del identificador de sesión recibido. De esta manera quedará almacenado en la persistencia de usuarios internos activos, el identificador de sesión definido para dicho usuario que está intentando iniciar sesión.
6. Después de dicha asignación, se pregunta si ese usuario es el administrador, ya que para este no es necesario controlar su último ingreso a la aplicación. Si no es el administrador, se lee el archivo *user.xml* que hace parte de los archivos de configuración del directorio del usuario descrito en el apartado [9.3.10](#). En este archivo se modifica el elemento *lastEntry* (elemento que almacena la fecha de último ingreso) con la fecha actual.
7. Ya sea que el usuario es el administrador o que ha culminado el paso 6, se procede a leer el archivo *sessionActives.xml*, que es el archivo más influyente en el control de sesiones.
8. Ya cargado en memoria el contenido del archivo, se hace un *split* por el carácter punto y coma, y así obtener un *array* que contiene parejas usuario-identificador de sesión, que están unidas por el carácter dos puntos.
9. Obtenido el *array*, se recorre en busca del usuario, ya que éste fue agregado al ser activado y se le asignó un identificador de sesión arbitrario como lo describe el paso 14 del proceso Activar Programador. En cada iteración de la búsqueda se hace un *split* a

- cada elemento del *array* por el carácter dos puntos, y así se obtiene por separado el usuario y el identificador.
10. Cuando el usuario es encontrado, se cambia el valor del identificador de sesión por el nuevo.
 11. Por último se reescribe el archivo *sessionActives.txt* para que los cambios queden efectuados, y aquí culmina el proceso de abrir sesión.

9.6.3.4. Validar Sesión

Validar Sesión ④ es el proceso encargado del control de las sesiones activas de los programadores. Como ya se ha descrito en los procesos anteriores, cada vez que un programador es activado, este usuario interno junto con un identificador de sesión arbitrario es almacenado en el archivo *sessionActives.txt*, y cuando dicho programador inicia sesión, es tomado el identificador de la sesión del objeto *request* y se modifica en dicho archivo, además de agregar ese mismo identificador en el elemento *session* del registro que corresponde al programador en el archivo *users.xml*. Tanto el archivo *sessionActives.txt*, como el elemento sesión del programador en el *users.xml* tienen la función de mantener actualizado el identificador de la sesión del programador para la verificación periódica de que éste no haya cambiado mientras el usuario interno esté en la aplicación. El anterior caso puede presentarse cuando un programador ha ingresado en la aplicación, por lo tanto al iniciar sesión se ha almacenado un identificador, pero si otra persona inicia sesión con la misma cuenta mientras el programador sigue usando la aplicación, será modificado el identificador ya que esa persona inicia sesión correctamente, entonces cuando el primer programador realiza otra acción, antes de ejecutarse, validará la sesión y se enterará de que el identificador que tenía en su sesión ya no corresponde con el que está almacenado en el archivo *sessionActives.txt*, por lo tanto esto resulta como una sesión inválida y lo sacará de la aplicación. La verificación del estado de la sesión de un determinado programador puede realizarse en el redireccionamiento de un *jsp* a otro antes de ejecutar la acción que está en desarrollo, y el otro caso que efectúa la validación, es en la interacción de las notificaciones de tiempo real que más adelante será descrito.

Es muy importante dejar claro que la aplicación tiene procesos que se ejecutan a nivel de Java y otros a nivel de *JavaScript*. Estos últimos tienen el

objetivo de mantener una gestión en la interacción y dinamismo con el programador, y que no tiene un impacto directo que pueda afectar en la funcionalidad de la aplicación. Por esta razón la validación de las sesiones es controlada en primera instancia por un componente *JavaScript*, pero si por algún motivo éste falla, no habrá ningún problema ya que en el modelo también se tiene un control sobre la validez de la sesión. El diagrama 20 muestra la manera como se determina la validez de la sesión:

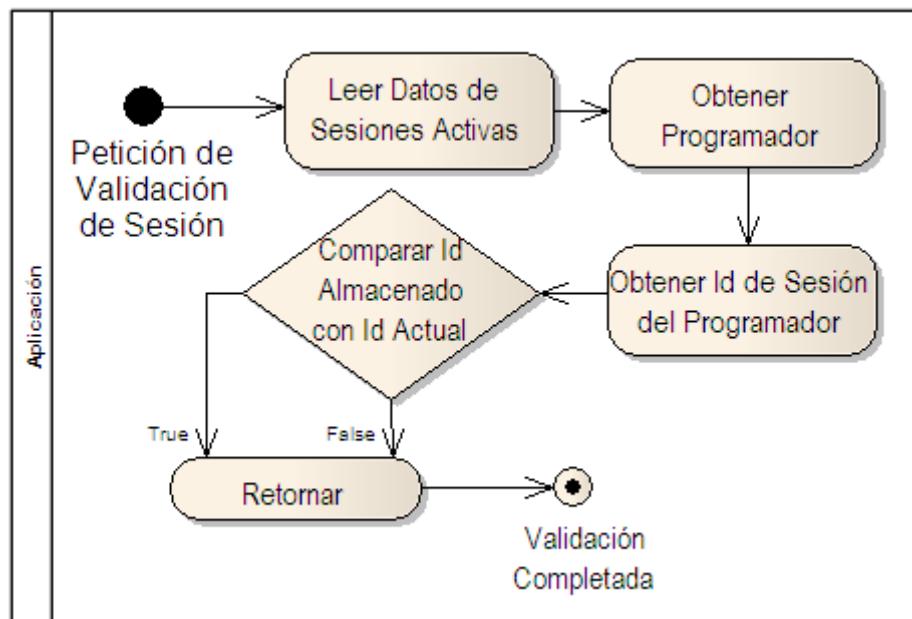


Diagrama 20. Descripción del Proceso Validar Sesión

Especificación

- Este proceso inicia en cualquier acción que el programador desea ejecutar en la aplicación. Esto debe ser así, ya que en cualquier momento su sesión puede ser inválida. Como se había mencionado anteriormente, el objetivo de este proceso es mantener una sesión activa por programador en un momento dado. Con todo lo dicho, si un programador se encuentra usando la aplicación y de repente le sale un mensaje diciéndole: "le han invalidado su sesión", y seguido a esto es redireccionado a la página *index*, quiere decir que otra persona ha ingresado a la aplicación con la misma cuenta.
- Leer los datos de sesiones activas es un paso que depende desde dónde se esté desarrollando la validación, es decir si la validación es realizada desde código *JavaScript* o desde un *JSP*. En el proceso de

abrir sesión se pudo observar que se toma el identificador de la sesión y se almacena en el *sesión* del registro del programador en *users.xml* y una vez hecho esto también se busca el programador en el archivo plano *sessionActives.txt* y se modifica el identificador de sesión. El archivo plano es necesario porque por seguridad era necesario aislar la persistencia de los componentes *JavaScript* que requieren alcanzar un dato de ésta. Dicho esto, la lectura o consulta de la id de sesión puede hacerse en el archivo plano, si *Socket.io* necesita validar la sesión y también se puede realizar consultando en la persistencia cuando el proceso lo requiera el modelo.

3. Una vez leído la correspondiente fuente de datos, se itera sobre sus registros en busca del programador.
4. Cuando el programador es encontrado se toma el valor del identificador de sesión que esté almacenado para él.
5. Ahora con el dato consultado y con el identificador que llega como parámetro que corresponde a la id de sesión recibida cuando se inició sesión, se comparan.
6. Ya sea que la comparación dé como resultado true o false, dicho valor será retornado a donde fue solicitado.

9.6.4. Restablecer y Cambiar Contraseña

La aplicación no tiene establecido ningún criterio de claves seguras, de manera que el visitante al registrarse o al cambiar la contraseña, tiene libertad de usar la combinación de caracteres que desee. Por seguridad en la persistencia no se almacena la clave real del programador, permitiendo tener un grado de seguridad, ya que la clave es codificada a través de un algoritmo y se almacena encriptada, y cuando esta es requerida se desencripta. Ahora, es probable que el programador olvide su contraseña o quiera cambiarla, en consecuencia la aplicación deberá ofrecerle dicha funcionalidad. A continuación se presenta la descripción de cómo se lleva a cabo dichas operaciones en la aplicación.

9.6.4.1. Restablecer Contraseña

Cuando el programador olvida su contraseña por la circunstancia que sea, se debe proporcionar el medio para que él la recupere, tomando importancia

de nuevo el correo electrónico que el programador registró en la aplicación. El diagrama 21 describe el proceso de restablecimiento de la contraseña.

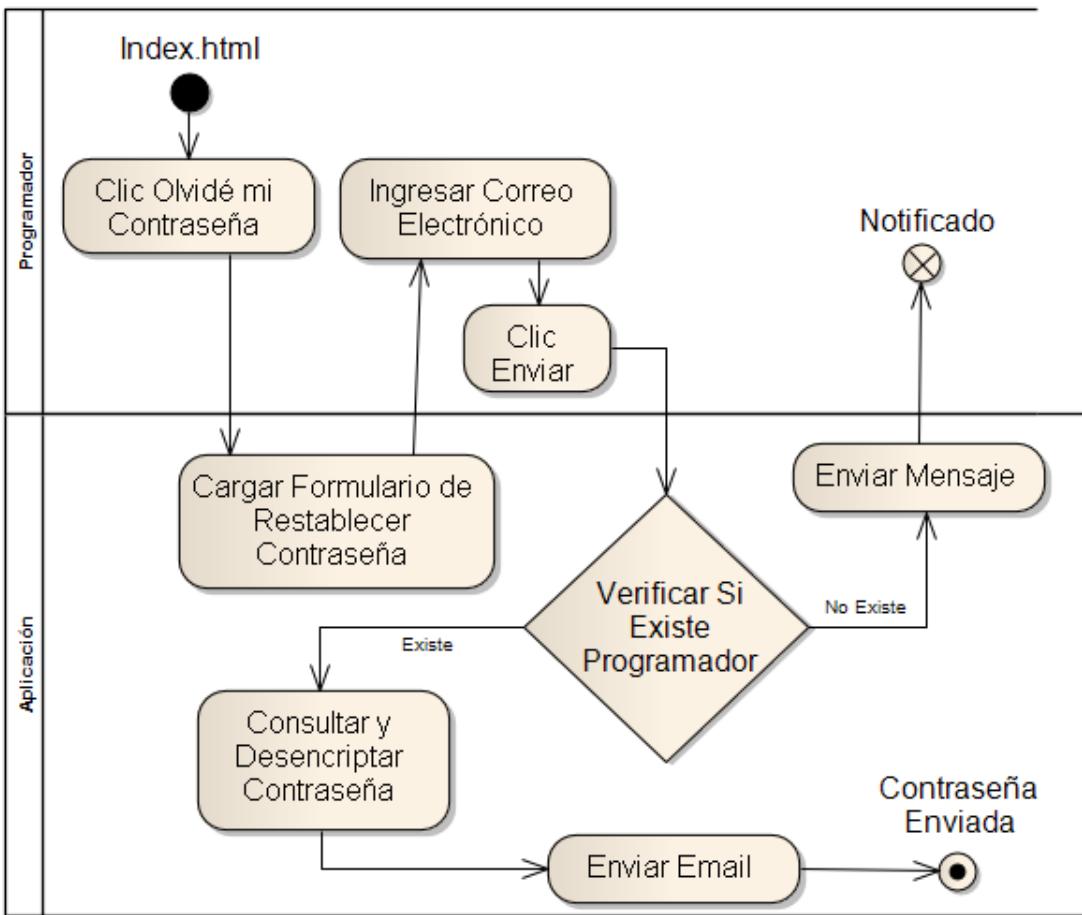


Diagrama 21. Descripción del Proceso Restablecer Contraseña

Especificación

1. El programador se encuentra en la página *index* de la aplicación, y ha olvidado su contraseña, por lo tanto no puede ingresar.
2. El programador pulsa clic en el enlace de ¿Olvidó Contraseña?
3. La aplicación cargará el formulario de restablecer contraseña, que incluye un campo para digitar el correo electrónico que el programador registró en la aplicación.
4. El programador ingresa el correo electrónico.
5. El programador pulsa clic en enviar.
6. La aplicación captura el correo enviado desde la solicitud, y procede a consultar en la persistencia si existe un programador registrado con

dicho correo electrónico. Si en el contexto de la aplicación no existe un programador registrado con dicho correo, se procede a enviar el mensaje de lo sucedido al programador y allí terminará el flujo.

7. Si el programador existe, se consulta la contraseña almacenada y se procede a desencriptarla.
8. Por último se envía desde la aplicación un email a dicho correo con la contraseña ya desencriptada.

9.6.4.2. Cambiar Contraseña

Se cual se la razón el programador debe tener la posibilidad de cambiar su contraseña, por tal motivo es necesario, ofrecerle un medio para que lleve a cabo esta necesidad. Es necesario tener en cuenta que para efectuar dicha operación el programador debe ubicarse en la vista del perfil dispuesta en el *index* del programador y además se debe proporcionar por seguridad la contraseña anterior. El diagrama 22 describe dicha operación:

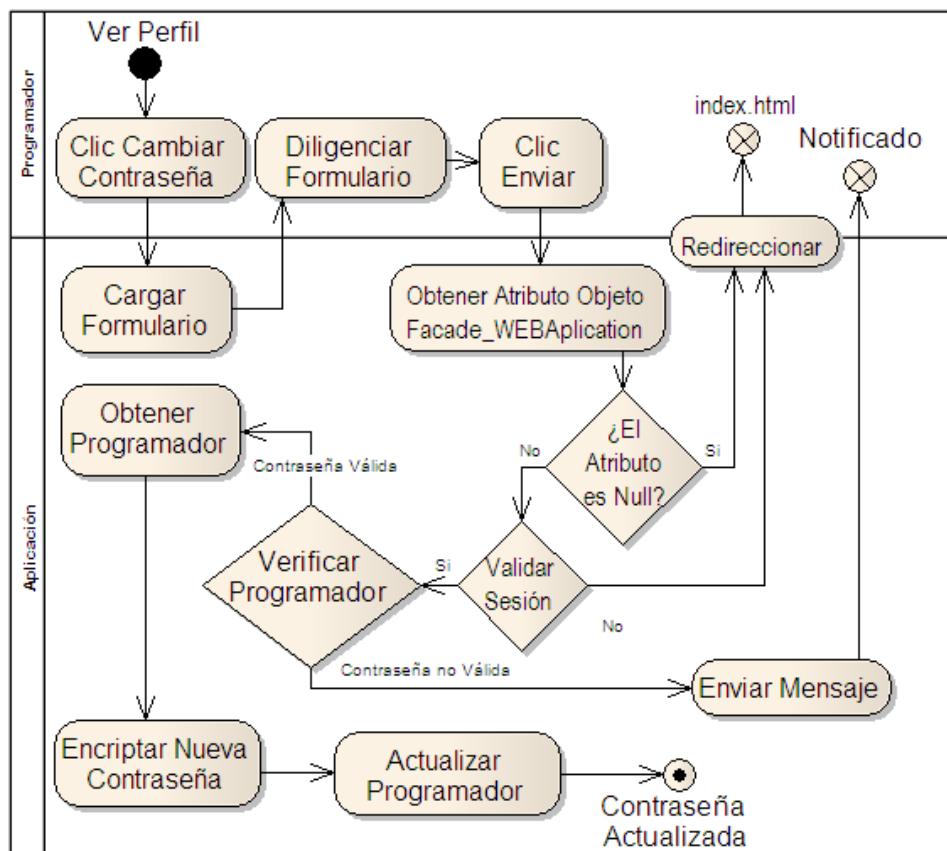


Diagrama 22. Descripción del Proceso Cambiar Contraseña

Especificación

1. Cambiar Contraseña es una operación que se puede llevar a cabo solo desde la aplicación, exactamente en la vista de ver perfil del programador, donde se muestran sus datos, además de la imagen de avatar. Desde allí el programador podrá cambiar su contraseña.
2. El programador pulsa el botón de cambio de contraseña.
3. La aplicación cargará el formulario que contienen los campos: contraseña anterior, y dos campos para ingresar la nueva contraseña.
4. El programador diligencia los campos respectivos.
5. El programador pulsa clic en enviar.
6. La aplicación obtiene el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* ([paso 11 del proceso login apartado 9.6.3.1](#)).
7. Verifica que este objeto facade no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
8. Luego de esto, se pregunta por el estado de la sesión, ya que si esta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado [9.6.3.4](#).
9. Después de dichas validaciones, la aplicación toma a contraseña anterior suministrada y procede a verificar el programador en la persistencia, es decir que la contraseña corresponda al programador que hace la solicitud. Si la contraseña no corresponde, se debe enviar un mensaje al programador y allí terminará el flujo.
10. Si la contraseña anterior es válida, se obtiene al programador.
11. La aplicación procede a encriptar la nueva contraseña diligenciada.
12. Por último se actualiza la contraseña del programador y se modifica el dato en la persistencia.

9.6.5. Gestionar Usuarios

La Gestión de Usuario o Programadores, es una actividad exclusiva del administrador de la aplicación. El administrador tiene el control sobre los

programadores y sus proyectos, además de los usuarios temporales. La gestión de usuario o programadores básicamente consiste en consultarlos, filtrarlos por último acceso en la aplicación y eliminarlos.

9.6.5.1. Mostrar Programadores y Usuarios Temporales

Esta operación consiste en una consulta en los respectivos archivos xml de la persistencia, ya sea de los usuarios temporales que se han registrado y aún no se han activado, o los programadores activos de la aplicación. El proceso de consultar ya sea usuarios temporales como programadores activos es similar, con la única diferencia de que consultan en archivos diferentes. El diagrama 23 describe el proceso general de dichas consultas:

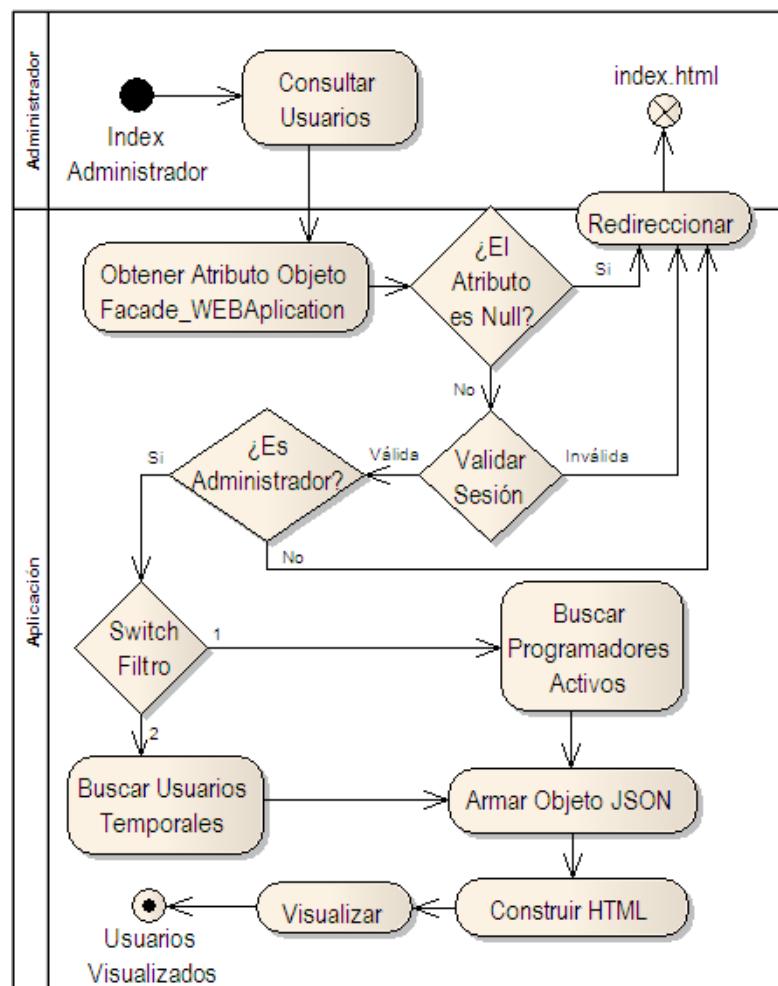


Diagrama 23. Descripción del Proceso Buscar Usuarios

Especificación

1. El administrador de la aplicación se encuentra en su *index* después de haber iniciado sesión, y tiene a disposición buscar los usuarios temporales o Programadores activos. Para ambos casos el proceso es similar a excepción de que la consulta se efectúa en archivos diferentes.
2. El administrador inicia la consulta.
3. La aplicación obtiene el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* ([paso 11 del proceso login apartado 9.6.3.1](#)).
4. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
5. Luego de esto, se pregunta por el estado de la sesión, ya que si esta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado [9.6.3.4](#).
6. Se verifica que el usuario sea el administrador ya que este es quién puede gestionar los usuarios. Si no es redirecciona y termina el flujo.
7. La aplicación filtra la petición con un *switch* a través de un parámetro que es enviado desde el enlace. Dicho parámetro indica 1 si desea consultar los programadores activos y 2 si desea consultar los usuarios temporales.
8. Independiente del filtro, la aplicación define qué archivo *xml* de la persistencia consultar, y almacena el resultado de dicha consulta en un objeto *Document*, producto de la lectura del archivo *xml*.
9. Una vez se obtiene el resultado, se construye un objeto JSON con dicho objeto *Document*, que embebe los datos de los usuarios o programadores consultados. En dicho objeto JSON se crea un *array* llamado *programers*, donde cada elemento tiene valores como el correo electrónico, el nombre, una descripción, una imagen de avatar, la fecha en que fue hecho miembro de la aplicación, y por último la fecha del último ingreso a la aplicación. Estos datos son utilizados, cuando se selecciona un usuario o programador y se muestre el detalle del mismo, que consiste básicamente de un espacio dispuesto para ver la información de dicho elemento seleccionado.

10. Una vez construido el objeto JSON, se itera en él y se construye una tabla HTML, calculando la cantidad de celdas que alcance contener el ancho dispuesto para dicha visualización.
11. Por último se obtiene el HTML generado y se coloca en el elemento dispuesto para la visualización.

9.6.5.2. Filtrar Programadores por Última Fecha de Ingreso

Este proceso consiste en ofrecerle al administrador un medio para conocer los usuarios que no han ingresado en la aplicación hasta una fecha determinada en un calendario. Esto le permitirá al administrador determinar qué usuarios llevan un alto período de inactividad, y así tener un criterio de decisión de eliminación del programador según él lo crea. Básicamente el proceso consiste en determinar una fecha, y a continuación realizar una búsqueda de aquellos programadores que hasta dicha fecha no han ingresado en la aplicación. El diagrama 24 describe el proceso:

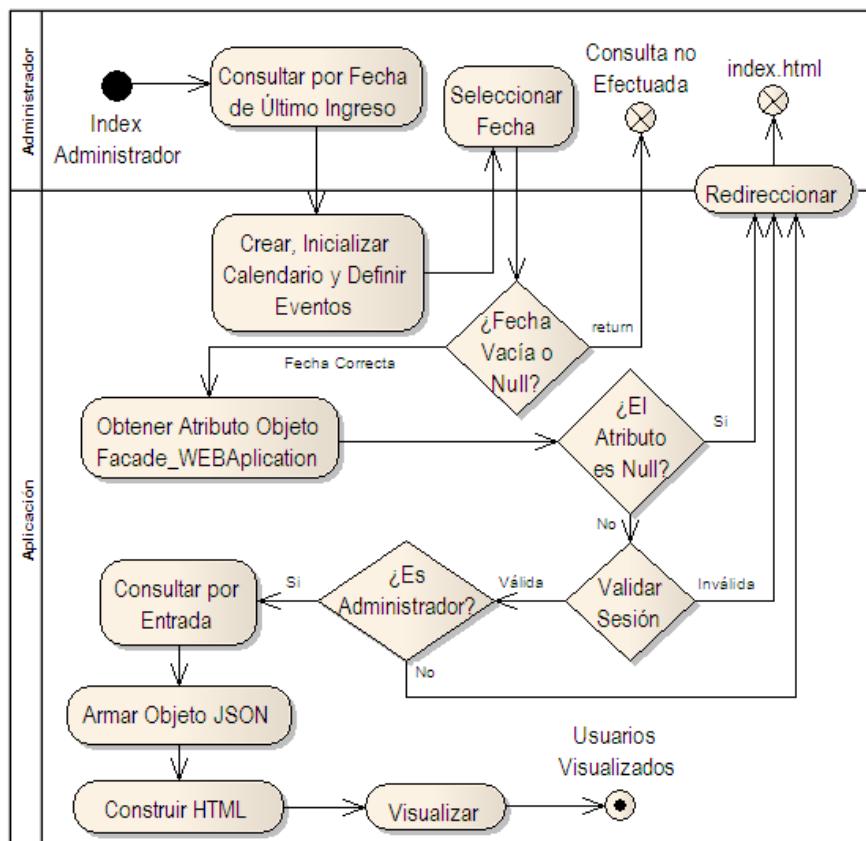


Diagrama 24. Descripción del Proceso Consultar por Última Entrada

Especificación

1. El administrador de la aplicación se encuentra en su *index* después de haber iniciado sesión.
2. El administrador tiene a disposición consultar usuarios por último ingreso a la aplicación.
3. La aplicación prepara el correspondiente html y calendario (componente *JQuery*) inicializándolo y agregándole ciertas operaciones con respecto a ciertos eventos, de manera que cuando se le seleccione una fecha ejecute cierta acción, que en este caso es consultar usuarios con fecha de ingreso anteriores o igual a la seleccionada.
4. El administrador selecciona una fecha.
5. La aplicación verifica que dicha fecha seleccionada no esté vacía o sea nula. Si lo es, terminará el flujo y no consultará.
6. Si la fecha es válida, la aplicación obtiene el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* ([paso 11 del proceso login apartado 9.6.3.1](#)).
7. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
8. Luego de esto, se pregunta por el estado de la sesión, ya que si esta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado [9.6.3.4](#).
9. Se verifica que el usuario sea el administrador ya que este es quién puede llevar a cabo esta operación. Si no es redirecciona y termina el flujo.
10. Ahora si el usuario es el administrador, procede a realizar la consulta en *users.xml* de dichos usuarios que tengan registro de último ingreso a la aplicación en una fecha igual o anterior a la que el administrador seleccionó en el calendario. El resultado de la consulta es almacenado en un objeto *Document*, producto de la lectura del archivo xml.

11. Una vez se obtiene el resultado, se construye un objeto JSON con dicho objeto *Document*, que embebe los datos de los usuarios o programadores consultados. En dicho objeto JSON se crea un *array* llamado *programers*, donde cada elemento tiene valores como el correo electrónico, el nombre, una descripción, una imagen de avatar, la fecha en que fue hecho miembro de la aplicación, y por último la fecha del último ingreso a la aplicación. Estos datos son utilizados, cuando se selecciona un usuario o programador y se muestre el detalle del mismo, que consiste básicamente de un espacio dispuesto para ver la información de dicho elemento seleccionado.
12. Una vez construido el objeto JSON, se itera en él y se construye una tabla HTML, calculando la cantidad de celdas que alcance contener el ancho dispuesto para dicha visualización.
13. Por último se obtiene el HTML generado y se coloca en el elemento dispuesto para la visualización.

9.6.5.3. Eliminar Usuarios

Uno de los roles importantes del administrador es eliminar programadores o usuarios temporales. El proceso de eliminación es similar para los programadores y usuarios temporales hasta cierto punto, el cual es el borrado de su registro en los respectivos archivos *xml* de persistencia. Después del borrado del registro, en el caso de un programador, se debe además borrar el registro y directorios de los proyectos

9.6.6. Notificaciones en Tiempo Real

Las notificaciones en tiempo real es la funcionalidad que logra que un grupo de programadores interactúen con ella concurrentemente, realizando cambios que todos verán en el mismo instante. Esta función junto con la edición concurrente, brindan un ambiente de colaboración en tiempo real, donde un grupo de programadores que comparten un proyecto, podrán trabajar sobre éste efectuando cambios y ediciones, todos al tiempo, y la versión de dicho proyecto siempre se mantendrá actualizada para todos los que lo comparten.

Las notificaciones en tiempo real consisten básicamente en mantener al tanto a los programadores que comparten un proyecto, de todos los cambios importantes que en un momento dado se estén presentando en el proyecto.

Para implementar dicha funcionalidad, fue necesario integrar a la aplicación el componente Javascript llamado Socket.io, que es el encargado de mantener ciertos nodos o clientes comunicados entre sí, a través de un canal establecido previamente, por el cual “viajará” un mensaje que haga referencia a una acción realizada por uno de los programadores que comparte el proyecto y que se encuentra suscrito a dicho canal. Dependiendo del evento, se enviará un tipo de mensaje a quienes estén suscritos a dicho canal.

Con el fin de proporcionar conectividad en tiempo real en todos los navegadores, Socket.IO elimina todas las diferencias que existen entre los mecanismos de transporte de cada uno de ellos y los unifica en una sola API, seleccionando en tiempo de ejecución, el método de transporte más adecuado, es decir, Socket.io dependiendo de la disponibilidad de mecanismos de transporte que tenga el navegador que se está usando, selecciona el más adecuado. Socket.io puede seleccionar de los siguientes métodos (ordenados de mejor a peor) de pendiendo de la disponibilidad del navegador⁸⁵.

- *WebSocket*: Protocolo que opera muy bien sobre una infraestructura web, permitiendo una conexión full-dúplex con un host remoto.
- *Adobe® Flash® Socket*: API de Action Script que permite a una aplicación cliente conectarse a un servidor, pero no detecta conexiones entrantes.
- *AJAX long polling*: Consiste mantener a través de AJAX una conexión abierta para que los tiempos de respuesta sean más rápidos y sensibles a responder.
- *AJAX multipart streaming*
- *Forever Iframe*
- *JSONP Polling*

9.6.6.1. Proceso

Socket.io es el componente que se ha integrado a la aplicación para llevar a cabo la conexión full-dúplex entre el cliente y el servidor, donde se transmitirán mensajes dependiendo de la acción efectuada sobre el

proyecto. El diagrama 25 muestra el despliegue del componente mencionado tanto en el servidor como en el lado del cliente:

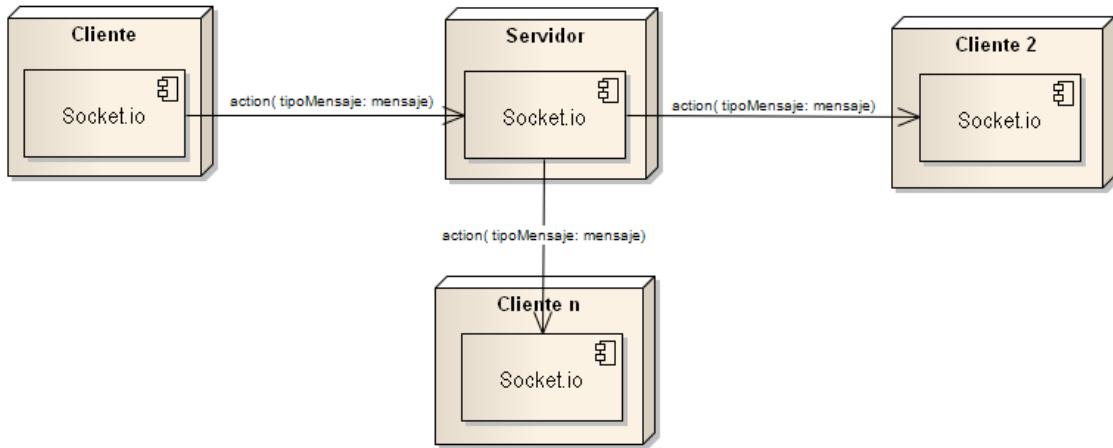


Diagrama 25. Descripción de Notificaciones En Tiempo Real

Especificación

La figura anterior muestra que Socket.io está instanciado tanto en el servidor como en los diferentes clientes, permitiendo entre estas instancias del componente establecer canales de conexión para emitir mensajes producidos por acciones en la aplicación. A continuación se describe el proceso que se ejecuta para llevar a cabo las notificaciones:

1. Cuando en el Cliente se produce un evento, éste procede a notificar al servidor de la acción a través de la emisión de un mensaje, qué básicamente está compuesto por un tipo de operación, un tipo de mensaje o evento y los datos necesarios para que al recibirla sea ejecutado.
2. El servidor una vez notificado se encarga de inmediato a verificar qué clientes están suscritos al canal y les notifica (incluyendo a quién emitió el evento).
3. Cuando el cliente es notificado procede a ejecutar el evento usando el objeto que viene como parámetro, y de esta manera todos los clientes que estén suscritos a un mismo canal ejecutarán las mismas acciones en el orden que son emitidas y recibidas.

Es importante mencionar que el cliente que emite el mensaje que produjo su evento, ejecutará la acción solo cuando la notificación llegue a él, al igual como lo harán los demás programadores que comparten. Esto asegura que en todo tiempo se mantenga un mismo estado para todos los programadores que en ese instante están trabajando sobre el proyecto.

9.6.6.2. Estructura del Mensaje

El contenido del mensaje puede ser una variable, un objeto *JSON* o una cadena formateada en *xml*, que al ser recibido tendrá el significado que el desarrollador quiso darle para su implementación. En este caso el mensaje fue formateado en *xml* y tiene la siguiente estructura como lo muestra la figura 71:

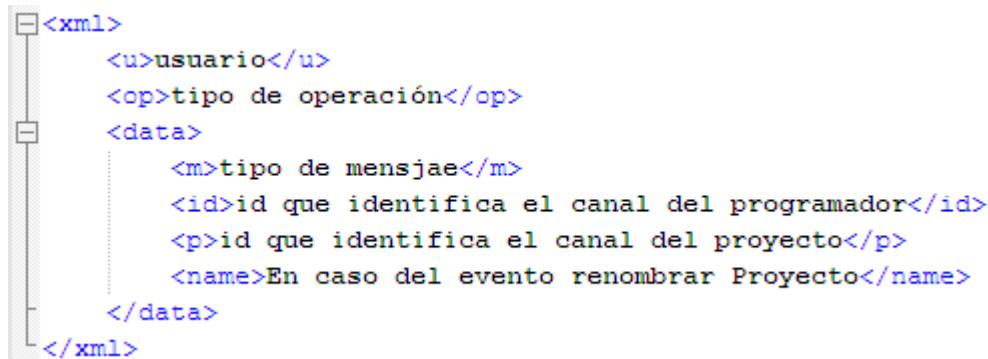


Figura 71. Estructura XML del Mensaje de Notificaciones

La anterior figura muestra las distintas etiquetas contenidas en el *xml* que representa el mensaje y a continuación se describen:

Etiqueta	Descripción
<u>	Etiqueta que identifica el programador que emite el mensaje.
<op>	Etiqueta que indica el tipo de operación que se ejecutará. El tipo de operación es una clasificación de eventos
<data>	Etiqueta que contiene los datos relevantes del evento.
<m>	Etiqueta que describe el tipo de mensaje que se debe ejecutar.

Las anteriores etiquetas descritas son las que todo mensaje debe tener, las demás son dependientes del evento que se desea ejecutar, es decir en algunos casos aparecerán unas y otras no, y también tendrán su significado dependiendo de la acción requerida.

9.6.6.3. Tipo de Operaciones

Las operaciones permiten clasificar los tipos de mensajes que se transmitirán por el canal, por ejemplo si el evento en la aplicación fue agregar una clase, la emisión del mensaje por el canal debe indicar que es una operación de tipo *rui* (*Refresh User Interface*) y que el tipo de mensaje es *ac* (Agregar Clase). La tabla 43 describe los diferentes tipos de operación que la aplicación usa para clasificar las acciones:

Id	Nombre	Descripción
ml	Modificar Lienzo	Indica que una operación que produzca cambios sobre el lienzo se ha presentado.
rui	<i>Refresh User Interface</i>	Indica que la interface gráfica será actualizada para pintar los cambios producidos.
chat	Chat	Mensaje que indica que existe un nuevo texto que se enviará a través del chat de un proyecto compartido.
cuser	Usuario Conectado	Notifica que un nuevo programador ha abierto un proyecto. Esto es útil cuando este proyecto es compartido ya que esto pone al tanto a los programadores que estén trabajando en ese instancia sobre el proyecto.

Tabla 43. Tipo de Operaciones en las Notificaciones

9.6.6.4. Tipos de Mensajes

Se ha mencionado que los eventos que se presentan en la aplicación emiten mensajes a través de un canal, que definen una acción particular para que los suscritos a éste lo ejecuten, todo esto gracias al componente Socket.IO.

La tabla 44 se describe los tipos de mensajes que pueden ser transmitidos por estos canales:

Id	Nombre	Descripción
rc	Renombrar Clase	Mensaje que indica que se renombrará una clase.
rcg	Renombrar Clase GUI	Este tipo de mensaje indica que se renombrará una clase que es de tipo Interface gráfica.
rf	Renombrar Fichero	Define la acción de renombrar ya sea archivos de texto plano, imágenes, o cualquier otro archivo que la aplicación permita.
rl	Renombrar Librería	En un proyecto se podrán integrar librerías externas, por lo tanto este mensaje indica que se renombrará una de estas.
rp	Renombrar Paquete	Mensaje que expresa que un paquete será renombrado.
ac	Agregar Clase	Indica la acción de crear una clase en el proyecto.
acg	Agregar Clase GUI	Indica la acción de crear una clase especial, que
af	Agregar Fichero	Indica que se creará un archivo de texto plano
ap	Agregar Paquete	Este mensaje notifica que se creará un nuevo paquete en el proyecto.
ec	Eliminar Clase	Indica que una clase será eliminada.
ecg	Eliminar Clase GUI	Indica que una clase de tipo interface gráfica se eliminará.
ef	Eliminar Fichero	Mensaje que emitirá la acción de eliminar un fichero (archivo de texto plano, imagen, entre otros)
ep	Eliminar Paquete	Notifica que un paquete del proyecto será eliminado.
el	Eliminar Librería	Notifica que una librería que se integrado al proyecto se eliminará.
cc	Cargar Clase	Mensaje que indica una clase externa será cargada al proyecto.
cf	Cargar Fichero	Indica que un fichero será cargado al proyecto.
cl	Cargar Librería	Mensaje que emite la acción de que una librería será integrada al proyecto.
ccpc	Cortar Copiar Pegar Clase	Es un mensaje que indica se realizará una operación de cortapapeles sobre una clase del proyecto.
ccpg	Cortar Copiar Pegar GUI	Notifica que se realizará una operación de cortapapeles sobre una clase de tipo interface gráfica que está en un proyecto.
ccpf	Cortar Copiar Pegar Fichero	Operación de cortapapeles para un fichero en el proyecto.
ccpl	Cortar Copiar Pegar Librería	Notificación de una operación de cortapapeles sobre una librería de un proyecto.

mcp	Modifica Clase Principal	Este mensaje indica que la clase principal del proyecto ha sido modificada.
sp	Share Project	Mensaje que emite la acción de que el proyecto se ha compartido.
ac	Agregar Componente	Mensaje que indica que un componente gráfico (botón, etiqueta- <i>label</i> , área de texto o campo de texto)que agregará al lienzo que representa un panel de un <i>JFrame</i>
mc	Mover Componente	Indica que el componente cambiará de posición (X,Y) con respecto al lienzo.
mct	Mover Componente Teclado	Indica que el componente cambiará de posición, pero este se activa por las teclas cursoras.
rl	Resize Lienzo	Notifica que el lienzo ha cambiado de tamaño.
rc	Resize Componente	Notifica que un componente gráfico cambiará de tamaño.
apc	Actualizar Propiedades del Componente	Mensaje que indica que las propiedades de un Componente se actualizarán, de manera que esto se debe ver reflejado tanto en el lienzo como en el código fuente generado.
anc	Actualizar Nombre Componente	Mensaje particular que modifica el nombre de la variable del componente.
avc	Actualizar Valor Componente	Mensaje que indica que el valor de la etiqueta del componente será modificado.
ec	Eliminar Componente	Indica que un componente seleccionado del lienzo será eliminado.
cs	Cerrar Sesión	Este es un mensaje que se emite cuando se determina que es necesario cerrar la sesión, aclarando que esta acción solo involucra un redireccionamiento y un mensaje ya que el cierre de la sesión fue realizado por la capa de lógica

Tabla 44. Tipo de Mensajes en las Notificaciones

9.6.6.5. Abstracción del Funcionamiento de Socket.IO

En este punto ya se ha presentado el componente Socket.io, el proceso general de la interacción cliente-servidor a través de este componente, se han descrito la estructura del mensaje que usa la aplicación, el tipo de operaciones y mensajes que incluye.

El diagrama 26 presenta una abstracción del funcionamiento de los componentes instanciados tanto en el servidor como en los nodos clientes que se encuentren conectados. Este caso particular incluye dos programadores (xx@xx.com y yy@yy.com) que se han logueado, tienen sesión activa y han abierto el proyecto compartido llamado “prueba” que es de propiedad de xx@xx.com.

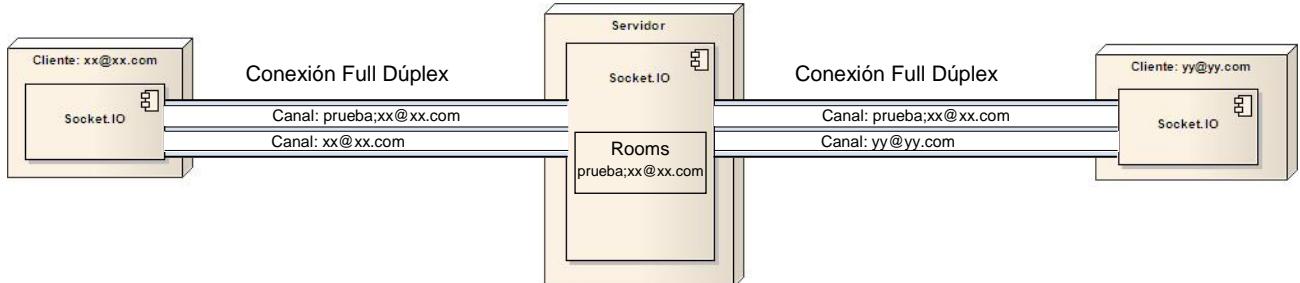


Diagrama 26. Abstracción de Funcionamiento de Socket.IO

En el anterior diagrama se puede observar que dos clientes o programadores están interactuando con la aplicación, pero en este caso solo se tendrá en cuenta la interacción con los respectivos componentes Socket.IO.

Especificación

1. Tanto el programador xx@xx.com como yy@yy.com inician sesión en la aplicación estableciendo para cada uno una conexión full dúplex a través del puerto que se configuró para el Socket.IO instanciado en el servidor.
2. En la figura se puede percibir que cada programador tiene un canal que está identificado con el respectivo correo electrónico. Este canal cumple una función de control de sesión a nivel de capa de vista, ya que la capa de lógica se encarga de la gestión de las sesiones activas.
 - 2.1. Antes de que este canal sea creado se verifica que ya no exista un canal instanciado para este programador ya que estos están identificados con el correo electrónico del programador
 - 2.2. En caso de que un programador inicie sesión con el usuario xx@xx.com y ya exista un canal identificado con el mismo correo, se emitirá un mensaje de tipo cierre de sesión para aquella sesión que en ese momento está activa en la aplicación. Esta acción es ejecutada después que haya terminado el proceso de inicio de sesión

en la capa de lógica como se describe en los puntos del apartado [9.6.3. Login y Gestión de Sesión](#)

- 2.3. Independientemente si existe o no un canal para ese programador, se le asigna el canal al programador que ha iniciado sesión. Se debe tener en cuenta que si existe un canal ya establecido, el que actualmente lo tiene le será quitado y asignado a la nueva sesión.
3. Cada vez que un proyecto es abierto (apartado [9.6.7.2.](#)) el programador se suscribirá a un canal nuevo que representa el proyecto, de manera que toda acción que se presente sobre este proyecto será notificado sobre este canal. Pero en este punto sucede algo importante:
 - 3.1. Si el proyecto es compartido y el programador al que le fue dado permisos lo abre también, entra un concepto llamado *rooms* que usa Socket.IO para definir salas compartidas, es decir varios programadores tienen una instancia del mismo canal de manera que todo lo que se emita por éste será recibido por cada uno de los programadores que se encuentren en dicha sala. Por eso en la figura anterior se puede percibir que los dos programadores tiene una instancia del mismo canal identificado como '*prueba;xx@xx.com*', que en este caso representa un proyecto llamado '*prueba*' y que es de propiedad de *xx@xx.com*.
 - 3.2. Entonces cuando se abre un proyecto se verifica si ya existe un *room* que lo represente. Si ya existe simplemente se instancia el canal para este programador, pero si no existe, se procede agregar una *room* y crear el respectivo canal.
 - 3.3. Cada vez que un evento sobre un proyecto que tiene un *room* asociado emita un mensaje, Socket.IO se encarga de emitir este mensaje a quienes pertenezcan a este *room*, y al recibirlo cada cliente de esta sala, se enterarán de la presencia de una nueva acción y procederán a ejecutarla.

9.6.7. Gestión de Proyectos

La gestión de proyectos es toda aquella operación que un programador pueda realizar sobre proyectos. Algunas operaciones de gestión de proyectos son única y exclusivamente del propietario del proyecto. La gestión de proyecto está estrechamente ligada a la gestión de notificaciones, ya que es necesario que gran parte de las operaciones sobre un proyecto

realice notificaciones. A continuación se presenta los procesos más importantes con respecto a la gestión de proyecto:

9.6.7.1. Crear Proyecto

Se puede decir que cada programador tiene un espacio de trabajo en el servidor, que corresponde al subdirectorio *projects* dentro de la carpeta creada para él al ser activado ([paso 9 de 9.6.1.2 Activar Programador](#)). En dicho subdirectorío se crearán tantos subcarpetas como proyectos haya creado en la aplicación. Un programador podrá crear un proyecto ya sea desde su respectivo *index*, o cuando se encuentre dentro de la IDE, y dependiendo de dónde se haya originado la solicitud de creación de proyecto, se efectuará una determinado proceso adicional una vez el proyecto haya sido creado. El diagrama 27 describirá el proceso:

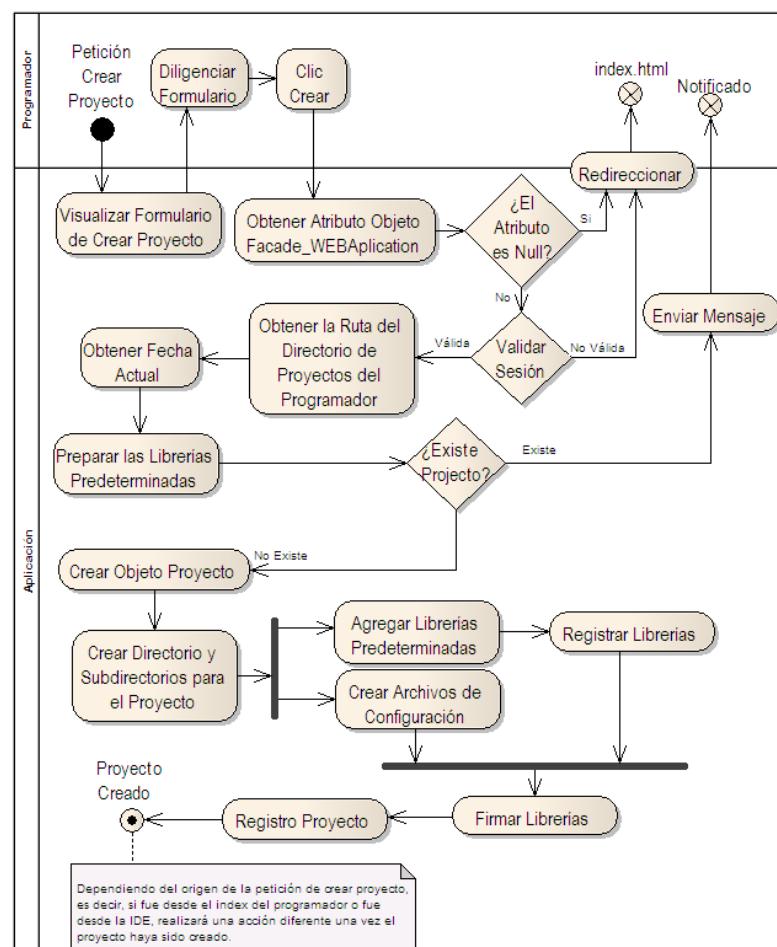


Diagrama 27. Descripción del Proceso Crear Proyecto.

Especificación

1. El programador realiza la petición de crear un proyecto ya sea desde su *index* o desde la IDE.
2. La aplicación cargará un pequeño formulario de creación de proyecto.
3. El programador diligencia el formulario.
4. Una vez diligenciado el programador dará clic en crear.
5. A partir de aquí la aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* ([paso 11 del proceso login apartado 9.6.3.1](#)).
6. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
7. Luego de esto, se pregunta por el estado de la sesión, ya que si esta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado [9.6.3.4](#).
8. Si la sesión es válida se obtiene la ruta de la carpeta de proyectos del programador.
9. Se obtiene la fecha actual.
10. Se preparan las librerías que por defecto tendrá cada proyecto, y se construye un *array* que contiene las rutas de todas aquellas librerías que dispone la aplicación para llevar a cabo cierta funcionalidad que desea que cada proyecto pueda tener a disposición.
11. Ahora con la ruta, la fecha, las librerías y el nombre del proyecto que el programador definió, inicia el proceso real de crear un proyecto con la validación de la existencia de un proyecto con el mismo nombre en el espacio de trabajo del programador. Si existe alguno, se debe notificar dicha novedad al programador, y por lo tanto el proyecto no podrá ser creado y allí terminará el flujo.
12. Si no existe un proyecto con el mismo nombre, se creará un objeto *Project* el cual se encarga de gestionar todas las operaciones que puedan efectuarse sobre un proyecto.
13. Es a través de este objeto que se crea la carpeta para el nuevo proyecto, y una serie de subdirectorios que contiene todo proyecto en

- la aplicación, como se describe en el apartado [9.3.3.](#) en sus descripciones 3, 4, 5, 6, 13, 14, 15 y 16.
14. Una vez se crean la carpeta y sus subdirectorios, se procede a crear los archivos xml de configuración para el proyecto, los cuales quedarán almacenados en el subdirectorio config, y se describen en los apartados [9.3.4](#), [9.3.5](#), [9.3.6](#), [9.3.7](#) y [9.3.8](#). Además de los archivos de configuración se agregan las librerías predeterminadas al subdirectorio *lib*, lo cual implica copiarlas desde el directorio que dispone la aplicación para ellas, hasta la ruta del subdirectorio *lib* del proyecto.
 15. Luego de agregar las librerías al proyecto, se deben registrar en el archivo *libs.xml* que fue creado junto con los demás archivos de configuración. Esto es necesario para que la aplicación puede conocer con qué librerías cuenta el proyecto.
 16. Terminado los paso 14 y 15, se procede a firmar aquellas librerías agregadas al proyecto. El firmado de librerías es necesario para la ejecución del proyecto, ya que en los [resultados encontrados](#) en el análisis de Java y las diferentes opciones de ejecución, se determinó que el medio para lanzar la ejecución del proyecto era a través de un *applet*, y que para evitar restringir el uso de librerías de terceros o incluso las disponibles en la API de Java, era necesario que la ejecución actuara sobre la máquina virtual del cliente. Para lograr acceder a los recursos del cliente para la ejecución, es necesario firmar las librerías e incluso el archivo JAR ejecutable que contiene el proyecto, para que el programador otorgue los permisos para dicho objetivo. El firmado de un archivo JAR se describe en el paso 4 del apartado [6.8.2.2. ¿Cómo Firmar un Applet?](#)
 17. Una vez se culmine lo anterior, se registra el proyecto en el archivo *projects.xml*, quién es el encargado de servirle a la aplicación como indexador de los diferentes proyectos que tiene el programador. En este momento el proyecto ya ha sido creado y la aplicación ya conoce de dicha novedad.
 18. Por último es necesario mencionar que dependiendo del origen donde se haya realizado la solicitud de crear un proyecto, se efectuará un proceso adicional. En caso de que el origen fuese desde el *index*, al terminar el proceso de la creación del proyecto se actualizará la vista que visualiza los proyectos. Si la solicitud se hizo desde la IDE, al terminar crear proyecto, se ejecuta un proceso que se encarga de

abrirlo en la IDE y esto implica una descripción aparte para lograr entender de qué se trata.

9.6.7.2. Abrir Proyecto

Abrir Proyecto es un proceso que puede ser iniciado una vez termine el proceso de crear un proyecto, esto último siempre y cuando este se haya realizado desde la IDE. Abrir un proyecto también puede ser iniciado cuando el programador desde la IDE quiere abrir un proyecto que no está cargado. En los siguientes diagramas se describen el proceso general de abrir un proyecto, y las acciones adicionales que implican que un programador desde la IDE desea abrir un proyecto:

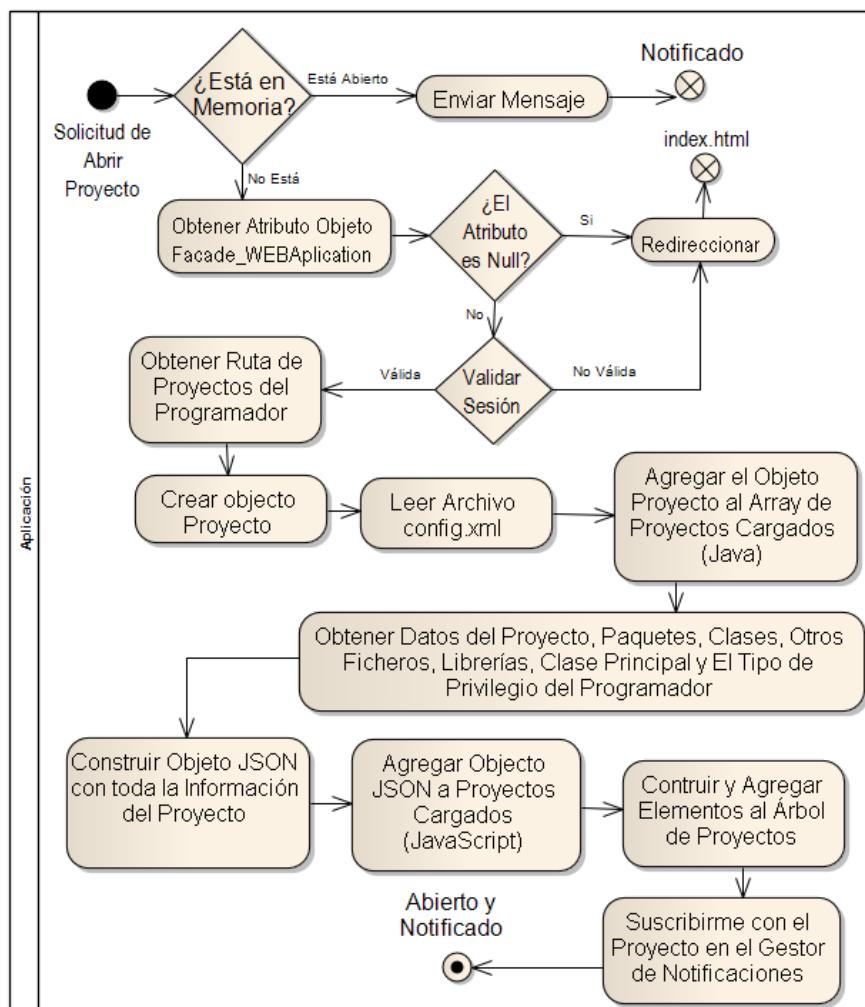


Diagrama 28. Descripción del Proceso de Abrir un Proyecto

Especificación

1. El proceso real que permite abrir un proyecto inicia desde la petición para efectuar esta tarea. Por ejemplo cuando un programador desea crear un proyecto desde la IDE, y este proceso se lleva a cabo, se procede a abrir el proyecto en la IDE, activando un nuevo proceso que cumplirá con esa función.
2. La aplicación mantiene en el *Control_IDE.js*, un array de objetos JSON donde cada uno contienen toda la información de un proyecto, y esto con el objetivo de evitar que intenten abrir un proyecto que ya se encuentra abierto en la IDE, o también para mantener los proyectos abiertos cargado de manera que si la página es recargada pueda ser reconstruido el árbol de proyectos. Entonces en este paso la aplicación verifica si este proyecto ya se encuentra abierto, y si lo está envía un mensaje al programador notificándole lo sucedido y terminará con el flujo.
3. A partir de aquí la aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* ([paso 11 del proceso login apartado 9.6.3.1](#)).
4. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
5. Luego de esto, se pregunta por el estado de la sesión, ya que si esta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado [9.6.3.4](#).
6. A partir del nombre del proyecto, el propietario del mismo, y el usuario de la sesión, los cuales son datos obtenidos de los parámetros enviados en la petición y del objeto *Facade_WEBApplication*, se procede consultando la ruta del subdirectorio de proyectos de la carpeta del usuario de la sesión.
7. La aplicación crea un objeto Proyecto, donde a través de éste se podrá llevar a cabo el proceso.
8. Con el objeto creado se inicia la carga del proyecto enviando como parámetros el nombre, el propietario del proyecto y la ruta de la

- carpeta de proyectos del programador, además del usuario de la sesión, y así continua leyendo el archivo config.xml que contiene la información del proyecto. De este archivo tomará el nombre y el propietario del proyecto y los asignará al objeto.
9. *Control_ProjectsManager* tiene un *array* de objetos *Project*, para mantener un control interno de los diferentes proyectos que han sido abierto y aún permanecen así. Ahora, luego de que se ha obtenido los datos de nombre y propietario del proyecto y se le asignaron al objeto se agrega a dicho *array*.
 10. Ahora es necesario obtener todos los demás datos que tiene un proyecto, por lo tanto se consulta toda la información referente a paquetes, clases, otros ficheros, librerías, clase principal y algo muy importante que es el tipo de privilegio que el programador tiene sobre el proyecto (solo lectura o lectura]/escritura). A pesar de que en el archivo *config.xml* existen más datos importantes del proyectos, estos no son tenido en cuenta en la clase *Project* como atributos ya que varios programadores pueden estar trabajando en el mismo proyecto generando cambios constantes que no podrían ser controlados con facilidad, si estos fueran asignados a la clase. Entonces para obtener los demás datos del proyecto, se deberá realizar consultas a los respectivos archivos xml para que la aplicación se dé por enterada de los distintos cambios que ha sufrido el proyecto a raíz de las actividades que los programadores hayan desarrollado sobre éste.
 11. Con todos los datos obtenidos del proyecto se construye un objeto JSON, en el cual estará embebida toda esa información consultada.
 12. Este objeto JSON será agregado en el array de proyectos que tiene *Control_ProjectsManager*.
 13. Luego de esto se construirán y agregarán los elementos gráficos que corresponde a un proyecto en el componente de árbol de proyectos.
 14. Por último se procede a suscribir en el gestor de notificaciones, el programador a ese proyecto. Esta suscripción es importante para mantener el control en tiempo real de todos los cambios que sufre el proyecto mientras dos o más programadores trabajan en éste, permitiendo así mantener al tanto a cada uno de ellos de las operaciones que todos hacen con respecto a ese proyecto.

En el diagrama 29 se describe el proceso de abrir un proyecto desde la IDE:

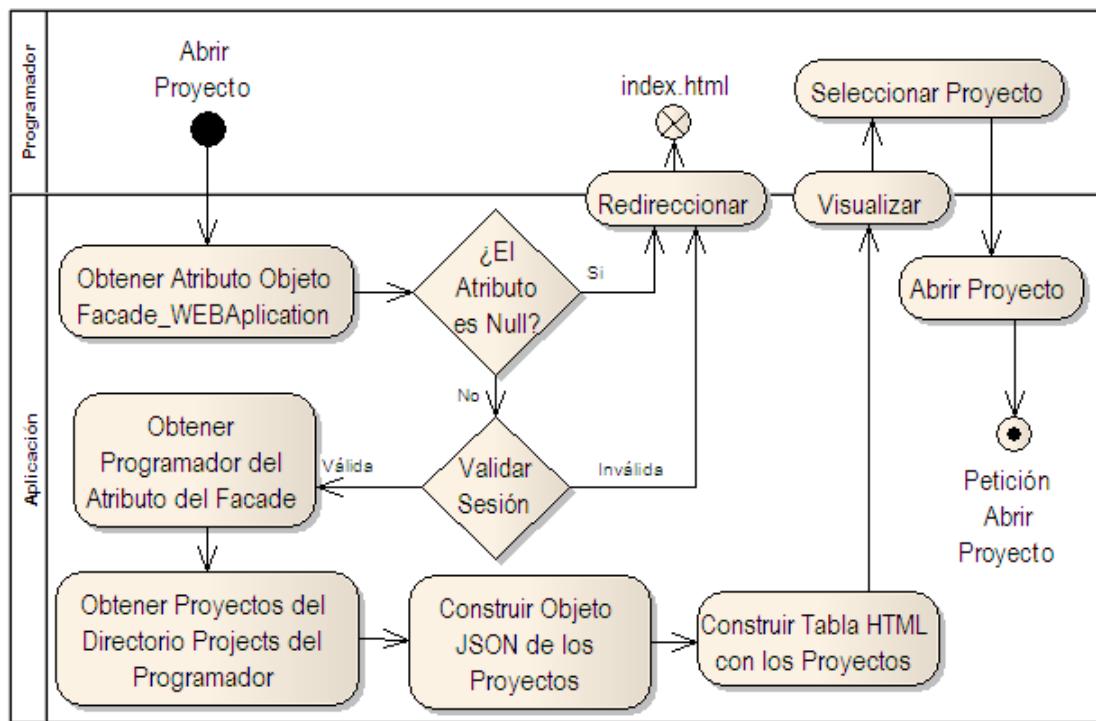


Diagrama 29. Descripción del Proceso Abrir Proyecto desde la IDE

Especificación

1. El programador se encuentra en la IDE y desea abrir un proyecto.
2. La aplicación como en todas las acciones obtiene el objeto *Facade_WEBAplication* que está almacenado como atributo en la sesión del objeto *request*.
3. Si este objeto es nulo se redirecciona al *index.html*.
4. Si dicho objeto no es nulo se procede a verificar la sesión. Si ésta es inválida se debe redireccionar al *index.html*.
5. Si la sesión es válida se obtiene el programador del objeto *Facade*.
6. Ahora se consultan todos los proyectos que se encuentran en el directorio *projects* de la carpeta del programador. Esta consulta retornará un objeto *Document* que embebe todos los proyectos encontrados.
7. Con el objeto que resultó de la consulta, se construye un objeto JSON.
8. Con toda la información almacenada en el objeto JSON se construye una tabla HTML donde se colocarán gráficamente los proyectos del programador.
9. Una vez construida la tabla se procede a visualizarla.

10. El programador escogerá un proyecto y dará clic en abrir.
11. En este punto se lanza la petición de abrir proyecto que se describió anteriormente.

9.6.7.3. Compartir Proyecto

Compartir un proyecto es una funcionalidad que sirve de antesala a la edición concurrente, es decir, para que dos o más programadores puedan trabajar colaborativamente en un proyecto, el propietario tiene que compartirlos y además asignarle privilegios de acceso a cada uno. De esta manera aquellos programadores que les fue asignado el privilegio de escritura son los únicos autorizados por la aplicación para participar en edición concurrente sobre el proyecto, pero aquellos que tienen privilegios de solo lectura, podrán ver los cambios que se producen en el proyecto en tiempo real, más no estará permitido para ellos efectuar ningún tipo de modificación sobre el proyecto. Compartir un proyecto ayuda a fomentar el trabajo en equipo, ya que un grupo de programadores podrán trabajar en un proyecto colaborativamente, manteniendo en todo momento una misma versión para todos, esto evita tener múltiples copias del mismo proyecto que al ser compactadas pueda resultar algo conflictivo.

Para compartir un proyecto es necesario que el propietario del mismo, agregue los programadores con los que desea trabajar y asigne a cada uno, los privilegios que tendrán sobre el proyecto. Existen dos tipos de privilegios que pueden ser asignados a un programador sobre determinado proyecto:

- *Sólo Lectura*: El programador no podrá realizar operaciones que implique modificación alguna en el proyecto. Con este privilegio solo podrá visualizar, compilar, ejecutar y descargar el proyecto.
- *Lectura/Escritura*: Este privilegio permite llevar a cabo todas las operaciones sobre un proyecto, excepto eliminarlo, renombrarlo y acceder a la función compartir proyecto, ya que estas funciones solo las puede realizar el propietario del mismo.

El propietario de un proyecto que quiera compartirlo deberá conocer el correo electrónico del programador al cual le quiere hacer partícipe, y si además desea compartir a varios deberá separarlos con un punto y coma.

Otro aspecto importante sobre esta función es que al compartir un proyecto a cierto programador(es), se mantendrá una única copia del mismo que corresponde al del propietario, pero se le agregará a cada uno de los participantes un nuevo nodo en su respectivo archivo de *projects.xml*, con el fin referenciar el proyecto que fue compartido, y mantener una copia centralizada que sufrirá cambios en cada operación que así lo requiera.

El diagrama 30 describe el proceso que el propietario de un proyecto debe realizar para compartir un proyecto.

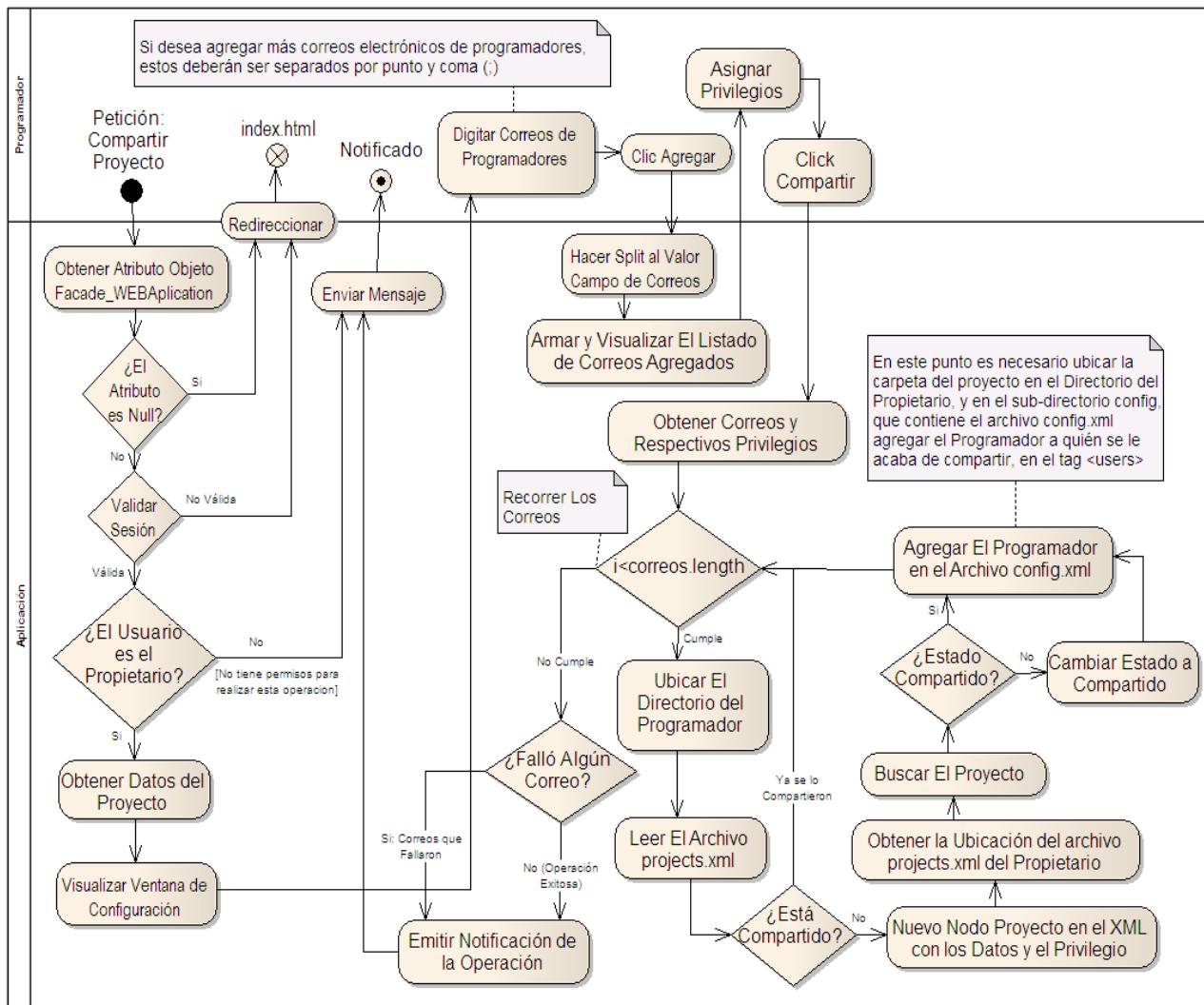


Diagrama 30. Descripción del Proceso Compartir Proyecto

Especificación

1. El proceso inicia cuando el propietario de un proyecto desea compartirlo con otros programadores, y hace su solicitud.
2. A partir de aquí la aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* (paso 11 del proceso login apartado 9.6.3.1).
3. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
4. Luego de esto, se pregunta por el estado de la sesión, ya que si ésta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado 9.6.3.4.
5. Si la sesión es válida, se debe verificar que el programador que solicito la operación es el propietario del proyecto que se pretende compartir. Si la aplicación identifica que dicho programador no es el propietario, se envía un mensaje que diga que no está autorizado para efectuar la operación y por tal razón el flujo terminará.
6. Si el programador es el propietario, se obtienen los datos del proyecto, como el nombre y el propietario.
7. Se visualiza una ventana de configuración, donde se agregarán los programadores a quienes se le compartirá el proyecto y cada uno con sus respectivos privilegios de acceso.
8. El propietario escribirá el o los programadores con quienes compartirá el proyecto. El propietario podrá ingresar dos o más programadores, siempre y cuando separe sus correos electrónicos con el carácter punto y coma.
9. Una vez ingresados los correos electrónicos, el propietario dará clic en el botón agregar.
10. El paso anterior activa un evento que realiza un *split* por el carácter punto y coma para separar cada correo electrónico, almacenándolos en un *array*.
11. Con el *array* generado se construye y visualiza un listado, donde cada elemento tendrá el correo digitado y al lado aparecerá un pequeño

- componente gráfico que permitirá seleccionar el tipo de privilegio que dicho programador tendrá sobre el proyecto.
12. El propietario define los privilegios de cada programador agregado.
 13. Luego se asignar privilegios el propietario dará clic en el botón de compartir.
 14. Se obtienen los correos y sus respectivos privilegios agregados y asignados en la ventana de configuración.
 15. Se inicia una iteración con cada uno de los correos, donde habrá un contador desde cero que aumentará en uno en cada vez que realice un ciclo, siempre y cuando no exceda la cantidad de correos a procesar.
 16. Mientras la condición de la iteración se cumpla se toma el correo a procesar y con éste se ubica el directorio del programador en el servidor.
 17. Se lee el archivo *projects.xml* que allí se encuentra.
 18. Cuando se cargue en memoria el contenido del archivo *xml* leído, se busca si ya el proyecto se le compartió anteriormente, es decir, si el nombre del proyecto y el propietario aparece en *projects.xml* indica que ya le fue compartido. Si ya se encuentra registrado se agrega el correo en un array de fallidos que será evaluado después de procesar todos los correos como se mencionaba en el paso 15, y continuará con una nueva iteración.
 19. Si el programador no tiene compartido el proyecto se procede a crear un nodo *project* en el archivo *projects.xml* del programador y en éste nodo se agregarán datos como el nombre del proyecto, el propietario, el estado compartido y el privilegio que tendrá sobre el mismo.
 20. Después de esto se ubica el archivo *projects.xml* pero ahora del propietario del proyecto.
 21. En este archivo se busca el proyecto que se está compartiendo.
 22. Una vez encontrado se verifica que si su estado es compartido o no.
Si no está compartido sencillamente se actualiza a compartido.
 23. Si su estado ya es compartido o se actualizó, se debe ubicar un archivo *config.xml* que se encuentra en el directorio del proyecto del propietario, en un subdirectorio llamado *config*. En este archivo *xml* se agregarán en la etiqueta *<users>* el programador a quién se le acaba de compartir el proyecto, almacenando el correo del programador y el privilegio que tiene sobre éste. Una vez se realice esta operación se continúa con una nueva iteración.

24. Finalmente, si todos los correos se han procesado en la iteración y la condición ya no se cumple más, se verifica si se presentó algún falló con uno o más de los correos procesados para armar un mensaje de correos fallidos que posteriormente se enviará al propietario. Luego de verificar si se presentaron fallos o no, se emiten notificaciones a todos aquellos programadores a quienes la operación de compartir proyecto fue exitosa, y por último se envía el mensaje de respuesta al programador propietario y será notificado para que el proceso finalice.

9.6.7.4. Dejar de Compartir Proyecto

En todo momento, el propietario de un proyecto compartido, puede decidir a cuál o cuáles programadores dejar de compartir. La aplicación debe ofrecerle dicha funcionalidad al propietario cuando lo crea conveniente, de manera que esto implica borrar todo registro que vincule al programador compartido con el proyecto.

El diagrama 31 describe el proceso que el propietario del proyecto debe realizar para dejar de compartir con un programador.

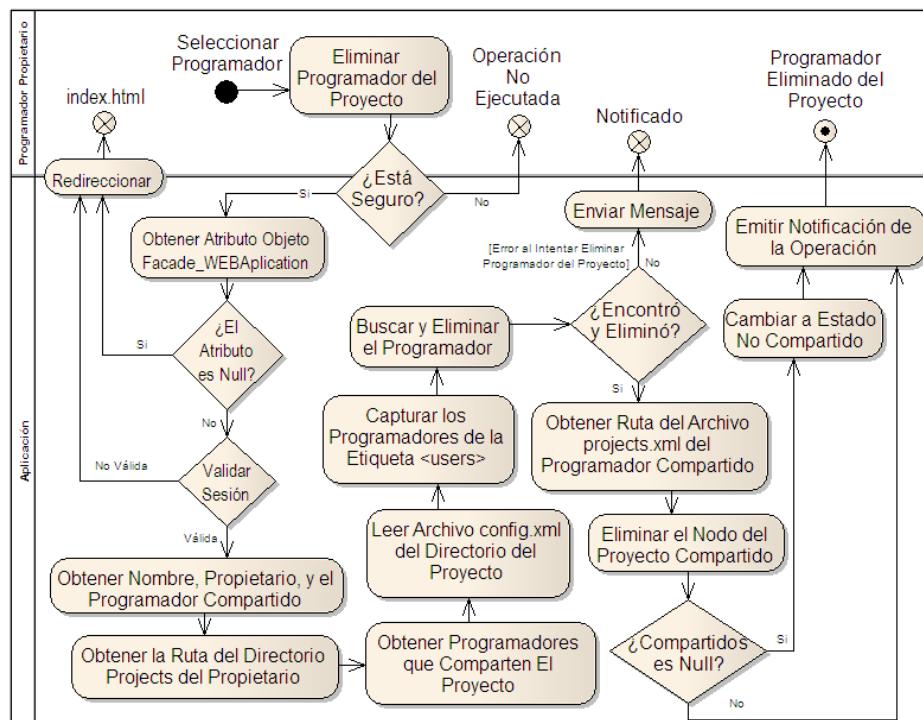


Diagrama 31. Descripción del Proceso Dejar de Compartir Proyecto

Especificación

1. El proceso inicia cuando el programador propietario del proyecto abre la pantalla de propiedades del proyecto y selecciona un programador con el que está compartiendo y solicita dejar de compartir con él.
2. La aplicación despliega un mensaje preguntándole al propietario si está seguro de llevar a cabo esta operación. Si no está seguro terminará el flujo y la aplicación no efectúa ninguna operación.
3. Si el propietario está seguro de proceder, la aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* ([paso 11 del proceso login apartado 9.6.3.1](#)).
4. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
5. Luego de esto, se pregunta por el estado de la sesión, ya que si ésta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado [9.6.3.4](#).
6. Si la sesión es válida, se obtienen los datos del nombre del proyecto, el propietario y el programador que desean dejar de compartir. Estos datos vienen como parámetros en el objeto *request*.
7. Con el dato capturado del propietario se obtiene la ruta del directorio *projects* ubicada en su directorio de usuario en el servidor.
8. La aplicación consulta los programadores con los que se está compartiendo el proyecto.
9. Con el dato del nombre del proyecto, se ubica el directorio que corresponde a dicho nombre en la ruta obtenida en el paso 8. Ya ubicado el proyecto se obtiene y se lee el archivo *config.xml*.
10. Se capturan los programadores con los que se está compartiendo el proyecto, leyendo el nodo *<users>* de dicho archivo *xml*.
11. Se busca el programador que desean dejar de compartirle y proceden a eliminarlo.

12. Verificar si se eliminó el programador. En caso de no haber encontrado un registro que correspondiera al programador, se envía un mensaje indicando que la operación sufrió un error y termina el flujo.
13. Si la eliminación fue exitosa, se obtiene la ruta del archivo *projects.xml* del programador que acaba de ser eliminado del proyecto.
14. Se lee el archivo *projects.xml* y se busca el proyecto que corresponda al que le dejarán de compartir y se eliminará el nodo encontrado.
15. Los pasos anteriores describen todo el proceso de dejar de compartir, pero es necesario verificar si aún existen programadores a los cuales se les esté compartiendo el proyecto.
16. Si ya no existen programadores compartidos, se debe cambiar el estado del proyecto a no compartido.
17. Independientemente del camino que haya tomado después de la verificación del paso anterior, se emite una notificación de que el programador ha sido eliminado del proyecto, y en este punto el proceso finalizará.

9.6.7.5. Dejar de Participar en un Proyecto

Un programador a quién se le ha compartido un proyecto, ya sea con privilegios de escritura/lectura o de solo lectura, está en condiciones de decidir no participar más en éste. Si el programador deja de participar en un proyecto que le ha sido compartido, la aplicación debe notificar a los demás participantes del acontecimiento.

El proceso de dejar de participar en un proyecto por parte del programador que comparte el proyecto, es muy similar a la función que puede realizar un programador propietario para dejar de compartirle a un programador.

El diagrama 32 describe el proceso que puede llevar a cabo un programador que le han compartido un proyecto, y que ha decidido dejar de participar en éste.

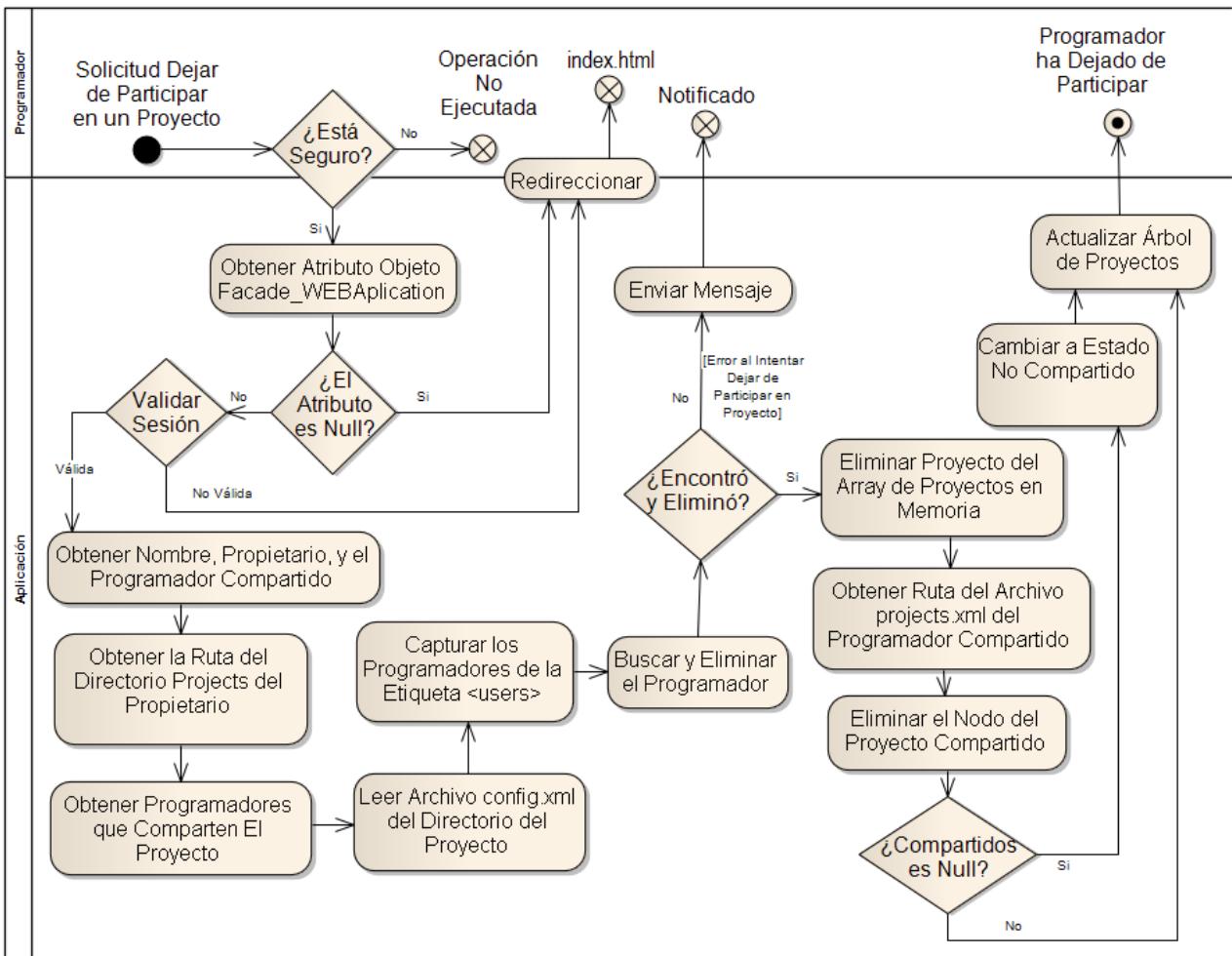


Diagrama 32. Descripción del Proceso Dejar de Participar en un Proyecto

Especificación

- Este proceso inicia cuando un programador que le han compartido un proyecto, y éste se encuentra abierto en la IDE, decide dejar de participar.
- La aplicación despliega un mensaje preguntándole al programador si está seguro de llevar a cabo esta operación. Si no está seguro terminará el flujo y la aplicación no efectúa ninguna operación.
- Si el programador está seguro de proceder, la aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* ([paso 11 del proceso login apartado 9.6.3.1](#)).

4. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
5. Luego de esto, se pregunta por el estado de la sesión, ya que si ésta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado 9.6.3.4.
6. Si la sesión es válida, se obtienen los datos del nombre del proyecto, el propietario y el programador que desean dejar de participar. Estos datos vienen como parámetros en el objeto *request*.
7. Con el dato capturado del propietario se obtiene la ruta del directorio *projects* ubicada en su directorio de usuario en el servidor.
8. La aplicación consulta los programadores con los que se está compartiendo el proyecto.
9. Con el dato del nombre del proyecto, se ubica el directorio que corresponde a dicho nombre en la ruta obtenida en el paso 7. Ya ubicado el proyecto se obtiene y se lee el archivo *config.xml*.
10. Se capturan los programadores con los que se está compartiendo el proyecto, leyendo el nodo *<users>* de dicho archivo *xml*.
11. Se busca el programador que desea dejar de participar y proceden a eliminarlo.
12. Verificar si se eliminó el programador. En caso de no haber encontrado un registro que correspondiera al programador, se envía un mensaje indicando que la operación sufrió un error y termina el flujo.
13. Si la eliminación se efectuó, se elimina el proyecto del array que está cargado en memoria con todos los proyectos que el programador tiene abiertos en la IDE. Éste array es creado en el momento que el programador abre la IDE y se agrega un proyecto cada vez que el programador lo abre.
14. Obtener la ruta el archivo *projects.xml* del programador que ha dejado de participar en el proyecto.
15. Se lee el archivo *projects.xml* y se busca el proyecto en el que dejará de participar y se eliminará el nodo encontrado.
16. Los pasos anteriores describen todo el proceso de dejar de compartir, pero es necesario verificar si aún existen programadores a los cuales se les esté compartiendo el proyecto.

17. Si ya no existen programadores que compartan el proyecto, se debe cambiar el estado del proyecto a no compartido.
18. Independientemente del camino que haya tomado después de la verificación del paso anterior, se actualiza la interface gráfica del árbol de proyectos de manera que ya no se visualice, y el proceso finalizará.

9.6.7.6. Renombrar Proyecto

Es importante que la aplicación le permita a un programador renombrar un proyecto siempre y cuando éste sea el propietario. Ésta funcionalidad debe actualizar:

- El valor de la etiqueta `<name>` del proyecto a renombrar. Ésta etiqueta está contenida en la etiqueta padre `<project>` que representa el proyecto que se renombrará y se encuentra almacenado en el archivo `projects.xml` del directorio del programador propietario. Es importante tener en cuenta que si el proyecto que se encuentra en un estado compartido, se debe actualizar el nombre del proyecto en los respectivos archivos `projects.xml` de cada uno de los programadores a quienes se les comparte.
- El valor de la etiqueta `<name>` contenida en el archivo `config.xml` almacenado en el subdirectorio `config` que se encuentra en el directorio del proyecto que se renombrará. Éste ajuste se realiza una vez, debido a que existe una única copia del proyecto que pertenece al programador propietario.

El diagrama 33 describe el proceso que el propietario de un proyecto debe realizar para renombrar un proyecto.

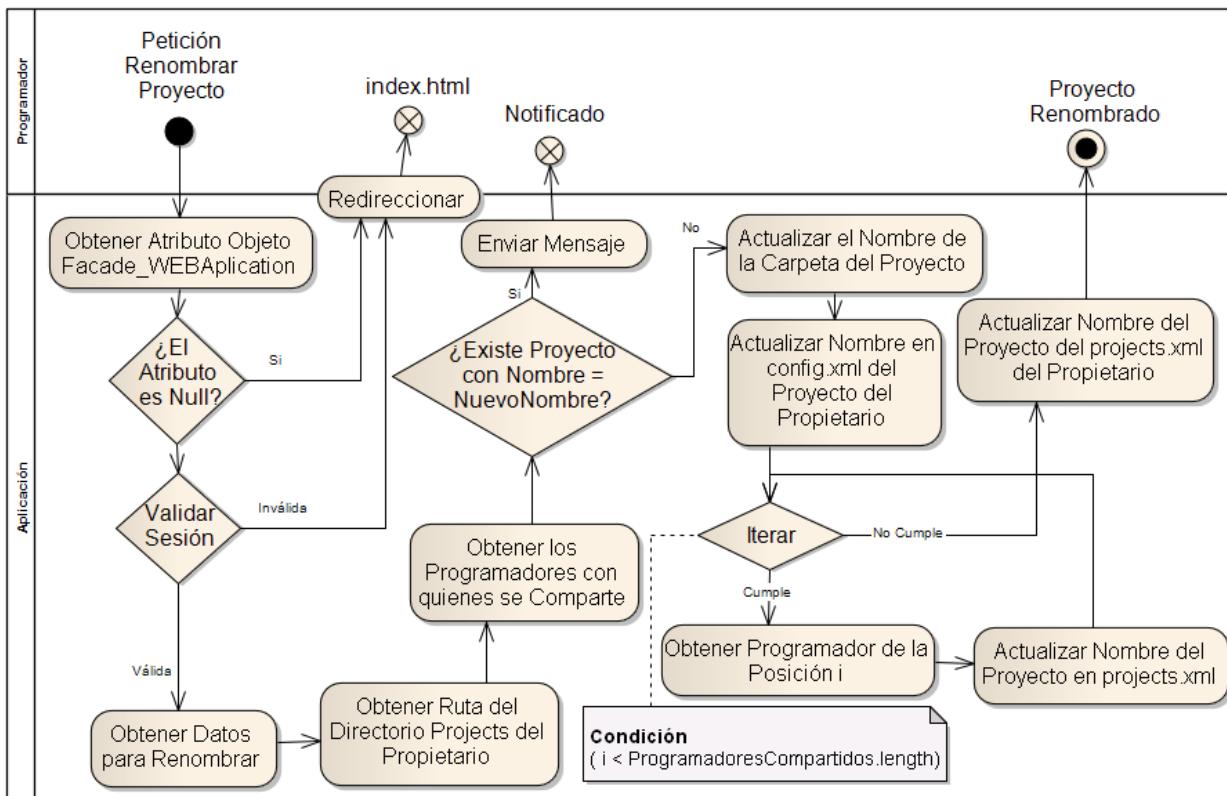


Diagrama 33. Descripción del Proceso Renombrar Proyecto.

Especificación

- El programador propietario del proyecto realiza la solicitud de renombrar el proyecto.
- A partir de aquí la aplicación obtendrá el objeto `Facade_WEBApplication` que al iniciar sesión fue guardado como atributo de la sesión del objeto `request` ([paso 11 del proceso login apartado 9.6.3.1](#)).
- Verifica que este objeto `facade` no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al `index.html` y terminará el flujo.
- Luego de esto, se pregunta por el estado de la sesión, ya que si ésta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al `index.html`. Esta

validación hace parte del control de sesión y se describe en el apartado [9.6.3.4](#).

5. Si la sesión es válida se capturan los datos del objeto *request* correspondientes al nombre y propietario del proyecto, además del nuevo nombre que recibirá.
6. Se obtiene la ruta del directorio *projects* de la carpeta que corresponde del programador propietario. (Importante recordar que cada programador cuenta con un directorio dispuesto en el servidor, y dentro de éste, se encuentra un sub-directorio llamado *projects*).
7. Ahora, con el nombre del proyecto, el propietario, la ruta anteriormente consultada, y el nuevo nombre, se consulta en el archivo *projects.xml* (contenido en el directorio *projects*) los programadores a quienes se les está compartiendo el proyecto.
8. Es necesario verificar si en el archivo *projects.xml* del propietario existe un proyecto cuyo nombre corresponda al nuevo nombre. Si existe un proyecto almacenado con dicho nuevo nombre, se le envía el mensaje al programador y terminará el flujo.
9. Si no existe un proyecto que concuerde con el nuevo nombre, se actualiza el nombre de la carpeta del proyecto a renombrar.
10. En el interior de la carpeta recientemente renombrada, se encuentra un directorio llamado *config*, que a su vez contiene el archivo de configuración *config.xml*, en donde se actualizará el valor de la etiqueta *<name>* con el nuevo nombre.
11. Es necesario ahora actualizar el nuevo nombre del proyecto, en los respectivo archivos *projects.xml* de cada programador a quién se le comparte el proyecto. Para llevar a cabo esto, es necesario iterar los programadores consultados en el paso 7 a través de un *for* que contiene la condición variable *i* sea menor que la cantidad de programadores encontrados.
12. Mientras la condición se cumpla, se obtiene el programador de la posición *i*, se le actualiza el valor de la etiqueta *<name>* de su respetivo archivo *projects.xml*.
13. Cuando la condición no se cumpla se debe actualizar el valor de la etiqueta *<name>* de su archivo *projects.xml* del programador propietario del proyecto. Fin del Proceso.

9.6.7.7. Eliminar Proyecto

Eliminar un proyecto es una operación exclusiva para el propietario del mismo y para el administrador de la aplicación, que consiste en borrar todo registro, referencia y directorios que lo conformen. El proceso de eliminación de un proyecto debe considerar dos aspectos:

- Directorios y archivos de configuración del proyecto, ubicados en la carpeta del programador propietario.
- Verificar si el proyecto está compartido para tener en cuenta las referencias al mismo que se encuentran en los respectivos archivos *projects.xml* de cada programador que es participe.

El diagrama 34 describe el proceso que se lleva a cabo para eliminar un proyecto.

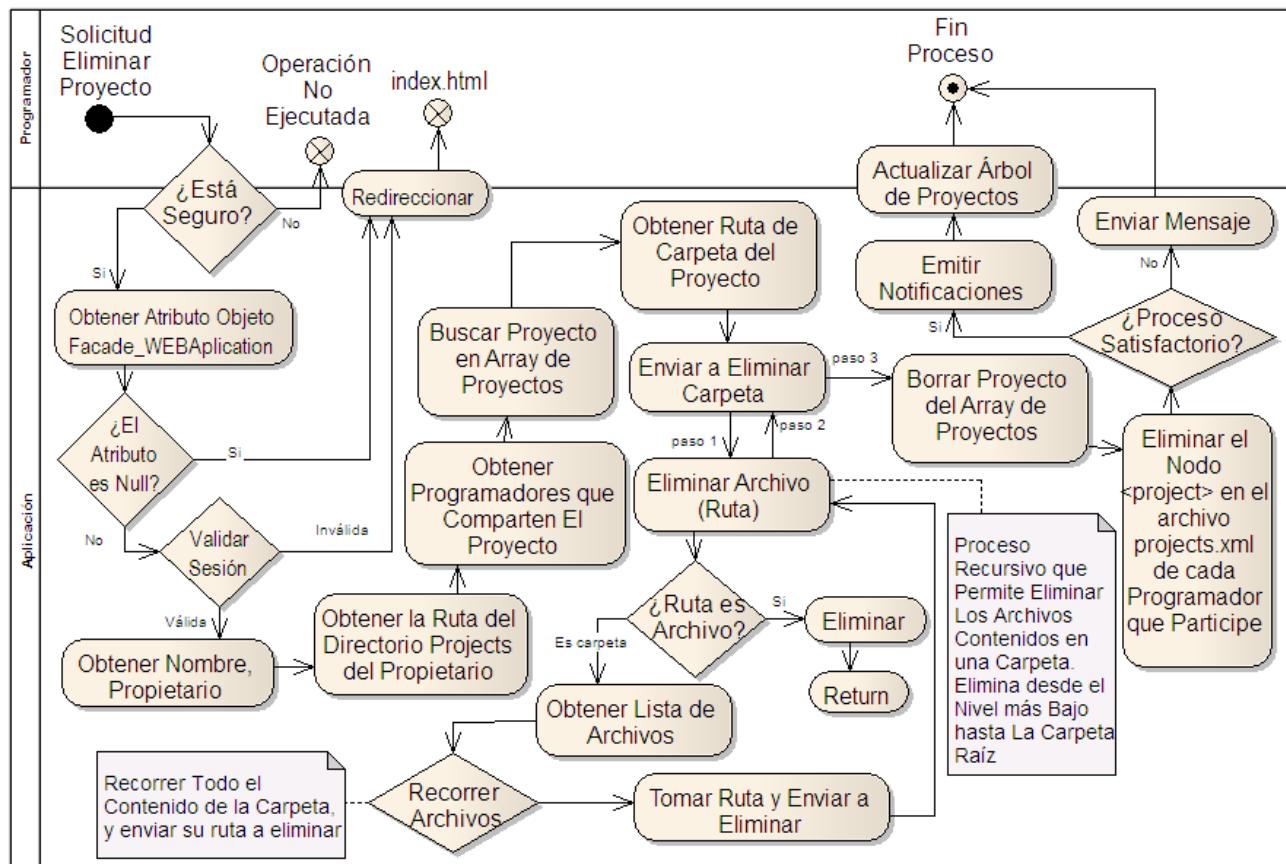


Diagrama 34. Descripción del Proceso Eliminar un Proyecto

Especificación

1. El Programador Propietario del proyecto inicia el proceso con la solicitud de eliminarlo.
2. La aplicación despliega un mensaje preguntándole al programador si está seguro de llevar a cabo esta operación. Si no está seguro terminará el flujo y la aplicación no efectúa ninguna operación.
3. Si el programador está seguro de proceder, la aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* ([paso 11 del proceso login apartado 9.6.3.1](#)).
4. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
5. Luego de esto, se pregunta por el estado de la sesión, ya que si ésta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado [9.6.3.4](#).
6. Si la sesión es válida, se obtienen los datos del nombre del proyecto, y el propietario del mismo. Estos datos vienen como parámetros en el objeto *request*.
7. Con el dato capturado del propietario se obtiene la ruta del directorio *projects* ubicada en su directorio de usuario en el servidor.
8. La aplicación consulta los programadores con los que se está compartiendo el proyecto.
9. Buscar el proyecto en el *array* de proyectos cargados que se crea al abrir la IDE.
10. Obtener la ruta de la carpeta del proyecto.
11. A partir de este paso comienza un subprocesso recursivo que eliminará todo el contenido de la carpeta del proyecto.
12. El subprocesso recibe la ruta del archivo.
13. Verifica si esa ruta corresponde a un archivo o a una carpeta. Si es un archivo lo elimina y retorna.
14. Si es una carpeta toma los archivos que contiene.

15. Iterar por cada archivo y enviar la respectiva ruta al paso 10. Este es el camino recursivo.
16. Los archivos son eliminados desde el nivel de subdirectorios más bajo hasta la carpeta raíz que corresponde a la del proyecto. Una vez el subprocesso recursivo termina se borra el proyecto del *array* de proyectos cargados.
17. Luego de eliminar el proyecto del *array*, continua accediendo a los archivos *projects.xml* de los programadores compartidos obtenidos en el paso 8 para eliminar el nodo que corresponde al proyecto. En este punto se considera que el proyecto ha sido eliminado.
18. Ahora es necesario verificar si el proceso fue satisfactorio. Si ocurrió algún error durante el proceso se debe enviar el mensaje de lo ocurrido al propietario y el flujo terminará
19. Si el proceso fue satisfactorio, se emite notificaciones a todos los que estén compartiendo el proyecto, describiendo que el proyecto ha sido eliminado.
20. Por último se actualiza la interface gráfica del árbol del proyecto, eliminándolo de la misma. Fin de Proceso.

9.6.7.8. Descargar Proyecto

Un programador de la aplicación tendrá la posibilidad de descargar un proyecto. La función descargar proyecto no es excluyente, es decir, sin importar el tipo de privilegios que un programador pueda tener sobre el proyecto (en caso de no ser propietario), podrá descargarlo, ya que esto no implica un mayor riesgo para la integridad del proyecto que está alojado en la aplicación.

El diagrama 35 describe el proceso que se lleva a cabo para la descarga de un proyecto.

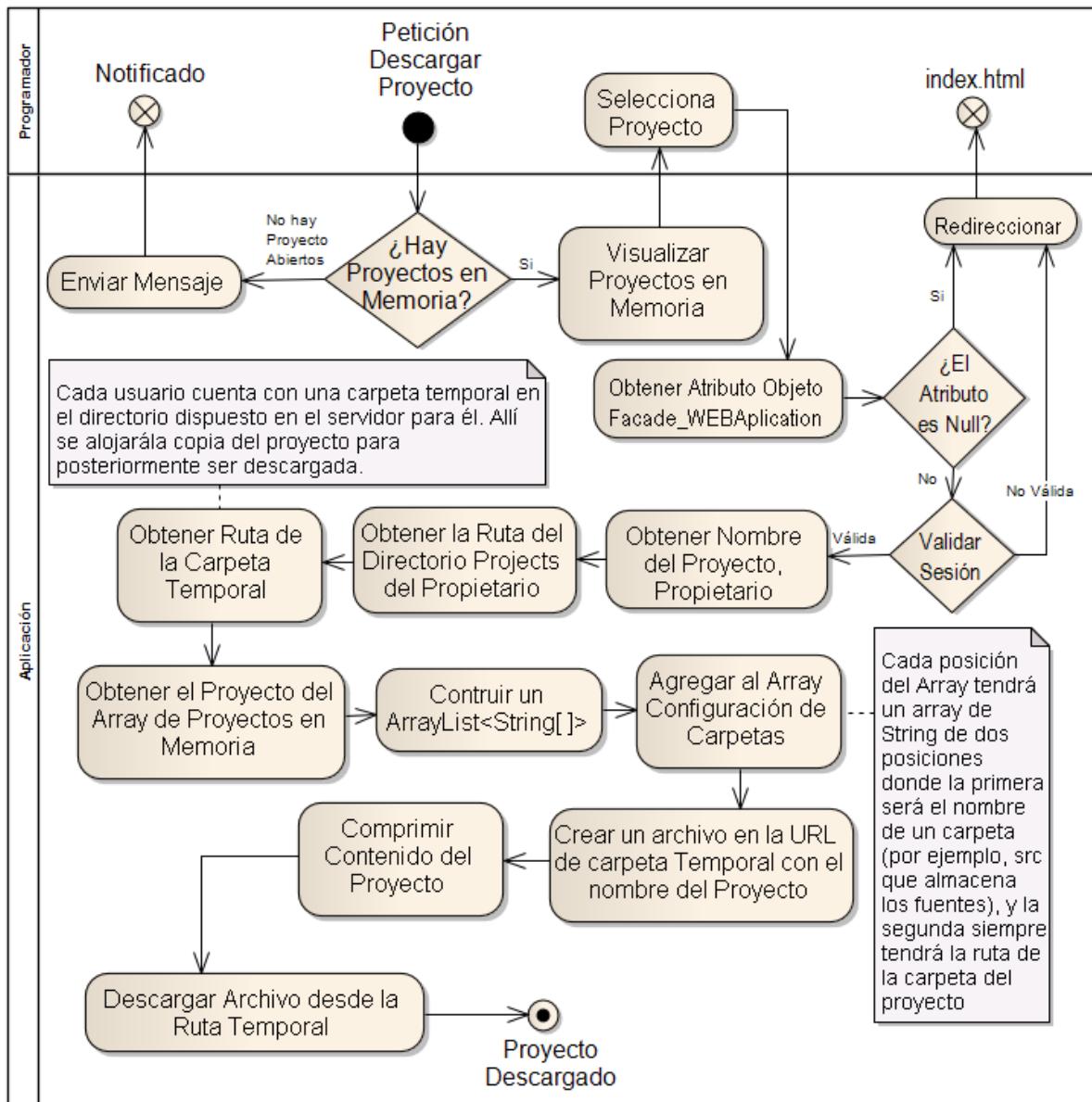


Diagrama 35. Descripción del Proceso Descargar un Proyecto

Especificación

1. El proceso inicia con la petición por parte del programador para descargar un proyecto.
2. La aplicación se encarga de verificar si existen proyectos en memoria (cuando un proyecto se abre, se almacena en memoria). Si no hay proyectos cargados se emite un mensaje el proceso culmina.

3. Si hay proyectos en memoria, la aplicación construye una ventana en donde lista los diferentes proyectos que en ese instante están abiertos en la aplicación
4. El programador debe seleccionar el proyecto que desea descargar
5. La aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* (paso 11 del proceso login apartado 9.6.3.1).
6. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
7. Luego de esto, se pregunta por el estado de la sesión, ya que si ésta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado 9.6.3.4.
8. Si la sesión es válida, se obtienen los datos del nombre del proyecto, y el propietario del mismo. Estos datos vienen como parámetros en el objeto *request*.
9. A partir de estos datos se obtiene la ruta del directorio *projects* que se encuentra en la carpeta del programador propietario del proyecto.
10. En este punto es necesario recordar que cada programador cuenta con una carpeta temporal en su directorio dispuesto en el servidor como se describe en el apartado 9.3.3. Teniendo claro esto la aplicación obtiene la ruta de esta carpeta para que en posteriores pasos ubiquen una copia del proyecto para ser descargada.
11. Se obtiene el proyecto de un array establecido para almacenar los proyectos abiertos. Con esto se evita ir a consultar en los archivos *xml* la existencia del proyecto.
12. Se crea un *array* que contendrá vectores de *String*, en el cual se podrá almacenar toda la información pertinente al proyecto.
13. Se agrega información de las diferentes carpetas que compondrán el proyecto que se descargará.
14. Se crea un archivo que llevará el nombre del proyecto, y se toman como valores de parámetros el *array* anteriormente creado, la ruta de la carpeta temporal, la ruta del directorio del proyecto.
15. Posterior a esto y con los valores obtenido en el paso anterior se procede a utilizar un componente que se encargará de comprimir los archivos.

16. Una vez se ha comprimido todo el contenido de las carpetas que indicaba el array, se procede a ubicar en la carpeta temporal del programador que realizó la petición.
17. Ya ubicado en dicha carpeta se procede a descargar desde el servidor hasta el cliente. Fin de proceso.

9.6.8. Gestión de Ficheros

Un fichero en la aplicación puede ser una clase, una clase de tipo interface gráfica (Clase GUI), un paquete, una librería, archivos planos o algún otro archivo que la aplicación soporte. Cabe mencionar que estas operaciones se realizan sobre un determinado proyecto, a excepción de las operaciones de mover ficheros que implican dos proyectos (origen – destino). También es necesario recordar que estas operaciones solo pueden ser realizadas por aquellos programadores que son propietarios o le han compartido el proyecto con permisos de lectura/escritura, y además toda operación que se efectúe sobre un fichero se llevará a cabo en un paquete, ya sea uno creado o el que el proyecto dispone por defecto.

En este apartado se describirá de manera general y genérica los diferentes procesos que pueden realizarse sobre los ficheros, ya que para todos los tipo de ficheros el comportamiento de sus operaciones son muy similares.

9.6.8.1. Crear Fichero

Crear fichero es una operación disponible para los ficheros de tipo clases, clases GUI, paquetes y archivos de texto plano. La aplicación se encargará de crear en el servidor el respectivo archivo o carpeta que represente un paquete.

El diagrama 36 describe el proceso general que lleva a cabo la aplicación para esta operación:

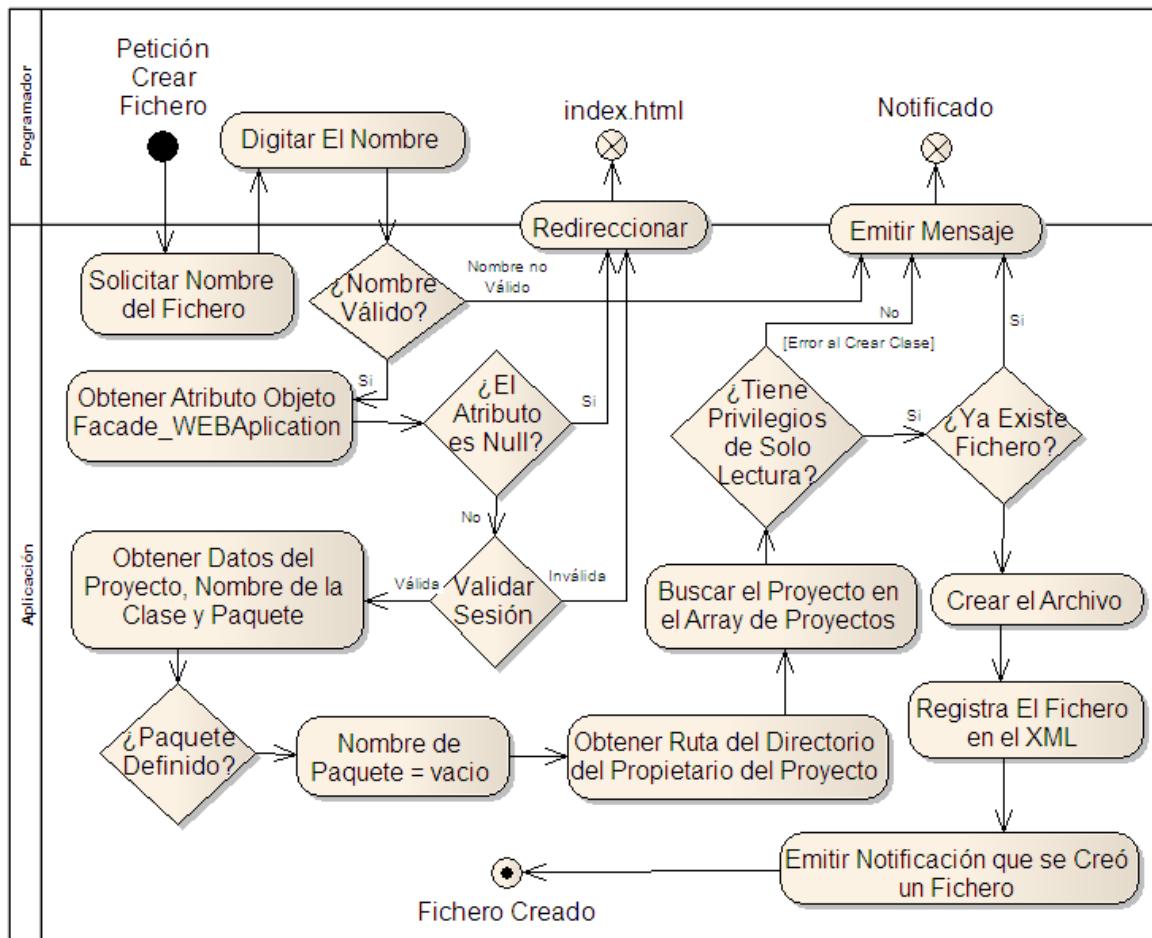


Diagrama 36. Descripción del Proceso Crear Fichero

Especificación

1. Un programador se encuentra en la IDE y pulsa clic derecho en uno de los paquetes visualizados en el árbol de proyectos.
2. La aplicación visualiza una ventana en donde se debe ingresar el nombre del fichero que se creará.
3. El programador ingresa un nombre para el fichero y da clic en aceptar.
4. La aplicación verifica que el nombre no esté vacío, y que además se ajuste a reglas establecidas para los nombres de ficheros a través de una expresión regular, que impide por ejemplo usar nombres que

coincidan con palabras reservadas del lenguaje y algunos caracteres especiales.

5. La aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* (paso 11 del proceso *login* apartado 9.6.3.1).
6. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
7. Luego de esto, se pregunta por el estado de la sesión, ya que si ésta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado 9.6.3.4.
8. Si la sesión es válida, se obtienen los datos del nombre del proyecto, y el propietario del mismo, el nombre que recibirá el fichero y el paquete en el cual se agregará. Estos datos vienen como parámetros en el objeto *request*.
9. La aplicación pregunta si el fichero se agregará en un paquete creado por el programador o se definirá sobre el paquete por defecto que la aplicación define para cada proyecto creado.
10. Si el paquete no está definido, la aplicación asume que el fichero será agregado sobre el paquete default, y por lo tanto el parámetro que identifica el nombre del paquete tendrá un valor vacío.
11. La aplicación consulta la ruta de la carpeta de proyectos de programador propietario, ya que es en la subcarpeta que corresponde al proyecto donde se creará el fichero.
12. Se busca la referencia del proyecto en el *array* que almacena aquellos que están abiertos en la IDE.
13. Se verifica los privilegios que el programador tiene sobre el proyecto para así definir si la operación puede llevarse a cabo o no. Si el programador no tiene privilegios de escritura se envía un mensaje de error al intentar crear el fichero y allí terminará el flujo del proceso.
14. Si el programador es el propietario o es un programador a quien le han compartido el proyecto con permisos de escritura, se procede a corroborar que el nombre que fue ingresado para dárselo al fichero nuevo no corresponda a uno creado ya en el proyecto. Si ya existe un fichero con este nombre se emite el mismo mensaje y el flujo terminará.

15. Se crea un objeto para crear el archivo. En el caso de una clase y clase GUI se escribirá en este archivo una plantilla de código base, es decir, al crear una clase la aplicación escribirá el código fuente de declaración de una clase JAVA.
16. Una vez creado el fichero en el servidor, específicamente en el paquete que se determinó, se debe registrar el nuevo fichero en los respectivos archivos de configuración *xml* que le sirven de indexador del contenido del proyecto a la aplicación.
17. Por último es necesario en el caso de que el proyecto sea compartido, y que en ese momento otros programadores estén trabajando en el proyecto, de notificar que un nuevo fichero ha sido creado. En este punto el proceso termina cuando todos los programadores que comparten el proyecto vean el cambio que se ha presentado.

9.6.8.2. Renombrar Fichero

Renombrar un fichero es una función disponible para aquellos programadores propietarios, o que comparten el proyecto con privilegios de escritura/lectura. El renombrar un fichero implica actualizar la interface gráfica, el nombre del archivo en el servidor y en los diferentes archivos de configuración que lo tiene referenciado, y emitir una notificación a los demás programadores que están trabajando colaborativamente para que se efectúe el cambio en su pantalla.

El diagrama 37 describe el proceso de renombrar un fichero que la aplicación debe llevar a cabo para renombrar un fichero..

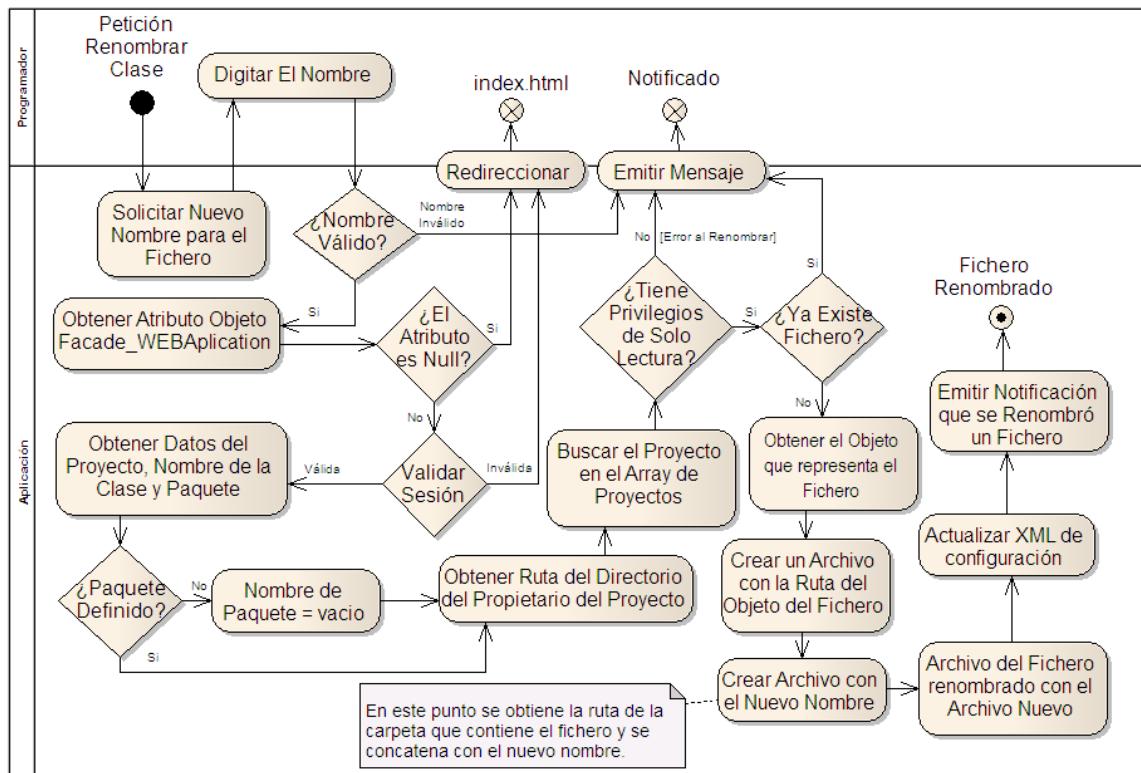


Diagrama 37. Descripción del Proceso Renombrar Fichero

Especificación

1. Un programador se encuentra en la IDE y pulsa clic derecho en uno de los ficheros visualizados en el árbol de proyectos.
2. La aplicación visualiza una ventana en donde se debe ingresar el nuevo nombre del fichero.
3. El programador ingresa un nombre para el fichero y da clic en aceptar.
4. La aplicación verifica que el nombre no esté vacío, y que además se ajuste a reglas establecidas para los nombres de ficheros a través de una expresión regular, que impide por ejemplo usar nombres que coincidan con palabras reservadas del lenguaje y algunos caracteres especiales.
5. La aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* ([paso 11 del proceso login apartado 9.6.3.1](#)).
6. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el

- objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
7. Luego de esto, se pregunta por el estado de la sesión, ya que si ésta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado 9.6.3.4.
 8. Si la sesión es válida, se obtienen los datos del nombre del proyecto, y el propietario del mismo, el nombre que recibirá el fichero y el paquete en el cual se agregará. Estos datos vienen como parámetros en el objeto *request*.
 9. La aplicación pregunta si el fichero que se renombrará está en un paquete creado por el programador o está en paquete por defecto.
 10. Si el paquete no está definido, la aplicación asume que el fichero será agregado sobre el paquete *default*, y por lo tanto el parámetro que identifica el nombre del paquete tendrá un valor vacío.
 11. La aplicación consulta la ruta de la carpeta de proyectos de programador propietario, ya que es en la subcarpeta que corresponde al proyecto donde se buscará y renombrará el fichero.
 12. Se busca la referencia del proyecto en el *array* que almacena aquellos que están abiertos en la IDE.
 13. Se verifica los privilegios que el programador tiene sobre el proyecto para así definir si la operación puede llevarse a cabo o no. Si el programador no tiene privilegios de escritura se envía un mensaje de error al intentar crear el fichero y allí terminará el flujo del proceso.
 14. Si el programador es el propietario o es un programador a quien le han compartido el proyecto con permisos de escritura, se procede a corroborar que el nuevo nombre que fue ingresado para el fichero no corresponda a uno creado ya en el proyecto. Si ya existe un fichero con este nombre se emite el mismo mensaje y el flujo terminará.
 15. Se crea un objeto de tipo *Class* en donde se almacenará el fichero que será renombrado.
 16. Con el objeto anteriormente creado se obtiene la ruta exacta del fichero, la cual se usa para crear un archivo.
 17. Ahora bien, con el objeto creado en el paso 15 se puede obtener la ruta del nivel padre del fichero, es decir, si el fichero está en un paquete, en este paso se obtiene la ruta de dicho nivel. Con la ruta captura y concatenándole el nuevo nombre se crea un nuevo archivo.

18. A través de un método llamado *renameTo* de la clase *File*, se logra realizar la operación de renombrar usando el archivo tipo *File* creado en el paso 16, y como parámetro el archivo creado para el nuevo nombre.
19. Luego de renombrar el fichero la aplicación actualiza los respectivos archivos *xml* en donde se esté referenciando el anterior nombre del fichero, remplazándolo por el nuevo.
20. Por último la aplicación debe emitir una notificación a todos aquellos programadores que comparten el proyecto, que un fichero ha sido renombrado. En este punto el proceso ha finalizado.

9.6.8.3. Eliminar Fichero

Un programador podrá siempre y cuando tenga permisos de escritura/lectura sobre el proyecto, eliminar ya sea una clase, un paquete o algún otro tipo de fichero. El cambio que produce esta operación sobre la interface gráfica debe verse reflejado en todos los programadores que en ese instante están trabajando sobre el proyecto. El diagrama 38 describe el proceso que implica eliminar un fichero:

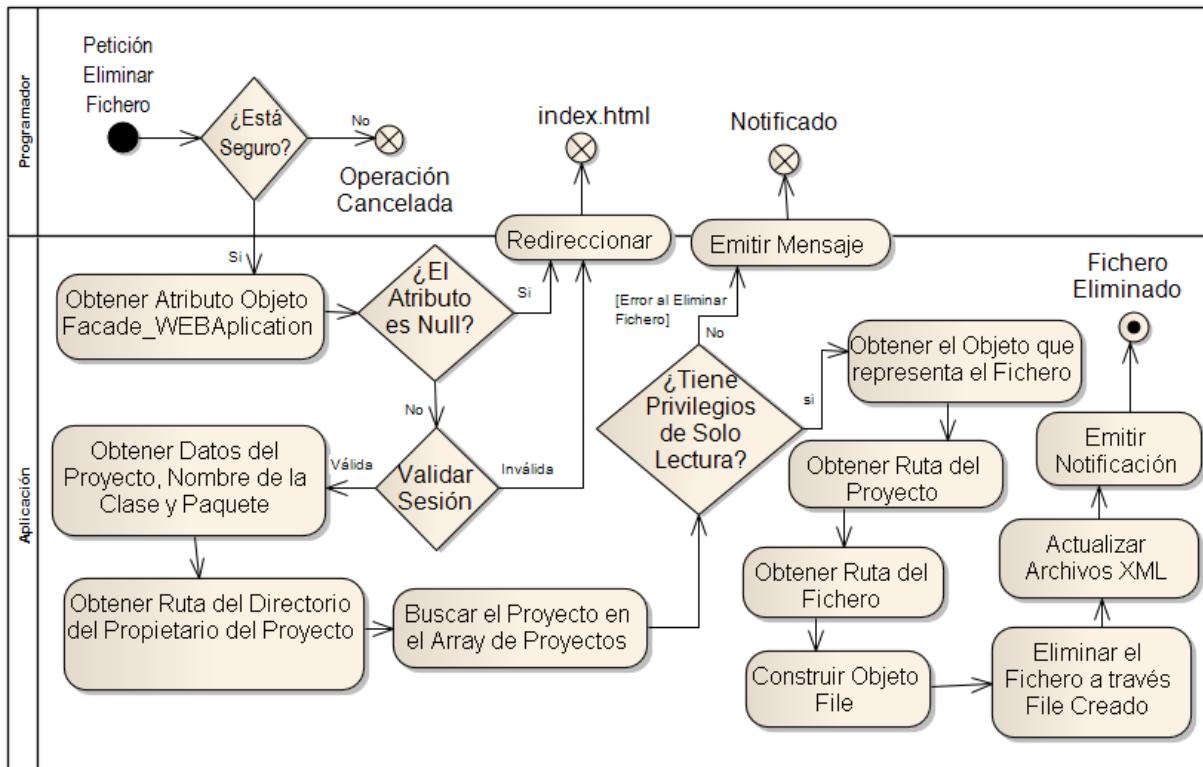


Diagrama 38. Descripción del Proceso Eliminar Fichero

Especificación

1. El programador desea eliminar un fichero que se encuentra en un determinado paquete del árbol del proyecto. Al pulsar clic derecho sobre el fichero seleccionará la opción eliminar iniciando de esta manera el proceso de eliminar un fichero.
2. La aplicación preguntará al programador si está seguro de seguir adelante con esta operación. Es necesario preguntar ya que esta es una operación irreversible, y además para evitar que al pulsar por accidente esta opción se elimine un fichero. Si el programador no está seguro la aplicación cancelará el proceso y terminará el flujo del mismo.
3. La aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* (paso 11 del proceso login apartado 9.6.3.1).
4. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
5. Luego de esto, se pregunta por el estado de la sesión, ya que si ésta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado 9.6.3.4.
6. Si la sesión es válida, se obtienen los datos del nombre del proyecto, y el propietario del mismo, el nombre del fichero y el paquete en el cual se encuentra. Estos datos vienen como parámetros en el objeto *request*.
7. La aplicación consulta la ruta de la carpeta de proyectos de programador propietario, ya que es en la subcarpeta que corresponde al proyecto donde se buscará y se eliminará el fichero.
8. Se busca la referencia del proyecto en el *array* que almacena aquellos que están abiertos en la IDE.
9. Se verifica los privilegios que el programador tiene sobre el proyecto para así definir si la operación puede llevarse a cabo o no. Si el programador no tiene privilegios de escritura se envía un mensaje de error al intentar eliminar el fichero y allí terminará el flujo del proceso.

10. Se crea un objeto de tipo *Class* en donde se almacenará el fichero que será eliminado.
11. Se obtiene la ruta del proyecto
12. Se obtiene la ruta exacta del fichero a través del objeto creado en el paso 10.
13. La aplicación crea un objeto *File* a través de la ruta del paso 12.
14. A través del objeto *File* creado se invoca un método *delete()* que se encargará de eliminar el fichero que se encuentra en el servidor.
15. Ahora es necesario actualizar los respectivos archivos *xml*, borrando todo registro que refiera al fichero que fue eliminado.
16. Por último es necesario que la aplicación notifique a los demás programadores que trabajan en el proyecto, que un fichero ha sido eliminado. Una vez la notificación del mensaje se haya enviado finaliza el proceso.

9.6.8.4. Mover Fichero

Mover un fichero consiste en llevar a cabo una operación copiar y pegar desde un origen a un destino. El movimiento de fichero puede realizarse entre los diferentes paquetes del proyecto, e incluso entre los paquetes de diferentes proyectos.

Mover un fichero requiere que quién solicite la operación tenga permisos de escritura sobre el proyecto, o en los dos proyectos si es el caso de mover entre proyectos. También es necesario mencionar que los ficheros de tipo librería solo puede llevarse a cabo entre proyectos y solo en el paquete dispuestas para estas.

9.6.9. Concurrencia

En un escenario real de programación se encuentran situaciones en donde en un proyecto intervienen dos o más programadores, que deben llevar a cabo tareas independientes; de manera que cada uno tendrá una copia del proyecto y trabajará en esta, pero al final deben realizar un empalme de todos los cambios para obtener una única copia funcional.

Los programadores pueden optar por alguna de estas opciones:

- Escoger un proyecto de todas las copias que existan, copiar y pegar cada uno de los cambios que cada programador realizó. Esta es una tarea muy tediosa y poco eficiente porque podrían cometerse muchos errores al agregar los cambios, o pueden resultar conflictos entre variables incluso métodos.
- Escoger un servidor subversión y alojar en éste una copia centralizada, y cada programador clonará el proyecto en su máquina, y cuando se realice cambios y quiera empalmarlos, simplemente realiza una operación *commit* al servidor. Pero para que los otros programadores puedan darse por enterado que hay una nueva versión, debe ejecutar una operación *update* para que el servidor ajuste la copia.
- Usar un sistema de control de versión de tipo GIT, el cual es similar al anterior pero difieren en que un programador puede realizar los *commit* que desee sin necesidad de notificar cambios al servidor. Cuando el programador quiere enviar los cambios al servidor debe ejecutar una operación *push*.

Las consideraciones que se acaban de presentar tienen un factor común, donde cada programador tiene una copia del proyecto y trabajará en esta independientemente, y solo se reflejarán cambios cuando se ejecuten operaciones de control de versiones.

9.6.9.1. Pioneros en Transformación Operacional

En el 2009 Google incursionó en la implementación de un concepto llamado Transformación Operacional (8.1.3.Familiarización Con Tecnologías, Herramientas y Prácticas) en proyectos como *Google Wave* y *Google Docs*. Este concepto básicamente consiste en un algoritmo que permite que desde varios nodos conectados entre sí, se logre editar un documento en tiempo real, asegurando que en cada cambio, todos los nodos tengan una misma versión.

En *Google Docs* que actualmente se llama *Google Drive*, se puede crear documentos, hojas de cálculo, presentaciones, editar de imágenes y crear formularios para compartirlos con otros usuarios asignándole privilegios de

acceso sobre estos. Por ejemplo, si un documento se ha compartido con permisos de edición, y los usuarios que lo comparten se conectan y lo abren, podrán editar simultáneamente y ambos verán los cambios que cada uno está efectuando en tiempo real.

Por su parte Google Wave presentado en la conferencia Google I/O del año 2009, fue el pionero en promover una aplicación pensada para la comunicación y colaboración en tiempo real entre múltiples usuarios conectados entre sí usando un navegador. Este proyecto integró servicios de email, mensajería instantánea, redes sociales, wikis, implementó *drag-and-drop* de archivos desde el escritorio al navegador, se compartían imágenes, y mantenía un historial de cambios que se realizaban en la aplicación. Este proyecto innovador no tuvo la acogida de usuarios que Google esperaba y decidieron progresivamente abandonar el proyecto, hasta que el 30 de abril de 2012 apagaron los servidores que alojaban esta aplicación. Pero a pesar de que el proyecto fracasó, no todo fue malo, ya que los componentes innovadores que desarrollaron se dispusieron en código libre para que los desarrolladores lo integraran en futuros proyectos, y Google se comprometió a que estas tecnologías podrían integrarlas en futuros productos, como lo lograron en Google Docs⁹⁹¹⁰⁰.

9.6.9.2. Integración

La idea de que un grupo de programadores pudiese trabajar en un proyecto simultáneamente, fue inspirada por la funcionalidad que ofrece Google Docs en edición colaborativa sobre sus documentos. Proporcionarle a la aplicación la edición concurrente fomentaría el trabajo en equipo en los desarrollos que en ella se hicieran, y agilizaría el trabajo ya que no sería necesario ningún tipo de sistema de control de versiones, ni mucho menos realizar empalmes de copiar y pegar.

Cuando la idea surgió no se contaba con conocimientos técnicos para llevarla a cabo, lo que obligó a realizar una búsqueda avanzada de todo

⁹⁹ Urs Hözle, Senior Vice President, Operations & Google Fellow. Update on Google Wave – Official Blog <http://googleblog.blogspot.com/2010/08/update-on-google-wave.html>

¹⁰⁰ Soporte de Google. Status of Google Wave
<http://support.google.com/bin/answer.py?hl=en&hlrm=en&answer=1083134&rd=1>

aquellos que aportará a la implementación o integración de esta funcionalidad a la aplicación.

Ya definido el componente que integrará el concepto de Transformación Operacional se ha implementado en la aplicación, se determina que un programador que ha creado un proyecto pueda compartirlo con otros y a cada uno asignarle el privilegio que tendrá sobre éste. Cuando dos programadores tienen permisos de escritura sobre un proyecto y lo abren, la aplicación les permitirá que editen concurrentemente a través de *ShareJS*, ya sea en la misma clase en donde verán los cambios que cada uno edita, o también en clases diferentes donde siempre se mantendrá una sola versión.

La aplicación tiene una única copia del proyecto que se encuentra en la carpeta de *projects* del propietario del mismo, y cada vez que un programador que le ha sido compartido con permisos de escritura lo abre, editará sobre los archivos que están alojados en el servidor para este proyecto.

9.6.9.3. Proceso

El componente *ShareJS* maneja un concepto llamado documento con el cual representa cada elemento en el cual se aplicará el algoritmo de transformación operacional, y además establece dos tipos de parámetros para interpretar la concurrencia:

- *Text*: Este parámetro le indica al componente que el recurso o el documento al cual se aplicará el algoritmo es un *array* de caracteres. *ShareJS* en este tipo de documentos responderá a posiciones en el array en el cual contralará la adición, modificación y eliminación de caracteres. La aplicación usa este tipo de documentos para manejar la concurrencia del código fuente que una clase contiene.
- *JSON*: Cuando el parámetro indica *JSON*, el componente interpretará los cambios simultáneos a través de atributos que el objeto incluye. Este tipo de concurrencia tiene utilidad en escenarios gráficos, que implica cambios y movimientos. En la aplicación este tipo de concurrencia es utilizada en el constructor de interfaces gráficas, en donde se maneja un archivo con extensión *fxml* que almacena las

propiedades del lienzo, los componentes gráficos agregados en él; y en cada uno, se guardan datos como ancho, alto y coordenadas de posición (x,y) con respecto al lienzo. Cada vez que un componente gráfico en el constructor es agregado, ShareJS se encarga de modificar las propiedades almacenadas en el archivo frml y además de notificar esos cambios a través de Socket.IO para que se vean reflejados en el lienzo de los programadores que están compartiendo el proyecto.

El proceso de la concurrencia independientemente del tipo de parámetro o documento a interpretar es igual, solo difieren en que uno indica que lo hará a través de caracteres y el otro que lo hará con objetos JSON. La figura X describe de manera general como funciona este componente:

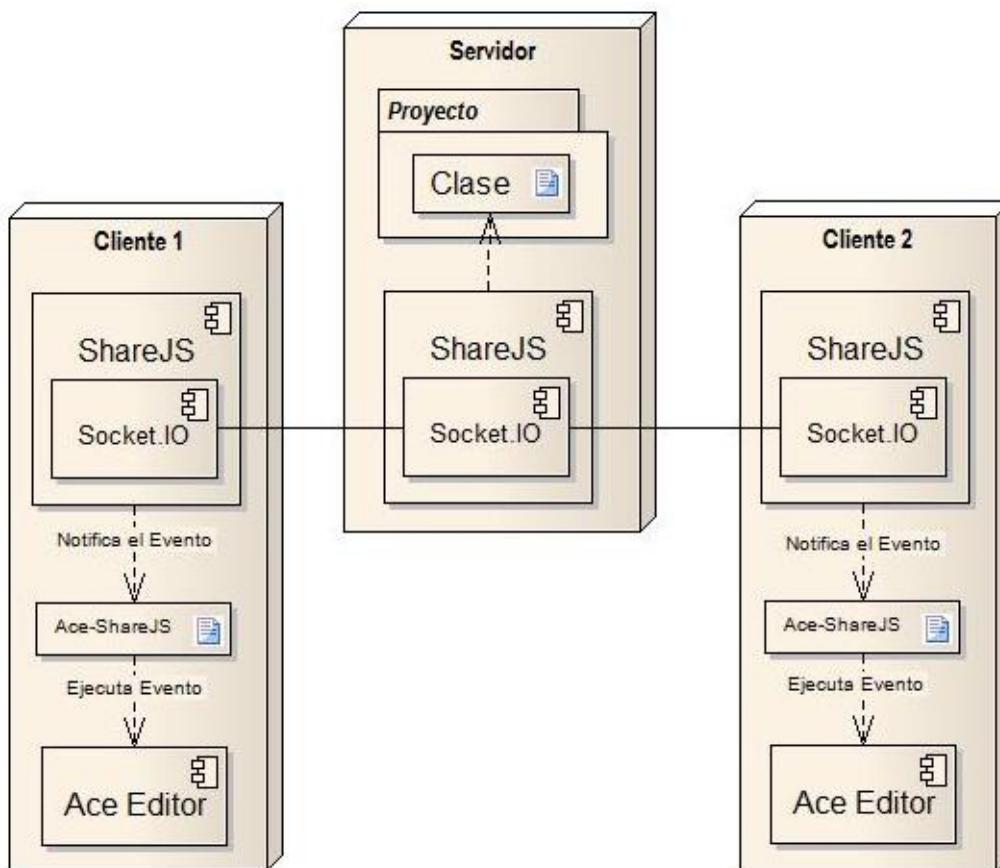


Diagrama 39. Descripción del Proceso de Concurrencia

Especificación

ShareJS al igual que Socket.IO está instanciado tanto en el servidor como en cada cliente que se conecte con la aplicación. ShareJS es el encargado de acceder al archivo que corresponda en el servidor, ya sea una clase, un archivo plano o un archivo frml. El rol que Socket.IO cumple en este contexto es el de transmitir el evento que a través del Ace Editor fue lanzado, y este último es el componente desde donde se harán las ediciones del código fuente. Ahora, para entender mejor el comportamiento de estos componentes es necesario presentarles un ejemplo donde se pueda explicar paso a paso lo que llega a suceder cuando se está trabajando concurrentemente sobre una clase. El ejemplo incluye dos clientes que en este caso son dos programadores que han ingresado a la aplicación, han abierto el mismo proyecto y a su vez también han abierto en la IDE la misma clase.

1. Cuando un fichero se abre en la IDE, la aplicación determina qué clase de fichero es, para definir el tipo de concurrencia que va a controlar. Por ejemplo si se trata de una clase, la aplicación deberá tratar la concurrencia con documentos de tipo texto. En este ejemplo se usará un clase como fichero.
2. Una vez la aplicación conoce cómo va a controlar la concurrencia, establece a través del Socket.IO una conexión con su homólogo instanciado en el servidor, por la cual se transmitirá los eventos de concurrencia. Socket.IO abre un canal por cada documento creado en el ShareJS, y a su vez registra este canal a un *room* que usará para realizar el *broadcast* de cada evento que acontezca.
3. Pero ahora es necesario integrar a Ace Editor con el ShareJS, para que este último escuche los eventos que en el editor suceda. Para cumplir con este objetivo, existe un archivo JavaScript Ace-ShareJS.js que escucha cada uno de los eventos de Ace Editor y ejecuta un evento de ShareJS, para que éste controle a través de su algoritmo de Transformación Operacional la versión del documento ya ha surgido un nuevo cambio.
4. Cuando un evento sucede en el editor, y el anterior archivo lo escucha y lo transfiere al ShareJS, se controla se ajusta la versión y se transmite el evento a través de Socket.IO.
5. La instancia del Socket.IO del servidor recibe el mensaje y los transfiere al ShareJS.

6. El *ShareJS* del servidor constantemente está editando sobre el archivo .java que corresponde a la clase que fue abierta en el editor. Es por esta razón que la aplicación no cuenta con un botón de guardar, porque siempre se está almacenando los cambios que se presentan en tiempo real. Cuando el *ShareJS* recibe el evento del *Socket.IO* procede a realizar los respectivos ajustes en el archivo de la nueva versión.
7. Después de esto *ShareJS* usa *Socket.IO* para realizar *broadcast* a los que están suscritos al *room* creado para dicha clase.
8. El *Socket.IO* de cada cliente que está suscrito al *room* recibe el evento, se lo transmite al su respectivo *ShareJS* para que ajuste la versión.
9. Por último, *ShareJS* debe reflejar el ajuste en el editor, de manera que reaparece el archivo Ace-Share.js para comunicar al *Ace Editor* del cambio.

9.6.10. Ejecutar Proyecto

Un programador que usa una IDE espera poder ejecutar su proyecto desde el mismo entorno, y no que debe acceder a una consola o terminal a escribir líneas de comando a la máquina virtual para lanzar la ejecución. Esta aplicación no debe ser la excepción.

En este punto es donde se puede ver la utilidad de la pequeña investigación realizada en el apartado 6.8 Ambientes de Ejecución y del 7. Resultado Obtenidos, en donde se describían las opciones de ejecución de programas de Java a través de Internet y el método seleccionado para llevar a cabo esta tarea. En los resultados obtenidos con respecto a la ejecución, se definió que el método a utilizar para lanzar la ejecución de los programas desarrollados en la aplicación, es a través de un applet.

El diagrama 40 describe el proceso que la aplicación lleva a cabo para que un proyecto sea ejecutado. Algunos de los pasos de este flujo serán ampliados más adelante.

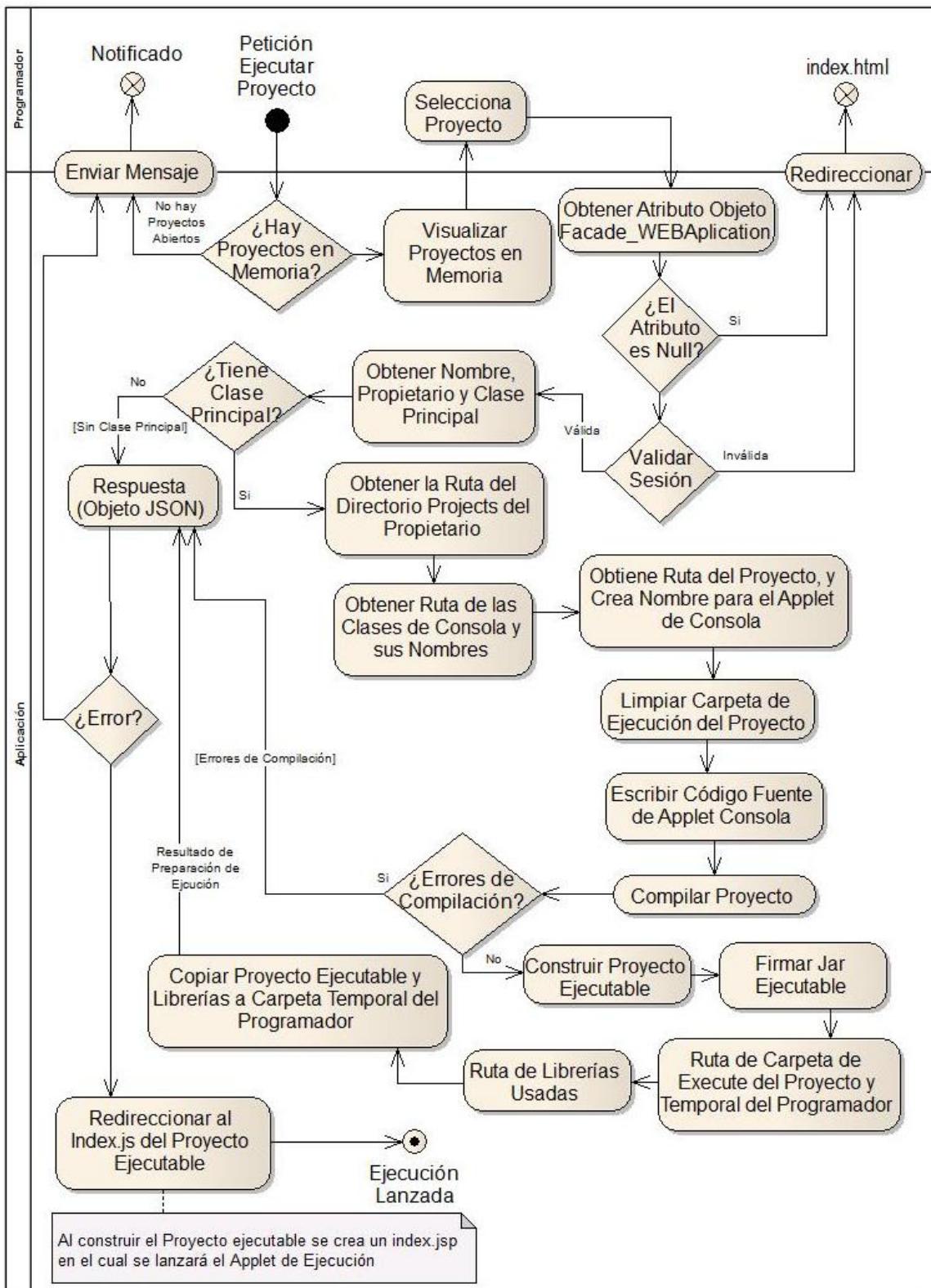


Diagrama 40. Descripción del Proceso Ejecutar Proyecto

Especificación

1. El programador inicia el proceso con una petición a la aplicación de ejecutar un proyecto.
2. La aplicación verifica si existen proyectos cargados en memoria. Si no hay proyectos se emite un mensaje diciéndole al programador que no hay proyectos abiertos y terminará el flujo.
3. Si al menos hay un proyecto en memoria, la aplicación hace un listado de ellos y se los visualiza al programador.
4. El programador selecciona el proyecto que desea ejecutar.
5. La aplicación obtendrá el objeto *Facade_WEBApplication* que al iniciar sesión fue guardado como atributo de la sesión del objeto *request* (paso 11 del proceso login apartado 9.6.3.1).
6. Verifica que este objeto *facade* no sea nulo para cerciorarse que no se haya solicitado esta acción sin antes haber iniciado sesión. Si el objeto es nulo se procede a redireccionar al *index.html* y terminará el flujo.
7. Luego de esto, se pregunta por el estado de la sesión, ya que si ésta es inválida quiere decir que otra persona ha iniciado sesión con la misma cuenta, y si es así se redirecciona al *index.html*. Esta validación hace parte del control de sesión y se describe en el apartado 9.6.3.4.
8. Si la sesión es válida, se obtienen los datos del nombre del proyecto, y el propietario del mismo y la clase principal. Estos datos vienen como parámetros en el objeto *request*.
9. La aplicación debe verificar que el proyecto seleccionado tenga en sus propiedades, definida una clase principal. Si no es así, se construye una respuesta de error, diciendo que el proyecto no tiene clase principal y terminará el flujo.
10. Si el proyecto tiene definida su clase principal, se obtiene la ruta de la carpeta de projects del propietario.
11. Se consulta las rutas de las clases y nombre de las mismas que permitirán crear una consola para entrada y salida de datos. Esta consola por ejemplo permitirá visualizar todo aquello que se imprima a través de *System.out.println*.
12. Obtener la ruta de la carpeta del proyecto a ejecutar y además establecer un nombre para el applet de consola.
13. La aplicación limpia el contenido de la carpeta de ejecución del proyecto, por si ya se había ejecutado.

14. La aplicación escribe el código fuente del applet de consola. A través de esta consola se lanzará la ejecución del proyecto.
15. Seguido a esto se debe compilar el proyecto. Este proceso será explicado con mayor detalle más adelante.
16. Si existen errores de compilación, se construye una respuesta de error diciendo que existen errores de compilación.
17. Si no hay errores se procede a construir el archivo JAR ejecutable del proyecto. Este proceso se describirá con mayor detalle más adelante.
18. Ese jar creado debe ser firmado para brindarle confianza al programador cliente de que es seguro. Esto implica solicitar autorización al usuario que permita usar su máquina virtual para ejecutar el proyecto.
19. Se obtiene las rutas de la carpeta de ejecución del propietario, la ruta de la carpeta de temporal del programador que quiere ejecutar el proyecto y la ruta donde está alojadas las librerías que usará el proyecto.
20. Una vez conseguidas las rutas se procede a copiar el proyecto ejecutable de la carpeta `execute` del propietario y las librerías, a la carpeta temporal del programador.
21. Cuando los archivos son copiados en la carpeta temporal se construye el resultado correcto.
22. Ahora la aplicación evalúa el resultado obtenido en el proceso, y si es de tipo error emite un mensaje de error y el flujo termina.
23. Si el resultado fue correcto entonces la aplicación redirecciona al `index.jsp` del proyecto construido, que contiene un applet que lanza toda la ejecución en el cliente una vez este otorgue permisos para usar su máquina virtual. Independientemente de que si el cliente autorice o no el proceso finaliza.

9.6.10.1 Compilar Proyecto

Compilar un proyecto es la operación que la aplicación le ofrece al programador para verificar si el código fuente escrito, esté sintácticamente correcto. Este proceso notificará si existen errores de sintaxis en el proyecto, dándole los detalles como la clase, la línea donde se presentó, y cuál es el error encontrado. La API de Java ofrece algunas clases que permiten ser importadas para crear rutinas que compilen determinado código fuente, de manera que reutilizándolo y realizando algunos ajustes fue posible

implementar esta funcionalidad en el proyecto. El diagrama 41 describe este proceso.

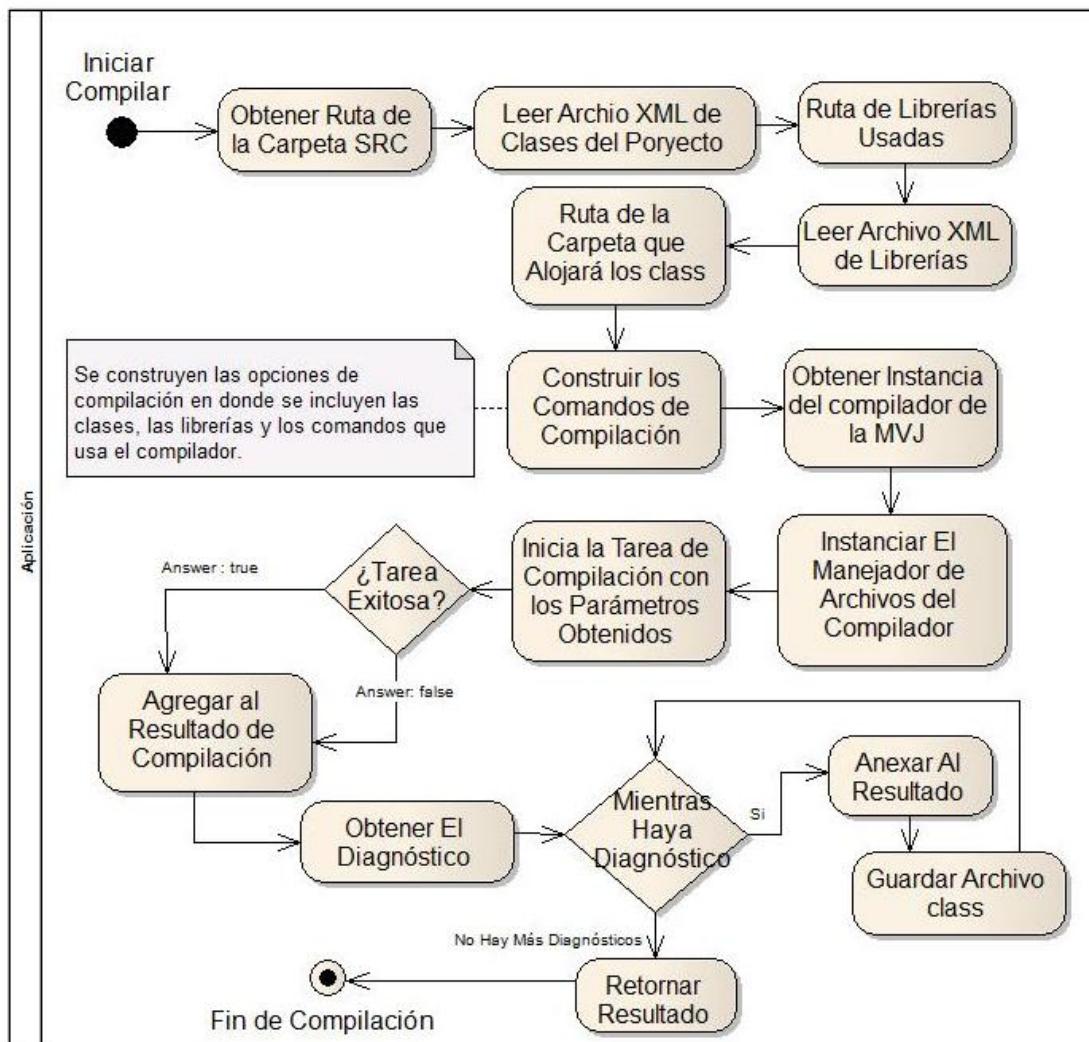


Diagrama 41. Descripción del Proceso Compilar

Especificación

1. El proceso de Compilación consiste en iterar sobre los diferentes archivos fuentes, y referencias a librerías, para verificar si su sintaxis es correcta.

2. En la aplicación llega la solicitud de compilar un proyecto, de manera que el proceso inicialmente conoce la ruta exacta del proyecto que será ejecutado.
3. Dentro la anterior ruta existe una carpeta llamada src que aloja todos los archivos fuente del proyecto. Entonces la aplicación obtiene la ruta de esta carpeta y además lee el archivo donde se almacena el índice de las clases y paquetes.
4. La aplicación además debe obtener la ruta de las librerías que el proyecto usa, y lee el archivo xml de librerías para obtener un índice de las mismas.
5. También es necesario conocer la ruta en la cual los archivos class generados por el compilador será almacenados.
6. La aplicación obtiene una instancia del compilador que está en JDK instalado en el servidor.
7. La aplicación a través de la anterior instancia ejecuta la tarea de compilar con todos los parámetros obtenidos en pasos anteriores.
8. Si la compilación fue exitosa, entonces al resultado se le agrega un atributo de que fue ok. Pero si no fue así el atributo almacenará un false.
9. Luego a esto se obtiene el diagnóstico de la compilación, ya sea porque se presentaron errores o advertencias.
10. Si hay diagnósticos se recorren para anexar la eventualidad al resultado.
11. Por último se guarda el archivo class generado en la ruta indicada para estos archivos. Con esto terminará el proceso.

9.6.10.2. Construir Jar Ejecutable

Construir el archivo JAR ejecutable es el recurso primordial para la ejecución del proyecto, porque finalmente tiene empaquetado todo lo necesario para que la ejecución se lleve a cabo. En este archivo ejecutable estarán todas las clases compiladas, un archivo de MANIFEST donde se especifica el nombre del JAR y la clase principal. El diagrama 42 describe el proceso de generar el ejecutable.

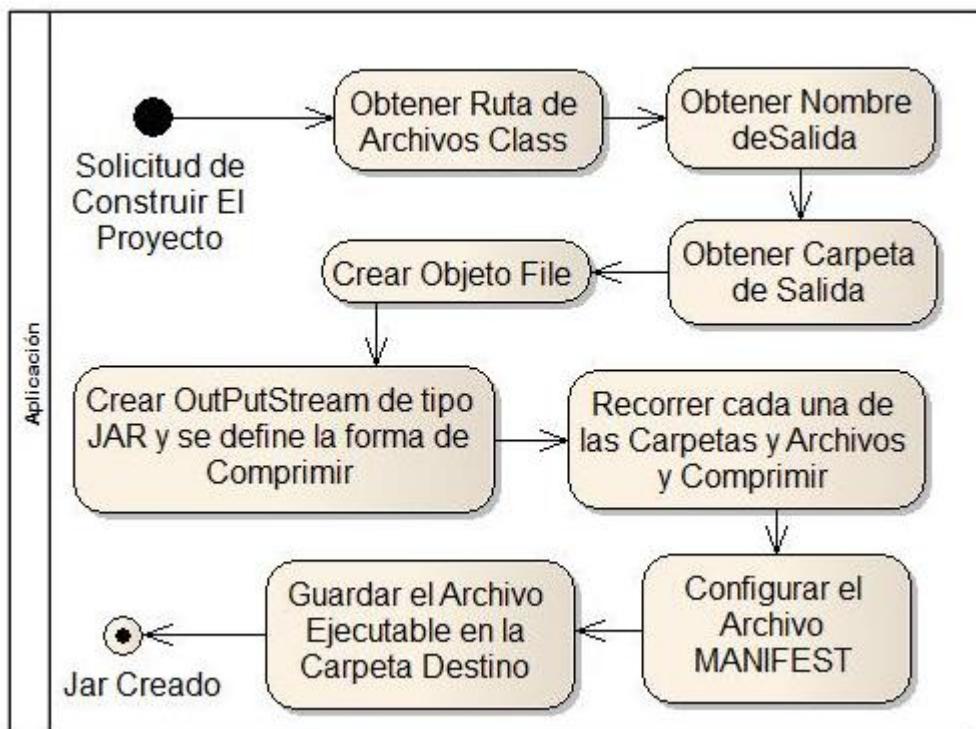


Diagrama 42. Descripción del Proceso Generar Proyecto Ejecutable

Especificación

1. El proceso parte de la solicitud por parte del proceso de ejecutar, que necesita que el proyecto se comprima en un archivo *JAR*, y que a su vez este sea configurado para que sea ejecutable.
2. Para esto es necesario las rutas de los paquetes, clases y librerías, donde en cada fichero encontrado será comprimido.
3. También es necesario la ruta en donde se alojará el archivo *JAR* generado.
4. Se crea un objeto de tipo *File*.
5. Se crea un objeto de tipo *Out Put Stream*, para que con ayuda de éste fuera posible construir el archivo y almacenarlos en la carpeta dispuesta.
6. El algoritmo inicia con un proceso iterativo que irá por cada uno de los ficheros y agregará al archivo que se está comprimiendo.
7. Se configura el archivo *MANIFEST* que dictará las propiedades que el archivo *JAR* tendrá para pueda ser ejecutado.
8. Por último el archivo generado será alojado en la carpeta que se le ha indicado y terminará el proceso.

9.7. DIAGRAMA DE DESPLIEGUE

El diagrama de despliegue ayuda a ver una abstracción de la distribución de los diferentes nodos y componentes que intervienen en la ejecución de una aplicación.

El diagrama 43 muestra la distribución de los diferentes nodos y componentes que intervienen en la aplicación.

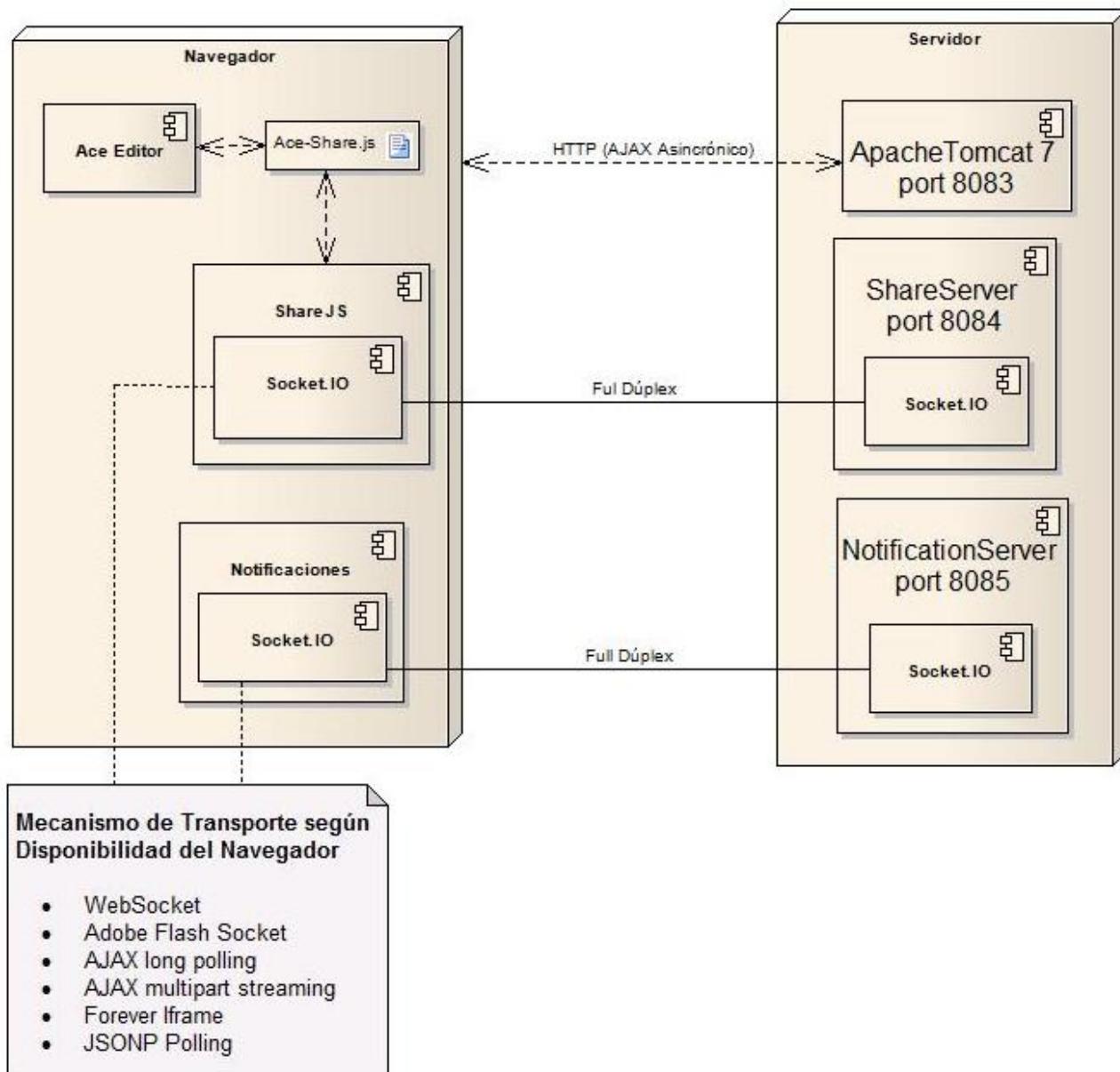


Diagrama 43. Diagrama de Despliegue de la Aplicación

El diagrama de despliegue anterior muestra claramente que la aplicación cuenta con un servidor físico, en donde se instancian esencialmente tres componentes, donde cada uno escucha por puertos diferentes y cumplen con roles específicos.

- *Apache Tomcat* dispuesto para atender todas las solicitudes que involucren acceso a la funcionalidad de la aplicación.
- *ShareServer* es el componente que se encarga de la gestión y control de la implementación de la concurrencia.
- *NotificationServer*, es el componente encargado de toda la gestión de las notificaciones de los cambios ocurridos en el proyecto.

La aplicación está pensada para usar a través de un navegador web, y cuando es abierta, se instanciarán los componentes que están involucrados con el funcionamiento de la aplicación.

- *Ace Editor*: Componente que ofrece un editor con resaltado de código con soporte de varios lenguajes.
- *Notificaciones-Socket.IO*. Este es el componente que establece un canal de comunicación, usando el mejor mecanismo de transporte para transmitir los eventos.
- *ShareJS*, es la instancia en el lado del cliente que modifica los cambios sufridos en el documento editado, cerciorándose de mantener una misma versión en todo momento.

Las comunicaciones entre los nodos se establecen de la siguiente manera:

- Entre el Apache Tomcat y el Navegador se establecerán peticiones HTTP a través de AJAX.
- Las notificaciones y la concurrencia tienen una instancia del componente Socket.IO, por el cual se transmitirán los eventos que acontecen entre servidor y cliente a través del mecanismo de transporte que el navegador tenga disponible.

CONCLUSIONES

Con el desarrollo de ECO – Entorno Colaborativo Online – se logró conseguir una herramienta bastante útil para el Plan de Estudios de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander. Con ECO podrán fomentar en los estudiantes, el trabajo en equipo, ya que esta aplicación integró un concepto conocido como Transformación Operacional, que básicamente les permite a un grupo de programadores que comparten un proyecto en la aplicación, editar todos en la misma copia del proyecto, y cada uno de ellos verá los cambios que se presentan durante la edición. Para explicar mejor esta funcionalidad, se usará Google Docs, hoy llamado Google Drive, y se creará un documento, y se le compartirá con privilegios de edición a determinado usuario. Cuando los dos usuarios abren el proyecto cada uno podrá escribir en el proyecto y siempre se mantendrá una única copia del mismo.

ECO les permite a sus programadores crear programas en ambientes desktop, que puedan cargar librerías externas, por ejemplo, si quisiera conectarse a una base de datos, debería cargar en el proyecto el archivo Jar que actúa de driver.

Se determinó durante el estudio de las generalidades de JAVA que el método más adecuado para lanzar la ejecución de los programas escritos en ECO, es a través de applet y jar firmados, que soliciten autorización al cliente para poder usar la Máquina Virtual del Cliente, donde finalmente se ejecutará el programa escrito.

Se determinó a demás que la forma en que ECO manejaría su persistencia de Datos es a través de archivos XML, tanto para los usuarios, como para los proyectos como tal.

ECO es la aplicación que puede ser usada en clases de programación para llevar a cabo actividades que impliquen construir algún programa como práctica de clase, y de esta manera fomentar la participación y el interés por programar.

Durante el trabajo investigativo realizado en el proyecto, se puede destacar que existe una clara incidencia en respuestas de que el IDE que usan se

demora en cargar, y que les gustaría contar con una herramienta en línea que les permitiera hacer lo mismo que con la IDE convencional.

Es necesario concluir que ECO no fue creado con fines de enseñanza y entrenamiento. Es una herramienta que estará disponible para que se registren y empiecen a disfrutar de sus funciones básicas, avanzadas, del constructor de interfaces gráficas, y las funciones novedosas de las notificaciones y edición concurrente.

ECO solo requiere de una conexión a internet y un navegador, y por supuesto asume que el cliente tiene instalado en su computador la máquina virtual.

En el desarrollo del proyecto se percibe la necesidad que tenemos como profesionales, en el área de tecnologías emergentes, y ver las oportunidades de documentarnos, ejecutar demos y lograr construir algunas implementaciones donde se pueda integrar y probar las nuevas tendencias en el desarrollo de software en el mundo.

RECOMENDACIONES

El proyecto parte con la idea de construir un entorno de desarrollo integrado online, disponible para los estudiantes del Plan de Estudios de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander. Inicialmente se planteó que esta aplicación debía ofrecer la funcionalidad básica que todo IDE comparte, y además que se convierta en la herramienta en donde los estudiantes del programa tuvieran la oportunidad de escribir pequeños programas, que los pudiesen compilar y ejecutar.

Durante el desarrollo del proyecto, se presentaron algunas tecnologías que al ser integradas a la aplicación proporcionaron un valor agregado a su funcionalidad, permitiendo de esta manera construir un producto cada vez más completo, e incluso con características innovadoras que los IDEs convencionales no ofrecen actualmente.

Es claro que todo proyecto debe definir un alcance, ya que este puede resultar interminable, porque el contexto tecnológico en el que vivimos es muy cambiante, y cada vez surgen nuevas herramientas, técnicas, metodologías, tecnologías que pueden ayudar a mejorar nuestros productos en versiones posteriores.

En el desarrollo del proyecto surgieron una variedad de ideas, algunas fueron implementadas y otras por limitaciones de tiempo fueron descartadas, pero el objetivo es que en nuevas versiones, aquellas ideas descartadas puedan ser integradas para conseguir un producto cada vez más completo.

Ahora, con el propósito de mejorar la aplicación, y que ésta pueda seguir evolucionando se recomienda que:

- El Plan de Estudios de Ingeniería de Sistemas adopte la aplicación en sus actividades académicas, para que los estudiantes cuenten con una herramienta en donde puedan desarrollar sus actividades y proyectos de sus clases de programación.

- Fomentar el trabajo en equipo, desarrollando proyectos colaborativos a través de la aplicación, en donde grupos de estudiantes puedan construir proyectos, usando la edición colaborativa y concurrente que la aplicación les permite realizar.
- Con el fin de obtener versiones más completas, se recomienda implementar en la aplicación el concepto de grupos académicos, en donde un docente de algún curso de programación, pueda establecer un grupo de estudiantes que trabajen en determinado proyecto. También sería útil que a través de la aplicación un docente tuviera la opción de crear actividades de programación, en donde él estableciera un requerimiento en un enunciado y que incluso pudiese a compartir un proyecto base, para que aquellos estudiantes que pertenezcan a su grupo tengan compartida la información para implementarlo.
- Es recomendable lograr que la IDE soporte más lenguajes, ya que actualmente solo permite desarrollar proyectos en JAVA.
- Similar a la anterior recomendación, dotar a la IDE de la capacidad de construir proyectos bajo arquitectura web, ampliaría su capacidad de desarrollo.
- También se recomienda brindarle al programador un ambiente más profesional, logrando integrar algún servidor git, que permita tener un control de historial de cambios, que le permita a los programadores volver a un estado anterior del proyecto. Esta recomendación junto con la concurrencia que ya está implementada, ofrecería un ambiente más completo a programadores avanzados, con experiencia en desarrollo y trabajo en equipo.

REFERENCIAS BIBLIOGRÁFICAS

1. AMAYA TORRADO, Yegny Karina y HERRERA ANGARITA, Lady Torcoroma. Identificación de las causas que generan problemas en el aprendizaje de fundamentos de programación de computadores en las facultades de ingeniería de las universidades de la ciudad de Cúcuta. Trabajo de Grado Ingeniería de Sistemas. Cúcuta N.de S. Universidad Francisco de Paula Santander. Facultad de Ingeniería Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura TIS 371.334A489i.
2. VARGAS C. José Gilberto. BUSTOS RIOS Ligia Stella y MORENO LAVERDE, Ricardo. Propuesta para Aumentar el Nivel Académico, Minimizar la Deserción, Rezago y Repitencia Universitaria por Problemas de Bajo Rendimiento Académico en la Universidad Tecnológica de Pereira, en el Programa Ingeniería de sistemas y Computación. Scientia et Technica Año XI No 28 Octubre de 2005 UTP. ISSN 0122-1701. Disponible: <http://revistas.utp.edu.co/index.php/revistaciencia/article/view/6841/4029>
3. Autoevaluación del programa de pregrado de Ingeniería de Sistemas de la Universidad Nacional de Colombia, Sede Bogotá. Facultad de Ingeniería. Departamento de Ingeniería de Sistemas e Industrial. 2005. 63p. Disponible: <http://www.ing.unal.edu.co/viceacad/acreditacion/sistemas/SistemasCNAv1.pdf>
4. RODRÍGUEZ RODRÍGUEZ, César. IDEWeb (Entorno de Desarrollo Integrado en Web). Proyecto de grado. Universidad de Oviedo. Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo. Disponible: http://www.di.uniovi.es/~juanrp/docencia/pfc/pfc/IDEWeb_Entorno_de_Desarrollo_Integrado_en_Web.pdf
5. MediaWiki: es un paquete de software libre de código abierto, escrito en PHP, escrito originalmente para Wikipedia. Es ahora utilizado por muchos más proyecto wikis existentes.
6. JENKINS, Jam. BRANNOCK, Evelyn y DEKHANE, Sonal. Innovation in an Online IDE. Tutorial Presentation. En: Journal of Computing Sciences in Colleges. Vol. 25 Issues 5, Mayo 2010. ACM Digital Library.
7. JNLP: (Por sus siglas en inglés Java Network Launching Protocol) El archivo JNLP es un XML donde se describe la forma de descargar y cargar una aplicación en particular.

8. CARAVACA MORA, Oscar Mauricio. Diseño de un Entrono de Desarrollo Integrado para una Unidad Controladora de Procesos. Proyecto de Grado Ingeniería en Electrónica. Instituto Tecnológico de Costa Rica. Escuela de Ingeniería en Electrónica. 132p. 2010. Disponible: http://www.ie.itcr.ac.cr/einteriano/control/Laboratorio/Proyectos/2010_IDE/IDE_Informe_Final.pdf
9. VILLALOBOS, Jorge A y CASALLAS Rubby. Fundamentos de Programación. Aprendizaje Activo Basado en Casos. Un enfoque moderno usando Java, UML, Objetos y Eclipse. 1^a Edición Pearson Prentice Hall. 384p. 2007. ISBN 9702608465.
10. CEBALLOS SIERRA, Francisco Javier. Java 2. Curso de programación. 3^a edición. Librería y Editorial Microinformática. 880p. 2005. ISBN 8478976866.
11. ORACLE. Java SE Documentation. Java SE 7 Documentation. <http://download.oracle.com/javase/7/docs/>
12. GROVE, Ralp F. *Web-Based Application Development*. Editorial Jones & Bartlett Pub. 329P. ISBN 0763759406.
13. Art. *Cloud Computing: Security Risk*. La'Quata Sumter. Florida A&M University. Department of Computer and Information Sciences. En: Proceedings of the 48th Annual Southeast Regional Conference. ACM Digital Library.
14. Art. Dave Durkee. Why Cloud Computing Will Never Be Free. En: Magazine Communications of the ACM. Vol. 53 Issues 5. Mayo 2010. ACM DIGITAL LIBRARY.
15. ALCOCER, Alberto. Blog Oficial SocieTIC Sociedad y Tecnología. Categoría Cloud Computing. <http://www.societic.com/category/cloud-computing>
16. Ronald E. Jeffries. *XProgramming.com An Agile Software Development Resource*. “What is Extreme Programming?”. <http://xprogramming.com/what-is-extreme-programming/>
17. Kent Benck, *Extreme Programming Explained. Embrace Change. Chapter 7 Four Values*. Addison-Wesley Professional ©2004.
18. Lisa Crispin, Tip House. *Testing Extreme Programming. Chapter 1 An Overview: Overview of XP*. Addison-Wesley Professional

19. Kent Beck. *eXtreme Programming eXplained: Embrace Change. 2nd Edition.* Addison-Wesley Professional. Chapter 21. *Lifecycle of an Ideal XP Project.*
20. VOSS Greg. Programación orientada a objetos. Una introducción. McGraw-Hill Companies Inc. 597p. 1194. ISBN 9701004930.
21. Creative Commons. Código Legal. Atribución no comercial compartir igual 2.5 (Colombia). <http://creativecommons.org/licenses/by-nc-sa/2.5/co/legalcode>
22. CONTRATO DE LICENCIA DE CÓDIGO BINARIO. Sun Microsystems, Inc. *Java 2 Platform Standard Edition Runtime Environment 5.0.* Disponible: http://www.java.com/es/download/license_jre5.jsp
23. GNU GENERAL PUBLIC LICENSE. <http://www.gnu.org/licenses/>
24. GNU AFFERO GENERAL PUBLIC LICENSE . <http://www.gnu.org/licenses/>
25. Universidad Francisco de Paula Santander. Estatuto Estudiantil. Acuerdo 065 26 de agosto de 1996.
26. ROSALES SALES, Euline Esperanza. VELASCO SÁNCHEZ, Diana Carolina y SUÁREZ GARCÍA, German Darío. Estudio sobre deserción - retención de los estudiantes del Plan de Estudios de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander y perfil ocupacional del egresado. Proyecto de Grado Ingeniería de Sistemas. Universidad Francisco de Paula Santander. Facultad de Ingeniería. Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura TIS 371.2913 R788E
27. AMAYA TORRADO, Yegny Karina y HERRERA ANGARITA, Lady Torcoroma. Identificación de las causas que generan problemas en el aprendizaje de fundamentos de programación de computadores en las facultades de ingeniería de las universidades de la ciudad de Cúcuta. Trabajo de Grado Ingeniería de Sistemas. Cúcuta N. de S. Universidad Francisco de Paula Santander. Facultad de Ingeniería Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura TIS 371.334A489i.
28. VERGEL ARÉVALO, Gina Paola y TARAZONA CLARO, Fernando Alberto. Creación de un software educativo para el desarrollo de las habilidades del razonamiento abstracto. Proyecto de Grado Ingeniería de Sistemas. Universidad Francisco de Paula Santander. Facultad de Ingeniería. Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura: TIS 005.1 V495c

29. VERA CONTRERAS, Milton de Jesús y Hernández Molina Mabel. Creación de un portal académico como herramienta didáctica de apoyo a los procesos educativos en los planes de estudio de pregrado modalidad presencial en la Universidad Francisco de Paula Santander que integre nuevas tecnologías de comunicación e información. Proyecto de Grado Ingeniería de Sistemas. Universidad Francisco de Paula Santander. Facultad de Ingeniería. Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura: TIS 005.1 H558c
30. *The Computer Science And Engineering-Handbook. Editor-In-Chief, Allen B. Tucker, Jr. A CRC Handbook Publishes in Cooperation with ACM, The Association for Computing.*
31. Nell B Dale, John Lewis. Computer Science Illuminated. 2th Edition. Editorial Jones & Bartlett Learning, 2004 ISBN 0763707996, 9780763707996, 699 págs.
32. Rice University JavaPLT. Sitio web oficial del grupo JavaPLT de Rice University, en donde se encuentra disponibles información acerca de proyectos e investigaciones llevadas a cabo por dicho grupo. <http://www.cs.rice.edu/~javaplt/>
33. Sitio web oficial de DrJava. <http://www.drjava.org/>
34. School of Computing. Teaching, learning, research and innovation. Sitio web de la escuela de computación de University of Kent, y donde se encuentra almacenada la información de proyectos e investigaciones informáticas llevadas a cabo en la universidad. <http://www.cs.kent.ac.uk/>
35. BlueJ – The interactive Java environment. Entorno integrado de Java diseñado específicamente para la enseñanza introductoria. Sitio web oficial <http://www.bluej.org/>
36. Sitio web oficial de JCreator. <http://www.jcreator.com/>
37. Sitio web oficial de JetBrains. En este apartado se encuentra información acerca de la compañía. <http://www.jetbrains.com/company/index.html>
38. Sitio web oficial de IntelliJ IDEA. <http://www.jetbrains.com>
39. Sitio web oficial de eclipse. En este apartado se encuentra información acerca de la historia del software entre otros. <http://www.eclipse.org/org/#about>

40. Sitio web oficial de Eclipse. Eclipse.org es un portal donde se pueden encontrar diversos plugins y recursos importantes para facilitar el desarrollo de software. <http://www.eclipse.org/>
41. Sitio web oficial de Netbeans. En este apartado se encuentra información acerca de la creación y evolución que ha tenido el software de NetBeans. <http://netbeans.org/about/history.html>
42. Sitio web oficial de NetBeans. netbeans.org es el portal de la comunidad de código abierto dedicado a construir un IDE de primera clase. <http://www.netbeans.org/>
43. Sitio oficial de JBuilder. <http://www.embarcadero.com/products/jbuilder>
44. Sitio oficial de Anjuta. Sección dedicada a compartir información acerca del equipo de desarrollo que se encuentran vinculados al proyecto. <http://www.anjuta.org/team.html>
45. Sitio oficial de Anjuta DevStudio. <http://www.anjuta.org/index.html>
46. Sitio oficial de Geany. Apartado dedicado a informar acerca de los autores de los autores del proyecto. <http://www.geany.org/Main/Authors>
47. Sitio oficial de Geany. Apartado dedicado acerca del proyecto. <http://www.geany.org/Main/About>
48. Sitio web oficial de SlickEdit Inc. En este apartado se encuentra información acerca de la compañía y sus productos software. <http://www.slickedit.com/about>
49. Sitio oficial de GreenFoot. En esta sección se especifican las entidades y personas involucradas con el proyecto. <http://www.greenfoot.org/about/contributors.html>
50. Sitio oficial de GreenFoot. Sección dedicada a especificar los aspectos y características del proyecto, más específicamente lo correspondiente a ¿Qué es GreenFoot?. <http://www.greenfoot.org/about/whatis.html>
51. METODOLOGÍA DE LA INVESTIGACIÓN. Roberto Hernández Sampieri, Carlos Fernández Collado, Pilar Baptista Lucio. Mc Graw Hill.

52. KRAMER Douglas. *The Java Platform A White Paper. A Look Inside the Java Platform.* 1996. <http://java.sun.com/docs/white/platform/javaplatform.doc2.html>
53. *The Java Virtual Machine Specification.* LINDHOLM Tim, YELLIN Frank. Second Edition. Chapter 1-Introduction. View HTML in: <http://docs.oracle.com/javase/specs/jvms/se5.0/html/Introduction.doc.html>
54. VENNERS, Bill. Inside the Java Virtual Machine. Capítulo 5.The Java Virtual Machine. The Architecture of the Java Virtual Machine.McGraw-Hill. 1997
55. KRAMER Douglas, The Java™ Platform A White Paper. Java Compile and Runtime Environments. 1996. Link:
<http://java.sun.com/docs/white/platform/javaplatform.doc3.html>
56. Aprenda Java como si estuviera en primero. Tecnun. Campus Tecnológico de la Universidad de Navarra. Escuela Superior de Ingenieros. SAN Sebastian 2000. Pág 3.
<http://www.tecnun.es/asignaturas/Informat1/Ayudalnf/aprendainf/Java/Java2.pdf>
57. The Apache Ant Project. Apache Ant™ 1.8.2 Manual. Description Javac.
<http://ant.apache.org/manual/Tasks/javac.html>
58. GREANIER, Todd. *Java Foundations.* Editorial John Wiley & Sons. ISBN 9780782143737. Pág 24.
59. NIEMEYER, Patrick; KNUDSEN Jonathan. Learning Java. Edición 3. Editorial O'Reilly Media Inc. 2005. ISBN 9780596008734. Capítulo 3. Pág. 72
60. VENNERS Bill. *Inside the Java Virtual Machine. McGraw-Hill. Chapter 6 – The Java Class File.* ISBN 0079132480
61. *The Java Virtual Machine Specification.* LINDHOLM Tim, YELLIN Frank. Second Edition. Chapter 4-The Class File Format.
62. Barte Van Rompaey. *Java and .NET: A Look Into Today's Virtual Machine Technology.* Universiteit Antwerpen. Departement Wiskunde en Informatica. Chapter 5 – File Formats. 2003-2004.
63. API - Java Application Programming Interface- definition-
<http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html>

64. Oracle Corporation. *Java Platform Standard Edition 7 Documentation*. Link:
<http://download.oracle.com/javase/7/docs/>
65. T. Ylonen, C. Lonwick. *The Seccure Shell (SSH) Protocol Architecture. Introduction*. <http://www.ietf.org/rfc/rfc4251.txt>
66. Oracle Corporation. *The Java Tutorials. Java Network Launch Protocol*.
<http://docs.oracle.com/javase/tutorial/deployment/deploymentInDepth/jnlp.html>
67. Sun Java Software. *Java Network Launching Protocol & API Specification (JSR-56). Application Environment*.
<http://jcp.org/aboutJava/communityprocess/first/jsr056/jnlp-1.0-proposed-final-draft.pdf>
68. Oracle Corporation. *Code Samples and Apps. Applets*.
<http://java.sun.com/applets/>
69. Jorge Sánchez. Java 2 incluye Swing, Threads, programación en red, JavaBeans, JDBC y JSP/Servlets. Applets.
<http://www.jorgesanchez.net/programacion/manuales/Java.pdf>
70. Oracle Corporation. *JAR File Specification. Introduction*.
<http://docs.oracle.com/javase/1.3/docs/guide/jar/jar.html#Intro>
71. Oracle Corporation. *What Applets Can and Cannot DO*.
<http://docs.oracle.com/javase/tutorial/deployment/applet/security.html>
72. Oracle Corporation. *Chapter 10: Signed Applets*.
<http://java.sun.com/developer/onlineTraining/Programming/JDCBook/signed.html>
73. SecurityManager.
<http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/SecurityManager.html>
74. Philippe Kruchten, *The Rational Unified Process An Introduction*, Addison Wesley, 2001
75. *The Unified Modeling Language Reference Manual*. RUMBAUGH James, JACOBSON Ivar, BOOCHE Grady. La guía completa del proceso unificado escrita por sus creadores. *Chapter 5, Use Case View*. Pág 63.

76. Oracle Corporation. JavaServer Pages Overview. Contents.
<http://www.oracle.com/technetwork/java/overview-138580.html>
77. Guía Breve de Tecnologías XML.
<http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML>
78. Sitio oficial de CoffeeScript. <http://coffeescript.org/>
79. Introducing JSON. <http://json.org/>
80. OT FAQ. Operational Transformation Frequently Asked Questions and Answers. What is OT?. <http://cooffice.ntu.edu.sg/otfaql/>
81. Pascal Molli, Gérald Oster, Hala Skaf-Molli, Abdessmad Imine. "Using the Transformation Approach to Build a Safe and Generic Data Synchronizer": Transformation Approach. [ACM](#) New York, NY, USA ©2003.
82. Nicolas Vidot, Michelle Cart, Jean Ferrié, Maher Suleiman. "Copies Convergence in a distributed real-time collaborative environment". General Problems. [ACM](#) New York, NY, USA ©2000
83. Reuven M. Lerner. At the Forge. Nodejs. Linux Journal, volumen May 2011 Issue 205.
84. Nagi Letaifa Dans. A Full Javascript Architecture, Part One-NodeJS. Zenika Architecture Informatique.
85. Introducing SocketIO v.8. <http://socket.io/>
86. Joseph Gentle. ShareJS-Live concurrent editing in your app. <http://sharejs.org/>
87. Joseph Gentle. ShareJS: library for Google Wave –style collaboration. ShareJS launch talk slider. The Universal Runtime
88. Jaroslav Pokorný, "NoSQLDatabases: a step to database scalability in Web environment" Charles University, Malostranske, Czech Republic. [ACM](#) New York, NY, USA ©2011
89. Rick Cattell. "Scalable SQL and NoSQL Data Stores". ACM SIGMOD Record. Volume 39 Issue 4, December 2010

90. JQuery write less, do more. <http://jquery.com/>
91. Ace Ajax.org Cloud9 Editor. <http://ace.ajax.org/>.
92. JavaScript Ajax Library for Building Great Web Apps.
<http://dhtmlx.com/docs/products/dhtmlxSuite/index.shtml>
93. Johannes Koskinen. *Metaprogramming in C++: Terminology*. March 9, 2004
94. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Sponsor: Software Engineering Standards Committee of the IEEE Computer Society. IEEE Std 1471-2000, Approved 21 September 2000
95. Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language*. Oxford University Press, New York, 1977.
96. Protocolo *Infinote*, Comunicación por Grupos
<http://infinote.org/Protocol/Groups/>
97. Implementación en Javascript del Protocolo *Infinote*.
<https://github.com/sveith/jinfinote>
98. Etherpad. Implementación de edición colaborativa en tiempo real.
<http://etherpad.org/>
99. Protocolo Google Wave OT.
<http://www.waveprotocol.org/whitepapers/operational-transform>
100. Urs Hözle, Senior Vice President, Operations & Google Fellow. Update on Google Wave – Official Blog <http://googleblog.blogspot.com/2010/08/update-on-google-wave.html>
101. Soporte de Google. Status of Google Wave
<http://support.google.com/bin/answer.py?hl=en&hlrm=en&answer=1083134&rd=1>